Internet tietovaraston lähteenä

Markku Kasila

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Erkki Mäkinen
Huhtikuu 2008

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Markku Kasila: Internet tietovaraston lähteenä
Pro gradu -tutkielma, 84 sivua, 37 liitesivua
Huhtikuu 2008

Tutkimus käsittelee automaattista tietojen keruuta internetistä tietovarastoon. Perinteisesti tietovaraston tiedot on koottu yritysten operatiivisista järjestelmistä, mutta tässä tutkimuksessa tietovarastoinnin periaatteita sovelletaan julkisten tietojen taltiointiin.

Tutkimuksessa selvitetään sekä tietovarastoinnin että internetistä tapahtuvan tiedonkeruun pääperiaatteita. Pääpaino tutkimuksessa on kuitenkin prosessin aikana syntyneissä toimintamalleissa ja ohjelmallisissa ratkaisuissa, joiden avulla tiedonkeruu on automatisoitavissa kohdealueen ollessa ennakkoon rajattu.

Käytännön esimerkkitietovarastona on palloilulajien, lähinnä jalkapallo tai jääkiekko, tilastotietokanta, jonka automaattiseen päivitykseen internet-haut ovat käytännössä ainoa mahdollisuus varsinaisten operatiivisten kantojen puuttuessa.

Sisällys

1. Johdanto

Tutkimuksen tavoitteena on kartoittaa niitä ongelmia, joita kohdataan poimittaessa tietovarastoon julkisia tietoja internetistä. Käytännön esimerkkinä on palloilulajien tilastotietoja keräävä tietovarasto, jonka tiedot joudutaan poimimaan internetistä varsinaisten operatiivisten järjestelmien puuttuessa.

Koska tietovaraston lähteenä ovat perinteisesti vain yrityksen omat operatiiviset järjestelmät, internetin käyttö tietolähteenä mahdollistaa aihealuetta koskevan tiedon yhdistelyn lukuisista eri lähteistä, mikä taas mahdollistaa kerätyn tiedon ympärille rakennettavat lisäarvopalvelut.

Tietovaraston latausprosessi sisältää yleensä kolme erillistä vaihetta: tiedon poiminta, muokkaus ja varsinainen siirto tietovarastoon. Tutkimus keskittyy ensisijaisesti näistä kahteen ensimmäiseen vaiheeseen, jossa tieto poimitaan alkuperäisestä lähteestä ja formalisoidaan sellaiselle tasolle, että se on siirrettävissä tietovarastoon.

Luvussa 2 käsitellään tietovarastoinnin pääperiaatteita yleisellä tasolla keskittyen tietovaraston määritelmiin ja käsitteistöön. Tietovaraston ominaisuudet kuuluvat tämän tutkimuksen piiriin ensisijaisesti vain niiltä osin, kuin ne liittyvät varsinaiseen tutkimusaiheeseen ja siksi tässä tutkimuksessa ei käsitellä esimerkiksi tietovaraston suunnitteluprosessia eikä metadataa.

Luvussa 3 tutustutaan internet-tiedonkeruun haasteisiin sekä yleisesti että tietovarastoinnin kannalta. Samalla perustellaan keruun tarpeellisuutta, vaikka internetiä jo voitaisiin pitää tietovarastona sinänsä.

Esimerkkitietovarastona oleva palloilulajien tietovarasto kuvataan tarkemmin luvussa 4. Palloilutietovaraston käytännön toteutus ei kuulu tämän tutkimuksen alueeseen. Sen sijaan sen todennäköisin tietokantarakenne on suunniteltu prosessin aikana.

Luvussa 5 esitellään ongelma-alueeseen liittyviä ohjelmointimalleja, joiden avulla tiedon keruuta internetistä voidaan automatisoida. Ohjelmointiesimerkit on toteutettu Microsoft Excelin makrokielellä (Visual Basic for Applications). Tärkeimmät funktiot on kuitenkin suunniteltu niin, että ne ovat siirrettävissä muihin ympäristöihin.

Luvussa 6 tarkastellaan esimerkkiprosesseja, jossa tietoja poimitaan internetistä ohjelmallisesti relaatiomallin mukaiseen tietokantamuotoon, josta ne myöhemmin on tarkoitus siirtää varsinaiseen tietovarastoon. Esimerkeissä hyödynnetään luvussa 5 esiteltyjä proseduureja ja lähestymistapoja.

2. Tietovarastointi

Termi ”tietovarasto” alkoi yleistyä 1990-luvulla, jolloin se käsitteellisesti irtautui perinteisistä tietokannoista, vaikka teknisesti tietovarastot edelleenkin perustuvat usein tietokantaratkaisuihin. Tietovarastoinnin perusajatus on tuottaa informaatiota, joka on erillään yrityksen varsinaisista operatiivisista kannoista ja joka on erityisesti muokattu valmiiksi juuri raportointitarpeita varten.

Jo 1970-luvulla esitettiin ensimmäisen kerran ajatuksia muokatun datan sijoittamisesta eri alustalle johdon päätöksenteon tukemista varten. Ajatusten taustalla olivat tuolloin seuraavat tarpeet [Gray & Watson 1998]:

- pääsy tietoihin ilman syvällistä atk-tuntemusta
- järjestelmän lyhentynyt vasteaika
- datan eheyden ja luotettavuuden paraneminen.

Ajatuksen konkretisoituminen vasta 1990-luvulla voidaan nähdä seurauksena kasvaneesta informaation määrästä kaikilla elämän osa-alueilla. Apuvälineitä tarvitaan vasta silloin, kun informaation määrä on paisunut niin suureksi, että operatiivisten järjestelmien tarjoamat keinot eivät enää riitä löytämään informaatiosta ydinkohtia päätöksenteon tueksi. Toinen käytännön syy on ollut se, että yrityksissä on yleensä useita operatiivisia järjestelmiä, joiden sisältöä on voitava tarkastella yhtenä kokonaisuutena.

2.1. Määritelmiä

Tietovarastolla tarkoitetaan järjestelmää, jossa yrityksen tai organisaation (tai vastaavan) operatiivisista tietokannoista kerätään dataa erilliseen tietokantaan ja samalla data organisoidaan uudelleen siten, että se palvelee operatiivista dataa paremmin päätöksentekoa tai muuta organisaatiota koskevaa analyysiä.

Hovi ja muut [2001] määrittelevät tietovarastolle seuraavat ominaisuudet:

- Tiedot ovat operatiivisten perusjärjestelmien tietokannoissa ja niitä käsitellään operatiivisilla järjestelmillä.
- Tiedot poimitaan ja muokataan tietovaraston edellyttämään muotoon. Samalla eri perusjärjestelmien tiedot yhdenmukaistetaan.
- Tietovarasto on nimenomaan tietojen helppoa ja nopeaa hakua varten suunniteltu tietokanta. Latauksessa (päivityksessä) uudet tiedot menevät edellisten perään, mikä mahdollistaa historiatietojen säilymisen.

- Tietovaraston tiedot määritellään ja kuvataan metatiedoissa.
- Tietovaraston tietoja kysellään, analysoidaan ja raportoidaan erilaisilla työkaluilla.

Inmon [2002] on määritellyt tietovarastolle seuraavat usein lainatut ominaisuudet: tietovarasto on aihesuuntautunut (subject-oriented), integroitu (integrated), aikariippuvainen (time-variant) ja pysyvä (nonvolatile).

Tietovarastossa data on organisoitu aihealueen mukaan sen sijaan, että se olisi organisoitu yksittäisten tapahtumien perusteella. Näin ollen se ylittää sovellusten ja osastojen rajat.

Integroinnilla tarkoitetaan tietovaraston yhteydessä ensisijaisesti sitä, että samaa asiaa tarkoittava tieto esitetään tietovarastossa vain yhdellä tavalla. Tuotaessa data tietovarastoon se integroidaan jo varastossa olevan tiedon kanssa siten, että siihen viitataan kaikkialla samalla tavalla. Lisäksi tiedolla on yhtenäinen esitystapa ja samat mittayksiköt tai muut attribuutit.

Aikariippuvuus tarkoittaa sitä, että tiedoilla on aikaleima. Kun operatiivinen järjestelmä kuvaa kohdealueen nykytilaa, tietovarasto kuvaa lisäksi kohdealueen aikaisemmat tilat määritetyillä hetkillä.

Pysyvyyden määre perustuu historiatiedon keskeiseen asemaan, sillä tietovarastosta ei päivitysajoissa poisteta tietoja, vaan aikaisempi tilanne siirretään historiatiedoksi uusien tietojen päivityksen yhteydessä.

2.2. Operatiivisen järjestelmän ja tietovaraston erot

Operatiivisista transaktiopohjaisista järjestelmistä käytetään usein nimitystä OLTP (on-line transactional processing) ja tietovarastopohjaisista ratkaisuista nimitystä OLAP (on-line analytical processing). Inmon [2002] luettelee syitä, joista johtuen jako operatiivisiin ja informatiivisiin kantoihin yleensä on tapahtunut:

- data, joka palvelee operatiivisia tarpeita, on fyysisesti täysin erilaista kuin informatiivisia ja analyyttisiä tarpeita palveleva data
- teknologia, joka tukee operatiivista datan käsittelyä, on perustaltaan erilainen kuin informatiivista datan käsittelyä tukeva
- datoilla on eri käyttäjäkunta
- datan prosessointiin liittyvät piirteet ja toimintatavat ovat erilaisia.

Operatiiviset sovellukset ovat perinteisesti suunniteltu toimintojen ja prosessien ympärille ja niiden data on siksi usein sidoksissa kyseiseen toimintoon. Lisäksi operatiivisen datan vaatimukset ovat riippuvaisia sovelluksen tarpeista ja data perustuu yritystoiminnan sääntöihin. Tietovarasto

taas sisältää dataa, joka on suunniteltu nimenomaan analyyttiseen käyttöön ja päätöksentekoon.

Westermanin [2001] mukaan tietovarastoista saatiin jo niiden ensimmäisissä kehitysvaiheissa kaksi merkittävää etua: tietovarasto ei ole suoraan yhteydessä operatiivisiin järjestelmiin ja tietovarasto mahdollistaa historiallisen datan keräämisen.

Operatiivisten ja informatiivisten tietokantojen tärkeimmät erot ovat Hanin ja Kamberin [2001] mukaan seuraavat:

- Käyttäjä- ja järjestelmäorientoituneisuus: OLTP on asiakasorientoitunut, sillä sitä käyttävät yleensä yksittäiset ihmiset tapahtumien käsittelyyn ja tapahtumakyselyihin. OLAP taas on markkinaorientoitunut, sillä sitä käyttävät johtajat ja analyytikot.
- Datan sisältö: OLTP hallitsee nykydataa, joka on yleensä liian yksityiskohtaista tukeakseen päätöksentekoa. OLAP hallitsee historiallista dataa ja mahdollistaa summaukset ja yhteenvedot eri tasoilla, mikä taas on välttämätöntä päätöksenteossa.
- Tietokantarakenne: OLTP noudattaa yleensä entity-relationship -mallia (ER) ja tietokannan muoto perustuu sovellusten tarpeisiin. OLAP taas perustuu tähti- tai lumihiutalemalliin tukien näin tietovaraston aiheorientoituneisuutta.
- Ajallinen näkemys: OLTP keskittyy ensisijaisesti yrityksen tai sen osaston nykydataan. OLAP taas kattaa tavallaan operatiivisen tietokannan useita versioita johtuen organisaation muutoksista vuosien aikana. OLAP myös sallii sellaisen tiedon liittämisen, joka on peräisin eri organisaatiosta.
- Saantitavat: OLTP:n käyttö koostuu lyhyistä tapahtumasarjoista, joiden osalta järjestelmältä vaaditaan samanaikaisuuden hallintaa ja palautusmekanismeja virhetilanteissa. OLAP-järjestelmään kohdistuvat pyynnöt ovat yleensä ”vain luku” -tyyppisiä.

Inmon ja muut [1999] nostavat esille lisäksi yhtenä merkittävänä erona sen, että operatiivisten järjestelmien pohjalta tehdyt päätökset ovat lyhyen tähtäimen päätöksiä, kun taas tietovaraston pohjalta tehdyt päätökset ovat pitkävaikutteisia ja strategisia.

2.3. Tietovaraston tehtävät

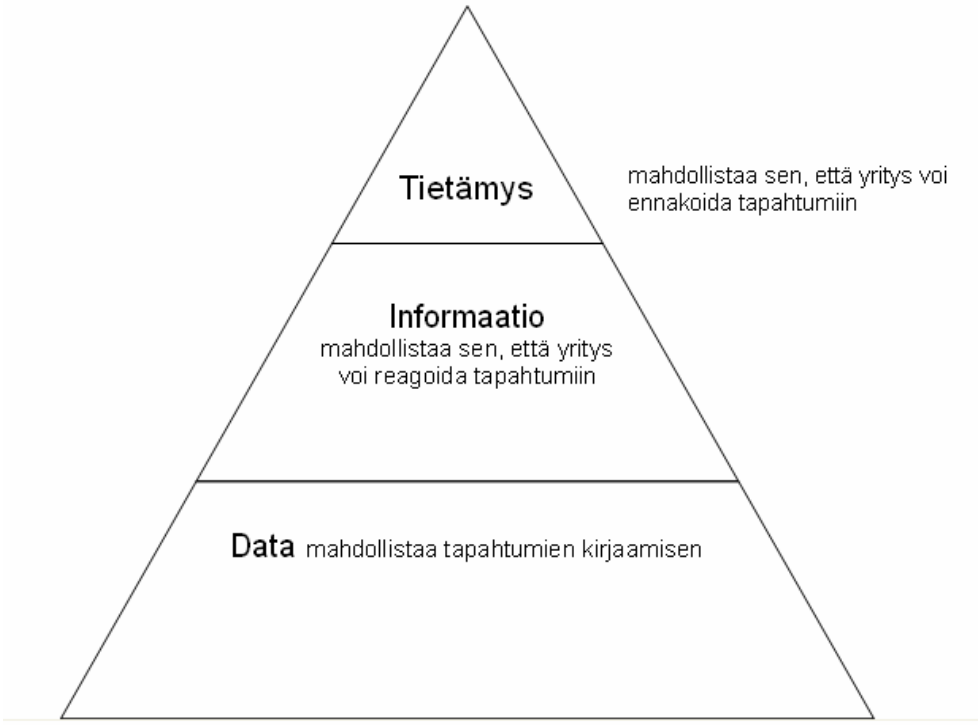
Tietovaraston oleellisin rooli on tuottaa tietoa päätöksenteon tueksi. Tällöin tietovaraston painopiste on informaatiossa eikä datassa itsessään [Gray & Watson 1998].

Hanin ja Kamberin [2001] mielestä tietovarastoa voidaan käyttää kolmeen eri tarkoitukseen. Ensimmäinen näistä on informaation käsittely, jossa pääasiallisia toimenpiteitä ovat kyselyt ja raportointi. Toinen käyttötapa on analyyttinen, jossa tietoa käsitellään OLAP-menetelmillä ja käytön painopiste on tietovaraston moniulotteisen esitystavan mahdollistamissa ominaisuuksissa. Kolmas käyttötapa on tiedon louhinta (data mining), jossa tavoitteena on uuden tietämyksen luominen siten, että tiedonlouhintamekanismit yrittävät havaita datassa uusia tietojen välisiä riippuvuuksia ja malleja. Han ja Kamber pyrkivät myös erottamaan OLAP-järjestelmän ja tiedon louhinnan toisistaan rajaten OLAPin tehtäväksi sen, että se on analyysiä helpottava työkalu kun taas tiedon louhinnan painopiste on sen automaattisessa tietämyksen etsimisessä.

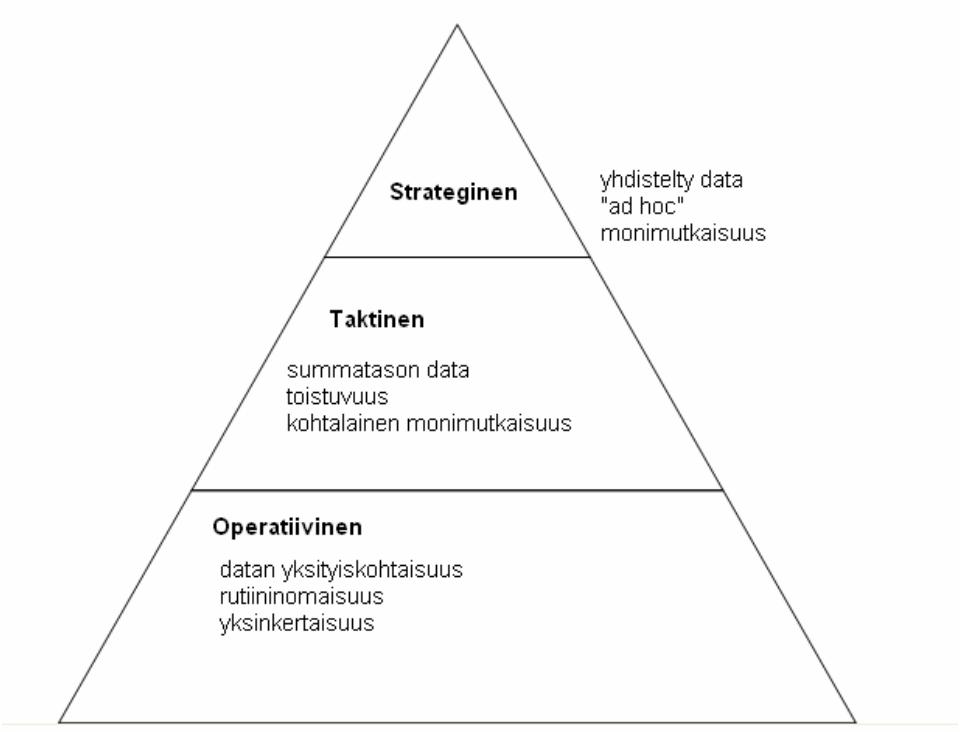
Tyypillisimpiä tietovaraston käyttötapoja ovat raportit ja kyselyt. Raportit voivat olla vakioraportteja tai interaktiivisia raportteja, joissa käyttäjä voi liikkua eri organisaatiotasolta toiselle porautumalla. Käyttäjillä tulee olla myös mahdollisuus määrittää itse tietovarastoon tehtäviä kyselyitä. Tällöin heillä on oltava käytössään tietovaraston rakenteen kuvaus tai sen osa.

Vaikka tietovarastoajattelun kehitys on ollut liiketoiminnan tarpeista ohjautuvaa, se ei käsitteellisesti eikä teknisesti ole sidottu vain yrityselämän tarpeisiin. Yhtä hyvin tietovarastoinnin käsitteistöä ja menetelmiä voidaan soveltaa mihin tahansa tietoon. Tietovarastoa voidaan koota myös julkisesta tiedosta. Tällöin tietojen lähteenä on ensisijaisesti internet, mikä asettaa tietojen poiminnalle erilaisia haasteita kuin operatiivista kannoista tapahtuva poiminta.

Tietovaraston rooli päätöksenteon apuvälineenä voidaan laajentaa seurannan apuvälineeksi. Seuranta johtaa havaintoihin, jotka taas antavat usein aiheen päätöksentekoon. Kuvat 1 ja 2 havainnollistavat tiedon eri tasojen suhdetta päätöksenteon eri tasoihin.



Kuva 1: Datan, informaation ja tietämyksen arvoketju [Kelly 1997].



Kuva 2: Organisaatioiden eri tasojen tietotarpeiden hierarkia [Kelly 1996].

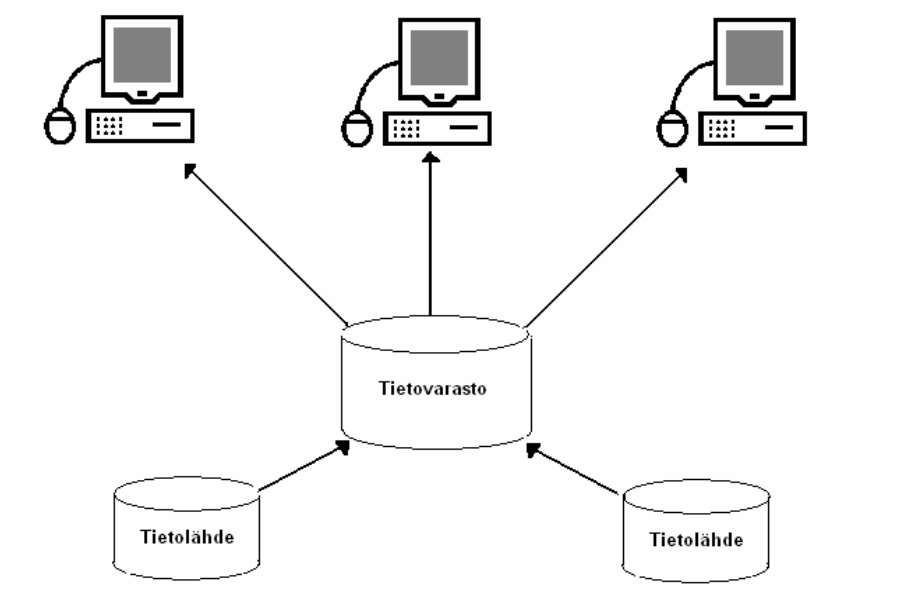
2.4. Tietovarastotyypit

Koko yrityksen datan kattavan tietovaraston rinnalla voi olla lukuisia paikallisia ja suppeampia tietovarastoja. Näiden kaksi pääasiallisinta eroa ovat seuraavat [Jarke et al. 2000]:

- Globaali tietovarasto on seurausta monimutkaisesta tiedon eristämis-, muokkaus- ja integrointiprosessista. Paikallisen tietovaraston vastaava prosessi lähtee liikkeelle vasta globaalista tietovarastosta.
- Globaalin tietovaraston tieto on volyymiltään suuri ja data on hyvinkin yksityiskohtaista, minkä seurauksena summaus on usein kevyttä. Paikallisessa tietovarastossa taas tiedon määrä on pienempi ja se on pidemmälle summattua.

2.4.1. Keskitetty tietovarasto

Keskitetty tietovarasto (central data warehouse, enterprise data warehouse) tarkoittaa koko yrityksenlaajuista tietovarastoa, jossa samaan tietovarastoon kerätään kaikki yritystä koskeva informaatio. Tällöin lähdejärjestelmät ja niiden laitealustat voivat olla hyvinkin kirjavia.



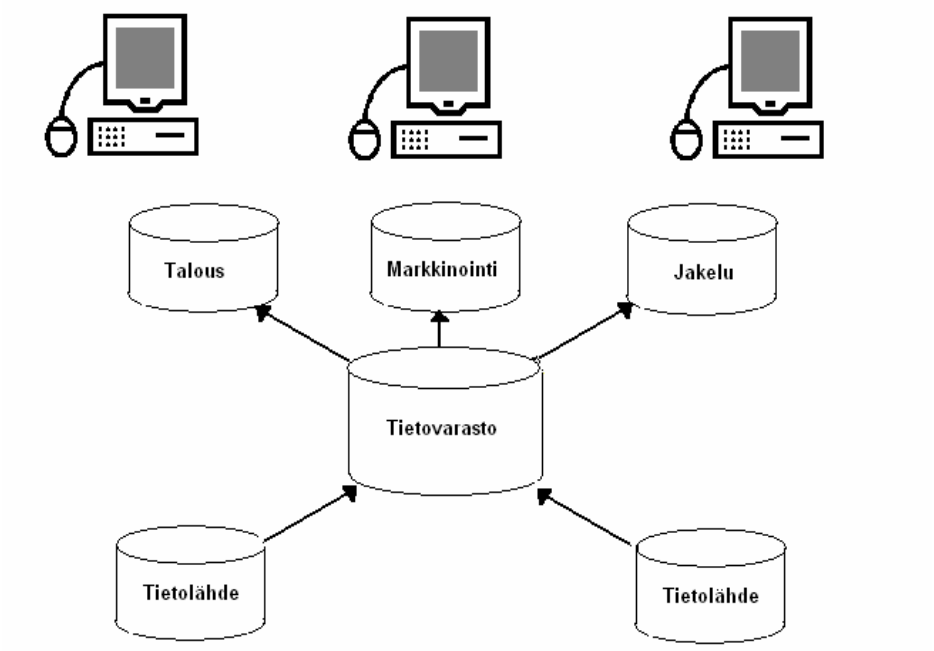
Kuva 3: Koko yrityksen tietovarasto [Jarke et al. 2000].

Kuvan 3 mukainen keskustietovarasto voi toimia pelkästään tietoa keskittävässä roolissa, jolloin siihen ei kohdisteta kyselyitä, vaan tietovaraston koko käyttö kohdistuu päätietovarastosta johdettuihin paikallistietovarastoihin.

Tällöin keskustietovaraston rakenne voi muistuttaa enemmän perinteistä relaatiotietokantaa kuin varsinaista tietovarastoa ja tietovarastoinnille ominaiset ja käyttöä tehostavat ratkaisut, kuten harkittu denormalisointi, toteutetaan vasta alitietovarastoissa.

2.4.2. Paikallistietovarasto

Paikallistietovarasto (alitietovarasto, ”data mart”) tarkoittaa tietovarastoa, joka sisältää suppeamman tietoelementtien joukon. Usein tietovarastoprojektit aloitetaan suunnittelemalla ensin paikallistietovarasto. Paikallistietovarasto on tarkoitettu esimerkiksi kattamaan vain tiettyjä yrityksen osa-alueita kuvan 4 tapaan tai se on suunnattu vain tietyille käyttäjille. Paikallinen tietovarasto voi olla varsinaisen tietovaraston tai laajemman paikallistietovaraston osa tai se voi olla täysin itsenäinen.



Kuva 4: Paikallistietovarastoja [Jarke et al. 2000].

Paikallistietovarasto voi tuoda datan lähemmäksi sitä tarvitsevaa käyttäjää, jolloin tarvittavaan dataan pääsy nopeutuu ja siihen pääsyyn liittyy vähemmän rajoituksia. Laajempaa tietovarastoa voidaan käyttää esimerkiksi tiedon summaamiseen, jonka jälkeen summattu data kopioidaan paikallistietovarastoihin.

Joskus myös tietoturvatekijät puoltavat tietovarastojen pilkkomista. Esimerkiksi franchising-periaatteella toimivat osastot tai yrittäjät eivät halua muiden vastaavien näkevän tietojaan, koska keskenään nämä ovat kilpailijoita.

Paikallistietovarastot voivat joskus olla myös täysin itsenäisiä. Jotta niitä kuitenkin voidaan pitää paikallistietovarastoina, niitä voi yhdistää jokin yhdenmukainen osa, joka on vertailukelpoinen muiden rinnakkaistten tietovarastojen kanssa. Tällainen yhdistävä tekijä voi olla esimerkiksi taloudellisten tunnuslukujen tiivistelmä.

2.4.3. Muita tietovarastotyyppejä

Yhtenä tietovarastomallina mainitaan myös virtuaalinen tietovarasto [Han & Kamber 2001; Elmasri & Navathe 2000], joka teknisesti on toteutettu joukkona erilaisia operatiiviseen kantaan rakennettuja näkymiä. Tällainen lähestymistapa vaatii operatiivisilta järjestelmiltä suurta kapasiteettia, koska kantaan liittyvät näkymät on jatkuvasti pidettävä ajan tasalla.

Tietovaraston yhdeksi lajiksi voitaisiin laskea myös tutkimustietovarasto (exploration warehouse) [Inmon et al. 1999]. Tässä tapauksessa yrityksen kattavasta tietovarastosta irrotetaan erillinen osa jonkun tietyn erityisalan analyysiä varten. Erillisen tietovaraston osan ideaan liittyy oleellisesti se, ettei sitä päivitetä, vaan myöhemmin voidaan tarvittaessa irrottaa uusi vastaava tai erilainen osuus.

Tietovarastointiprosessiin kuuluu usein muitakin tietokantoja kuin operatiiviset järjestelmät ja itse tietovarasto. Jo tietojen siirron ja muokkauksen takia tarvitaan erilaisia välivarastoja, joiden tehtävä on toimia työkaluina siirrettäessä eri järjestelmien dataa tietovarastoon.

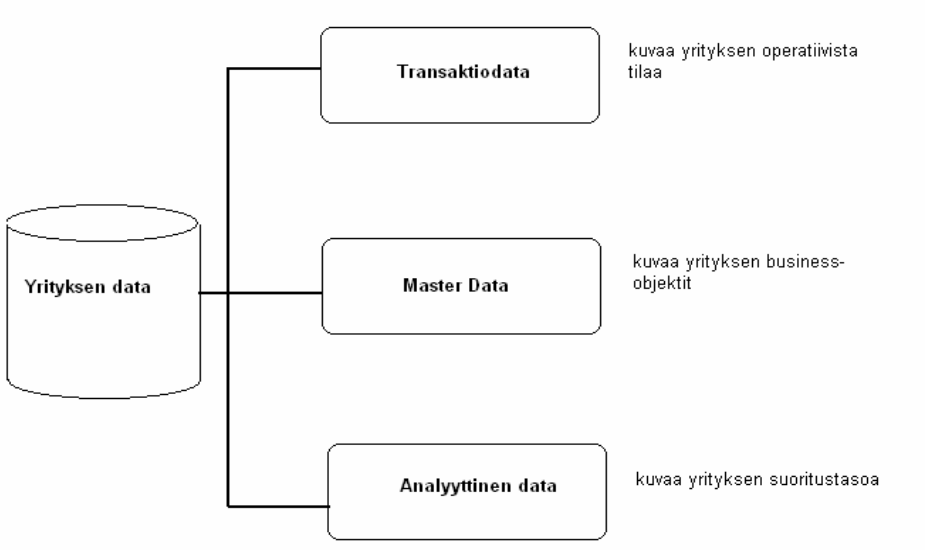
Joissakin tapauksissa on ollut tarvetta rakentaa operatiivisten kantojen ja tietovaraston väliin erillinen tilannekanta (operational data store, ODS), joka perustuu operatiivisen järjestelmän tietoihin, mutta on operatiivisesta järjestelmästä irrallinen. ODS voi tarjota tietovaraston rinnalla rajatun määrän ajantasaista informaatiota asioista, joita ei päivitetä tietovarastoon riittävän usein tai se voi toimia tietovaraston syötteenä sellaisen datan osalta, joka tarjoaa yritystä kiinnostavaa ajantasaista informaatiota myös muille prosesseille. Varsinaisesta tietovarastosta ODS eroaa seuraavasti [Jarke et al. 2000]:

- päivittyy useammin kuin tietovarasto
- sisältää vain ajan tasalla olevaa tietoa
- karkeisuus on vähäinen eli sen data on korkeintaan kevyesti summattua.

Tilannekannan lisäksi tietovarastoarkkitehtuuriin kuuluu usein erillinen työtietokanta (staging area), joka on erillään sekä operatiivisista kannoista että tietovarastosta. Työtietokanta on Kimballin ja muiden [1998] määritelmän mukaan itse välivarasto ja joukko prosesseja, jotka puhdistavat, muokkaavat, yhdistelevät ja muilla tavoin esiprosessoivat dataa sen varsinaista tietovarastokäyttöä varten. Kimball ja muut painottavat erityisesti työtietokannan merkitystä yhtenä kolmesta tietovarastoarkkitehtuurin ydinosasta lähdejärjestelmän ja varsinaisen tietovaraston ohella.

Työtietokantoja voi olla lukuisia ja ne voivat sijaita eri laitealustoilla eikä niiden tarvitse teknisesti olla relaatiotietokantoja. Työtietokannan perusidea on se, että data poimitaan ensin yhdenmukaiselle alustalle, jossa sitä voidaan tehokkaimmin työkaluin jalostaa ja josta se myöhemmin siirretään varsinaiseen tietovarastoon. Esimerkiksi Microsoft Excel soveltuu hyvin tietovaraston työtietokannaksi, koska tiedon muokkaaminen on Excel-taulukossa usein oleellisesti helpompaa kuin varsinaisessa tietokantataulussa.

Yrityksen data on perinteisesti jaettu transaktionaaliseen ja analyyttiseen niiden käyttötarkoituksen perusteella. Kolmantena yrityksessä käsiteltävän datan muotona on tuotu kuvan 5 mukaisesti esiin hallintadata (masterdata) [Butler & Stackowiak 2006]. Hallintadatalla tarkoitetaan liiketoimintaolioita (business objects), joita transaktiopohjaiset sovellukset käyttävät. Hallintadatan tarkoitus on ylläpitää yksikäsitteistä versiota liiketoimintaan liittyvistä olioista.



Kuva 5: Yrityksen datan lajit Butlerin ja Stackowiakin mukaan [2006].

Masterdatan käsite on yleistynyt vasta viime vuosina, vaikka sen perusajatus tietojen yhtenäistämisestä liittyy oleellisesti tietovaraston käsitteeseen. Yritysten sovellusympäristöjen laajentuminen on johtanut siihen, että tiedon yhdenmukaistaminen (harmonisointi) on eriytynyt omaksi osa-alueekseen ja siihen on kehitetty sekä metodeja että työkaluja.

2.5. Tietovaraston rakenne

Tietovarastoa tarkastellaan yleisesti moniulotteisena mallina, jota usein kuvataan kuutiona, vaikka todellisuudessa mallissa onkin enemmän ulottuvuuksia. Toisaalta taas tietojen esitystapa on usein juuri kolmiulotteinen: myyntiä seurataan usein halutulla ajanjaksolla toisaalta alueittain tai myymälöittäin, toisaalta tuotteittain tai tuoteryhmittäin. Eli useista dimensioista kolme on kerrallaan aktiivisia.

Moniulotteisen tiedon keskeisiä tarkastelutapoja on porautuminen (drilling down), jossa käyttäjä voi liikkua dimensioiden tasolta toiselle. Jos esimerkiksi alueen myyntiluvut herättävät käyttäjässä kysymyksiä, on luontevaa tarkastella alueen myymälöiden lukuja. Toisaalta käyttäjä voi myös verrata alueen lukuja saman alueen lukuihin eri aikajaksolla.

Tietovarastointi rakentuu teknisestä tietokantaratkaisusta riippumatta usein perusmalliin, jossa pääelementteinä ovat faktataulut ja dimensiotaulut. Tällainen rakenne mahdollistaa moniulotteisen tiedon relaatiomallia tehokkaamman käsittelyn.

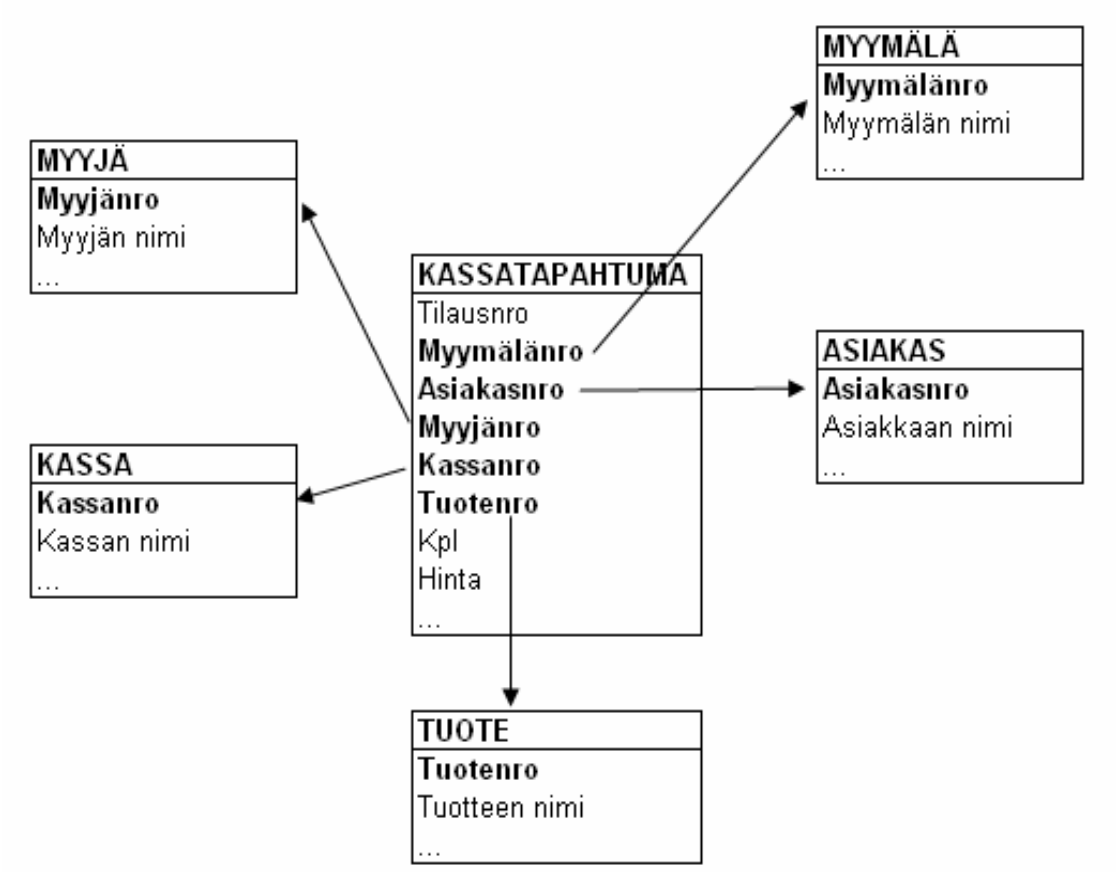
Faktataulu koostuu viittauksista dimensioihin (esimerkiksi päivämäärä, toimipiste, tuote) ja mitattavista suureista (measures), jotka käytännössä ovat usein numeerisia (esimerkiksi myynti kappaleina ja euroina) ja siten ne ovat summattavissa tai muuten laskennallisesti käsiteltävissä.

Kaikissa dimensiotauluissa on yksikäsitteinen avainkenttä, joihin faktatauluissa viitataan vierasavainten avulla. Yleinen suositus on käyttää tietovarastossa surrogaatteja eli keinotekoisia avaimia alkuperäisten operatiivisten kantojen avainten sijaan, koska alkuperäiset voivat muuttua.

Dimensiotauluja ovat usein yksi kutakin dimensiota kohti ja niissä on dimensioiden käyttäjille näkyvät selitteet ja mahdollisia muita täydentäviä määritteitä. Dimensiotauluissa voi myös olla viittauksia muihin dimensiotauluihin.

Usein tietovaraston faktataulujen viittaukset dimensiotauluihin kuvataan ns. tähtimallilla (star schema), jossa faktat ovat keskellä ja dimensiot sen ympärillä. Lumihiutalemalli (snowflake schema) on tähtimallin muunnelma,

jossa dimensiotauluissa on viittauksia muihin dimensiotauluihin, ja siksi graafinen esitys muistuttaa enemmän lumihiutaletta kuin tähteä. Tähtimallissa dimensiotaulut ovat denormalisoituja, mikä mahdollistaa yksinkertaisemman rakenteen. Kuvassa 6 on yksinkertainen esimerkki tähtimallista.



Kuva 6: Esimerkki tähtimallista.

Tähti- ja lumihiutalemallien lisäksi tunnetaan faktakonstellaatio-malli (fact constellation), jossa joukolla faktatauluja on yhteiset dimensiotaulut. Malli kuitenkin rajoittaa jonkin verran tietovarastoon tehtäviä kyselyitä verrattuna tilanteeseen, jossa kaikki faktat ovat samassa taulussa.

2.6. Summautuvuus ja karkeisuus

Summautuvuus (additiivisuus) on yksittäisen tietoalkion tärkeimpiä ominaisuuksia tietovaraston kannalta, sillä hyvin harvoin tietovaraston käyttäjä edes näkee yksittäistä faktaa vaan ainoastaan niiden summauksia. Han ja

Kamber [2001] jaottelevat mittayksiköt niiden summautuvuuden perusteella kolmeen eri kategoriaan:

- Distributiivisen funktion arvo koko datan osalta voidaan laskea partitioidusta datasta laskettujen arvojen (esimerkiksi välisummien) perusteella. Tällaisia funktioita ovat esimerkiksi summa, lukumäärä, minimi- tai maksimiarvot.
- Algebrallisen funktion arvo voidaan laskea sellaisten argumenttien avulla, jotka voidaan tuottaa distributiivisilla funktioilla. Keskiarvo on esimerkki algebrallisesta funktiosta, sillä sen parametreina ovat distributiiviset summa ja lukumäärä.
- Holistista funktiota ei voida kuvata distributiivisesti eikä algebrallisesti. Tyyppiesimerkkejä holistisista funktioista ovat mediaani ja moodi.

Karkeisuus (granulariteetti) tarkoittaa tarkkuustasoa, jolla tietty data esitetään. Tietovaraston perusajatus on tiivistää informaatiota sellaiseen muotoon, että se tukee päätöksentekoa. Näin ollen operatiivista dataa harvoin on tarvetta kopioida tietovarastoon tapahtumatasolla.

Yksi karkeisuuden astetta ohjaava tekijä on tietovaraston fyysinen koko. Tuoreimman datan sisältämä tietomäärä voi hienojakoisimmalla tasolla tallennettuna kasvaa suureksi. Uuteen tietoon kohdistuu kuitenkin usein eniten kyselyitä ja siksi se yleensä varastoidaan tarkemmalla tasolla kuin historiadata.

Summauksista on eniten hyötyä sellaisten tietojen kohdalla, joihin kohdistuu paljon kyselyitä. Summaamisella saavutetaan tietovarastoinnissa seuraavia etuja [Hovi 1997]:

- kyseltävä massa pienenee, koska summatauluissa on vähemmän rivejä kuin perustauluissa
- summatauluissa on usein valmiiksi liitetty yhteen useiden taulujen liitoksia, jolloin kyselyt yksinkertaistuvat
- perustiedoista johdetut tiedot lasketaan summauksen yhteydessä
- monimutkaiset summaukset tehdään keskitetysti ja oikein
- raportointi helpottuu, koska haut tulevat parhaimmillaan yhdestä taulusta eikä tarvita juurikaan käsittelylogiikkaa.

2.7. Toisto ja denormalisointi

Tietovarasto sisältää usein tarkoituksellisesti denormalisoitua dataa. Toisin sanoen ainakin kyselyiden kannalta tärkeimpiä koodituksia on purettu auki tietovarastokannoissa. Denormalisointi ovat perusteltua esimerkiksi seuraavista syistä:

- Viittaukset muuttuvat usein ajan kuluessa, joten historiallinen tarkkuus edellyttää tietojen denormalisointia. Esimerkiksi myyjän piiri voi muuttua ja tietovarastoon näin ollen päivittyy myöhemmin samalle myyjälle eri piiritieto.
- Datan helppokäyttöisyyden lisääminen etenkin ad hoc -kyselyissä. Relaatiomallin mukaisesti esimerkiksi koodeja vastaavat selitteet ovat operatiivisissa kannoissa eri tauluissa ja jo alkeellinen kysely edellyttäisi vastaavalla rakenteella käyttäjältä SQL-osaamista.
- Tiedon eheys ei vaarannu, koska käyttäjät eivät päivitä tietovarastoa.

Tietovaraston tehokkaampi käyttö edellyttää sitä, että käyttäjä joutuu kosketuksiin fyysisten tietokantojen kanssa. Tämän vuoksi on perusteltua, että käyttäjä voi ainakin ohjeistuksen avulla laatia yksinkertaisia SQL-lauseita, joissa haetaan tietoa yhdestä taulusta esimerkiksi WHERE-ehdolla. Kyselyehdossa voidaan käyttää esimerkiksi tuotekoodia, mutta kyselyn tuloksessa tietovaraston on osattava näyttää myös tuotteen nimi ilman tietokantaliitoksia.

2.8. Tietojen lataus tietovarastoon

Lataus tarkoittaa tietojen siirtoa tietovarastoon. Latauksesta käytetään usein termiä ETL (extract – transform – load). Toisinaan latauksella tarkoitetaan ainoastaan vaihetta, jossa tieto fyysisesti siirtyy tietovarastoon, toisinaan taas käsitettä käytetään laajemmin kuvaamaan koko prosessia, jossa tieto poimitaan operatiivisista järjestelmistä ja jossa se päätyy tietovarastoon enemmän tai vähemmän muokattuna. Suppeampi merkityssisältö jakaa prosessin neljään vaiheeseen, jotka ovat poiminta operatiivisista järjestelmistä, siirto tietovarastopalvelimelle (tai muulle sovitulle alustalle), tietojen muokkaus ja varsinainen lataus tietovaraston tietorakenteisiin.

Operatiivisessa kannassa päivitykset (lisäykset, muutokset, poistot) tehdään yleensä välittömästi tapahtumahetkellä. Tietovarastoja päivitetään yleensä eri lähtökohdista. Tieto ladataan tietovarastoon usein aikataulutetusti, minkä jälkeen käyttäjillä vasta on pääsy tietoihin.

Voidaan erottaa myös saman tietovaraston eri kehitysvaiheissa laajuudeltaan erityyppisiä latauksia: tietovaraston perustamisen yhteydessä saattaa olla tarve alustaa tietovarasto historiatiedolla, jolloin lataus on volyymiltään suuri. Toisaalta historiatietojen lataus ei ole aikakriittinen kuten esimerkiksi päivittäinen lataus, joten ladattavan aineiston määrä ei ole välttämättä ongelma. Peruslatauksella puolestaan tarkoitetaan kannan perustamisvaiheen latausta, jossa järjestelmään tuodaan operatiivisista

järjestelmistä kaikki tarvittavat tiedot. Myöhemmin perustiedot päivitetään yleensä vain muuttuneiden tietojen osalta. Jatkuvalla latauksella vastaavasti tarkoitetaan tietovaraston säännöllisesti tapahtuvaa latausta sen ollessa samanaikaisesti käytössä.

Tietojen poiminnassa lähdejärjestelmistä voidaan nähdä kuusi erilaista tekniikkaa [Devlin 1997]:

- Staattinen poiminta on otos lähdejärjestelmästä tietyllä hetkellä. Sen alalajeja ovat kokonaisuuden (esimerkiksi yksittäinen taulu), attribuuttijoukon (taulukon tietyt kentät) tai esiintymäjoukon (taulukon tietyt rivit) poiminta.
- Sovellusavusteisessa poiminnassa lähdejärjestelmä tuottaa tietovarastoa varten tarvittavan poiminnan.
- Liipaisinpohjaisessa (triggered) poiminnassa lähdetietokantaan on määritelty liipaisin, jossa tietty muutos generoi määrätyn toimintosarjan, joka voi olla esimerkiksi tietovaraston poimintakannan päivitys.
- Lokiin perustuvassa poiminnassa, jossa tietovarastopäivitys jäljittää operatiivisten kantojen muutokset tapahtumalokin perusteella.
- Aikaleimaan perustuva poiminta edellyttää, että lähdejärjestelmän tietotauluissa tallentuu niiden muutosaika, jonka perusteella on mahdollista poimia tietovarastoon vain muuttuneet tiedot.
- Tiedostovertailu on käyttökelpoinen vain silloin, kun muita keinoja ei ole käytettävissä. Se edellyttää, että edellisen poiminnan lähteenä olleesta tiedostosta on sen hetkinen versio tallessa, jolloin tiedoston uusinta versiota voidaan verrata siihen ja poimia muuttuneet tiedot.

Datan tuonti lähdejärjestelmästä tietovarastoon tapahtuu käytännössä tiedostona tai tietovirtana. Kimball ja muut [1998] muistuttavat tiedostosiirron yhtenä etuna olevan sen, että siirto voidaan aloittaa uudelleen eri kohdista. Lisäksi lähde- ja kohdetiedoston fyysinen vertailu helpottaa siirron tarkistusta.

2.8.1. Latauksen ongelmakohtia

Tietojen siirto operatiivisista järjestelmistä on monimutkainen prosessi, sillä siihen liittyy usein ainakin seuraavia ongelmakohtia [Inmon 2002]:

- Usein operatiivisen datan eristäminen ja siirto tietovarastoon merkitsee myös sitä, että joudutaan toimimaan eri käyttöjärjestelmien ja laitealustojen kanssa. Samassa yhteydessä voidaan joutua konvertoimaan tietoja merkkijärjestelmästä toiseen (esimerkiksi EBCDIC:n muuntaminen ASCII:ksi).

- Datan valintaproseduuri operatiivisista järjestelmistä voi olla kompleksinen, sillä tietoa voidaan joutua poimimaan lukuisista eri lähteistä.
- Operatiivisten järjestelmien avaimet on usein järjestettävä uudelleen ja muunnettava tietovarastoon vietäessä. Jos tietolähteillä on eri avainnukset, ne on myös limitettävä.
- Data on tietovarastoon vietäessä usein muokattava uuteen esitysasuun. Lisäksi data on puhdistettava (cleansing) tietovarastoon vietäessä.
- Eri tietolähteissä voi olla osin samaa tietoa ja ne on limitettävä yhteen ennen tietovarastoon siirtoa.
- Yksi valintaprosessi voi tuottaa useita tulosjoukkoja, sillä tietovarastoon tulee tuottaa sama datasisältö useilla eri summaustasoilla.
- Joillekin tietoalkioille tulee tuottaa oletusarvo, ellei niitä ole saatavissa kaikissa tapauksissa lähdejärjestelmistä.
- Valintaprosessin tehokkuus muodostuu usein merkittäväksi tekijäksi prosessissa, koska poiminnalla voi olla käytettävissä vain rajallinen aika ja toisaalta se voidaan joutua tekemään operatiivisen järjestelmän käytön aikana eikä se näin ollen saa kuormittaa operatiivista järjestelmää liikaa.
- Tietoalkiot nimetään yleensä uudelleen tietovarastossa. Nimeämisketju on aina dokumentoitava.
- Lähdejärjestelmien datan riippuvuudet on selvitettävä ennen niiden lukua tietovarastoon. Usein tämä vaihe voi olla erittäin työläs dokumentaation puuttuessa tai ollessa vanhentunut.
- Käsiteltävä datan määrä on usein massiivinen ja sen vuoksi tulee huomioida esimerkiksi mahdollisuus ajaa latauksen eri vaiheita rinnakkain.
- Koska tietovarasto sisältää myös historiallisen näkökulman, tietoon on lisättävä aikaelementti tietovarastoon siirrettäessä.

2.8.2. Datan muokkaus

Muokkausvaiheen aikana operatiivisten järjestelmien tiedot muunnetaan tietovaraston muotoon. Yleisimpiä tehtäviä tässä vaiheessa on eri järjestelmistä tulevien tietojen kooditusten yhdenmukaistaminen. Lisäksi tiedoille tehdään tarpeen mukaan denormalisointia kyselyiden helpottamiseksi. Tässä vaiheessa on myös tarkistettava relaatioiden viite-eheydet ja yleensä tulee tarkistaa kaikki, mikä on koneellisesti tarkistettavissa. Muokkausvaiheessa voidaan

määritellä myös tietoalkioiden tietotyyppejä tai arvoja uudelleen tai luoda tietovarastoon uusia luokitteluita (datan rikastaminen).

Datan puhdistukseen liittyy usein datan validointi, jossa datan laatua tarkistetaan vielä erikseen vertaamalla sitä annettuihin raja-arvoihin tai annettuihin arvojoukkoihin. Lisäksi voidaan verrata datan eri arvoja keskenään ja tällöin validointiprosessi voi tavalla tai toisella hälyttää havaitessaan yksittäisen tietoalkion eroavan merkittävästi muista vastaavista arvoista.

Laadukkaan datan tunnuspiirteiksi ovat Kimball ja muut [1998] määritelleet seuraavat:

- Täsmällisyys eli data vastaa lähdejärjestelmien kirjauksia. Mahdolliset erot voivat johtua vain sovituista laskentatavoista ja ne on dokumentoitava.
- Täydellisyys eli tietovaraston data edustaa relevanttia dataa.
- Konsistenssi eli eri tasoiset tiedot keskenään ristiriidattomia (summaukset vs. detaljit).
- Ainutlaatuisuus eli samaa tarkoittavat asiat on esitetty vain yhdellä tavalla.
- Ajantasaisuus eli dataa päivitetään sellaisella aikataululla, joka palvelee käyttäjiä.

3. Internet tiedonkeruun kohteena

Internetiä itsessään voitaisiin pitää jättiläismäisenä tietovarastona. Käytännössä se ei kuitenkaan täytä Inmonin neljästä tunnusmerkistä kuin yhden eli aihesuuntautuneisuuden. Senkin lähinnä sitä kautta, että internetin voidaan katsoa olevan aihesuuntautunut kaikkiin mahdollisiin aiheisiin. Minkäänlaista integraatiota ei eri sivustojen välillä ole. Aikariippuvuuden ja pysyvyyden puute puolestaan on merkittävimpiä ongelmia esimerkiksi suunniteltaessa verkkosisällön arkistointia, koska sivustot muuttuvat koko ajan. Lisäksi olemassa olevia sivustoja poistuu ja uusia tulee lisää.

Toinen tietovaraston periaatteiden kanssa ristiriidassa oleva internetin ominaisuus on se, ettei tiedon luotettavuutta voida mitenkään taata. Varsinainen tietovarasto pyrkii myös määrittämään sisältämänsä tiedon viralliseksi totuudeksi [Inmon 2002]. Internetin luonteen mukaisesti samasta aiheesta voi olla lukematon määrä sivustoja, joiden tietojen paikkansapitävyyttä on mahdoton arvioida. Kun lisäksi yhden sivuston lähteenä on saattanut olla toinen sivusto, ei voida arvioida tiedon luotettavuutta edes sen perusteella, että se mainitaan useassa lähteessä.

Internetistä voidaan kerätä tietoja useaan eri tarkoitukseen. Tiedonkeruulla tarkoitetaan tässä yhteydessä sellaisia toimintoja, joiden tuloksena tieto siirtyy pysyvästi toiseen järjestelmään tai se tallennetaan paikallisesti.

Internet-tiedonkeruussa voidaan nähdä kolme erilaista päälinjaa: web-arkistointi, web-sisällön louhinta ja datan eristäminen. Arkistoinnissa pääpaino on sivustojen mahdollisimman kattavalla taltioinnilla sisällön louhinnan ja datan eristämisen keskittyessä sivustojen sisältöön. Louhinta eroaa datan eristämisestä siinä, että se pyrkii keräämään uutta tietämystä jälkimmäisen pyrkiessä ennen kaikkea datan formalisointiin ja integrointiin. Lisäksi jälkimmäisessä sekä kohteena oleva aihealue että kohdesivustot ovat varsin rajattuja.

3.1. Web-arkistointi

Web-arkistointi pyrkii jäljittelemään perinteistä arkistointia taltioimalla otoksen internetistä säännöllisin väliajoin. Viime vuosina internetiin on syntynyt erilaisia web-arkistoja, jotka yrittävät tallentaa tiettyjen aihealueiden sivustoja.

Internetin luonteesta johtuen sen täydellinen arkistointi on kuitenkin mahdotonta.

Web-arkistointijärjestelmät yrittävät yleensä arkistoida kokosivuston koko sisällön, mukaan lukien sen ulkoasu ja toiminnallisuus. Yksi tunnetuimpia on http://www.archive.org, jossa voi tarkastella annetun url-osoitteen (universal resource locator) sisältöjä eri aikakausina.

Lyman [2002] erottelee neljä erillistä ongelma-aluetta koskien web-arkistointia. Samat ongelmat koskevat myös perinteistä arkistointia, mutta internetin luonteesta johtuen monet näistä korostuvat yrityksissä arkistoida webin sisältöä:

- Kulttuurillisen ongelman lähtökohta on se, ettei tunnisteta informaation historiallista arvoa, minkä vuoksi sen tallentamista ei katsota tarpeelliseksi.
- Tekninen ongelma taas korostuu erityisesti sähköisessä informaatiossa toisaalta digitaalisten asiakirjojen heikon säilyvyyden vuoksi ja toisaalta teknisen kehityksen takia, mistä johtuen vanhoja sähköisiä asiakirjoja tukevia sovelluksia tai laitteita ei ole helposti saatavilla.
- Taloudellinen ongelma liittyy taloudellisten näkökohtien lisäksi vastuukysymyksiin. Ellei ole selkeästi määritelty, kenen vastuulla arkistointi on, siihen ei voida kohdentaa resurssejakaan. Arkistoinnin kannattavuus perinteisillä talouden mittareilla on heikko, mutta silti se edellyttäisi laajoja investointeja.
- Neljäs ongelma on laillisuusongelma. Vaikka web-sisältöä pidetään yleisesti julkisena informaationa, sitä koskevat kuitenkin tekijänoikeudet, joten tarkasti ottaen tietojen kopiointi edes arkistointitarkoituksissa ei ole aina kiistattoman sallittua.

Internetin tietoja voidaan arkistoida useilla tavoilla. Etäharavointi (remote harvesting) simuloi käyttäjää, sillä metodi siirtyy selainkäyttäjän tapaan sivulta toiselle linkkien avulla. On-demand -metodin mukaiset arkistointipalvelut sen sijaan antavat käyttäjän perustaa omia arkistoja omilla arkistointisäännöillään. Tietokanta-arkistointi (database archiving) keskittyy nimensä mukaisesti taltioimaan tietokantapohjaisten sivustojen datasisältöä. Usein tällä tarkoitetaan xml-pohjaisia palveluita, jolloin kantojen rakenne on xml-skeeman kautta jo valmiiksi määritelty. Transaktiopohjaisen arkistoinnin tarkoituksena on arkistoida palvelimelle kohdistettuja pyyntöjä.

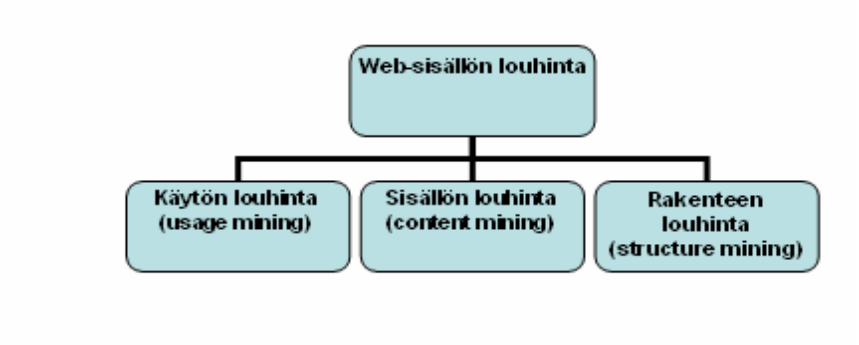
Web-haravointi on nousemassa keskeiseksi metodiksi verkkoaineiston arkistoinnissa, sillä se muuttaa perinteisen arkistoinnin vastuujakoa. Euroopan

unionin suosituksessa kulttuuriaineiston digitoinnista ja sähköisestä saatavuudesta sekä säilyttämisestä todetaan seuraavasti [EU 2006]: *"Tiedon haravointi (web harvesting) on uusi tekniikka, jolla kerätään materiaalia internetistä säilytettäväksi. Siinä valtuutetut laitokset kokoavat aktiivisesti aineistoa sen sijaan, että odottaisivat, että ne tallennetaan. Tällä tavoin vähennetään digitaalisten aineiston tuottajien hallinnollista taakkaa, ja kansallisessa lainsäädännössä olisi sen vuoksi säädettävä asiasta."*

3.2. Web-sisällön louhinta

Tiedon louhinta (data mining) liittyy käsitteenä erityisesti tietovarastoon. Sillä tarkoitetaan metodeja, joiden avulla tietovarastoista etsitään trendejä ja malleja. Louhinta eroaa perinteisistä tietovarastokyselyistä myös siinä, että louhinnassa käyttäjän ei välttämättä tarvitse olla perillä tietovaraston teknisestä rakenteesta.

Web-sisällön louhinnalla (web content mining) tarkoitetaan vastaavan kaltaisia tiedonlouhintatekniikoita, jotka ovat keskittyneet analysoimaan internetissä olevaa tietoa. Web-sisällön louhinta jaetaan yleisesti kolmeen eri lajiin kuvan 7 mukaisesti.



Kuva 7: Web-sisällön louhinnan eri lajit.

Internet-käytön louhinta on kiinnostunut analysoimaan webin käyttötietoa. Tällöin esimerkiksi palvelimen lokiin tallentuvien käyttäjätietojen perusteella voidaan profiloida käyttäjiä sen mukaan, milloin käyttäjät ovat vierailleet palvelimella, mitä palveluita he ovat käyttäneet jne.

Sisällön louhinta taas etsii sivustoilta merkityksellistä tietosisältöä. Sisällön ei välttämättä tarvitse olla tekstimuotoista sisältöä vaan se voi olla myös ääntä, kuvaa tms. Se on metodiikaltaan lähellä datan louhintaa, koska monia datan louhintatekniikoita voidaan soveltaa web-sisällön louhintaan. Toisaalta web-sisällön louhinta eroaa datan louhinnasta, koska internetin data

on perusluonteeltaan heikosti strukturoitua, kun perinteinen tiedon louhinta on keskittynyt käsittelemään strukturoitua dataa.

Sisällön louhinta voi myös keskittyä mielipiteiden louhintaan. Tällöin tiedonkeruu tuottaa arvokasta markkinainformaatiota erilaisten palveluiden tuottajille ja myös niiden käyttäjille.

Rakenteen louhinta mahdollistaa sen tutkimisen, miten muut sivustot viittaavat johonkin tiettyyn sivustoon. Kun sivustoon viitataan laajalti, se lisää sivun painoarvoa.

Liun [2005] mukaan internet tarjoaa valtavat mahdollisuudet ja haasteet datan louhinnalle muun muassa seuraavista syistä:

- web on dynaaminen (elää jatkuvasti) ja informaation määrä valtava
- web sisältää lähes kaikissa mahdollisissa muodoissa olevaa dataa (tekstin lisäksi taulukoita, multimediaa jne)
- webissä on ”kohinaa”, sillä sivu sisältää usein datan lisäksi mainoksia, navigointipaneeleita yms.

3.3. Tiedon eristäminen

Internet-dokumenttien kuvauskieleksi on vakiintunut tekstipohjainen html (hypertext markup language), joka mahdollistaa sivun käsittelyn tekstinä. Tiedon eristämisellä (data extraction, information extraction) tarkoitetaan olennaisen tiedon eristämistä dokumenteista**.** Tietoa eristettäessä järjestelmä suodattaa yksittäiset faktat tekstistä, minkä jälkeen niistä muodostetaan yhä suurempia kokonaisuuksia.

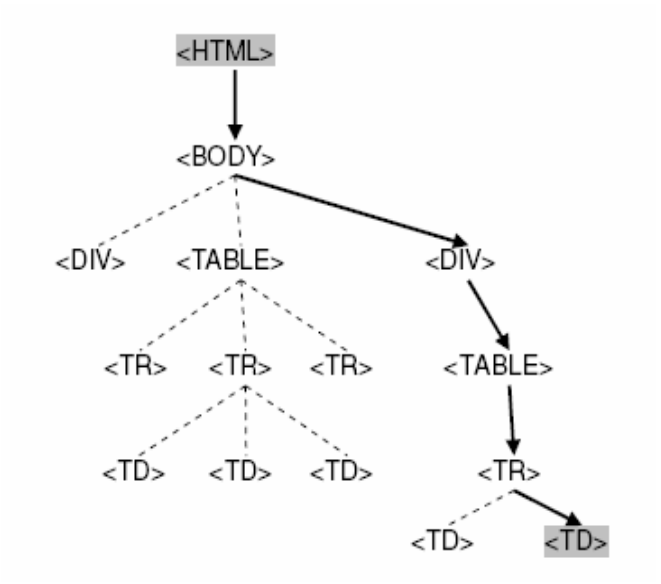
Adams [2001] erottaa informaation eristämisen (information extraction, IE) ja informaation haun (information retrieval, IR). Jälkimmäisessä käyttäjälle palautuu dokumenttijoukko, joka vastaa tehtyä kyselyä. IE taas palauttaa dokumenteissa olevia faktoja.

Eristäminen tapahtuu luomalla malleja ja sääntöjä, jotka kertovat, miten haluttu tieto on esitetty tekstissä. Eristämisessä käytetään ns. alku- ja loppusääntöjä, jotka kertovat, mistä kohde alkaa ja mihin se loppuu. Säännöstöjen luonnissa yritetään käyttää hyväksi sisällön rakenteisuutta ja tunnistaa sen eri tasot ja niiden väliset suhteet.

Kowalkiewicz ja muut [2006] erottelevat kolme erilaista lähestymistapaa webin sisällön eristämiseen:

- uniikkiin tunnisteeseen (id) perustuvat
- kontekstuaaliseen informaatioon perustuvat
- dokumentin puurakenteeseen perustuvat.

Tunnisteen käyttö perustuu sivuston html-lohkoille annettaviin tunnisteisiin, joilla voidaan osoittaa eristettävät data-alueet. Se soveltuu parhaiten intranet-ratkaisuihin [Kowalkiewicz et al. 2006], joissa tuotettavaan html-sivustoon voidaan vaikuttaa siten, että se sisältää tarvittavat tunnisteet html-elementeille. Kontekstuaalinen lähestymistapa käsittelee dokumenttia merkkijonona ja se eristää tekstistä informaatiota sitä ympäröivien merkintöjen avulla. Kolmannessa lähestymistavassa dokumentti yritetään jäsentää kuvan 8 tyyppiseksi puurakenteeksi ja se on ehkä käytetyin menetelmä webin sisällön eristämisessä.



Kuva 8: Esimerkki html-sivun puurakenteesta [Kowalkiewicz et al. 2006].

3.4. Web-keruuprosessien tekniikoita

Web-ryömijä (web crawler, web spider tai web robot) tarkoittaa sovellusta, joka selaa internet-sivustoja automaattisesti jonkun määritellyn metodin mukaisesti. Erityisesti hakukoneet käyttävät web-ryömintää päivittäessään tietokantojaan. Lisäksi sitä voidaan käyttää pienempiin tehtäviin kuten tuotetun sivuston linkkien tarkistukseen tai keräämään sähköpostiosoitteita roskapostitusta varten. Keskitetty ryömintä (focused crawling) lähtee liikkeelle sivujoukosta, joihin linkitetyiltä sivulta se kerää tietoa. Alkuperäinen sivujoukko on ennakkoon määritelty käyttäjän mielenkiinnon mukaan.

Perinteiseen web-ryömintään liittyy kaksi keskeistä ongelmaa [Nelson et al. 2006]: ryömijä ei voi tietää, onko se käynyt läpi jonkun verkkosivuston kaikki resurssit (ns. laskentaongelma) ja toisaalta ihmisen luettavissa oleva esitystapa ei ole aina sopiva koneelliselle käsittelylle (ns. representaatio-ongelma).

Internetin laaja volyymi ja alituinen muutosprosessi tekevät koko internetin ajantasaisen indeksoinnin tai arkistoinnin käytännössä mahdottomaksi. Lisäksi dynaaminen sivujen generointi laajentaa volyymiä entisestään ja erityisesti se aiheuttaa sen, ettei sivujen määrää ole edes mahdollista määrittää. Webissä erotellaankin usein pintaweb (surface web) ja syvä web (deep web). Pintaweb tarkoittaa sivustoja, joihin pääsee selaimella, ja syvä web sivuja, jotka generoituvat esimerkiksi tietokantakyselyiden kautta.

Tietovarastokeruun kannalta laaja volyymi ei ole välitön ongelma, koska jo keruuprosessissa tiedetään tarkasti, mitä kerätään. Sen sijaan prosessin luonteeseen kuuluu oleellisesti seurata muuttuvia sivustoja ja päivittää niiden uudet tiedot tietokantaan riittävän nopeasti. Sivujen dynaaminen generointi taas saattaa aiheuttaa runsaastikin työtä poiminnassa, sillä tieto voi olla jaoteltu niin, että suoraviivainen poiminta esimerkiksi url-osoitteessa vaihtuvan numeerisen tunnisteen perusteella ei onnistu.

Web-ryöminnässä samoin kuin muissakin tiedonhakuprosesseissa tulisi ottaa huomioon seuraavat neljä menettelytapaa (policy) [Wikipedia]:

- valinta (selection policy)
- tarkistussykli (re-visit policy)
- huomaavaisuus (politeness policy)
- rinnakkaisuus (parallelization policy).

Valinta tarkoittaa kerättävien tietojen valintaa, mikä tietovarastoon tietoja haettaessa toteutuu jo automaattisesti aihealueen ollessa tarkasti rajattu. Esimerkiksi arkistoinnissa sen sijaan on ennen tietojen keruuta päätettävä strategia, miltä sivustoilta tietoja haetaan.

Tarkistussykli viittaa siihen aikaväliin, jolla kohdesivustoja haravoidaan. Kerättäessä vain ennakkoon määriteltyjä sivujoukkoja voidaan tarkistussyklin määrityksessä huomioida sivun rakenne. Jos sivustolla on jo oma arkistointijärjestelmänsä, haravointi voidaan tehdä harvemmin, ellei ole tärkeää arkistoida sivustoa juuri sellaisena, kuin se näkyi selaimessa tietyllä hetkellä.

Huomaavaisuus tarkoittaa sitä, ettei ole tarpeen kuormittaa ladattavaa sivustoa kohtuuttomasti. Vaikka etenkin keruuprosessin suunnitteluvaiheessa

on tarve ladata samaa sivua lukuisia kertoja, on perusteltua tallentaa se ensin paikallisesti ja vasta sen jälkeen testata datan käsittelyproseduureja.

Rinnakkaisuudella tarkoitetaan ryömijän kykyä ajaa rinnakkain useita prosesseja. Tietovarastohakujen kannalta sillä voitaisiin tarkoittaa yhtä hyvin erillisten poimintojen ajoa eri työasemilta.

3.5. Tiedonkeruun vaiheet

Suuri osuus internetin informaatiosta esitetään rakenteisessa muodossa, joten on ollut mahdollista kehittää erilaisia lähestymistapoja informaation keruun automatisoimiseksi. Osa perustuu koneellisen oppimisen eri variaatioihin, joissa prosessin alussa käyttäjä kertoo keruusovellukselle, mitkä alueet internet-sivustoista sisältävät kiinnostavaa informaatiota. Automatisoidut prosessit taas perustuvat keruusovelluksen omaan päättelyyn.

Tietojen automatisoitu poiminta internetistä tietovarastoon edellyttää välttämättä esitutkimusta, jossa manuaalisesti ladataan haluttuja sivustoja ja analysoidaan niiden rakenne automaattisesti kerättäviksi tarkoitettujen tietojen osalta. Kun sivujen rakenne on pääpiirteittäin selvitetty, jatko koostuu tekniseltä kannalta kolmesta erilaisesta prosessista:

- Ensimmäinen vaihe on html-sivujen luku niiden alkuperäisestä sijainnista ja samalla niiden tallennus levylle datan eristämistä ja myöhempää käyttöä varten.
- Toisessa vaiheessa erotetaan luetusta aineistosta ne linkit, joiden kautta sivustojen tiedot yhdistellään. Yksinkertaisimmissa toteutuksissa tätä vaihetta ei tarvita, jos kaikki kohdedata on samalla sivulla tai yksi kohdesivu vastaa suoraan kohdetietokannan yhtä riviä.
- Kolmas vaihe on kohteena olevan tiedon eristäminen ja jäsentäminen niin pitkälle, että se voidaan ladata tietovarastoon. Vaikka jokainen sivusto on erilainen ja jokaista sivustoa varten on laadittava oma proseduurinsa, on silti mahdollista kehittää yleiskäyttöisiä proseduureja, joiden avulla tietoa voidaan jäsentää.

Tietojen haku tietovarastoon merkitsee usein operatiivisten kantojen uudelleen konstruointia html-muotoon puretun esityksen pohjalta. Vaikka taustalla olevasta operatiivisesta kannasta ei ole saatavana tarkkaa tietoa, usein esitystavan perusteella sen pääpiirteet ovat pääteltävissä, jos tunnetaan myös aihealueeseen liittyvä käsitteistö.

3.6. Ongelmakohtia

Internet-tiedonkeruun suurimmat haasteet liittyvät toimintaympäristön luonteeseen:

- sivuston rakenne on muuttunut, mikä varsin usein tarkoittaa sitä, että tiedon poiminta sivustolta on ohjelmoitava uudelleen
- pienemmät selaimelle näkymättömät muutokset, esimerkiksi sivulla oleva linkki samalle palvelimelle muutetaan viittaamaan esimerkiksi toiseen kansioon tai erinimiseen alisivuun
- suojaukset (estävät myös leikkaa-liimaa -käsittelyn)
- haluttujen tietojen näyttö kuvana tai pdf-liitteenä
- jokaiselle eri lähteelle tehtävä erilainen poiminta, vaikka peruslogiikka voi säilyä samanlaisena.
- eri granulariteetti eri lähteissä (tiedot eri karkeisuustasoilla)
- historiatietojen katoaminen aktiivisen kauden vaihtuessa
- ristiriitaisuudet samasta tiedosta eri lähteissä
- loogisesti yhteenkuuluvia tietoja kerättävä useilta eri sivustoilta.

Eri lähteistä tulevia tietoja yhdisteltäessä suurimmaksi sisältötekniseksi ongelmaksi muodostuu se, että samaa tarkoittava tieto on esitetty usein eri tavalla eri lähdejärjestelmissä. Strukturoidun ja strukturoimattoman datan yhdistäminen on käytännössä mahdollista vain tekstuaalisen vertailun avulla. Suora vertailu johtaa kuitenkin ainakin seuraaviin ongelmiin [Inmon 2005]:

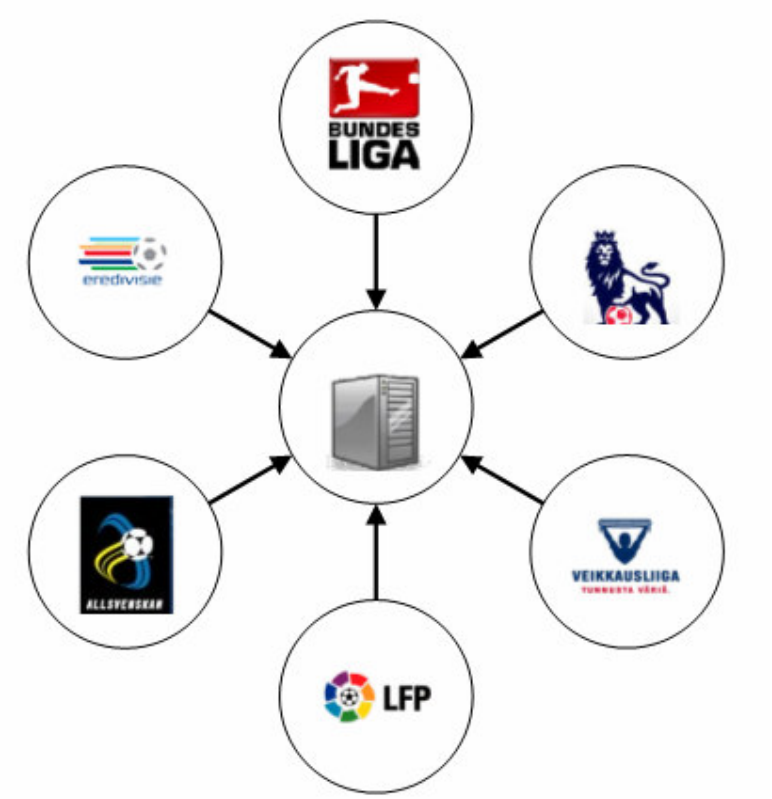
- kirjoitusvirheet ja eri kielialueiden erilaiset translitteroinnit
- sanat voivat eri asiayhteyksissä tarkoittaa eri asioita
- nimiongelmat (Matti Virtasia ei voi automaattisesti määritellä samaksi henkilöksi)
- nimien erilaiset kirjoitustavat (William Inmon ↔ Bill Inmon) ja eri taivutusmuodot tai eri sanajärjestys (Jari Litmanen ↔ Litmasen Jari)
- lyhenteet (Hämeenkatu ↔ Hämeenk.)
- epätäydelliset nimet (E.Mäkinen ↔ Erkki Mäkinen)

Inmon [2005] tuo perinteisen tietueen tunnisteen (identifier) rinnalle termin lähes-tunniste (close identifier). Kun tunnisteen (esimerkiksi henkilötunnus) avulla henkilö voidaan identifioida varmuudella, lähes-tunnisteen avulla tunnistus voidaan tehdä suurella todennäköisyydellä. Tällaisia tunnistetietoja ovat esimerkiksi nimi- ja osoitetiedot. Identifioinnin todennäköisyys paranee entisestään, mikäli käytettävissä on useita samaa tietuetta koskevia lähes-tunnisteita.

4. Esimerkkitietovarasto: palloilulajien tietovarasto

Internetissä olevat palloilulajien tilastoja sisältävät palvelut ovat alkaneet yleistyä viime vuosina. Se, että näin on tapahtunut myöhemmin kuin useimmilla muilla palvelualueilla, johtuu siitä, että tilastotietojen hyötyarvo ei ole yhtä keskeinen kuin yrityselämässä. Toisena syynä on ollut tietojen saatavuus, koska eri palloilusarjat ovat usein vasta 2000-luvulla julkaisseet tilastotietoja ottelutapahtumista internetissä. Tietoja ei aikaisemmin edes ole välttämättä tallennettu mihinkään tietojärjestelmään, koska tapahtumat kirjataan usein manuaalisesti erotuomarin ottelun aikana käsin täyttämän ottelupöytäkirjan ollessa virallisin dokumentaatio.

Palloilulajien tietovaraston pääperiaate on kuvan 9 mukainen: tilastotietoja pyritään keräämään globaalisti maiden ja sarjojen rajoista riippumatta.



Kuva 9: Palloilulajien tietovarasto.

Useimmille palloilusarjoja seuraaville sarjan itsensä tuottama perusinformaatio on riittävä. Tämä informaatio itsessään on jo pienimuotoinen tietovarasto. Perusinformaatioon kuuluvat myös kullekin lajille tyypilliset summaukset, joista tunnetuimmat ovat sarjataulukko ja maalintekijätilasto.

Vaativamman tilastoseurannan kannalta peruspalvelu ei kuitenkaan riitä, sillä kantaan ei voida kohdistaa omia kyselyitä. Esimerkiksi tieto siitä, missä otteluissa tietty pelaaja on pelannut, joudutaan usein etsimään manuaalisesti porautumalla ottelu kerrallaan jokaiseen pelaajan seuran pelaamaan otteluun. Koska taustalla oleva tietokanta on käyttäjälle näkymättömissä, sinänsä yleensä riittävällä tarkkuustasolla oleva perusdata ei mahdollista tiedon jalostamista tai lisäarvon luomista sille ilman manuaalista työtä.

Jo yksittäisen seuran syvemmälle menevä tilastoseuranta edellyttäisi tietojen keruuta useista eri lähteistä, ellei seuran omalla kotisivulla ole jo valmiiksi koostettu ottelutietoja yhdenmukaiseen formaattiin. Esimerkiksi Tampere Unitedin sarjaotteluiden tilastotiedot on tallennettu Veikkausliigan tietojärjestelmään, mutta seuran kansainväliset ottelut Euroopan jalkapalloliiton (UEFA) turnauksissa ovat saatavilla varmuudella vain kyseisen liiton sivuilta. Kansallisen cup-kilpailun (Suomen cup) tiedot taas ovat saatavana Suomen Palloliiton verkkosivuilla.

Datan haku internetistä on avainasemassa tutkimuskohteena olevan tietovaraston ajan tasalla pitämisen ja sitä kautta sen palvelevuuden kannalta, koska varsinaista operatiivista kantaa ei ole olemassa. Koska yksittäisten otteluiden informaatiota koskevat html-sivut muodostetaan ohjelmallisesti, niiden presentaatio on joka ottelun (tai ”pelaajakortin” tms.) osalta samankaltainen. Näin ollen sivujen oleellinen tietosisältö voidaan kerätä ohjelmallisesti, kun ensin analysoidaan yksittäisen sivuston rakenne.

Osa informaatiosta voidaan joutua poimimaan manuaalisesti ”leikkaa ja liimaa” -periaatteella. Toisinaan ei edes ole järkevää rakentaa automaattista poimintaa, jos on tarve kertaluonteisesti kerätä pienehkö määrä informaatiota rajatulta sivujoukolta.

4.1. Toiminnot ja palvelut

Rakennettavaa tietovarastoa hyödyntämään on mahdollista kehittää esimerkiksi seuraavia palveluita:

- Pelaajahistoriakanta (tunnetuin http://www.playerhistory.com), joka mahdollistaa yksittäisten pelaajien uratietojen kyselyn mahdollisimman kattavina. Kansallisten sarjojen palvelut eivät pysty yhtä kattavaan

esitykseen, koska pelaajat liikkuvat maasta ja sarjasta toiseen yhä useammin.

- Seuratietokanta. Koska palloilulajien ottelutuloksissa ovat vastakkain seurat, niiden tiedot tallentuvat järjestelmään. Pakollisten nimitietojen ja perustamisvuoden lisäksi voi tallentaa muutakin seuroja koskevaa tietoa.
- Oy Veikkaus Ab:lle tai muille peliyhtiöille tarkoitettu tilastotyökalu, joka kerää kannasta yksittäistä pelikohdetta koskevat tärkeimmät sen arviointiin vaikuttavat tiedot.
- Liittojen ja seurojen tilastoinnin ulkoistaminen. Voidaan rakentaa web-service -tekniikalla toimiva palvelu, joka lähettää pyytävälle sovellukselle tarvittavat tilastot (esimerkiksi sarjataulukko). Tällöin seuran tai lajiliiton ei tarvitse itse huolehtia tilastointipalveluistaan, vaan esimerkiksi liiton sivulla oleva sarjataulukkotoiminto voi kannasta haun ja laskennan sijasta lähettää pyynnön palvelulle. Web-service palauttaa pyydetyn tiedon xml-sanomana, joka kutsuvan sovelluksen tarvitsee vain purkaa sivulleen haluamaansa esitystapaan. Tämä toimintamalli tosin edellyttäisi tietojen alkutallennusta palveluntarjoajan toimesta, koska niitä ei olisi enää mahdollista päivittää automaattisesti ainakaan kyseisen palvelun tilaajan sivuilta.
- Automatisoitu veikkausvihjepalvelu, jossa sovellus laskee eri muuttujien avulla otteluiden todennäköisyysarvioita perustuen kannassa oleviin seuran aikaisempiin otteluihin.
- Internetissä toimiva työkalu, jolla käyttäjä voi rakentaa oman tietovarastonsa, joka toimii samoilla periaatteilla kuin laajempi tietovarasto. Tällöin seuran ei tarvitse rakentaa sivustolleen omia tilastointipalveluja, vaan se voi tämän palvelun avulla räätälöidä tarkoitukseensa sopivan palvelun ja linkittää sen seuran sivustolle.
- Otteluennakot ja informaatiopaketit. Usein merkittävien otteluiden edellä on tarvetta taustoittaa tulevaa ottelua tavanomaista tarkemmalla tasolla. Tietovaraston omistaja voi hyödyntää varastoa siten, että sen avulla hän voi poimia erilaista faktatietoa johonkin tulevaan otteluun liittyen.
- Kuvan 10 esimerkin mukainen tulospalvelu internetissä. Palvelu seuraa automaattisesti tärkeimpiä palloilusarjoja ja perustilastot päivittyvät sivustolle automaattisesti heti kun ottelu on päivitetty järjestelmään.

http://www.score24.com/stats/statistics2.jsp?partner=iltasanomat&country=Finland&lang=fin&tz=2&eventId=6117

ILTA-SANOMAT IS Veikkaaja TULOSPALVELU

Suomi | Jalkapallo | Muut lajit | Takaisin | Arsenal | Arsenal | Valitse

Jalkapallo - Englanti - Premier League Otteluohjelma: Kaikki Tulevat

Viimeisin kierros

Päivä	Ottelu	Tulospalvelu
2008-02-02	Manchester C - **Arsenal**	1-3
2008-02-02	Birmingham - Derby	1-1
2008-02-02	Blackburn - Everton	0-0
2008-02-02	Portsmouth - Chelsea	1-1
2008-02-02	Reading - **Bolton**	0-2
2008-02-02	Tottenham - Manchester U	1-1
2008-02-02	**Wigan** - West Ham	1-0
2008-02-02	**Liverpool** - Sunderland	3-0
2008-02-03	Newcastle - Middlesbrough	
2008-02-03	Fulham - Aston Villa	

Seuraava kierros

Päivä	Ottelu
2008-02-09	Aston Villa - Newcastle
2008-02-09	Bolton - Portsmouth
2008-02-09	Derby - Tottenham
2008-02-09	Everton - Reading
2008-02-09	Middlesbrough - Fulham
2008-02-09	Sunderland - Wigan
2008-02-09	West Ham - Birmingham
2008-02-10	Manchester U - Manchester C
2008-02-10	Chelsea - Liverpool
2008-02-11	Arsenal - Blackburn

Saat lisää tilastoja klikkaamalla joukkuetta taulukossa.

Sarjataulukko

		Kotona							Vieraissa							Yhteensä							
Sija	Joukkue	Ott	V	T	H	M	PM	P	Ott	V	T	H	M	PM	P	Ott	V	T	H	M	PM	ME	P
1	Arsenal	13	11	2	0	29	8	35	12	7	4	1	23	10	25	25	18	6	1	52	18	34	**60**
2	Manchester U	13	12	1	0	31	3	37	12	6	3	3	18	9	21	25	18	4	3	49	12	37	**58**
3	Chelsea	12	8	4	0	23	8	28	13	8	2	3	15	9	26	25	16	6	3	38	17	21	**54**
4	Everton	12	7	2	3	23	11	23	13	6	3	4	17	12	21	25	13	5	7	40	23	17	**44**
5	Liverpool	12	5	6	1	26	9	21	12	6	4	2	14	8	22	24	11	10	3	40	17	23	**43**

Kuva 10: Esimerkki Ilta-Sanomien tulospalvelusta.

4.2. Palloilutietovaraston laajuus

Tilastotietovarastolle on ominaista, ettei sen tietosisältöä aleta kerätä varaston käyttöönotosta alkaen, vaan varaston käyttötarkoituksen kannalta on jopa välttämätöntä, että sitä laajennetaan koko ajan sekä ajallisen ulottuvuuden että karkeisuuden suhteen. Varastoon ajetaan viimeisimpien tietojen lisäksi myös vanhaa dataa yhä vanhemmalta ajalta ja myös toisaalta tuoretta dataa yhä laajemmalta alueelta (esimerkiksi uusien maiden tai sarjojen tietoja).

Tietovaraston teoreettisena tavoitteena on kattaa kaikkien palloilusarjojen kaikki tiedot kaikilta ajoilta. Käytännössä kuitenkin järjestelmää on rakennettava vaiheittain siten, että uusia sarjoja tai kilpailuita otetaan mukaan toisaalta yleisen kiinnostuksen, toisaalta tietojen saatavuuden ohjaamana. Jälkimmäinen näkökohta tulee vastaan esimerkiksi silloin, kun huomataan, että tietyn sarjan tilastopoimintaan tehdyllä proseduurilla saadaan muutoksitta tai vähäisin muutoksin poimittua myös toisen sarjan tiedot.

Tietovaraston kokoon yleensä vaikuttavat käytännössä seuraavat tekijät [Hovi et al. 2001]:

- tietovaraston kattavuus (mistä järjestelmistä tietoa)
- tietovaraston laajuus (paljonko tietoja eri järjestelmistä)
- karkeisuus (millä tasolla tiedot tuodaan eri järjestelmistä)
- säilytysaika (kuinka monen vuoden tiedot säilytetään kannassa).

Erona yrityksen tietovarastoon palloilutietovaraston karkeisuuden tulee olla dynaaminen, sillä jopa saman tietolajin sisällä tietovaraston tulee sallia eri karkeisuustasoilla olevia tietoja. Koska tietojen yleisistä saatavuusongelmista johtuen etenkin vanhoista historiatiedoista saattaa puuttua esimerkiksi otteluiden kokoonpanoja tai maalintekijöitä, järjestelmän tulee mahdollistaa se, että vain osalle vanhoista tiedoista nämä tiedot ovat saatavilla. Yrityksen tietovarastoissa granulariteetti normaalisti on aikajanan tietyssä kohdassa sama kaikkien tietojen osalta. Esimerkiksi kuluvan kuukauden kaikki myyntitiedot ovat päivätasolla, mutta sitä vanhemmat kuukausitasolla.

Näin ollen historiadatan ongelmista johtuen ei ole mahdollista noudattaa tietovarastoinnin yhtä suositusta: kaikkien lopullisen faktataulun rivien tulisi olla samalla tarkkuustasolla, koska se mahdollistaa faktataulun luontevan laajentamisen uusia faktoja tai dimensioita lisäämällä. Sen ansiosta myös kaikki tiedot ovat käyttäjän nähtävissä kaikilla dimensiotasoilla [Kimball et al. 1998].

Historiadatan keruuprosessin vaatimukset voivat olla osin erilaisia kuin aktiividatan. Koska aktiividata elää koko ajan, keruuprosessin on kyettävä muokkaamaan tieto suoraan tietovarastomuotoon. Historiadatan osalta riittää, että automatisointi jalostaa tiedon sellaiseen muotoon, että se voidaan kohtuullisella vaivalla saattaa manuaalisesti lopulliseen muotoonsa.

4.3. Summaukset ja yhteenvedot

Palloilulajien tietovarastossa historiatiedon osuus on keskeisempi kuin yritysten tietovarastoissa. Yleinen tiedontarve on esimerkiksi saada tietää Evertonin ja Aston Villan keskenään pelaamien otteluiden osalta se, kumpi on voittanut kautta aikojen enemmän keskinäisiä otteluita. Seurat kohtasivat ensimmäisen kerran pelikaudella 1888-89, joten tiedon laskenta edellyttää kaikkien tietojen säilyttämistä. Koska on muita samantyyppisiä tiedontarpeita, vanhoja detaljitason tietoja ei voida korvata summatauluilla.

Summataulut eivät kuitenkaan ole järjestelmän kannalta merkityksettömiä vaan erittäin oleellisia. Aihealueeseen liittyen on vakiintunut joukko esitystapoja, joiden perusluonne on juuri summaava. Kaikki perinteiset tilastot

ovat summatauluja, vaikka niiden laskennassa käytetty logiikka usein onkin mutkikkaampaa kuin yhteenlasku.

Palloilutietovaraston tyypillisin summataulu on sarjataulukko, jossa joukkueet on lajiteltu paremmuusjärjestykseen niiden otteluista keräämien pisteiden perusteella. Muita sarjataulukon perustietoja ovat lajista riippumatta joukkueen pelikauden aikana saavuttamien voittojen, tasapelien tai häviöiden lukumäärä. Sarjataulukosta toinen yleisesti käytetty versio on ns. kuntopuntari, johon on laskettu mukaan vain joukkueen viimeisimmät ottelut, jolloin se koko kauden taulukkoa paremmin kuvastaa joukkueen vireystilaa tarkasteluhetkellä.

Muita perinteisiä summatauluja ovat maalintekijä- ja ottelutilasto, joista käy ilmi yksittäisten pelaajien pelaamien otteluiden ja maalien kokonaismäärät. Lisäksi tietovarastokannasta voidaan generoida myöhemmin esimerkiksi minkä tahansa aikaisemman pelipäivän jälkeinen sarjataulukko tai muu tilasto. Historiatiedot mahdollistavat myös sarjataulukon tai pelaajatilastojen laskennan siten, että sarjan koko historia on huomioitu. Sarjataulukkoa, jossa ovat mukana kaikki sarjassa kautta aikojen pelatut ottelut, kutsutaan usein maratontaulukoksi.

Summataulujen fyysinen tallennus järjestelmään riippuu tietovaraston käyttötarkoituksesta. Jos tietovarastoa käyttää työasemasovellus, tarvittavat summaukset voidaan suorittaa aina uudelleen sovellukselta niitä pyydettäessä, eikä summatietoja ole siten tarpeellista tallentaa erillisiin summatauluihin. Sen sijaan verkkopalvelussa on järkevää palauttaa käyttäjälle valmiiksi laskettu sarjataulukko tai muu tilasto sen sijaan, että se generoitaisiin kaikille käyttäjille aina uudelleen. Valmiiksi summatun tilaston palauttaminen on perusteltua myös kuormitussyistä, sillä järjestelmään saattaa esimerkiksi tärkeän ottelun päättyessä kohdistua lukemattomia kertoja sama kysely lyhyen ajan sisällä.

Tietojen puutteellisesta saatavuudesta johtuen jotkut tiedot voivat olla järjestelmässä tallennettuna vain summatauluina. Esimerkiksi alempien sarjatasojen tiedot voivat olla tietovarastossa vain kausikohtaisina sarjataulukoina ottelutulosten puuttuessa tai ollessa liian työläitä siirtää järjestelmään suhteessa niiden hyötyarvoon.

4.4. Tietovaraston ajantasaisuus

Esimerkiksi mediajakeluun tarkoitettu tulevaa merkittävää ottelua koskeva ennakkopaketti edellyttää, että järjestelmä tuottaa aiheesta niin suuren määrän informaatiota, että siitä voidaan manuaalisesti editoida kiinnostavimmat kohdat eri medioihin. Valmiiksi jatkojalostettu aineisto taas edellyttää, että

aikaa sen jalostamiseen on riittävästi, jolloin on tärkeää saada ajantasainen perusdata nopeasti ja helposti.

Siihen, miten ajan tasalla tietovaraston tulisi olla, vaikuttavat ainakin seuraavat tekijät [Elmasri & Navathe 2000]:

- voiko tietovarasto olla välillä pois käytöstä
- miten pitkään tietovarasto voi olla pois käytöstä
- mitkä ovat datan keskinäiset riippuvuudet
- tietovaraston teknisen alustan saatavilla olo
- jakeluun/julkaisuun liittyvät vaatimukset
- lataukseen kuluva kokonaisaika (mukaan lukien käsittely- ja siirtoaika).

Haun vasteajan kriittisyys riippuu kerättävien tietojen laajuudesta, ja tietovaraston tarjoamista palveluista riippuu, miten tärkeää on sen ajantasaisuus. Yksittäisen sarjan yhden tai useamman kauden kaikkien otteluiden tietojen lataus saattaa olla pitkäkestoinenkin prosessi, mutta kun aikaisempi historia on kerran päivitetty, riittää vain viimeisimpien otteluiden päivitys.

Tiettyinä ajankohtina monissa sarjoissa pelataan lyhyen ajan sisällä suuri määrä otteluita. Tällöin on kiinnitettävä huomiota latauksen kestoaikaan, jos tietovaraston tietoja on tarkoitus näyttää esimerkiksi internetin verkkopalvelussa. Rinnakkaisten tiedonkeruuprosessien tarve voi tulla, jos suuri määrä aineistoa on samalla hetkellä päivitettävä ajan tasalle. Esimerkiksi lauantaisin päättyy usein suuri määrä otteluita liki samanaikaisesti ja tietovaraston käyttötarkoitus voi vaatia, että päivitysprosessi on hajautettu tehokkuussyistä. Koska eri sarjoille on käytännössä laadittava eri poimintaohjelmat, niitä voidaan helpostikin ajaa yhtä aikaa eri työasemilta.

4.5. Tietokannan tekninen rakenne

Internetissä olevien erilaisten tilastotietokantojen perusrakenne on muotoutunut varsin standardiksi. Mahdollisten esivalintojen (seura, kausi tms.) jälkeen käyttäjälle avautuu lista, jossa näytetään valintaehdot täyttävien otteluiden perustiedot (päivämäärä, vastakkain olevat joukkueet ja lopputulos). Valitsemalla yksittäinen ottelu listasta voidaan porautua tarkemmin ottelua koskeviin tietoihin (joukkueiden kokoonpanot, maalintekijät, tuomarit jne.).

Internetissä valmiina olevat tilastotietokannat ohjaavat osaltaan kohdetietovaraston rakennetta, sillä datan määrästä johtuen ei ole yleensä mahdollista täydentää tietoja manuaalisesti. Sellaisia otteluun tai pelaajaan

liittyviä faktoja ei kantaan ainakaan laajamittaisesti kannata varastoida, joita ei ole ladattavissa internetistä.

Lajien rajat ylittävää tietovarastoa ei ole tarkoitus muodostaa, vaikka etenkin menneinä vuosikymmeninä oli kohtalaisen yleistä, että sama pelaaja on harrastanut useita lajeja jopa maajoukkuetasolla asti. Nykypäivänä lajit ovat ammattilaistuneet siinä määrin, että useassa lajissa esiintyminen huipputasolla ei ole enää mahdollista. Poikkeuksena ehkä ovat jotkut ”sukulaislajit” kuten jalkapallo ja futsal.

Teknisesti eri lajien tietovarastot pyritään kuitenkin pitämään samankaltaisina, jolloin niitä voidaan käsitellä joko samoilla sovelluksilla tai vain vähän niitä muokattuna eri lajeja varten. Käytännössä suurin osa tietoalkioista ja tietojen esittämistavoista on yhteisiä kaikille palloilulajeille. Kaikissa on vastakkain kaksi joukkuetta, joiden keskinäisestä ottelusta kirjataan numeerinen tulos. Useimmiten tulos tarkoittaa joukkueiden maalimääriä ottelussa, mutta joissakin lajeissa se tarkoittaa voitettuja eriä tai jaksoja (lentopallo, pesäpallo). Palloilulajien tietovaraston rakennetta on kuvattu yksityiskohtaisemmin liitteissä 1 ja 2.

4.6. Ongelma-alueita esimerkkitietovarastossa

Seurojen nimenmuutokset ovat tietovaraston muodostuksen ja myös käytettävyyden kannalta haasteellinen ongelma. Etenkin Suomessa, jossa seuranimet eivät ole muodostuneet kaupallisiksi tavaramerkeiksi, seuroja on fuusioitu sattumanvaraisesti ja usein on tulkintakysymys, katsotaanko joku uusi seura jonkun aikaisemman toiminnan jatkajaksi. Nimenmuutoshistorialla on merkitystä etenkin historiavedoksissa, joissa voidaan selvittää jonkun seuran aikaisempaa menestystä.

Yksinkertainen, mutta suoraviivaisuudessaan käyttökelpoinen ratkaisu on sellainen, jossa seuran eri nimivariaatiot on kirjattu eri seuroiksi. Tällöin seuratietoon liittyy kenttä, joka sisältää viittauksen samassa rekisterissä olevaan seuran nykyiseen nimeen. Tällöin kytkentä kahden erinimisen seuran välillä voidaan tarvittaessa purkaa viittaus poistamalla.

Pelaajien tai muiden henkilöiden nimenmuutokset ovat harvinaisempia. Seuranimiin liittyvä jatkumon tulkinnanvaraisuus puuttuu henkilötasolta, joten ratkaisuna pelaajien näyttäminen nykyisillä tai heidän viimeksi käyttämillään nimillä olisi perusteltua. Tällöin kuitenkin olisi pelaajaa koskevaan lisäinformaatioon tallennettava tieto nimenmuutoksesta.

Vaikka lähdetiedoissa yksittäiset dimensiot (seura, pelaaja) esiintyvät vaihtelevissa kirjoitusasuissa, tietovaraston päivitysvaiheen on silti osattava tunnistaa, onko pelaaja tai seura jo tietovarastossa. Esimerkiksi Suomen Veikkausliigassa pelanneen Sami Hyypiän tietojen tulee integroitua Englannin Valioliigasta tuleviin Sami Hyypian tietoihin. On löydettävä tunnistussääntöjä, joilla latausproseduuri kykenee esimerkiksi syntymäajan perusteella päättelemään suurimman osan sellaisista tapauksista, jotka ihminen ymmärtää luonnostaan samaksi henkilöksi.

Yhtenä haasteena ovat esimerkiksi Espanjassa tai Brasiliassa käytetyt pelaajien taiteilijanimet. Esimerkiksi Adriano-nimellä yleisesti tunnettuja jalkapalloilijoita on useita. Jääkiekkoa ongelma ei käytännössä koske ainakin lajin huipun keskittyessä perinteisten nimikäytäntöjen maihin.

Seurojen nimillä voi myös olla useita erilaisia kirjoitusasuja. Jos tietovarasto on suunnattu kansainväliseen käyttöön, sen on kyettävä näyttämään esimerkiksi suurimpien eurooppalaisten kaupunkien huippuseurojen nimet kansallisilla kirjoitustavoilla. Esimerkiksi Inter Milan tunnetaan Saksassa yleisemmin nimellä Inter Mailand. Vastaavasti FC Kööpenhaminan nimeä ei voida tallentaa yksinomaan tuossa muodossa, jos tietovaraston tietoja on tarkoitus käyttää maamme rajojen ulkopuolella.

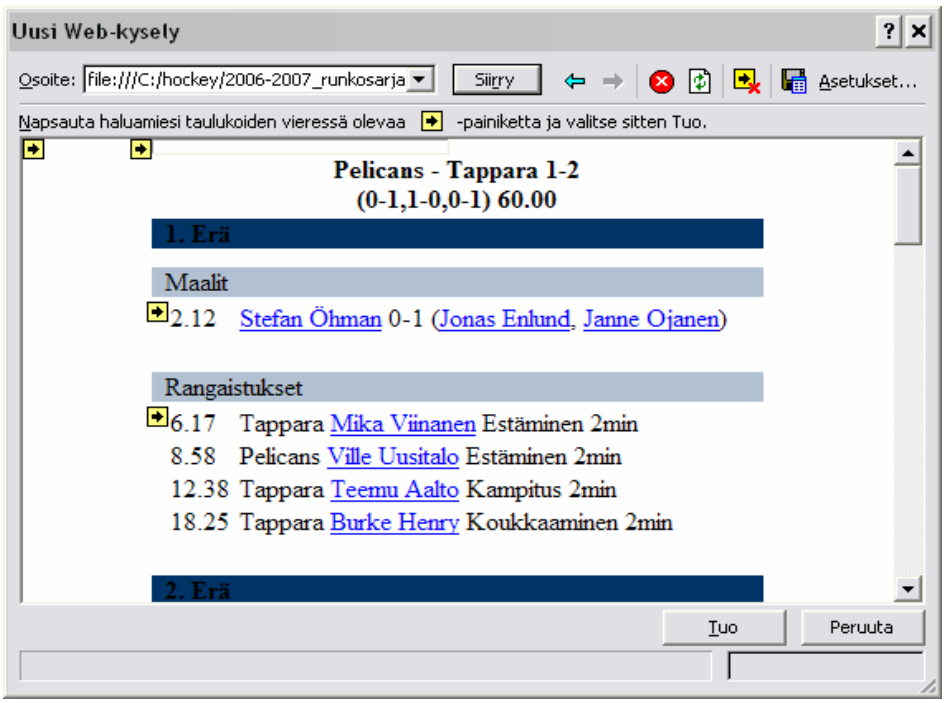
Ongelmia tuottavat myös muut kuin länsieurooppalaisia merkistöjä käyttävät kielet. Tällöin pelaajan nimi on translitteroitava erikseen ainakin niiden kielialueiden kirjoitusasuun, joissa sovellusta on tarkoitus käyttää.

5. Teknisiä ratkaisuja ja toimintamalleja

5.1. Tietojen haku internetistä Excelin web-kyselyn avulla

Helpoin tietojen poiminnan automatisointiin soveltuva menetelmä on Excelin web-kysely (Web Query). Se mahdollistaa jopa tietoalueen automaattisen päivityksen Excel-taulukkoon internetistä, mikä tosin ei tietovaraston latauksen kannalta ole merkityksellinen ominaisuus, koska tietovarastoon siirrettävän datan tulee nimenomaan olla tietyn hetken otos lähdejärjestelmästä.

Toiminto jäsentää kuvan 11 esimerkin mukaisesti annetun html-sivun taulukkoalueet valmiiksi, joista käyttäjä voi valita yhden tai useamman alueen siirrettäväksi Exceliin. Asetukset-painikkeella voi lisäksi määritellä oletusarvot korvaavia tietojen jäsennysohjeita, joissa voi määritellä mm. sen, tuodaanko Excel-taulukkosivulle vain data vai myös muotoilut. Asetuksissa voidaan myös estää Excelin automaattinen päivämäärätunnistus, jonka vuoksi esimerkiksi ottelutulos 1-2 jäsentyy oletusarvoisesti muotoon ”1. helmikuuta”.



Kuva 11: Esimerkki Excelin web-kyselyn käyttöliittymästä.

Nauhoitettaessa Excelin toimintosarja saadaan koodikatkelman 1 mukainen koodirunko, joka toistorakenteeseen sijoitettuna mahdollistaa

tietojen haun sivu kerrallaan Excelin taulukkoon, josta tiedot sitten erillisellä proseduurilla voidaan siirtää varsinaiseen tietokantaan.

```
With ActiveSheet.QueryTables.Add(Connection:= _
    "URL;http://… tapahtumat_34451.htm", Destination:=Range("A1"))
    .Name = "…tapahtumat_34451"
    .FieldNames = True
    .RowNumbers = False
    .FillAdjacentFormulas = False
    .PreserveFormatting = True
    .RefreshOnFileOpen = False
    .BackgroundQuery = True
    .RefreshStyle = xlInsertDeleteCells
    .SavePassword = False
    .SaveData = True
    .AdjustColumnWidth = True
    .RefreshPeriod = 0
    .WebSelectionType = xlEntirePage
    .WebFormatting = xlWebFormattingNone
    .WebPreFormattedTextToColumns = True
    .WebConsecutiveDelimitersAsOne = True
    .WebSingleBlockTextImport = False
    .WebDisableDateRecognition = True
    .WebDisableRedirections = False
    .Refresh BackgroundQuery:=False
End With
```

Koodikatkelma 1: Excelin web-kyselyn nauhoituksen tuottama makro.

Valittaessa kaikkien taulukkoalueiden (*WebSelectionType* = *xlEntirePage*) sijaan vain osa sivun taulukkoalueista (*WebSelectionType* = *xlSpecifiedTables*) toiminnolla on lisäparametri *WebTables*, joka sisältää pilkulla erotettuna merkkijonona haluttujen taulukkoalueiden järjestysnumerot (esimerkiksi *WebTables* = "*2,3*").

Ainakin vanhemmissa html-aineistoissa voi esiintyä <*pre*>-komennolla alkavia esimuotoiltuja taulukkoalueita, jotka eivät teknisesti ole taulukoita vaan vapaamuotoista tekstiä, joka on välilyönneillä ja tasavälisellä fontilla muotoiltu

siten, että selaimessa se näyttää taulukolta. Asetusten parametreilla *WebConsecutiveDelimitersAsOne* = *True* ja *WebPreFormattedTextToColumns* = *True* saadaan usein esimuotoiltu aineistokin avattua Exceliin taulukoksi jäsennettynä.

Tulevia tarpeita varten kerran Exceliin ladattu aineisto kannattaa tallentaa paikallisesti. Tällöin tietojen jatkokäsittelyssä ei enää tarvitse jäsentää html-dataa tietojen ollessa valmiiksi esijäsennettynä Excel-taulukoissa.

5.2. Tiedon luku ja tallennus internetistä

Internet-taustaisen datan jäsentämistä yksinkertaisempi, mutta prosessin kannalta yhtä oleellinen vaihe, on haluttujen html-sivujen lataus ohjelmallisesti joko muistiin tai levytaltioksi. Ohjelmallinen toteutus on mahdollista useallakin eri tavalla. Tässä esitellään niistä kolme, joiden rajoituksena on se, että ne ovat käytettävissä vain Windows-ympäristössä.

Windowsin vakiokirjastoihin kuuluvat muun muassa koodikatkelmassa 2 luetellut API-ohjelmointirajapinnan (application program interface) funktiot, joiden avulla voidaan varsin helposti lukea objekteja internetistä. Ladattavan tiedoston ei tarvitse olla html-sivu vaan se voi olla yhtä hyvin mikä tahansa dokumentti tai mediatiedosto.

```
Public Declare Function InternetOpen Lib "wininet"
Public Declare Function InternetConnect Lib "wininet"
Public Declare Function InternetOpenUrl Lib "wininet.dll"
Public Declare Function InternetReadFile Lib "wininet.dll"
Public Declare Function InternetCloseHandle Lib "wininet.dll"
Public Declare Function URLDownloadToFile Lib "urlmon"
```

Koodikatkelma 2: Windowsin API-rajapinnan funktioita.

Vakiokirjastojen palveluiden ympärille voidaan rakentaa funktioita, joiden kaikkien tehtävänä on palauttaa html-sivun koko sisältö yhtenä merkkijonona. Erilaiset vaihtoehdot ovat perusteltuja siksi, että eri tavoissa erilainen osuus funktioiden toiminnoista on koottu Windowsin palveluista ja niissä voi siten proseduuria muokkaamalla vaikuttaa eri kohtiin.

Koodikatkelman 3 funktio on variaatio yleisestä perusratkaisusta, jossa tiedoston luku suoritetaan vaiheittain avaamalla ensin internet-yhteys, minkä

jälkeen avataan yhteys haluttuun url-osoitteeseen. Tämän jälkeen tietoa luetaan niin paljon kuin sitä on saatavilla ja lopuksi internet- yhteys suljetaan.

```
Public Function GetHTMLFromURL(URL As String) As String
   - - -
  'Avataan ensin internet-yhteys ja sen jälkeen URL
  hOpen = InternetOpen("", INTERNET_OPEN_TYPE_PRECONFIG, _
    vbNullString, vbNullString, 0)
  hOpenUrl = InternetOpenUrl(hOpen, URL, vbNullString, 0, _
    INTERNET_FLAG_RELOAD, 0)
  'Luetaan niin kauan kuin merkkejä on luettavissa
  Do
    Call InternetReadFile(hOpenUrl, ReadBuffer, _
      Len(ReadBuffer), NumberOfBytesRead)
    StrVar = StrVar & Left$(ReadBuffer, NumberOfBytesRead)
  Loop Until (NumberOfBytesRead = 0)
   - - -
```

Koodikatkelma 3: Html-sivun luku merkkijonoksi.

Koodikatkelman 4 funktio perustuu API-funktioon *URLDownloadToFile*, jonka avulla voidaan yhdellä lauseella ladata tiedosto internetistä levylle. Sen ympärille on rakennettu sovelluskehys, jossa osa parametreista käsitellään oletusarvoina ja kutsussa välitetään vain ladattava url-osoite ja tallennettavan paikallisen tiedoston nimi. Funktion rajoituksena on se, että se on käytettävissä vain niissä ympäristöissä, joihin on asennettu Internet Explorer –selain.

```
Public Function DownloadFile(SourceUrl As String, _
  LocalFile As String) As Boolean

  'URLDownloadToFile pelkistettynä (lisäparametrit oletusarvoina)
  DownloadFile = (URLDownloadToFile(0&, SourceUrl, LocalFile, _
    0&, 0&) = ERROR_SUCCESS)
   - - -
```

Koodikatkelma 4: Html-sivun talletus levylle *UrlDownLoadFile*-funktion avulla.

Kolmas tapa hakea tiedosto perustuu Internet Explorerin mukana tulevaan ActiveX-luokkaan *MSXML2*. Sen ensisijainen tarkoitus on mahdollistaa xml-sanoman lähetys annettuun osoitteeseen, mutta käytettäessä HTTP-metodia GET, se palauttaa kohdesivun sisällön. Tämän menettelytavan etuja on se, että sitä voidaan käyttää esimerkiksi asp-ohjelmoinnissa, mikä ei ole suoraan mahdollista kahdella muulla tallennustavalla.

```
Public Function SendXML(ByVal XML As String, ByVal HTTPMethod, _
  ByVal URL As String) As String
   - - -
   'Luodaan ActiveX-objekti
   Set X = CreateObject("Msxml2.XMLHTTP")
   - - -
   'Lähetetään mahdollinen XML-sanoma
   X.Open HTTPMethod, URL, False
   X.SetRequestHeader "Content-Type", "text/xml"
   X.Send Trim(XML)

   'Paluusanoma on annetun URLin sisältö GET-metodilla
   SendXML = X.ResponseText
   - - -
```

Koodikatkelma 5: Sivun lataaminen MSXML2-luokan metodin avulla.

Koodikatkelmassa 6 esitelty funktio *DownLoadFiles* on tarkoitettu hieman mutkikkaampien url-osoitteiden tallennukseen. Funktiolle voidaan välittää parametrit X ja Y, joiden avulla voidaan varioida url-osoitteessa kahta juoksevaa numeroa (tai kirjainta A-Z). Lisäksi ladattavalle sivulle voidaan antaa minimikoko tavuina, jolloin funktio ohittaa ne vaihtuvassa url-osoitteessa olevat linkit, joiden takana ei ole tietoa. Funktion tällöin vastaanottama virheilmoitus (”sivua ei löydy” tms.) on yleensä tavumäärältään oleellisesti pienempi kuin tavoiteltavat datasivut.

Funktiossa voidaan määritellä myös ladattavien tiedostojen enimmäismäärä. Tällöin voidaan asettaa vaihtuvan url-osoitteen numerosarjat ilman tarkkaa tietoa numeroinnista, jos on tiedossa, että tiedostoja voi olla enintään tietty lukumäärä, vaikka tunnisteen numerointi ei olisi aukottomasti

juokseva. Tällöin proseduuri pysähtyy onnistuneiden latausten määrän tultua täyteen.

```
Public Sub DownloadFiles(ByVal SourceUrl As String, ByVal LocalFile As _
  String, ByVal X_Start As Variant, ByVal X_End As Variant, _
  ByVal Y_Start As Variant, ByVal Y_End As Variant, ByVal LeadingZeros _
  As Boolean, ByVal MinimumSize As Long, ByRef FileCounter As Long, _
  ByVal MaxFiles As Long, ByVal ReplaceFiles As Boolean, _
  ByVal ExitInnerLoop As Boolean)
  - - -
  For Counter1 = X_Start To X_End
    For Counter2 = Y_Start To Y_End
      - - -
      If X_Char Then
        URL = Replace(URL, "%X%", Chr(Counter1), , , vbTextCompare)
        LocalFileName = Replace(LocalFileName, "%X%",
          Chr(Counter1), , , vbTextCompare)
      Else
        URL = Replace(URL, "%X%", Format(Counter1, FormatString1), , _
          , vbTextCompare)
        LocalFileName = Replace(LocalFileName, "%X%", Format(Counter1, _
          FormatString1), , , vbTextCompare)
      End If
      - - -
      'Parametrilla voidaan myös ohittaa jo aikaisemmin talletettu tiedosto
      If ReplaceFiles Or (Dir(LocalFileName) = "") Then
        Call DownloadFile(URL, LocalFileName)
      End If
      'Jos ladattu tiedosto alle minimikoon, tuhotaan se
      If (MinimumSize > 0) And (FileLen(LocalFileName) < MinimumSize) Then
        Kill LocalFileName
      - - -
      'Jos maksimimäärä kelvollisia tiedostoja talletettu, poistutaan
      If (MaxFiles > 0) And (FileCounter >= MaxFiles) Then Exit Sub
      - - -
```

Koodikatkelma 6: Sivujen erälatausfunktio *DownLoadFile*.

Url-osoitteen variointiparametrien vaihteluvälit annetaan erillisinä argumentteina. Parametrien sijainti osoitetaan korvaamalla argumenttina välitettävässä url-osoitteessa X-argumentti merkinnällä %X% ja Y-argumentti merkinnällä %Y%. Proseduurin kutsu voi olla esimerkiksi koodikatkelman 7 mukainen.

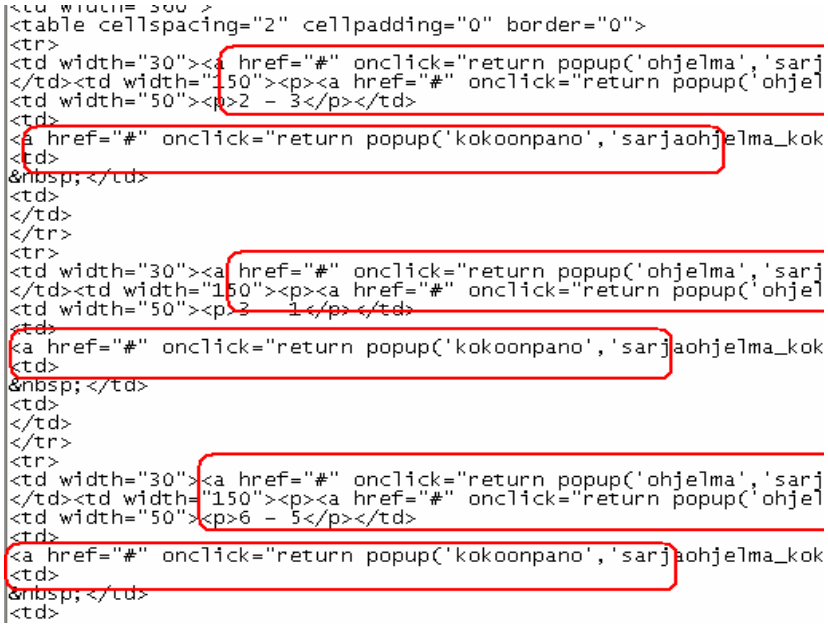
```
DownloadFiles("http://www.mtv3.fi/urheilu/arkisto.shtml/arkistot/futis/%X
%-%Y%", "C:\DATA\DATA%N.HTM", 2000, 2007, 1, 12, true, 0, FileCounter, 0,
false, false)
```

Koodikatkelma 7: Parametroidun url-osoitteen lataaminen kerralla.

Tallennettavan tiedoston nimessä voidaan käyttää vastaavia *%X%- ja %Y%* -merkintöjä ja lisäksi merkintää *%N%*, jolla saadaan tiedostot nimettyä juoksevalla numeroinnilla. Lisäksi proseduuria voidaan ohjata käyttämään tunnisteissa etunollia aloitusparametrin alkuarvon muodon määrätessä etunollien määrän. Parametri *ExitInnerLoop* aiheuttaa sen, että sisimmästä silmukasta poistutaan ehdot täyttävän sivun löydyttyä. Optio on käyttökelpoinen, kun tiedetään, että loogisesti silmukoita on vain yksi, mutta jokin toinen muuttuja url-osoitteessa vaihtelee joissakin rajoissa.

5.3. Html-sivun linkkien poiminta

Usein datan poiminta edellyttää sivulla olevien linkkien jäljittämistä (esimerkki kuvassa 12), sillä osa asiakokonaisuuteen liittyvistä tiedoista on selainkäytön selkeyttämiseksi saatavissa erillistä linkkiä seuraamalla. Rakenne mukailee näin osin taustalla olevaa relaatiotietokantaa, sillä pääsivulla on ainoastaan tietojen otsikot, joista sitten voidaan porautua yksityiskohtaisiin tietoihin.



Kuva 12: Html-sivun poimittavia linkkejä.

Koodikatkelman 8 funktio etsii linkit html-sivulta merkkijonohakuna. Koska usein tietokantaan viittaavat linkit ovat rakenteeltaan samanmuotoisia, funktiolle voidaan antaa viittauksen alkua niin paljon kuin on tarpeen oikeiden viittausten löytämiseksi. Argumentin alussa on annettava myös “href=…”-osuus. Kolmannella argumentilla ohjataan sitä, otetaanko mukaan vain ne linkit, joilla on myös merkkijonona ilmaistava arvo. Parametrin ollessa *true* esimerkiksi viittaukset kuviin ohitetaan.

```
Function GetHRefs(ByVal HTML As String, ByVal FindHRef As String, _
  ByVal TextLinksOnly As Boolean) As Variant
   - - -
  'Etsitään href-alkuisen viittauksen ensimmäinen esiintymä
  Position1 = InStr(Position1 + 1, HTML, FindHRef, vbTextCompare)

  'Kerätään kaikki annetulla merkkijonolla alkavat linkit
  'ensin merkkijonoon (ASCII-koodi 255 erottaa merkkijonossa
  'eri osat/linkit)
  While (Position1 > 0)
    HRef = Mid(HTML, Position1, Len(HTML))
    Position2 = InStr(1, HRef, ">", vbTextCompare)
    If (Position2 > 0) Then
```

```
      'Linkkiviittaus päättyy lopulta <-merkkiin
      Position3 = InStr(1, HRef, "<", vbTextCompare)
      'Jos arvona ei-tyhjä (ei ><) tai otetaan kaikki linkit
      '(muuten vain ne, joilla teksti, jota klikkaamalla linkki avautuu)
      If (Position3 > Position2 + 1) Or (TextLinksOnly = False) Then
        HRefList = HRefList & Chr(255) & Left(HRef, Position3)
      End If
    End If
    Position1 = InStr(Position1 + 1, HTML, FindHRef, vbTextCompare)
  Wend
  - - -
```

Koodikatkelma 8: Html-sivun linkkien poiminta taulukkomuuttujaan.

Edellä esiteltyjen funktioiden avulla voidaan rakentaa yksinkertainen web-arkistointijärjestelmä, joka kerää annetun aloitusosoitteen sisällön linkkeineen paikalliseen taltioon. Algoritmissa tulisi tällöin ottaa kantaa ainakin seuraaviin asioihin:

- sen tulisi osata ohittaa jo aikaisemmin prosessin aikana taltioidut sivut
- sen tulisi ohittaa linkit, jotka johtavat eri pääsivuille, koska silloin prosessi saattaisi olla päättymätön
- sen tulisi tarvittaessa ohittaa kuvien lataus tilan säästämiseksi (ellei haluta arkistoida myös sivuston visuaalista olemusta).

5.4. Taulukkomuotoisen datan eristäminen html-dokumentista

Internetistä kerättävä ja formaaliin muotoon muunnettava data on usein jo valmiiksi taulukkomuotoisena html-dokumentissa. Alun perin html-komennot <*table*> ja <*/table*> on tarkoitettu vain taulukkomuotoisen datan esittämiseen internetissä, mutta myöhemmin niitä on alettu käyttää muunkin sisällön jäsentämisen apuvälineenä.

Taulukon alkua ja loppua ilmaisevat <*table*> ja <*/table*> sijoittuvat html-dokumentissa <*body*>- ja <*/body*>-komentojen sisään, ja niitä voi olla useita sisäkkäin. Komento <*tr*> aloittaa taulukon rivin ja <*td*> uuden solun. Vastaavat lopetussymbolit ovat <*/tr*> ja <*/td*>. Näiden lisäksi komentopari <*th*> ja <*/th*> aloittaa ja päättää taulukon otsikon.

```
    <td background="img/common/horizLine.gif"><img src="img/common/horizLine
</tr>
<tr>
        <td align="center">
                <br>
                <!-- joukkueet -->
<table width="370" cellspacing="0" cellpadding="0" border="0">
<tr>
<td width="53" rowspan="2" bgcolor="#efefef" valign="top"><a name=HPK> </a><
<td width="1" background="img/joukkueet/titlebg.gif"><img src="img/common/tr
<td width="316" valign="top" background="img/joukkueet/titlebg.gif"><span cl
</tr>
<tr>
<td colspan="2" bgcolor="#efefef">
<table cellspacing="0" cellpadding="0" border="0">
<tr>
<td align="center" width="30"><p class="gray">1</p></td>
<td><p><a href="#" onclick="return popup('pelaaja','pelaajainfo.asp?id=23057
</tr>
<tr>
<td align="center" width="30"><p class="gray">29</p></td>
<td><p><a href="#" onclick="return popup('pelaaja','pelaajainfo.asp?id=1277&
</tr>
<tr>
<td align="center" width="30"><p class="gray">4</p></td>
<td><p><a href="#" onclick="return popup('pelaaja','pelaajainfo.asp?id=19392
</tr>
<tr>
<td align="center" width="30"><p class="gray">6</p></td>
<td><p><a href="#" onclick="return popup('pelaaja','pelaajainfo.asp?id=1109&
```

Kuva 13: Taulukkoalue html-sivun lähdekoodissa.

Html-taulukko (esimerkki kuvassa 13) voi sisältää datan lisäksi mitä tahansa muuta informaatiota tai uusia taulukoita. Suurin osa taulukon html-koodauksesta on tiedonkeruun kannalta epäoleellisia taulukkoa koskevia attribuutteja tai muotoilumääritteitä. Tietovarastoinnin kannalta tavoitteena on suodattaa dokumentin taulukkoalueista vain varsinainen datasisältö. Ohjelmoitavalle datan eristämiseen soveltuvalle proseduurille asetettiin seuraavat tavoitteet:

- yksinkertaisuus
- yleisyys
- siirrettävyys
- vikasietoisuus
- testattavuus.

Yksinkertaisuuden lähtökohtana on se, että html-dokumenttia ei yritetä jäsentää rakenteellisesti, vaan sen sisällöstä yritetään poimia oleellisin osuus minimimäärällä operaatioita. Koska dokumentin taulukoissa on tässä tapauksessa kiinnostavaa vain sen tietosisältö, kaikki soluihin liittyvä muotoilutieto ja metatieto ohitetaan. Data eristetään merkkijono-operaatioilla, joten proseduuri edustaa näin ollen kontekstuaalista lähestymistapaa [Kowalkiewicz et al. 2006].

Yleisyys tarkoittaa tässä yhteydessä sitä, että proseduuria ei ole tehty minkään määritellyn dokumenttijoukon käsittelyyn. Se on sovellettavissa minkä tahansa html-muotoiseen asiakirjan taulukkoalueisiin. Proseduuri ei myöskään ole tarkoitettu käsittelemään ainoastaan tiettyjen html-standardimääritysten mukaisia dokumentteja.

Siirrettävyydessä on pyritty siihen, että proseduuri olisi joko suoraan käytettävissä muissa ohjelmointiympäristöissä tai ainakin tarvittavat muutokset olisivat vähäisiä. Alkuperäinen toteutus tehtiin Excelin VBA-makrokielellä, joka kuitenkin on syntaktisesti sama kuin Visual Basic silloin, kun ei käytetä Excelin oliorakenteita. Proseduurissa hyödynnettiin Visual Basicille ominaista joustavuutta tietotyypeissä, sillä html-dokumentista puretun taulukon alkion arvo on silloin totuusarvotyyppinen *false*, kun taulukon kyseinen alkio on tyhjä tai sitä ei ole. Kaikilla kehitysvälineillä ei sama ole mahdollista, mutta arvon *false* tilalla voi käyttää muuta merkintätapaa

Vikasietoisuuden vaatimus perustuu siihen, että html-dokumentin ei voida olettaa olevan ”well-formed”. Koska selain osaa ohittaa useat dokumentissa olevat virheet näyttäen tiedot jossain muodossa, myös poiminta-algoritmin tulee kyetä jäsentämään kaikki dokumentin taulukkomuotoinen data riippumatta siitä, ovatko kyseeseen tulevien html-komentojen loppumerkinnät loogisesti oikein.

Testattavuus on tavoitteena siksi, että uusia internet-hakuja suunniteltaessa proseduuria on voitava suorittaa osina, jotta voidaan tarkastella sen poimimaa tietoa tiedon edelleen käsittelyä varten. Excel valittiin proseduurin alustaksi ennen kaikkea simuloinnin helpottamiseksi, koska käännettävä kieli on liian raskas poiminta-ajon suunnitteluun oleellisesti kuuluvaan iterointiin. Excel-ympäristössä kohdistimen alla olevan makron voi ajaa suoraan edellyttäen, että se on syntaktisesti virheetön. Tällöin ei tarvitse rakentaa yksittäisen toiminnon testauksen kannalta usein tarpeetonta pääohjelmaa tai käynnistyslomaketta.

Tavoitteiston ulkopuolella proseduuri tuo lisäetuina ainakin poiminnan tehokkuuden ja eräiden Excel-ongelmien (esimerkiksi päivämääräpäättely) poistuminen. Tehokkuus on suora seuraus siitä, että html-dokumentti luetaan muistiin tekstitiedostona ja käsitellään merkkijonona ilman että se jäsennetään ohjelmallisesti sen sisältöä vastaavaksi puurakenteeksi. Tällä on ajankäytöllisesti suuri merkitys jo poiminnan kehitysvaiheessa, jolle on tyypillistä iterointi. Tehokkuuden merkitys varsinaisessa tuotantovaiheessa riippuu sitten käsiteltävän tiedon määrästä ja poiminta-ajojen ajankohdasta.

5.4.1. Taulukkoelementtien käsittelyfunktio

Taulukkodatan jäsennys perustuu koodikatkelmassa 9 esiteltyyn funktioon *GetTableData*, joka palauttaa yksidimensioisena taulukkona html-dokumentin tai sen osan sisältämät taulukkoalueet tai niiden osat. Proseduuri jakaa parametrina saadun html-syötteen merkkijonotaulukoksi komennon alkusymbolin perusteella. Funktio hyödyntää muissakin ohjelmointikielissä nykyisin olevaa *Split*-funktiota, jolla merkkijonosta voidaan erotinmerkkijonon perusteella muodostaa merkkijonotaulukko. Kutsuvassa proseduurissa funktion avulla jäsennetään ensin html-dokumentin taulukkoalueet erilliseen merkkijonotaulukkoon, minkä jälkeen haluttu taulukkoalue saman funktion avulla jäsennetään riveiksi ja rivit edelleen rivin soluiksi.

Funktio on tarkoitettu ensisijaisesti taulukkoelementtien <*table*>, <*tr*>, <*td*> ja <*th*> käsittelyyn, eikä sen toimintaa muilla html-elementeillä ole määritelty. Sitä voidaan kuitenkin käyttää tapauskohtaisesti taulukkoa muistuttavien html-rakenteiden käsittelyyn. Yksi esimerkki yhteensopivista määritteistä on lista (<*li*> ja </*li*>).

```
Function GetTableData(ByVal HTML As String, ByVal Tag As String) As Variant
    - - -
  Position = InStr(1, HTML, "<" & Tag, vbTextCompare)
    - - -
  'Pilkotaan koko HTML halutun tagin alkumerkinnän perusteella
  TableData = Split(Mid(HTML, Position + Len(Tag) + 1, Len(HTML)), "<" _
    & Tag, , vbTextCompare)
   - - -
  'Varmistetaan, että taulukon viimeinen alkio päättyy viimeiseen löytyvään
  'loppumerkintään
  PrevPosition = 0
  'Position on valmiiksi >0 aikaisemman operaation jäljiltä
  While (Position > 0)
    Position = InStr(PrevPosition + 1, TableData(UBound(TableData)), "</" _
      & Tag, vbTextCompare)
    If (Position > 0) Then PrevPosition = Position
  Wend

  'Poistetaan viimeisen loppumerkinnän jälkeinen osuus
```

```
If (PrevPosition > 0) Then
  TableData(UBound(TableData)) = Left(TableData(UBound(TableData)), _
    PrevPosition - 1)
- - -
```

Koodikatkelma 9: Taulukkoelementtien jäsennysfunktio.

Html:n taulukkomääritykseen liittyy suuri määrä muitakin elementtejä [W3C], mutta tässä esityksessä ne ohitetaan. Taulukon otsikko <*th*> käsitellään taulukon rivin tavoin. Koska taulukon tai sen solujen attribuutit ohitetaan, niiden avulla mahdollisesti pääteltävissä olevaa lisäinformaatiota taulukosta ei hyödynnetä tässä toteutuksessa.

Koska on mahdollista, että alkumerkintöjä vastaavia loppumerkintöjä joko puuttuu, on liikaa tai ne ovat väärässä kohdassa html-dokumentissa, funktio huolehtii siitä, että vain viimeinen lohkon lopetussymboleista huomioidaan. Sen ylijäävä osuus poistetaan, mikä estää taulukon ulkopuolisen datan tulemasta tulkituksi taulukon solun arvoksi. Mahdollisia sisäkkäisiä taulukkorakenteita ei huomioida erikseen vaan kaikki taulukkodata jäsentyy ylimmän tason taulukkoalueen kanssa samaan kaksiulotteiseen taulukkomuuttujaan.

Taulukon keskeisistä html-elementeistä taulukkomääre <*table*> on ainoa, jolle loppumerkintä on pakollinen [W3C]. Koska taulukon rivit ja sarakkeet erotetaan proseduurissa aloitussymbolin avulla, loppusymbolin puuttuminen ei aiheuta vakavia jäsennysvirheitä. Loppumerkin puuttuminen ohjaa taulukkoalueen solun kaksiulotteisessa matriisissa joko väärälle riville tai väärään sarakkeeseen.

Aloitussymbolin mukaan aineistoa jäsennettäessä on huomioitava, että siihen liittyy usein muita määritteitä, esimerkiksi ”<*table width*=…”, jolla määritetään taulukon leveys. Tämän vuoksi syötemerkkijonosta etsitään osamerkkijonoja ”<*table* ” (ei ”<*table*>”), ”<*tr* ” tai ”<*td* ”.

5.4.2. Taulukon solun arvon eristäminen

Kun taulukkoaineisto on jäsennelty riveiksi ja sarakkeiksi, koodikatkelman 10 funktiolla *GetCellValue* voidaan etsiä solun koko sisältöä kuvaavasta merkkijonosta solun varsinainen arvo tai arvot. Funktio palauttaa arvon aina merkkijonona jättäen tiedon tarkemman tulkinnan funktiota hyödyntävälle sovellukselle. Silloin, kun solulle ei löydy arvoa, palautetaan *false*.

Koska solun arvo voi sisältää loogisesti useita arvoja, parametrina voidaan antaa erotinmerkki, jolla lähdekoodissa esimerkiksi eri muotoilukoodeilla erotetut solun arvon osat liitetään yhteen. Kutsussa määriteltävä erotinmerkki mahdollistaa sen, että palautuva solun arvo voidaan tarvittaessa pilkkoa pienempiin osiin erottimen avulla.

```
Function GetCellValue(ByVal td As String, ByVal Separator As String) _
  As Variant
  - - -
  'Pilkotaan merkkijono taulukon arvoa edeltävän >-merkin kohdalta
  TblData = Split(td, ">", , vbTextCompare)

  'Käydään kaikki löytyneet taulukon alkiot läpi
  'ensimmäistä lukuun ottamatta (ei voi sisältää arvoa)
  For Counter = LBound(TblData) + 1 To UBound(TblData)
    'Jos alkiotaulukon alkio alkaa <-merkillä, ohitetaan
    '(aloittaa muotoilun tms)
    Position = InStr(1, Trim(TblData(Counter)), "<", vbTextCompare)
    'Leikataan merkkijono seuraavaan <-merkkiin asti
    If (Position >= 1) Then
      TblData(Counter) = Left(TblData(Counter), _
        InStr(1, TblData(Counter), "<", vbTextCompare) - 1)
    End If
    If (Trim(TblData(Counter)) <> vbNullString) Then
      'Kootaan kelvolliset arvot yhdeksi
      If (VarType(CellValue) = vbBoolean) Then
        CellValue = TblData(Counter)
      Else
        CellValue = CellValue & Separator & TblData(Counter)
      End If
    End If
  - - -
```

Koodikatkelma 10: Taulukon solun arvon eristäminen solun sisällöstä.

Funktion toimintaa voidaan havainnollistaa koodikatkelmien 11-15 avulla. Esimerkissä on html-dokumentin (jääkiekon SM-liigan ottelutapahtumat)

taulukon yksittäinen rivi, jossa on kaksi solua, joiden arvot ovat ”55.47” ja ”Joonas Vihko Kampitus 2min”.

```
<tr>
<td width="40" valign="top"><p>55.47</p></td>
<td width="310"><p>HPK <a href="#" onclick="return
popup('pelaaja','pelaajainfo.asp?jid=76&id=37631',400,300);"">Joonas
Vihko</a> Kampitus 2min</p></td>
</tr>
```

Koodikatkelma 11: Taulukon yksittäisen rivin solut.

Solusymbolin ”<*tr* ” mukaan eroteltuna rivi muotoutuu koodikatkelman 12 mukaiseksi. Erotinmerkki ei tule mukaan S*plit*-funktion generoimiin merkkijonoihin.

```
> <td width="40" valign="top"><p>55.47</p></td> <td width="310"><p>HPK <a
href="#" onclick="return
popup('pelaaja','pelaajainfo.asp?jid=76&id=37631',400,300);"">Joonas
Vihko</a> Kampitus 2min</p></td>.
```

Koodikatkelma 12: Taulukon rivi eristettynä taulukosta.

Symbolin <*td* mukaan eroteltuna koodikatkelman 12 rivi jakaantuu koodikatkelman 13 mukaisesti kahdeksi soluksi.

```
(1) width="40" valign="top"><p>55.47</p></td>

(2) width="310"><p>HPK <a href="#" onclick="return popup('pelaaja',
'pelaajainfo.asp?jid=76&id=37631',400,300);"">Joonas Vihko</a> Kampitus
2min</p>
```

Koodikatkelma 13: Taulukon rivin solujen arvot eroteltuina.

Merkin ”>” mukaan eroteltuna koodikatkelman 13 solusta (1) muodostuvat koodikatkelman 14 mukaiset erilliset merkkijonot.

```
(1) width="40" valign="top"
(2) <p
(3) 55.47</p
(4) </td.
```

Koodikatkelma 14: Taulukon rivin solun (1) arvo eristettynä.

Koodikatkelman 14 merkkijono (1) ohitetaan automaattisesti funktiossa, koska ensimmäinen ”>”-merkki päättää merkinnän ”<*td* ” ja sen mahdolliset attribuutit. Merkkijono (2) ohitetaan sillä perusteella, että sen ensimmäiseksi merkiksi jää ”<”, jonka jälkeen muuta arvoa ei voi enää tulla ilman ”>”-merkkiä, koska se siirtäisi arvon seuraavan merkkijonotaulukon alkion sisällöksi.

Merkkijono (3) on kelvollinen solun arvoksi merkkiä ”<” edeltävältä osaltaan. Jos merkki ”<” puuttuu merkkijonosta, koko merkkijonotaulukon alkio liitetään solun arvoon. Merkki voi puuttua esimerkiksi taulukon viimeisessä sarakkeessa, jossa purkualgoritmi on jo poistanut merkinnän </*td*> eikä muotoilukoodeja ole käytetty.

Solun (1) arvoksi saadaan näin merkkijono ”55.47”. Koodikatkelman 13 solusta (2) vastaavasti muodostuvat koodikatkelman 15 mukaiset erilliset merkkijonot.

```
(1) width="310"
(2) <p
(3) HPK <a href="#" onclick="return popup('pelaaja',
'pelaajainfo.asp?jid=76&id=37631',400,300);""
(4) Joonas Vihko</a
(5) Kampitus 2min</p.
```

Koodikatkelma 15: Taulukon rivin solun 2 arvot eristettynä.

Solun (2) arvoksi saadaan merkkijono *”HPK :Joonas Vihko: Kampitus 2min”*, kun funktion *Separator*-parametriksi on kutsuttaessa määritelty kaksoispiste. Esimerkkitapauksessa html-dokumentin tuottava järjestelmä on yhdistänyt taulukon samaan soluun seuran nimen, pelaajan nimen (linkki pelaajaan) sekä rangaistuksen syyn ja kestoajan. Pilkottaessa solun sisältö sopivan erotinmerkin

avulla voidaan luotettavasti poimia siitä haluttu osa, mikä ei tässä tapauksessa olisi mahdollista alkuperäisenä erottimena käytetyn välilyönnin avulla.

5.4.3. Tietojen jäsentäminen kaksiulotteiseksi taulukoksi

Funktioita *GetTableData* ja *GetCellValue* hyödyntämällä voidaan muodostaa yksittäisestä html-dokumentin taulukosta kaksiulotteinen taulukkomuuttuja, jota sitten voidaan käsitellä kutsuvassa sovelluksessa. Koodikatkelman 16 funktio *GetHTMLTable* erottelee dokumentista ensin taulukkoalueet, joista se sitten parametrilla kerrotusta yksittäisestä taulukkoalueesta taulukoi solujen arvot.

Koska samaa html-dokumenttia käsitellään mahdollisesti moneen kertaan, funktio tekee taulukoiden purkamisen vain kerran, ellei *Reload*-parametrilla määrätä toisin (näin on meneteltävä dokumentin vaihtuessa). Taulukoiden lukumäärää voidaan funktiolta kysyä asettamalla *Count*-parametrille arvo t*rue*. Käytännössä ensimmäisellä kutsukerralla voivat parametrit *Count* ja *Reload* olla molemmat *true*, minkä jälkeen myöhemmät kutsut saman dokumentin käsittelyn aikana voivat käyttää molempia arvolla *false* ja lisäksi voidaan html-dokumenttina välittää jatkokäsittelyillä tyhjä merkkijono, koska dokumentti on jo jäsennelty muistissa.

Funktio muokkaa dokumenttia ottaessaan sen käsittelyyn ensimmäisen kerran siten, että se korvaa näkymättömät erikoismerkit välilyönneillä, jolloin niihin ei enää tarvitse varautua taulukointialgoritmissa.

```
Function GetHTMLTable(ByVal HTML As String, ByVal TableIndex As Long, _
  ByVal ReLoad As Boolean, ByVal Header As Boolean, _
  ByVal Count As Boolean, ByVal ValueSeparator As String, _
  ByVal Links As Boolean) As Variant
  - - -
  'Kootaan HTML-taulukon rivit erilliseen taulukkoon
  RowData = GetTableData(TableData(TableIndex), "tr")
  - - -
  If (VarType(RowData) <> vbBoolean) Then
    For RowCounter = LBound(RowData) To UBound(RowData)
      'Haluttaessa taulukon otsikko otetaan mukaan yhdeksi riviksi
      If (Header) Then
        HeaderData = GetTableData(RowData(RowCounter), "th")
        Header = False
```

```
    End If
    'Otsikko käsitellään normaalina taulukon rivinä
    'muutoin taulukoidaan sarakkeiden arvot
    If (Header And (VarType(HeaderData) <> vbBoolean)) Then
      ColumnData = HeaderData
    Else
      ColumnData = GetTableData(RowData(RowCounter), "td")
    End If
    If (VarType(ColumnData) <> vbBoolean) Then
      TableRowCounter = TableRowCounter + 1
      TableColumnCounter = LBound(Table, 2) - 1
      For ColumnCounter = LBound(ColumnData) To UBound(ColumnData)
        'Oletusarvoisesti palautetaan solun arvo
        CellValue = GetCellValue(ColumnData(ColumnCounter), _
          ValueSeparator)
        'Mutta jos löytyy linkki, palautetaan se
        '(mahdollinen solun arvo mukana)
        If Links Then CellValue = GetHRefs(ColumnData(ColumnCounter), _
          "href=", False)(0)
        TableColumnCounter = TableColumnCounter + 1
        'Pidetään kirjaa suurimmasta arvon sisältävästä sarakenumerosta
        If (MaxColumnCounter < TableColumnCounter) Then _
          MaxColumnCounter = TableColumnCounter
        Table(TableRowCounter, TableColumnCounter) = CellValue
- - -
```

Koodikatkelma 16: Taulukkoalueen jäsennysfunktio.

Header-parametrilla voidaan määrätä taulukossa oleva mahdollinen otsikkotieto yhdeksi taulukkomuuttujan riviksi. Jos noudettujen tietojen käsittelyn helpottamiseksi kaikki taulukot yhdistetään peräkkäin, välissä olevat otsikot mahdollistavat tietojen luotettavamman tunnistamisen. Funktiossa kuitenkin oletetaan, että <*th*> voi tulla vain kerran taulukkoa kohti.

Palautettavan kaksiulotteisen taulukon kooksi asetetaan taulukon rivien (<*tr*>-komentojen) lukumäärä, koska kaksiulotteisen taulukon ensimmäistä dimensiota ei voida enää jälkeenpäin ohjelmallisesti muuttaa pienemmäksi mahdollisten tyhjien rivien vuoksi. Sarakemäärä on oletetusarvoisesti 255,

mutta funktio uudelleendimensioi taulukon lopuksi todellista sarakemäärää vastaavaksi.

Funktion viimeinen argumentti mahdollistaa *true*-arvolla sen, että taulukossa mahdollisesti olevat linkit muihin dokumentteihin palautetaan arvojen ohella, jolloin soluun liittyvää lisätietoa voidaan noutaa linkissä määritellystä osoitteesta tai muun linkin avulla pääteltävän avaimen avulla. Linkin mukana palautuu sen mahdollinen teksti, joka on siis samalla taulukon solun arvo. Tällöin se kuitenkin pitää vielä erikseen jäsentää linkistä.

Taulukon rivit tai solut voivat *rowspan*- ja *colspan*–määreiden avulla kattaa useita soluja tai rivejä. Koska nämäkin attribuutit ohitetaan, taulukon rakenne saattaa jäsentyä erilaiseksi kuin miltä se näyttää selaimella. Esimerkiksi koodikatkelman 17 taulukko jäsentyy selaimeen kuvassa 14 vasemmalla esitetyllä tavalla, mutta taulukkomuuttujaan kuvassa oikealla esitetyllä tavalla.

```
<TABLE>
<TR><TD>1<TD rowspan="2">2<TD>3
<TR><TD>4<TD>6
<TR><TD>7<TD>8<TD>9
</TABLE>
```

Koodikatkelma 17: Taulukon rivin solun 2 arvot eristettynä.

1	2	3
4		6
7	8	9

1	2	3
4	6	FALSE
7	8	9

Kuva 14: Koodikatkelman 15 taulukon jäsentyminen selaimessa (vasemmalla) ja *GetHTMLTable*-metodilla (oikealla).

5.5. Tietojen edelleenjäsennys

Jäsennettyjen taulukkoalueiden ympärille voidaan rakentaa erilaisia jatkokäsittelyfunktioita, joilla voidaan käsitellä muistiin luettuja html-taulukoita. Useinkaan taulukoitu data ei sellaisenaan vielä ole kelvollista tietovarastoon vietäväksi. Liitteen 3 esimerkkisovelluksessa on esimerkkejä funktioista, joiden avulla esitaulukoidusta datasta voidaan noutaa tietoja.

Funktio *ArrayLookup* etsii kaksiulotteisen taulukon annetusta sarakkeesta haluttua merkkijonoa. Poimittava tieto ei aina esiinny suoraan taulukkoalueen tietyllä rivillä vaan sen sijainti voi riippua edellä olevasta otsikkotiedosta tai muusta selitteestä. Esimerkiksi pörssikursseja ladattaessa (esimerkki kuvassa 15) funktiolla voidaan etsiä taulukosta ensin merkkijonoa ”Nimi”, minkä jälkeen vasta sarakkeen seuraavia arvoja luettaessa voidaan olla varmoja, että tieto tarkoittaa yrityksen nimeä. Funktion avulla voi myös hakea kuvan taulukkoalueelta yksittäisen yrityksen sijaintirivin sen nimellä.

	A	B	C	D	
23	21	Nimi	Viim.	Muutos	Val.
24	21	Neste Oil	19.14	-2.45%	EUR
25	21	EPÄTOSI			
26	22	EPÄTOSI	Perusteollisuus	EPÄTOSI	
27	22	EPÄTOSI			
28	22	Nimi	Viim.	Muutos	Val.
29	22	Ahlstrom	16.15	-0.80%	EUR
30	22	Exel	10.50	-4.02%	EUR
31	22	Huhtamäki	7.13	+6.10%	EUR
32	22	Kemira	9.00	+7.14%	EUR
33	22	Kemira GrowHow (T)	12.15	+0.16%	EUR

Kuva 15: Esimerkki pörssikurssien hakutuloksesta.

Funktio *ArrayColumnOffset* etsii kaksiulotteisen taulukon annetusta rivistä ja annetusta sarakkeesta halutun määrän allekkain olevia solujen arvoja. Yhtenä argumenttina on palautettavien solujen määrä ja lisäksi funktio voidaan määrätä lopettamaan tietojen keruu sarakkeesta, mikäli vastaan tulee solu ilman arvoa. Tällöin on tiedossa, että kaikki haluttu data on taulukon peräkkäisillä riveillä.

6. Esimerkkejä datan poiminnasta internetistä

Koska palloilulajien tietovaraston kantaratkaisu on tätä kirjoitettaessa avoinna eikä kantaa siksi ole vielä olemassa lopullisessa muodossaan, tiedot haetaan kausikohtaisesti Excel-työkirjoihin, joiden rakenne kuitenkin vastaa mahdollisimman pitkälle tulevaa tietovarastoa. Microsoft Excel on sopiva alusta erityisesti tietokantoja koostettaessa, sillä sovelluksen toiminnoilla on helppo manipuloida dataa manuaalisesti ennen lopullista siirtoa varsinaiseen relaatiotietokantaan. Lisäksi työkirjojen monisivuisuus mahdollistaa relaatioiden suunnittelun valmiiksi taulujen ja kentästöjen osalta. Lopulliseen tietokantaan siirryttäessä joudutaan perusasioista määrittämään uudelleen mahdollisesti vain kenttien tietotyypit ja avainnukset.

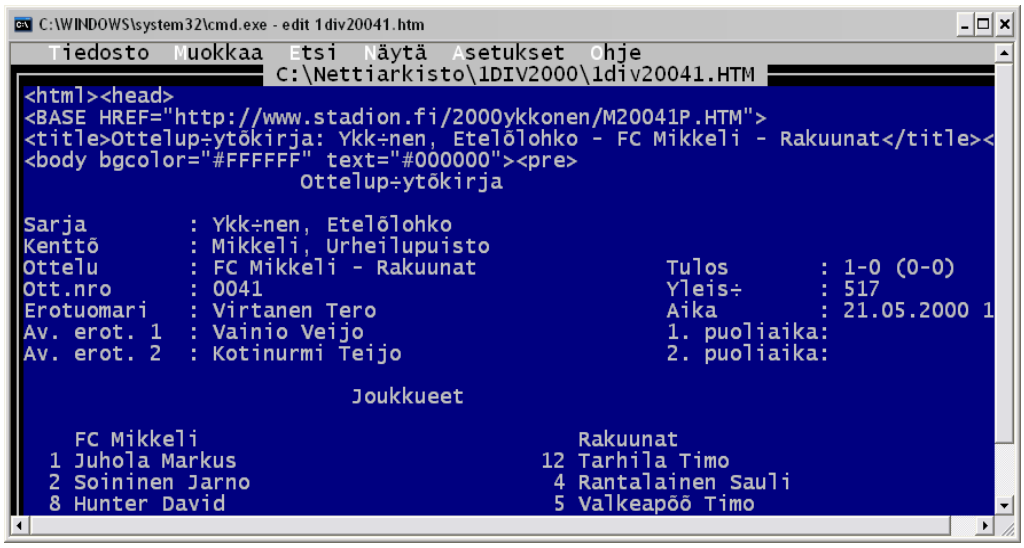
Jokaista tietoalkiota ei esimerkeissä pyritä muokkaamaan lopulliseen muotoonsa vaan tavoitteena on jäsentää html-sivuston sisältämä data sellaiseen tilaan, jossa sitä voidaan jalostaa edelleen minkä tahansa sovelluskehitysvälineen perustoiminnoilla.

6.1. Esimerkki Excelin web-kyselyn käytöstä

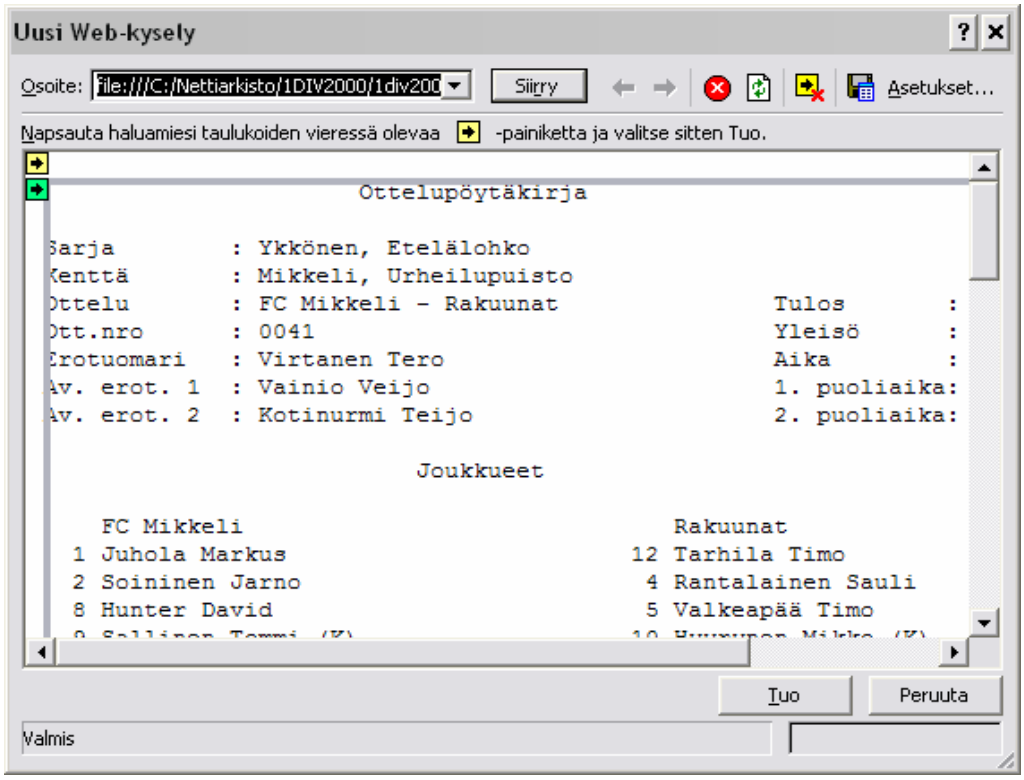
Seuraava esimerkki havainnollistaa web-kyselyn toimintaa sekä <*table*> että <*pre*>-määritteiden kanssa. Samalla esimerkkiaineisto osoittaa myös sen, ettei voida olla varmoja, että sivusto esittää kaikki loogisesti yhteenkuuluvat tiedot yhdenmukaisina edes tietyllä hetkellä.

Esimerkissä avataan web-arkistosta (http://www.archive.org) levylle aikaisemmin tallennettuja jalkapallon yksittäisen sarjan vanhoja ottelutilastoja. Osa ottelutilastoista sisältää <*pre*>-määreen, mutta osassa on käytetty <*table*>-määrityksiä. Yksittäisten sivujen erilainen rakenne selittyy sillä, että ne on tallennettu lähdejärjestelmään staattisina html-sivuina sen sijaan, että ne generoituisivat kyselylomakkeen kautta.

Kuvissa 16 ja 17 on sama kohdetiedosto avattuna sekä tekstieditorissa että Excelin web-kyselyssä.



Kuva 16: Avattava tiedosto tekstieditorissa.



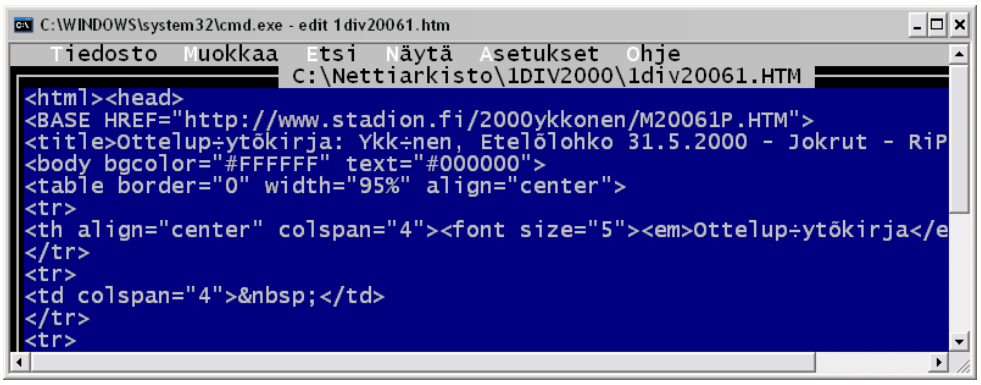
Kuva 17: Avattava tiedosto Excelin web-kyselyssä.

Web-kysely tunnistaa <*pre*>-määritellyn alueen taulukkomuotoiseksi. Asetuksilla *WebPreFormattedTextToColumns* ja *WebConsecutiveDelimitersAsOne* tiedosto avautuu Exceliin kuvan 18 mukaisesti.

	A	B	C	D	E	F
1	Ottelupöytäkirja					
2						
3	Sarja	:	Ykkönen, Etelälohko			
4	Kenttä	:	Mikkeli, Urheilupuisto			
5	Ottelu	:	FC Mikkeli - Rakuunat	Tulos :	1-0 (0-0)	
6	Ott.nro	:	41	Yleisö :	517	
7	Erotuomari	:	Virtanen Tero	Aika :	21.5.2000	18:30
8	Av. erot. 1	:	Vainio Veijo	1. puoliaika:		
9	Av. erot. 2	:	Kotinurmi Teijo	2. puoliaika:		
10						
11	Joukkueet					
12						
13		FC Mikkeli		Rakuunat		
14	1	Juhola Markus	12	Tarhila Timo		
15	2	Soininen Jarno	4	Rantalainen Sauli		
16	8	Hunter David	5	Valkeapää Timo		
17	9	Sallinen Tommi (K)	10	Hyyrynen Mikko (K)		
18	10	Niiranen Marko	11	Hovi Pasi		
19	13	Sariola Tommi	14	Pirhonen Jere		
20	14	Nuija Pasi	15	Kotilainen Mikko		
21	16	Lahtonen Janne	16	Kallio Markku		

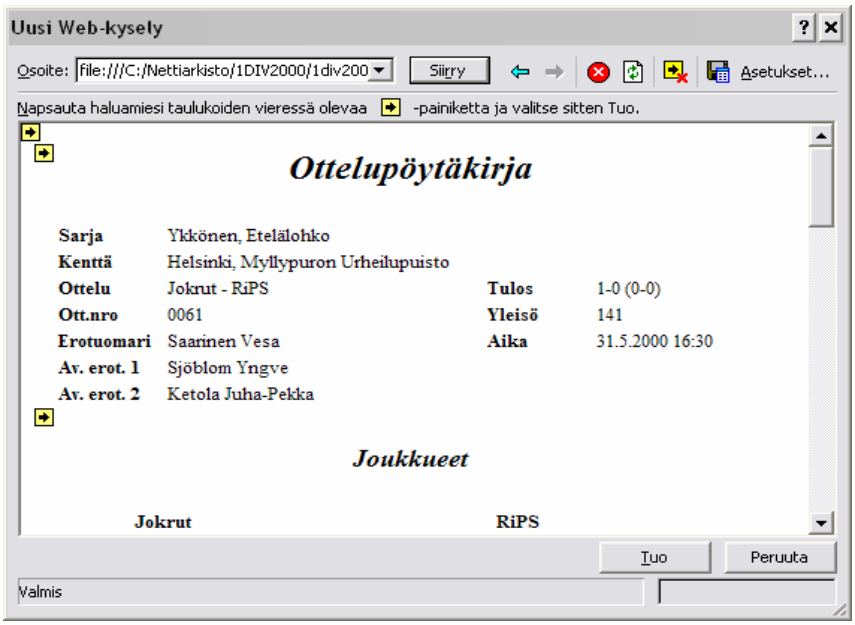
Kuva 18: Web-kyselyllä ladattu tiedosto Excel-taulukossa

Ajallisesti vain 10 päivää myöhemmin tuotettu toisen ottelun tilastodata on järjestelmämuutoksista johtuen tallennettu jo erilaisessa muodossa kuten kuva 19 osoittaa.



Kuva 19: Web-kyselyllä ladattu tiedosto tekstieditorissa.

Nyt web-kysely jäsentää sivun taulukkoalueet kuvan 20 mukaisesti. Ilman erillisiä aluevalintoja haku tuo Exceliin html-sivun kaikki taulukot. Taulukko jäsentyy Exceliin sarakkeiston osalta (kuva 21) hieman eri tavalla kuin kuvien 16 ja 17 taulukko, mutta erot ovat kuitenkin niin vähäisiä, että alkukauden ja loppukauden otteluiden tiedot voidaan poimia molemmissa tapauksissa samaa proseduuria hieman muunnellen.



Kuva 20: Web-kyselyn käynnistysruutu.

A1 Ottelupöytäkirja

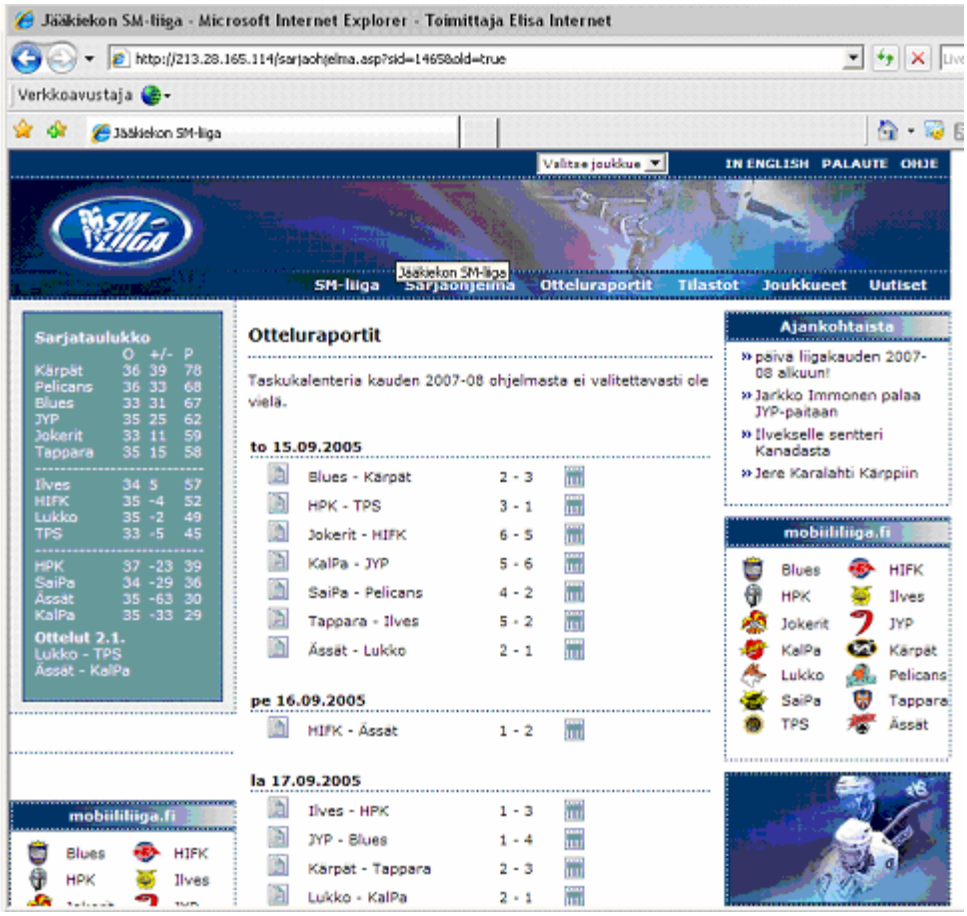
	A	B	C	D	E
1	Ottelupöytäkirja				
2					
3	Sarja	Ykkönen, Etelälohko			
4	Kenttä	Helsinki, Myllypuron Urheilupuisto			
5	Ottelu	Jokrut - RiPS	Tulos	1-0 (0-0)	
6	Ott.nro	61	Yleisö	141	
7	Erotuomari	Saarinen Vesa	Aika	31.5.2000 16:30	
8	Av. erot. 1	Sjöblom Yngve			
9	Av. erot. 2	Ketola Juha-Pekka			
10					
11	Joukkueet				
12					
13		Jokrut			RiPS
14	1	Laitila Tommi		1	Peltonen Tuomas
15	5	Latvanen Antti		2	Leinonen Mikko
16	6	Kuusela Ari		3	Kinnaslampi Jussi
17	8	Heilala Roope		9	Huuhtanen Peke
18	9	Karhu Tero (K)		11	Onyekaba Chika
19	10	Hiukka Matti		14	Odigbo Isaac

Kuva 21: Web-kyselyn jäsentämä kuvan 20 tiedosto Excel-taulukossa.

6.2. Jääkiekon SM-liigan aikaisempien kausien tilastot

Jääkiekon SM-liigan internet-sivuilla (http://www.sm-liiga.fi) ovat nähtävissä (tätä kirjoitettaessa) ainoastaan meneillään olevan pelikauden (2007-2008) tilastot. Ottelut-alisivulla on valintalistassa valittavissa aikaisempiakin kausia, mutta niiden tietoja ei ole päivitetty tietokantaan. Aikaisempien kausien ottelutiedot vuodesta 1999 lähtien ovat kuitenkin toistaiseksi olleet nähtävillä osoitteessa http://213.28.165.114/sarjaohjelma.asp?sid=1465&old=true.

Sivuston rakenne noudattaa palloilutilastojen web-esitystavaksi vakiintunutta käytäntöä eli tilasto-osion pääsivulla (kuva 22) on lista, josta valitaan ottelu, jota halutaan tarkastella lähemmin. Yksittäistä ottelua klikkaamalla voidaan sitten porautua yksittäisiin ottelutapahtumiin.



Kuva 22: Esimerkki jääkiekon SM-liigan tulostiedoista.

6.2.1. Sivujen tallennus paikallisesti

Sivujen paikallinen tallennus ennen varsinaisen datan jäsennystä korostui tietojen poiminnassa, sillä tiedossa oli sivujen poistumisen uhka. Talletusta varten rakennettiin liitteessä 3 kokonaisuudessaan esitetty proseduurisarja, jossa aliohjelma *VanhatOttelut* vastaa yksittäisen kauden talletuksesta saaden argumenttinaan urlissa olevan sarjan tunnisteen ja kauden. Kausi annetaan muodossa ”2006-2007” ja sitä tarvitaan ainoastaan nimettäessä yksittäisen ottelun tietoja, jotta prosessin tuloksena syntyvät tiedostot voidaan erotella kausittain.

Proseduurissa ladataan ensin kunkin kauden ottelulista, josta sen jälkeen erotellaan otteluihin viittaavat linkit. Ottelutapahtumien url ei kuitenkaan suoraan sisälly linkkiin, koska ottelua klikattaessa kutsutaan funktiota, jolle välitetään ottelun numeerinen tunniste. Selaimen avulla on kuitenkin selvitettävissä ottelutapahtumien todelliset url-osoitteet. Ottelun tapahtumiin viittaava linkki on koodikatkelman 18 mukainen, mutta otteluraporttia klikattaessa avautuu osoite http://213.28.165.114/sarjaohjelma_historia.asp?id=33613.

```
href="#" onclick="return
popup('ohjelma','sarjaohjelma_historia.asp?id=33613',400,400);">Tappara -
SaiPa
```

Koodikatkelma 18: Ottelutapahtumiin viittaava linkki.

Tallennettaessa on tärkeätä säilyttää ottelulistan ja ottelutapahtumien välinen linkki ristiriidattomana. Tässä esimerkissä se on toteutettu nimeämällä tapahtuma- ja kokoonpanosivut tallennuksen yhteydessä siten, että ottelun alkuperäinen tunniste sisältyy tiedostonimeen. Toinen vaihtoehto asian toteuttamiseksi olisi ollut esimerkiksi nimetä yksittäinen tapahtumatiedosto sisällön perusteella niin, että se viittaa yksikäsitteisesti sisältämäänsä otteluun. Koska joukkueet kohtaavat jääkiekon SM-liigassa kauden aikana useita kertoja, nimessä olisi oltava tavalla tai toisella esillä sekä joukkueet että ottelupäivä. Kolmas vaihtoehto olisi tallentaa alisivut mielivaltaisilla nimillä, mutta tällöin pitäisi pääsivulla olevia otteluiden tunnisteita manipuloida tietoja talletettaessa siten, että ne viittaisivat oikeisiin alisivuihin. Poiminnassa ilmenevien

ongelmien tai myöhempien tarkistusten tai täydennysten varalta on kuitenkin perusteltua tallentaa html-sivut sellaisenaan alkuperäisessä muodossaan.

Poiminta tapahtuu kutsumalla proseduuria *VanhatOttelut* (koodikatkelma 19) sarjatunnisteen ja kausiselitteen avulla. Kausien ns. play-off –ottelut on lähdejärjestelmässä merkitty eri sarjoiksi.

```
VanhatOttelut "1483", "2006-2007_runkosarja"
VanhatOttelut "1484", "2006-2007_playoffs"

VanhatOttelut "1465", "2005-2006_runkosarja"
VanhatOttelut "1466", "2005-2006_playoffs"

VanhatOttelut "1303", "2004-2005_runkosarja"
VanhatOttelut "1304", "2004-2005_playoffs"
```

Koodikatkelma 19: Otteluiden tallennus kausittain.

6.2.2. Datan eristäminen

Datan jäsentäminen html-sivuista mukailee etenemisjärjestyksen osalta sivuston tallennusta. Kultakin ottelulistan sisältävältä pääsivulta poimitaan ensin haluttu informaatio ja sen jälkeen käydään läpi ottelulistan mukaisten yksittäisten otteluiden data. Ottelulistan voisi käsitellä myös ilman pääsivun tietoja, sillä suurin osa ottelun perustiedoista sisältyy myös alisivuihin ottelupäivää lukuun ottamatta.

Jääkiekkoa koskevissa sivustototeutuksissa porautuminen alimmalle tietotasolle edellyttää usein kahden eri linkin seuraamista, sillä tilankäyttösyistä joukkueiden kokoonpanot ja varsinaiset pelitapahtumat avautuvat eri linkistä. Jalkapallon vastaavilla sivustoilla kokoonpanot ja tapahtumat ovat useissa toteutuksissa samalla alisivulla, koska sekä pelaajia että tapahtumia on vähemmän.

Kuvan 22 esimerkkisivu on kuvassa 23 jäsennetty kaksiulotteiseksi taulukoksi Exceliin siten, että html-sivun taulukkoalueen soluista tuodaan vain arvot. Kuvassa 24 sama sivu on purettu niin, että käytetty *GetHTMLTable*-proseduuri palauttaa myös soluissa olevat linkit. Excel-esimerkeissä sarakkeessa A oleva luku tarkoittaa taulukkoalueen eli <*table*>…</*table*> -lohkon juokseva järjestysnumeroa 0…n.

	A	B	C	D	
49	19	to 15.09.2005			
50	19	EPÄTOSI			
51	20	EPÄTOSI	Blues - Kärpät	2 - 3	EP
52	20	EPÄTOSI	HPK - TPS	3 - 1	EP
53	20	EPÄTOSI	Jokerit - HIFK	6 - 5	EP
54	20	EPÄTOSI	KalPa - JYP	5 - 6	EP
55	20	EPÄTOSI	SaiPa - Pelicans	4 - 2	EP
56	20	EPÄTOSI	Tappara - Ilves	5 - 2	EP
57	20	EPÄTOSI	Ässät - Lukko	2 - 1	EP
58	20	pe 16.09.2005			
59	20	EPÄTOSI			
60	21	EPÄTOSI	HIFK - Ässät	1 - 2	EP
61	21	la 17.09.2005			
62	21	EPÄTOSI			
63	22	EPÄTOSI	Ilves - HPK	1 - 3	EP
64	22	EPÄTOSI	JYP - Blues	1 - 4	EP
65	22	EPÄTOSI	Kärpät - Tappara	2 - 3	EP
66	22	EPÄTOSI	Lukko - KalPa	2 - 1	EP
67	22	EPÄTOSI	Pelicans - Jokerit	3 - 1	EP
68	22	EPÄTOSI	TPS - SaiPa	2 - 3	EP
69	22	ti 20.09.2005			
70	22	EPÄTOSI			
71	23	EPÄTOSI	Blues - Ilves	1 - 0	EP
72	23	EPÄTOSI	HPK - Lukko	5 - 1	EP
73	23	EPÄTOSI	Jokerit - JYP	0 - 3	EP
74	23	EPÄTOSI	KalPa - Kärpät	1 - 9	EP

Kuva 23: Kauden 2005-2006 otteluiden lista Exceliin jäsennettynä.

	A	B	C
49	19	to 15.09.2005	
50	19	EPÄTOSI	
51	20	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33572',400,400);">Blues - Kärpät
52	20	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33573',400,400);">HPK - TPS<
53	20	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33574',400,400);">Jokerit - HIFK<
54	20	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33575',400,400);">KalPa - JYP<
55	20	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33576',400,400);">SaiPa - Pelica
56	20	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33577',400,400);">Tappara - Ilves
57	20	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33578',400,400);">Ässät - Lukko
58	20	pe 16.09.2005	
59	20	EPÄTOSI	
60	21	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33579',400,400);">HIFK - Ässät<
61	21	la 17.09.2005	
62	21	EPÄTOSI	
63	22	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33580',400,400);">Ilves - HPK<
64	22	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33581',400,400);">JYP - Blues<
65	22	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33582',400,400);">Kärpät - Tappa
66	22	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33583',400,400);">Lukko - KalPa
67	22	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33584',400,400);">Pelicans - Jok
68	22	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33585',400,400);">TPS - SaiPa<
69	22	ti 20.09.2005	
70	22	EPÄTOSI	
71	23	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33586',400,400);">Blues - Ilves<
72	23	href="#" onclick="return popup('ohjelma','sarjaohje	href="#" onclick="return popup('ohjelma','sarjaohjelma_historia.asp?id=33587',400,400);">HPK - Lukko<

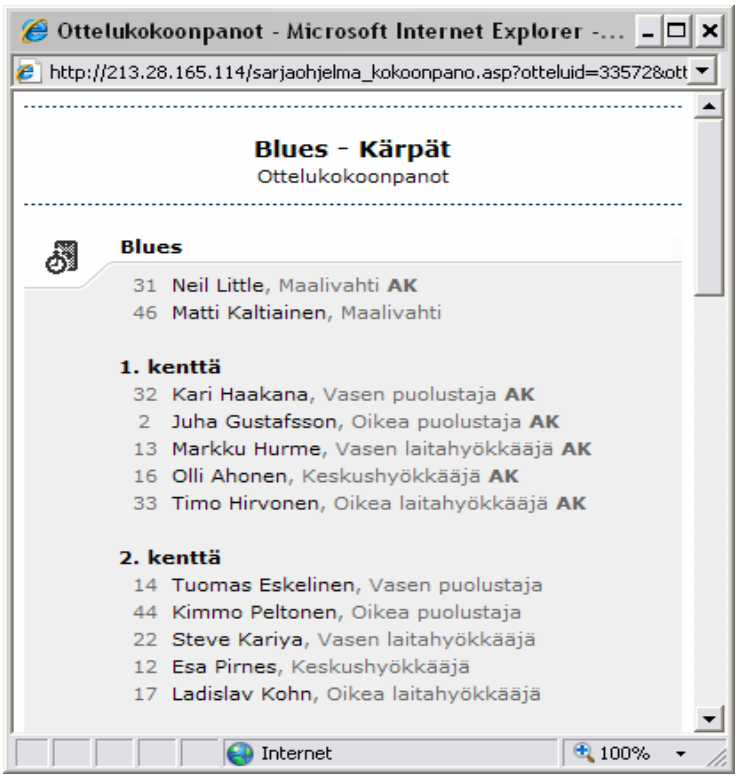
Kuva 24: Kauden 2005-2006 otteluiden lista linkeittäin Exceliin jäsennettynä.

Kuvan 22 ottelulistassa yksittäisen ottelun molemmin puolin olevia kuvakkeita klikkaamalla voi porautua ottelun yksityiskohtiin. Kuvissa 25 ja 26 on valitun ottelun pelitapahtumat selainesityksenä ja *GetHTMLTable*-metodilla jäsennettynä taulukkoesityksenä. Kuvissa 27 ja 28 on vastaavasti saman ottelun kokoonpanotiedot molemmilla esitystavoilla.

Kuva 25: Ottelulistasta valitun ottelun tapahtumat selainesityksenä.

	A	B	C	D	E
1	0	EPÄTOSI			
2	0	EPÄTOSI	Blues - Kärpät 2-3:(0-0,1-1,1-1,0-0,0-	EPÄTOSI	
3	0	EPÄTOSI			
4	0	1. Erä			
5	0	EPÄTOSI			
6	0	EPÄTOSI	EPÄTOSI	EPÄTOSI	
7	0	EPÄTOSI			
8	0	Rangaistukset			
9	0	EPÄTOSI			
10	1	0.24	Blues :Markku Hurme: Kampitus 2min		
11	1	0.30	Kärpät :Lasse Kukkonen: Estäminen 2min		
12	1	5.44	Blues :Tuomas Eskelinen: Koukkaaminen 2min		
13	1	8.25	Kärpät :Ari Vallin: Kampitus 2min		
14	1	17.28	Kärpät :Lasse Kukkonen: Väkivaltaisuus 2min		
15	1	EPÄTOSI	EPÄTOSI		
16	1	EPÄTOSI			
17	1	2. Erä			
18	1	EPÄTOSI			
19	1	EPÄTOSI	EPÄTOSI	EPÄTOSI	
20	1	EPÄTOSI			
21	1	Maalit			
22	1	EPÄTOSI			
23	2	31.39	Esa Pirnes: 1-0 (:Ladislav Kohn:) YV		
24	2	32.56	Juhamatti Aaltonen: 1-1 (:Lasse Kukkonen:, :Jari Viuhkola:)		
25	2	EPÄTOSI	EPÄTOSI		
26	2	EPÄTOSI			

Kuva 26: Ottelulistasta valitun ottelun tapahtumat taulukkoesityksenä.



Kuva 27: Ottelulistasta valitun ottelun kokoonpanot selainesityksenä.

	A	B	C	D
1	0	EPÄTOSI		
2	0	Ottelukokoonpanot		
3	0	EPÄTOSI		
4	1	EPÄTOSI	EPÄTOSI	Blues
5	2	31	Neil Little:, Maalivahti: AK	
6	2	46	Matti Kaltiainen:, Maalivahti	
7	2	1. kenttä		
8	2	32	Kari Haakana:, Vasen puolustaja: AK	
9	2	2	Juha Gustafsson:, Oikea puolustaja: AK	
10	2	13	Markku Hurme:, Vasen laitahyökkääjä: AK	
11	2	16	Olli Ahonen:, Keskushyökkääjä: AK	
12	2	33	Timo Hirvonen:, Oikea laitahyökkääjä: AK	
13	2	2. kenttä		
14	2	14	Tuomas Eskelinen:, Vasen puolustaja	
15	2	44	Kimmo Peltonen:, Oikea puolustaja	
16	2	22	Steve Kariya:, Vasen laitahyökkääjä	
17	2	12	Esa Pirnes:, Keskushyökkääjä	
18	2	17	Ladislav Kohn:, Oikea laitahyökkääjä	
19	2	3. kenttä		
20	2	42	Arto Laatikainen:, Vasen puolustaja	
21	2	21	Tero Määttä:, Oikea puolustaja	
22	2	78	Kari Kalto:, Vasen laitahyökkääjä	

Kuva 28: Ottelulistasta valitun ottelun kokoonpanot taulukkoesityksenä.

Poiminnassa käytetään koodikatkelman 20 mukaisia tietorakenteita. Rakenne *Ottelu* vastaa tietovaraston tauluun *Matches* määriteltyjä kenttiä ja

rakenne *Tapahtuma* tauluun *Match_details* määriteltyjä kenttiä. Taulujen kentästöstä on tietorakenteisiin kuitenkin otettu mukaan vain ne, jotka on mahdollista poimia internetistä. Osa kentistä voidaan tietovarastossa joko jättää tyhjiksi tai ne voidaan täyttää ohjelmallisesti. Tällaisia kenttiä ovat esimerkiksi ottelun pelipaikka tai valmentaja, jotka oletusarvoisesti ovat koko kauden ajan joukkueittain samansisältöiset.

```
Type Ottelu
  Match_id As String
  Competition_id As String
  Division_id As String
  Season As String
  Match_date As Variant
  Home_id  As String
  Away_id As String
  Home_score As Long
  Away_score As Long
   - - -

Type Tapahtuma
  Match_id As String
  Detail_type_id As String
  Detail_subtype_id As String
  Team_id As String
  Person_id As String
  - - -
```

Koodikatkelma 20: Otteluiden ja ottelutapahtumien tietorakenteet.

Sovelluksen runko on samankaltainen kaikissa vastaavanmuotoisten sivustojen poiminnoissa, mutta yksittäisten tietokenttien jäsennystä joudutaan käytännössä muuttamaan jokaisessa uudessa poiminnassa.

Keruusovellukselle välitettävät argumentit ovat tässä tapauksessa yleisiä tietoja, jotka ovat kaikille *Matches*-kannan riveille samat eli paikallisesti tallennettujen sivujen sijaintikansio, pelikausi, sarjan nimi ja kilpailun nimi.

Ottelun tunnisteeksi generoidaan ”SM” + kauden päättymisvuosi + viisinumeroinen juokseva numero (otteluita ei kauden aikana ole kuin

korkeintaan satoja) siksi, että näin varmistetaan, ettei numerointi voi mennä päällekkäin toisen kauden tai toisen sarjan tietojen kanssa dataa lopulliseen tietovarastokantaan siirrettäessä. Relaatiosta johtuen ottelunumeroinnin muuttaminen poiminta-ajon jälkeen on työläs ja virhealtis prosessi.

Tässä tapauksessa rikotaan tarkoituksellisesti yhtä tietovarastoinnin sääntöä, sillä surrogaatin tulisi Kimballin ja muiden [1998] mukaan olla merkityksetön eikä se saisi sisältää upotettuja viittauksia dataan. Koska tietoja ei voida välittömästi siirtää varsinaiseen tietovarastoon, ei ole mahdollista käyttää esimerkiksi juoksevaa numerointia.

Koska ottelulistan osalta ei voida olla varmoja, mistä taulukkoalueesta halutut tiedot alkavat ja mihin ne päättyvät, proseduuri käy läpi kaikki alueet ja virtuaalisesti ne muodostavat yhden taulukon. Toiminnossa voitaisiin käyttää Excelin omaa web-kyselyä, jolloin Excel-taulukkoon voitaisiin tuoda koko sivun sisältö kerralla, jolloin samalla periaatteella Excel-taulukon sisältö käytäisiin läpi proseduurissa.

Proseduuri ohittaa taulukoiden ne rivit, jotka eivät ole ottelutietoja. Ottelu voidaan luotettavasti päätellä tässä tapauksessa sen perusteella, että sekä ottelussa (vastakkain olevien joukkueiden nimet) että tuloksessa (vastakkain olevien joukkueiden maaliluvut) esiintyy väliviiva, esimerkiksi ”Tappara – Lukko” tai ”7-4”.

Ottelulista ladataan taulukkorakenteeseen linkkeineen, sillä se mahdollistaa taulukon soluun otteluparin lisäksi sisältyvän tunnistenumeron löytämisen solun sisältämästä linkistä. Jokaisen otteluparin kohdalla määritetään tunnistenumeron avulla kokoonpanot ja ottelutapahtumat sisältävät tiedostot, jotka on tallennettu aikaisemmin siten, että tiedostonimessä on säilytetty kunkin ottelun alkuperäinen tunnistenumero.

Kun yhden ottelun perustiedot ja tapahtumat on koottu tietorakenteisiin, ne tallennetaan tietovarastokantaa simuloivan Excel-työkirjan sivuille. Ohjelmallisesti havaituista virheistä kootaan informaatiota kannassa tätä varten olevaan kenttään.

Poiminnassa tarkistettavat asiat riippuvat käsiteltävästä palloilulajista. Jääkiekon osalta voidaan tarkistaa ainakin seuraavat asiat:

- maalivahtien määrä ottelupöytäkirjassa on kaksi
- pelaajia kentällistä kohti on normaalisti viisi
- erien yhteenlaskettu maalimäärä vastaa ottelun tulosta
- tapahtumissa on maalitapahtumia ottelun tulosta vastaava määrä.

Mahdollinen lisätarkistus, jota tässä prosessissa ei ole huomioitu, olisi esimerkiksi se, ettei tapahtumissa ei saa esiintyä pelaajaa, joka ei ole kokoonpanossa. Lisäksi myös pelaajien tehopisteitä tai jäähyjä kerätyissä ottelutapahtumissa olisi mahdollista verrata tapahtumasivun lopussa valmiina oleviin yhteenvetotietoihin.

Tietovaraston lopullisen latauksen yhteydessä tulee vielä tarkistaa esimerkiksi se, kuuluuko pelaaja, valmentaja tai erotuomari henkilörekisteriin. Edelleen on tarkistettava, kuuluuko seura seurarekisteriin. Jääkiekossa ei kirjata ottelun aikana suoritettuja pelaajavaihtoja, mutta jalkapallotietoja kerätessä tulisi latauksen yhteydessä tarkistaa, että pelaajavaihtojen määrä ottelussa on sääntöjen mukainen.

Proseduuri *HaeOtteluTiedot* poimii html-datasta kunkin ottelun kokoonpanot ja tapahtumat *Match_details*-kantaan. Tapahtumia ottelua kohti on kymmenittäin, sillä joukkueiden kokoonpanotkin käsitellään tapahtumina, mikä mahdollistaa tietovaraston tehokkaat ja yksinkertaiset kyselyt. Excel-työkirjan rivimäärärajoite (65535 riviä aina versioon 2003 asti) aiheuttaa sen, että käytännössä vain yksi kausi kerrallaan voidaan poimia yhteen työkirjaan.

Ottelutiedoista käsitellään ensin kokoonpanotiedot, joiden on edeltävässä sivustokartoituksessa todettu löytyvän taulukkoalueista 2 ja 4. Funktion *ArrayColumnOffset* avulla poimitaan halutut tiedot taulukkomuuttujan sarakkeesta 2. Maalivahtien tiedetään olevan heti taulukkoalueen alussa. Sen sijaan kenttäpelaajat on etsittävä kutakin kentällistä edeltävän otsikon (järjestysnumero ja merkkijono ”kentällinen”) avulla.

Poiminta-ajo käynnistetään esimerkkisovelluksessa koodikatkelman 21 mukaisesti.

```
Sub PoimiOttelut()
- - -
HaeOttelut "C:\Hockey_1999-2007\Matches\", Kausi, "Runkosarja", "SM-liiga"
```

Koodikatkelma 21: Datan jäsennysajon käynnistys.

Prosessin lopputuloksena syntyvät tietovarastokannan taulut *Matches* ja *Match_Details* on esitetty kuvissa 29 ja 30.

	A	F	H	I	J	K	L	M	N	O	R	T
1	Match_id	Match_date	Home_id	Away_id	Home_score	Away_score	Periods	Extra_time	Penalties	Total	Referee1	Other referees
2	SM200700001	14.9.2006	HIFK	Jokerit	3	5	1-1,1-2,1-2			60.00	Levonen Jari	Kekäläinen Mikko,Terho Jussi
3	SM200700002	14.9.2006	Ilves	Tappara	2	0	1-0,0-0,1-0			60.00	Kruus Markku	Lajunen Antti,Tervo Mika
4	SM200700003	14.9.2006	JYP	KalPa	2	1	0-0,1-1,0-0,0-0,1-0	*	*	65.00	Favorin Timo	Myllynen Tuomas,Nieminen Pasi
5	SM200700004	14.9.2006	Kärpät	HPK	5	1	1-0,1-0,3-1			60.00	Rantala Aleksi	Isometsä Janne,Kautto Juha
6	SM200700005	14.9.2006	Lukko	Ässät	3	5	1-2,1-2,1-1			60.00	Rönn Jyri	Hämäläinen Antti,Väisänen Kare
7	SM200700006	14.9.2006	Pelicans	Blues	3	6	0-0,2-4,1-2			60.00	Ringbom Reijo	Lindroos Seppo,Salminen Ilkka
8	SM200700007	14.9.2006	TPS	SaiPa	7	3	1-1,4-1,2-1			60.00	Partanen Sami	Fonselius Stefan,Suoraniemi Juha
9	SM200700008	15.9.2006	Jokerit	Kärpät	4	2	2-0,1-2,1-0			60.00	Henriksson Hannu	Heikura Pekka,Lajunen Antti
10	SM200700009	16.9.2006	Blues	TPS	3	0	1-0,2-0,0-0			60.00	Salminen Teemu	Terho Jussi,Viikari Veikko
11	SM200700010	16.9.2006	HPK	JYP	3	1	1-0,0-1,2-0			60.00	Laaksonen Tom	Levänen Mika,Nieminen Pasi
12	SM200700011	16.9.2006	KalPa	Ilves	2	1	0-1,2-0,0-0			60.00	Savolainen Tatu	Hämäläinen Antti,Juntunen Antti
13	SM200700012	16.9.2006	SaiPa	Lukko	7	6	2-2,2-3,2-1,0-0,1-0	*	*	65.00	Vuohtoniemi Ville	Kekäläinen Mikko,Tervo Mika
14	SM200700013	16.9.2006	Tappara	HIFK	4	6	0-4,3-0,1-2			60.00	Rönn Jyri	Peltola Mika,Väisänen Kare
15	SM200700014	16.9.2006	Ässät	Pelicans	1	2	0-2,1-0,0-0			60.00	Sinkkonen Mika	Brännare Aaro,Tukia Henri
16	SM200700015	19.9.2006	HIFK	KalPa	7	0	3-0,0-0,4-0			60.00	Sinkkonen Mika	Fonselius Stefan,Suoraniemi Juha
17	SM200700016	19.9.2006	Ilves	Jokerit	1	4	1-0,0-3,0-1			60.00	Levonen Jari	Seppälä Jari,Tuomisto Petri

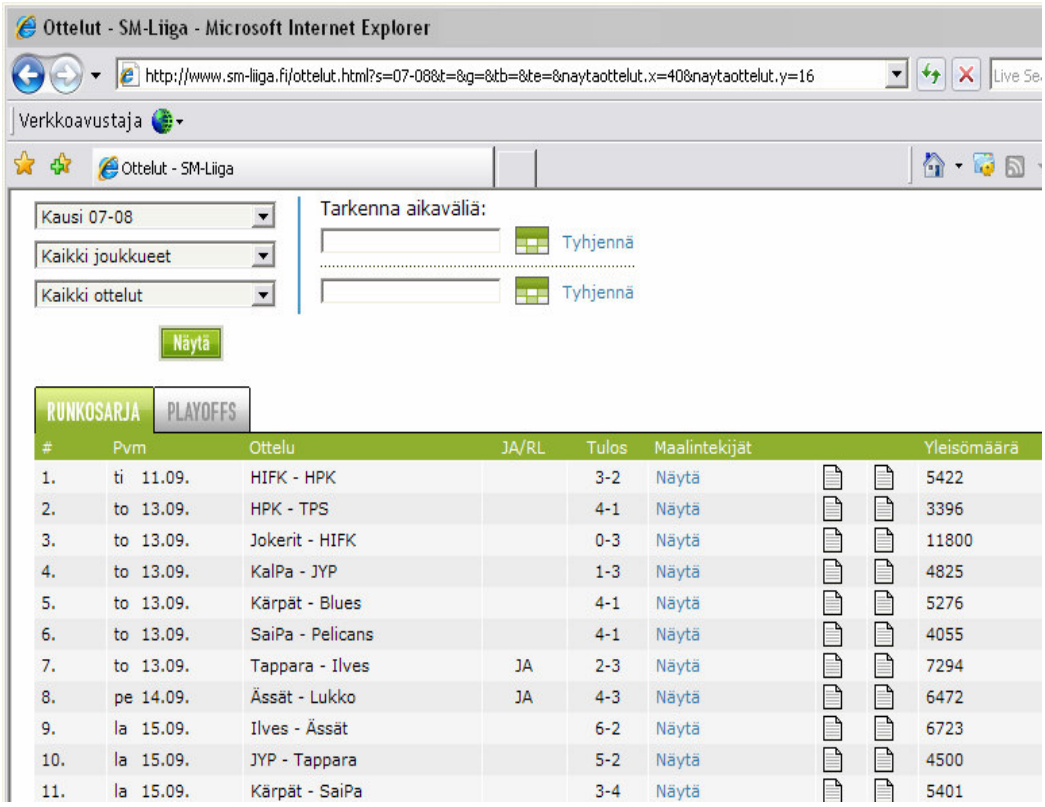
Kuva 29: Otteluiden otsikkotiedot sisältävä taulu *Matches*.

	A	B	C	D	E	F	G	H	I	J	K
1	Match_id	Detail_type_id	Detail_subtype_id	Team_id	Person_id	Order_nr	Time	Amount	Duration	Score	Info
127	SM200700002	L		Tappara	Viinanen Mika	32					2/VL
128	SM200700002	L		Tappara	Hancock Quinn	33					2/KH
129	SM200700002	L		Tappara	Venäläinen Sami	34					2/OL
130	SM200700002	L		Tappara	Henry Burke	35					3/VP
131	SM200700002	L		Tappara	Mäntymaa Ville	36					3/OP
132	SM200700002	L		Tappara	Virkkunen Teemu	37					3/VL
133	SM200700002	L		Tappara	Ojanen Janne	38					3/KH
134	SM200700002	L		Tappara	Kiilholma Juha	39					3/OL
135	SM200700002	L		Tappara	Pukka Mikko	40					4/VP
136	SM200700002	L		Tappara	Pyymäki Jarkko	41					4/H
137	SM200700002	L		Tappara	Laine Teemu	42					4/VL
138	SM200700002	L		Tappara	Öhman Stefan	43					4/KH
139	SM200700002	L		Tappara	Ojanen Marko	44					4/OL
140	SM200700002	G		Ilves	Viitakoski Vesa	45	00.40			1-0	AV,VoM
141	SM200700002	A		Ilves	Petriläinen Pasi	46	00.40			1-0	AV,VoM
142	SM200700002	P		Ilves	Jääskeläinen Teemu	47	00.08		02.00		Koukkaaminen
143	SM200700002	P		Ilves	Lehtivuori Joonas	48	02.27		02.00		Koukkaaminen
144	SM200700002	P		Ilves	Jerofejev Aleksandr	49	04.30		02.00		Kampitus
145	SM200700002	P		Tappara	Aalto Teemu	50	14.14		02.00		Kiinnipitäminen
146	SM200700002	P		Tappara	Mäntylä Tuukka	51	15.28		02.00		Pelin viivyttäminen - kiekko katsomoon
147	SM200700002	P		Ilves	Tuomanen Teppo	52	19.38		02.00		Kiinnipitäminen
148	SM200700002	P	TP	Ilves	Sandell Sami	53	21.26		02.00		Liian monta pelaajaa jäällä

Kuva 30: Otteluiden tapahtumatiedot sisältävä taulu *Match_Details*.

6.3. Jääkiekon SM-liigan kuluvan kauden tilastot

Kohdan 6.1 esimerkissä mainitun SM-liigan sivustot uusiutuivat syksyllä 2007 (kuvassa 31). Samalla myös internet-tilastopalvelun rakenne muuttui. Vaikka rakenne on pääpiirteittäin sama, etenkin ottelutapahtumien merkintätavoissa tapahtuneet muutokset aiheuttivat runsaasti muutoksia tiedonkeruulogiikkaan. Ottelulistan poiminta sen sijaan jopa helpottui, sillä ottelupäivämäärä on nyt joka otterivillä eikä keruusovelluksen näin ollen tarvitse säilyttää muistissa edellistä löydettyä päivämäärätietoa.



#	Pvm	Ottelu	JA/RL	Tulos	Maalintekijät			Yleisömäärä
1.	ti 11.09.	HIFK - HPK		3-2	Näytä			5422
2.	to 13.09.	HPK - TPS		4-1	Näytä			3396
3.	to 13.09.	Jokerit - HIFK		0-3	Näytä			11800
4.	to 13.09.	KalPa - JYP		1-3	Näytä			4825
5.	to 13.09.	Kärpät - Blues		4-1	Näytä			5276
6.	to 13.09.	SaiPa - Pelicans		4-1	Näytä			4055
7.	to 13.09.	Tappara - Ilves	JA	2-3	Näytä			7294
8.	pe 14.09.	Ässät - Lukko	JA	4-3	Näytä			6472
9.	la 15.09.	Ilves - Ässät		6-2	Näytä			6723
10.	la 15.09.	JYP - Tappara		5-2	Näytä			4500
11.	la 15.09.	Kärpät - SaiPa		3-4	Näytä			5401

Kuva 31: Esimerkki SM-liigan ottelutilastojen muuttuneesta ulkoasusta.

Vaikka joukkueiden kokoonpanot näyttävät pääpiirteittäin samalta kuin aikaisempien kausien tiedoissa, tietojen sisäinen esitystapa on kuitenkin erilainen. Jäsennettäessä taulukkoalueet Exceliin huomataan, että kentälliset ovat taulukossa osin samoissa soluissa. Lisäksi esitystapa on käännetty niin, että hyökkääjät luetellaan ennen puolustajia. Kuvissa 32 ja 33 on esimerkki ottelun kokoonpanosta selainesityksenä ja taulukkoesityksenä.

Joukkueiden kokoonpanot

1. Kenttä	**1. Kenttä**
39 Viitakoski, Vesa	21 Kivenmäki, Marko
41 Helminen, Raimo	7 Häppölä, Toni
10 Sandell, Sami	52 Saarela, Pasi
8 Jääskeläinen, Teemu	6 Peltonen, Pasi
3 Niemi, Toni	20 Konttinen, Tero
2. Kenttä	**2. Kenttä**
22 Koivisto, Toni	18 Joensuu, Jesse
79 Bishai, Mike	26 Hisey, Rob
51 Määttänen, Pasi	39 Tähtinen, Markku
81 Tukio, Arto	47 Sammalkangas, Tapio
12 Werner, Eric	10 Toivonen, Marko
3. Kenttä	**3. Kenttä**
27 Torkki, Sami	23 Takala, Tuomas
9 Peltola, Mikko	22 Kuparinen, Matti
11 Männikkö, Miikka	33 Huhtala, Tommi
33 Kantee, Kevin	59 Petre, Henrik
6 Lehtivuori, Joonas	24 Kuiper, Nick
4. Kenttä	**4. Kenttä**
20 Korhonen, Ville	32 Tasku, Petteri
36 Mattila, Lassi	36 Kemppainen, Joonas
61 Anttila, Marko	19 Salminen, Sakari

Kuva 32: Ottelun kokoonpano selainesityksenä.

	A	B	C	D
1	0	1. Kenttä	1. Kenttä	
2	0	39 :Viitakoski, Vesa: 41 :Helminen, Raimo: 10 :Sandell, Sami	21 :KivenmÃ¤ki, Marko: 7 :HÃ¤ppÃ¶lÃ¤, Toni: 52 :Saarela, Pasi	EPÄTOSI
3	0	8 :JÃ¤Ã¤skelÃ¤inen, Teemu: 3 :Niemi, Toni	6 :Peltonen, Pasi: 20 :Konttinen, Tero	EPÄTOSI
4	0	2. Kenttä	2. Kenttä	
5	0	22 :Koivisto, Toni: 79 :Bishai, Mike: 51 :MÃ¤Ã¤ttÃ¤nen, Pasi	18 :Joensuu, Jesse: 26 :Hisey, Rob: 39 :TÃ¤htinen, Markku	EPÄTOSI
6	0	81 :Tukio, Arto: 12 :Werner, Eric	47 :Sammalkangas, Tapio: 10 :Toivonen, Marko	EPÄTOSI
7	0	3. Kenttä	3. Kenttä	
8	0	27 :Torkki, Sami: 9 :Peltola, Mikko: 11 :MÃ¤nnikkÃ¶, Miikka	23 :Takala, Tuomas: 22 :Kuparinen, Matti: 33 :Huhtala, Tommi	EPÄTOSI
9	0	33 :Kantee, Kevin: 6 :Lehtivuori, Joonas	59 :Petre, Henrik: 24 :Kuiper, Nick	EPÄTOSI
10	0	4. Kenttä	4. Kenttä	
11	0	20 :Korhonen, Ville: 36 :Mattila, Lassi: 61 :Anttila, Marko	32 :Tasku, Petteri: 36 :Kemppainen, Joonas: 19 :Salminen, Sakari	EPÄTOSI
12	0	19 :Kuukka, Mikko: -	38 :Heikkinen, Eetu: 42 :Savinainen, Veli-Matti: (h)	EPÄTOSI
13	0	Maalivahdit:	Maalivahdit:	
14	0	35 :Leinonen, Tero:Varalla:: 37 :HÃ¤kkinen, Pasi	37 :KilpelÃ¤inen, Eero:Varalla:: 35 :Kapanen, Kimmo	EPÄTOSI

Kuva 33: Kuvan 32 tiedot *GetHTMLTablella* tuotuna taulukkoesityksenä.

Sen sijaan Excelin web-kysely jäsentää datan siten, että sen taulukointi vastaa paremmin selaimella näkyvää tapaa (kuva 34). Lisäksi huomataan, että

kuvan 33 merkistöongelma on poistunut. Poimintasovelluksessa on näin ollen harkittava web-kyselyn käyttöä jäsennyksen yksinkertaistamiseksi.

	A	B
1	1. Kenttä	1. Kenttä
2	39 Viitakoski, Vesa	21 Kivenmäki, Marko
3	41 Helminen, Raimo	7 Häppölä, Toni
4	10 Sandell, Sami	52 Saarela, Pasi
5		
6	8 Jääskeläinen, Teemu	6 Peltonen, Pasi
7	3 Niemi, Toni	20 Konttinen, Tero
8		
9	2. Kenttä	2. Kenttä
10	22 Koivisto, Toni	18 Joensuu, Jesse
11	79 Bishai, Mike	26 Hisey, Rob
12	51 Määttänen, Pasi	39 Tähtinen, Markku
13		
14	81 Tukio, Arto	47 Sammalkangas, Tapio
15	12 Werner, Eric	10 Toivonen, Marko
16		
17	3. Kenttä	3. Kenttä
18	27 Torkki, Sami	23 Takala, Tuomas
19	9 Peltola, Mikko	22 Kuparinen, Matti
20	11 Männikkö, Miikka	33 Huhtala, Tommi
21		
22	33 Kantee, Kevin	59 Petre, Henrik
23	6 Lehtivuori, Joonas	24 Kuiper, Nick

Kuva 34: Kuvan 32 tiedot taulukkoesityksenä web-kyselyllä jäsennettynä.

Koodikatkelmasta 22 käy ilmi, että joukkueen kentälliset ja maalivahdit ovat html-taulukossa kukin yhdessä solussa. Tästä voidaan päätellä web-kyselyn huomioivan myös taulukon solun muotoilut jäsentäessään tietoa. Käytettäessä *GetHTMLTable*-metodia maalivahdit ja kenttäpelaajat on jäsennettävä loppuun ohjelmallisesti.

```
class="grey">Leinonen, Tero</a><br />
<br/>
<b>Varalla:</b><br/>
37 <a href="/pelaajat/07-08/ilves/hakkinen-pasi.html"
class="grey">Häkkinen, Pasi</a><br />
</td><td>37 <a href="/pelaajat/07-08/assat/kilpelainen-eero.html"
class="grey">Kilpeläinen, Eero</a><br />
<br/>
```

```
<b>Varalla:</b><br/>
35 <a href="/pelaajat/07-08/assat/kapanen-kimmo.html"
class="grey">Kapanen, Kimmo</a><br />
</td>.
```

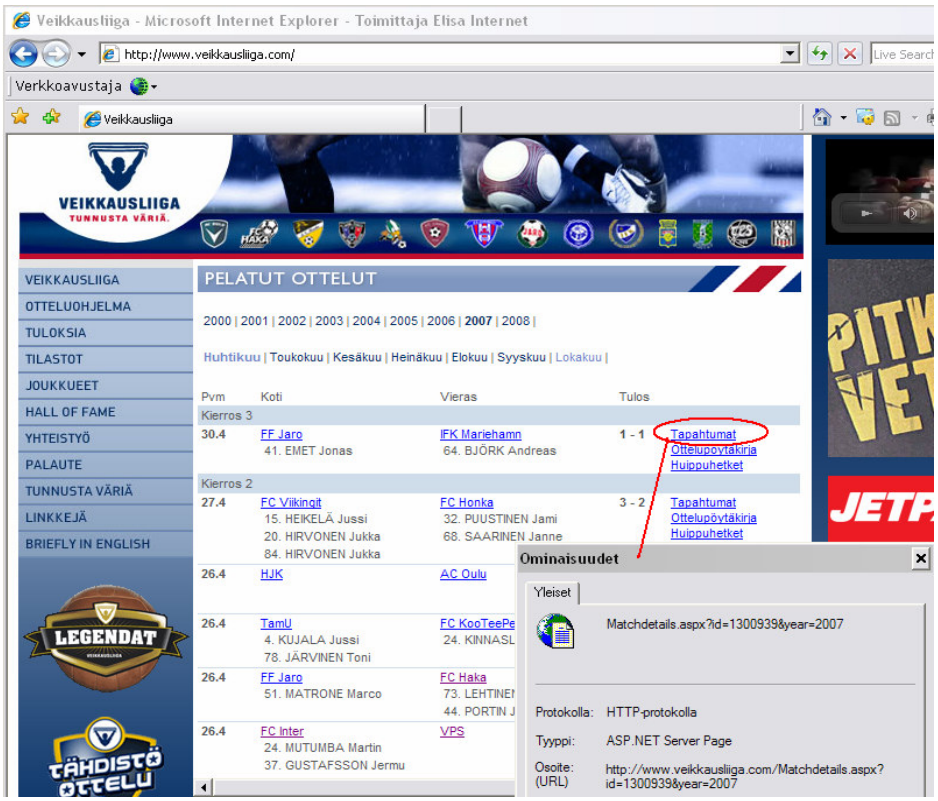
Koodikatkelma 22: Kuvien 32-34 tietoja html-koodina.

Koska kausi on tässä tapauksessa vielä kesken, ei ole mahdollista tallentaa html-sivuja ensin erikseen ja poimia dataa vasta sen jälkeen. Tässä ratkaisussa ottelulista luetaan aina uudelleen http-osoitteesta, mutta mikäli alisivut (otteluiden tiedot) on jo sovelluksen valitsemalla nimeämislogiikalla tallennettu sovittuun kansioon, käytetään levyllä olevaa versiota. Teoriassa ottelupöytäkirjat voivat vielä muuttua jälkeenpäin esimerkiksi virhekorjauksen vuoksi, mutta haluttaessa ladata sivu uudelleen internetistä riittää, että sen levylle tallennettu versio poistetaan.

Otteluiden erotuomarit on nyt määritelty html-listoina (merkitty <*li*>- ja <*/li*>-komennoilla) eikä taulukoina, joten ne joudutaan käsittelemään uudella tavalla. Ensin etsitään alkuperäisestä html-virrasta merkkijono ”Tuomarit”, minkä jälkeen voidaan käyttää alun perin html-taulukoiden käsittelyyn tarkoitettua *GetTableData*-funktiota, koska lista muistuttaa loogiselta rakenteeltaan taulukkoa. Näin saadusta tuomarilistasta erotetaan sitten vielä merkkijonoanalyysin avulla päätuomari ja linjatuomarit. Myös ottelun yleisömäärä ja erälukemat ovat html-sivun taulukkoalueen ulkopuolella, joten ne on etsittävä merkkijonohaulla. Yleisömäärä voidaan paikallistaa etsimällä merkkijonoa ”Yleisöä” ja eräluvut taas tunnistetaan html-koodissa merkkijonosta ”game_periods” ja tarkempi tunnistus voidaan tehdä erälukemia ympäröivien sulkujen perusteella.

6.4. Jalkapalloilun Veikkausliiga

Jalkapalloilun Veikkausliigan ottelutapahtumien poiminta on joiltakin osiltaan jääkiekon poimintoja yksinkertaisempi. Pelatuista otteluista ei ole saatavilla kaikki ottelut sisältävää listaa, sillä ottelut näytetään kuukausittain (kuva 35). Se ei ole kuitenkaan ongelma, sillä kauden ensimmäisten ja viimeisten otteluiden numerotunnisteita tutkimalla voidaan rakentaa silmukka, joka lukee kaikkien pelattujen otteluiden tiedot. Koska kaikki yhtä ottelua koskeva tieto löytyy tapahtumasivulta, varsinaisen ottelulistan käsittely voidaan ohittaa.



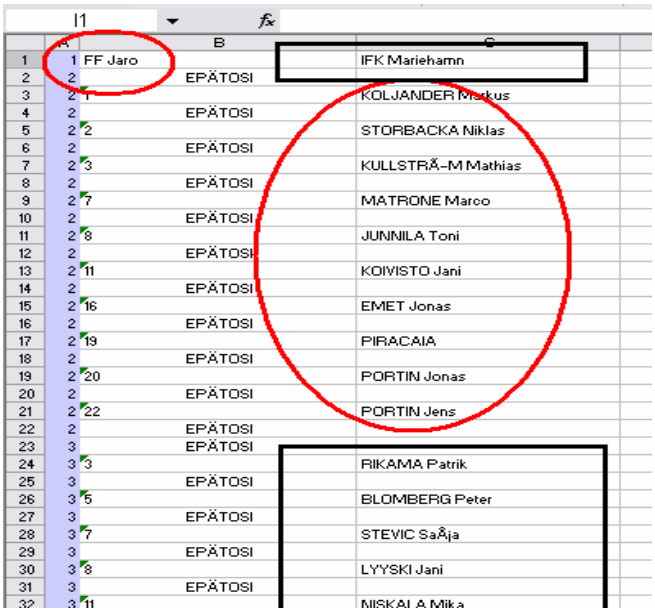
Kuva 35: Veikkausliigan ottelulista linkkeineen

Veikkausliigan tapahtumakanta sisältää muutakin kuin perustilastoja ja siksi html-sivujen tallennus mahdollisia laajennuksia varten on suositeltavaa. Esimerkiksi laukauksille tai kulmapotkuille ei ole tällä hetkellä varattu paikkaa tilastotietovarastossa, koska ne on nähty tietona, joiden hyötyarvo on lyhytkestoinen. Esimerkiksi yksittäisen seuran maalintekijät 10 vuoden takaa on kiinnostavaa tietoa, mutta ei enää joukkueen laukaustilasto tuolta ajalta. Toinen tekijä niiden pois jättämiselle tietovarastosta on se, ettei kyseisiä erikoistietoja ole saatavilla yleensä muiden sarjojen osalta. Tarpeen vaatiessa tietovarastoon voidaan kuitenkin rakentaa laji- ja sarjakohtaisia erikoiskantoja, joihin talletetaan tämän tyyppisiä tietoja.

Ottelulistassa on kunkin ottelun kohdalla kaksikin linkkiä – tapahtumat ja ottelupöytäkirja – joiden tietosisällöt ovat lähes identtiset. Tietojen keruu tietovarastoon olisi näin ollen mahdollista kummasta tahansa. Mutta tässä tapauksessa on päädytty Tapahtumat-osioon (kuva 36) siksi, että joitakin yksityiskohtia on siinä ilmaistu tarkoituksenmukaisemmin.

Kuva 36: Kuvan 35 linkeistä tapahtumat avattuna.

Ottelutapahtumissa olevat joukkueiden kokoonpanot jäsentyvät *GetHTMLTable*-proseduurilla kuvan 37 mukaisesti. Kotijoukkueen tiedot on ympyröity ja vierasjoukkueen tiedot kehystetty suorakaiteella.



Kuva 37: Kuvan 36 tapahtumasivu Exceliin jäsennettynä.

6.5. Uutispalstat

Vaikka internetin uutispalstoja ei olekaan mahdollista levittää edelleen eikä niiden pohjalta ole automaattisesti generoitavissa uutta informaatiota, ne kuitenkin voivat palvella faktatietojen tarkistuksen apuvälineenä. Vaikka tietoja voi selailla suoraan internetissä, voi olla perusteltua niiden keruu omaan järjestelmään esimerkiksi seuraavista syistä:

- nopeampi haku esimerkiksi Excel-taulukosta ajankohdan perusteella
- täsmähaku ei Googlen kautta edes aina ole mahdollinen
- tietojen säilyvyys internet-arkistossa on epävarma.

Kuvassa 38 oleva informaatiotietokanta on kerätty yksinkertaisella metodilla osoitteesta http://www.mtv3.fi/urheilu/arkisto.shtml/arkistot/futis/arkistot.

Kuva 38: Uutispalsta Exceliin koottuna.

Kuva 39: Yksittäisen uutisen rakenne MTV3:n uutisarkistossa.

Varsinainen uutisteksti ei ole html-taulukkona, vaan se on merkitty <*p*> ja </*p*>-komennoilla. Funktiolla *GetTableData* se voidaan kuitenkin jäsentää, koska jäsennettävä data on loogiselta rakenteeltaan taulukkomuotoista. Jotta jäsennys alkaisi vasta halutusta kohdasta, funktiolle välitetään html-merkkijonosta vain uutisotsikon jälkeinen osa.

Koska uutistekstin jälkeen sivustolla on vielä mainoksia tai muita linkkejä, katkaistaan uutinen ennen riviä, joka alkaa sanalla ”Takaisin” ja kootusta taulukkomuuttujasta myös leikataan ylimääräiset alkiot pois. Uutistekstin rivit liitetään lopuksi takaisin toisiinsa erottamalla ne välilyönnillä ja koska tarkoitus on tallentaa koko uutisteksti yhteen merkkijonomuuttujaan, siitä poistetaan mahdolliset rivinvaihtomerkit ja ylimääräiset välilyönnit.

7. Johtopäätökset

Vaikka julkisiin internet-tietoihin perustuvan tietovaraston aihealueena olikin palloilulajien tilastodata ensisijaisesti henkilökohtaisten intressien takia, aihealue osoittautui myös tutkimuksen kannalta hedelmälliseksi, koska aihealueen lähdesivustojen lukumäärä on valtava ja näin ollen ne kattavat käytännössä kaikki tietojen keruussa ilmenevät ongelmat.

Lisäksi tieto on perusrakenteeltaan sellaista, että eri kielialueilta lähtöisin olevat tiedot ovat saatettavissa yhteen. Suurin syy tähän ei kuitenkaan ole tekninen vaan se, että eri urheilulajeille on määritelty tarkat säännöt, mikä edistää informaation formalisointia. Palloilutietojärjestelmän peruselementtejä ovat seurat ja pelaajat, joiden nimet pääasiassa ovat kieliriippumattomia. Tarkka formaali data voi jopa korvata ottelun sanallisen arvioinnin kokonaan.

Rakennettaessa tietovarastoa, jonka lähdejärjestelmänä on internet, on kiinnitettävä huomiota aihealueen rajaukseen, sillä tietojen latausprosessi vaatii perinteistä tietovarastoa enemmän resursseja koko tietovaraston elinkaaren ajan. Käytännössä jokaiselle tietolähteelle on tehtävä oma latausproseduurinsa ja osa latauksista joudutaan välillä jopa rakentamaan kokonaan uudelleen – usein ilman valmistautumisaikaa – lähdesivuston muuttuessa.

Lisäksi internetistä puuttuu perinteisten lähdejärjestelmien tiedon laatukontrolli, joten tiedon muokkaus tietovarastoon soveltuvaksi vaatii usein enemmän työtä kuin yritysten operatiivisten järjestelmien osalta. Tiedonkeruuta vaikeuttaa myös se lähtökohta, että tiedot joudutaan keräämään html-presentaation kautta ja jo sivun generointivaiheessa alkuperäinen oikea data on saattanut muuttua virheelliseksi tai esitystapa vaikeasti tulkittavaksi.

Käytännössä tässä esitelty tietovarasto olisi parhaiten toteutettavissa itsenäisinä paikallistietovarastoina, joista kukin keskittyy vain yhden maan ja lajin palloilusarjoihin. Paikallisvarastoilla tulisi olla eri vastuuhenkilöt, jotka lajikohtaisen asiantuntemuksensa avulla pystyisivät huolehtimaan datan virheettömyydestä sen määrän ollessa rajallinen.

Tutkimuksen merkittävin anti tekijälle oli sellaisten ohjelmointimallien löytäminen, joiden avulla tietojen keruuta internetistä oli mahdollista tehostaa jopa dramaattisesti. Aikaisemmin käytetyt tietojen jäsennystavat jäivät lopulta kokonaan pois tutkimuksesta niiden heikon suorituskyvyn ja vaikean ylläpidettävyyden takia.

Lähteet

[Adams 2001]
Katherine C. Adams: The Web as Database: New Extraction Technologies and Content Management. ONLINE, March 2001.
http://www.onlinemag.net/OL2001/adams3_01.html.

[Butler & Stackowiak 2006]
David Butler & Bob Stackowiak: Master Data Management. An Oracle White Paper, October 2006. Oracle Corporation.
http://www.oracle.com/data_hub/mdm-white-paper.pdf.

[Devlin 1997]
Barry Devlin: *Data Warehouse: from Architecture to Implementation*. Addison Wesley Longman, 1997.

[Elmasri & Navathe 2000]
Ramez Elmasri & Shamkant B. Navathe: *Fundamentals of Database Systems*. Addison-Wesley, 2000.

[EU 2006]
Euroopan unionin virallinen lehti, komission suositus kulttuuriaineiston digitoinnista ja sähköisestä saatavuudesta sekä digitaalisesta säilyttämisestä (2006/585/EY).
http://ec.europa.eu/information_society/activities/digital_libraries/doc/recommendation/recommendation/fi.pdf.

[Gray & Watson 1998]
Paul Gray & Hugh J. Watson: *Decision Support in the Data Warehouse*. Prentice Hall, 1998.

[Han & Kamber 2001]
Jiawei Han & Micheline Kamber: *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, 2001.

[Hovi 1997]
Ari Hovi: *Data warehousing - tietovarastotekniikka*. Espoo: Suomen atk-kustannus, 1997.

[Hovi et al. 2001]
Ari Hovi, Heikki Koistinen & Jari Ylinen: *Tietovarastot liiketoiminnan tukena.* Helsinki: Satku, 2001.

[Inmon 2002]
W.H.Inmon: *Building the Data Warehouse*. Third Edition. Wiley Computer Publishing, 2002.

[Inmon 2005]
W.H.Inmon: *Building the Data Warehouse*. Fourth Edition. Wiley Publishing Inc., 2005.

[Inmon et al. 1999]
W.H.Inmon, Ken Rudin, Christopher K. Buss & Ryan Sousa: *Data Warehouse Performance*. Wiley & Sons, 1999.

[Jarke et al. 2000]
Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, Panos Vassiliadis: *Fundamentals of Data Warehouses*. Second, Revised and Extended Edition. Springer 2000.

[Kelly 1996]
Sean Kelly: *Data Warehousing, The Route to Mass Customization*. John Wiley & Sons, 1996.

[Kelly 1997]
Sean Kelly: *Data Warehouse in Action*. John Wiley & Sons, 1997.

[Kimball et al. 1998]
Ralph Kimball, Laura Reeves, Margy Ross, Warren Thornthwaite: *The Data Warehouse Lifecycle Toolkit*. Wiley Computer Publishing. 1998.

[Kowalkiewicz et al. 2006]
Marek Kowalkiewicz, Tomasz Kaczmarek, Witold Abramowicz: Robust Extraction and Aggregation of Web Content. VLDB '06, September 12–15, 2006, Seoul, Korea.
http://www.vldb.org/conf/2006/p1219-kowalkiewicz.pdf.

[Liu 2005]
Bing Liu: Web Content Mining. The 14th International World Wide Web Conference (WWW-2005). May 10-14, 2005, Chiba, Japan.
www.cs.uic.edu/~liub/Web-Content-Mining-2.pdf.

[Lyman 2002]
Peter Lyman: Archiving the World Wide Web. In Building a National Strategy for Preservation: Issues in Digital Media Archiving. Council on Library and Information Resources and the Library of Congress, April 2002.
http://www.digitalpreservation.gov/library/pdf/es_web.pdf.

[Nelson et al. 2006]
Michael L. Nelson, Joan A. Smith, Ignacio Garcia del Campo, Herbert Van de Sompel, Xiaoming Liu: Efficient, Automatic Web Resource Harvesting. Proceedings of WIDM 2006, November 10, 2006, Arlington, USA.
http://public.lanl.gov/herbertv/papers/f140-nelson.pdf.

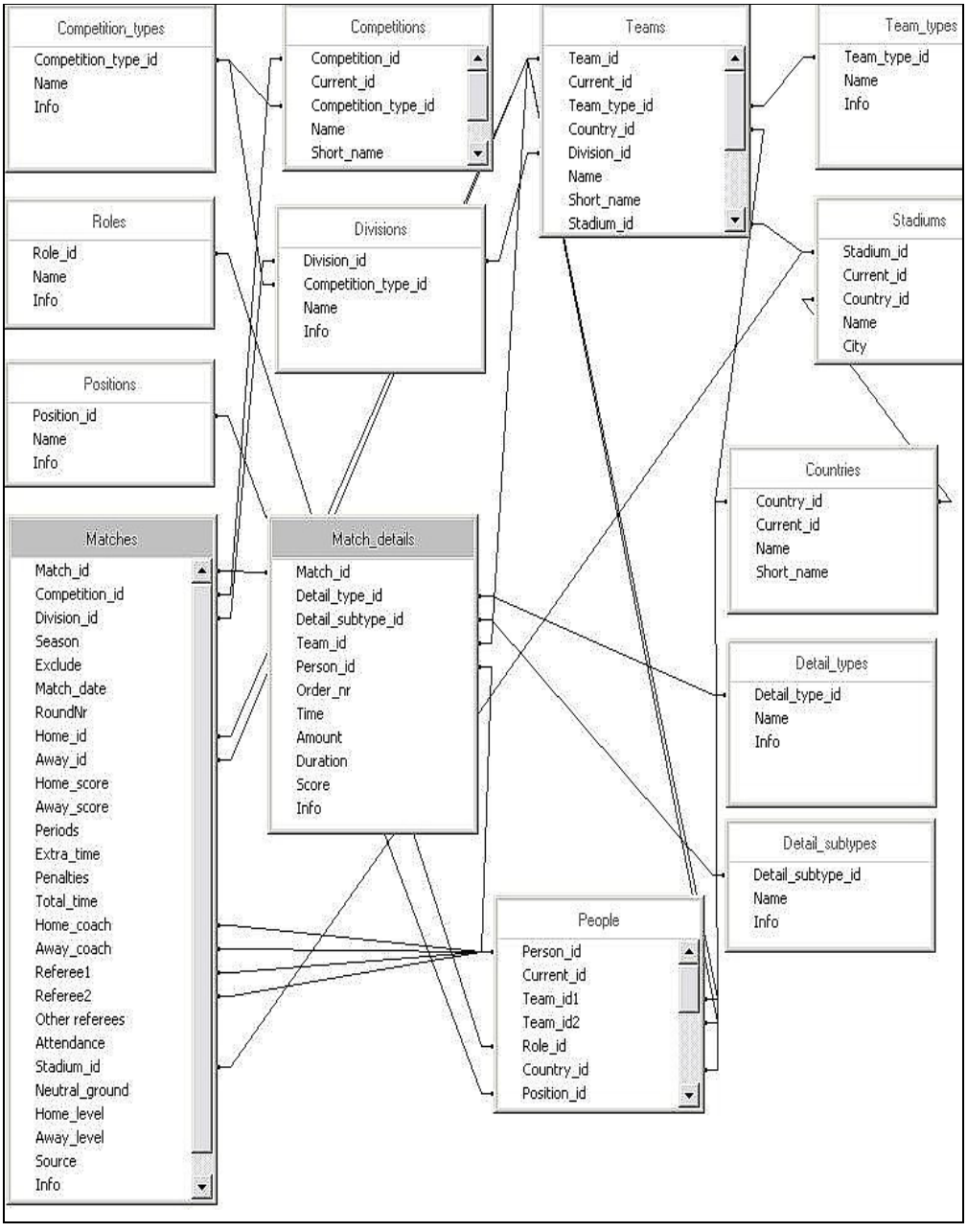
[W3C]
World Wide Web Consortium: HTML 3.2. Reference Specification, 1997 (Dave Raggett).
http://www.w3.org/TR/html401/struct/tables.html.

[Westerman 2001]
Paul Westerman: *Data Warehousing, Using the Wal-Mart Model*. Morgan Kaufmann Publishers, 2001.

[Wikipedia]
Web Crawler.
http://en.wikipedia.org/wiki/Web_crawler

Liitteet

Liite 1: Palloilusarjojen tietovaraston tietokantakaavio.



Liite 2: Palloilusarjojen tietovaraston tietokuvaukset.

Matches	Ottelut
Match_id	Ottelun tunnus
Competition_id	Kilpailun tunnus
Division_id	Kilpailun tarkenne (esimerkiksi alkusarja, play-offs tms)
Season	Kausi (esimerkiksi 2006 tai 2004-2005)
Exclude	Ohitetaanko ottelu tilastoinnissa? (ei lasketa sarjataulukkoon tms.)
Match_date	Ottelupäivä
RoundNr	Kierros (juokseva numero tai muu tunnus)
Home_id	Kotijoukkueen tunnus
Away_id	Vierasjoukkueen tunnus
Home_score	Kotijoukkueen maalimäärä ottelussa (tai pistemäärä)
Away_score	Vierasjoukkueen maalimäärä ottelussa (tai pistemäärä)
Periods	Erien tai pelijaksojen tulokset
Extra_time	Tarvittiinko jatkoaika?
Penalties	Tarvittiinko rangaistuslaukauskilpailu?
Total_time	Pelin kestoaika yhteensä
Home_coach	Kotijoukkueen valmentaja
Away_coach	Vierasjoukkueen valmentaja
Referee1	Päätuomari
Referee2	Mahdollinen toinen päätuomari
Other referees	Muut tuomarit (linjatuomarit tms.)
Attendance	Yleisömäärä
Stadium_id	Pelipaikan tunnus
Neutral_ground	Pelattiinko ottelu puolueettomalla kentällä?
Home_level	Kotijoukkueen taso (esimerkiksi sarjataso cup-ottelussa [1-n])
Away_level	Vierasjoukkueen taso (esimerkiksi sarjataso cup-ottelussa [1-n])
Source	Ottelun tietojen lähde (html-sivu tms)
Info	Lisätietoja
Check_list	Poiminnassa havaitut virheet

Match_details	**Ottelutapahtumat**
Match_id	Ottelun tunnus
Detail_type_id	Tapahtumalaji (esimerkiksi maali, syöttö tai vaihto)
Detail_subtype_id	Tapahtumalajin tarkenne (esimerkiksi rangaistuslaukausmaali)
Team_id	Tapahtuman joukkueen tunnus
Person_id	Tapahtuman pelaajan tunnus
Order_nr	Juokseva järjestysnumero otteluittain
Time	Tapahtuman aika
Amount	Tapahtumien lukumäärä
Duration	Tapahtuman kesto
Score	Ottelun tilanne tapahtumahetkellä
Info	Lisätietoja
Competitions	**Kilpailut**
Competition_id	Kilpailun tunnus
Current_id	Kilpailun nykyinen tunnus
Competition_type_id	Kilpailun tyyppi
Name	Kilpailun nimi
Short_name	Kilpailun lyhenne
Division_level	Kilpailun sarjataso (1-n)
Info	Lisätietoja
Teams	**Joukkueet**
Team_id	Joukkueen tunnus
Current_id	Joukkueen nykyinen tunnus
Team_type_id	Joukkueen tyyppi (esimerkiksi seurajoukkue tai maajoukkue)
Country_id	Joukkueen maatunnus
Division_id	Joukkueen sarja- tai lohkotunnus
Name	Joukkueen nimi
Short_name	Joukkueen lyhenne
Stadium_id	Joukkueen kotikenttä
City	Joukkueen kotipaikka
Founded	Perustamisvuosi
WWW	Kotisivut
Info	Lisätietoja

People	**Henkilöt**
Person_id	Henkilön tunnus
Current_id	Henkilön nykyinen tunnus
Team_id1	Henkilön joukkue 1 (esimerkiksi seurajoukkue)
Team_id2	Henkilön joukkue 2 (esimerkiksi maajoukkue)
Role_id	Henkilön rooli (pelaaja, valmentaja tms)
Country_id	Henkilön kotimaan tunnus
Position_id	Pelaajan pelipaikka (esimerkiksi puolustaja)
Name	Pelaajan nimi
Full_name	Pelaajan koko nimi
Surname	Pelaajan sukunimi
First_name	Pelaajan etunimi
Date_of_birth	Syntymäaika
Birth_place	Syntymäpaikka
Height	Pituus
Weight	Paino
Left_right	Kätisyys/jalkaisuus (left/right)
Info	Lisätietoja
Team_types	**Joukkuetyypit**
Team_type_id	Joukkuetyypin tunnus
Name	Selite (esimerkiksi seurajoukkue tai maajoukkue)
Info	Lisätietoja
Competition_types	**Kilpailutyypit**
Competition_type_id	Kilpailutyypin tunnus
Name	Selite (esimerkiksi sarja tai cup-kilpailu)
Info	Lisätietoja
Roles	**Roolit**
Role_id	Roolin tunnus
Name	Selite (esimerkiksi pelaaja tai erotuomari)
Info	Lisätietoja

Positions	**Pelipaikat**
Position_id	Roolin tunnus
Name	Selite (esimerkiksi pelaaja tai erotuomari)
Info	Lisätietoja
Detail_types	**Tapahtumatyypit**
Detail_type_id	Tapahtumatyypin tunnus
Name	Selite (esimerkiksi maali, syöttö tai varoitus)
Info	Lisätietoja
Detail_Subtypes	**Tapahtuman alatyypit**
Detail_subtype_id	Tapahtuman alatyypin tunnus
Name	Selite (esimerkiksi rangaistuslaukaus)
Info	Lisätietoja
Countries	**Maat**
Country_id	Maatunnus
Current_id	Maan nykyinen tunnus
Name	Nimi
Short_name	Lyhyt nimi
Info	Lisätietoja
Divisions	**Sarjojen alalajit/lohkot/tarkenteet**
Division_id	Alalajin tunnus
Competition_type_id	Kilpailutyyppi (esimerkiksi cup-kilpailu)
Name	Tarkenteen nimi (esimerkiksi runkosarja)
Info	Lisätietoja
Stadiums	**Kentät/pelipaikat**
Stadium_id	Kentän tunnus
Current_id	Kentän nykyinen tunnus
Country_id	Kentän kotimaan maatunnus
Name	Kentän nimi
City	Kotipaikka
Capacity	Yleisökapasiteetti
Measures	Pelialueen koko
Info	Lisätietoja

Liite 3: Jääkiekon SM-liigan aikaisempien pelikausien keruusovellus.

```
Option Explicit

'Esitellään Windowsin API-funktioita

Public Declare Function GetTempDir Lib "kernel32" Alias "GetTempPathA" (ByVal nBufferLength As Long, _
  ByVal lpBuffer As String) As Long

Public Declare Function InternetOpen Lib "wininet" Alias "InternetOpenA" (ByVal lpszAgent As String, _
  ByVal dwAccessType As Long, ByVal lpszProxyName As String, ByVal lpszProxyBypass As String, _
  ByVal dwFlags As Long) As Long

Public Declare Function InternetConnect Lib "wininet" Alias "InternetConnectA" (ByVal hInternet As Long, _
  ByVal lpszServerName As String, ByVal nServerPort As Long, ByVal lpszUserName As String, _
  ByVal lpszPassword As String, ByVal dwService As Long, ByVal dwFlags As Long, ByVal dwContext As Long) As Long

Public Declare Function InternetOpenUrl Lib "wininet.dll" Alias "InternetOpenUrlA" _
 (ByVal hOpen As Long, ByVal sUrl As String, ByVal sHeaders As String, ByVal lLength As Long, _
 ByVal lFlags As Long, ByVal lContext As Long) As Long

Public Declare Function InternetReadFile Lib "wininet.dll" _
 (ByVal hFile As Long, ByVal sBuffer As String, ByVal lNumBytesToRead As Long, _
 lNumberOfBytesRead As Long) As Integer

Public Declare Function InternetCloseHandle Lib "wininet.dll" (ByVal hInternetHandle As Long) As Integer

Public Declare Function URLDownloadToFile Lib "urlmon" Alias "URLDownloadToFileA" (ByVal pCaller As Long, _
  ByVal szURL As String, ByVal szFileName As String, ByVal dwReserved As Long, ByVal lpfnCB As Long) As Long

Public Const INTERNET_OPEN_TYPE_PRECONFIG = 0
Public Const INTERNET_FLAG_RELOAD = &H80000000
Public Const ERROR_SUCCESS = 0&

Public Const Kansio = "C:\HOCKEY\"
Const PerusURL = "http://213.28.165.114"
```

```
'Ottelutietojen poiminnassa käytettävät tietorakenteet

Type Ottelu
  Match_id As String
  Competition_id As String
  Division_id As String
  Season As String
  Match_date As Variant
  Home_id  As String
  Away_id As String
  Home_score As Long
  Away_score As Long
  Periods As String
  Extra_time As Boolean
  Penalties As Boolean
  Total_time As Variant
  Referee1 As String
  Referee2 As String
  Other_referees As String
  Attendance As Long
  Source As String
  Check_list As String
End Type

Type Tapahtuma
  Match_id As String
  Detail_type_id As String
  Detail_subtype_id As String
  Team_id As String
  Person_id As String
  Order_nr As Long
  Time As Variant
  Amount As Long
  Duration As Variant
  Score As String
  Info As String
End Type
```

```
Public Function GetTextFile(ByVal TextFile As String) As Variant

  'Luetaan tekstitiedosto yhteen merkkijonomuuttujaan

  Dim FileNumber As Long
  Dim ErrCode As Long
  Dim TextRow As String
  Dim TextStr As String

  On Error Resume Next

  GetTextFile = False

  FileNumber = FreeFile(0)
  Err = 0
  Open TextFile For Input As FileNumber
  ErrCode = Err

  'Paluuarvoksi jää false, jos ei löydy
  If (ErrCode <> 0) Then Exit Function

  TextStr = vbNullString
  TextStr = Input(FileLen(TextFile), #FileNumber)
  Close FileNumber
  GetTextFile = TextStr

End Function
```

```
Public Function GetHTMLFromURL(URL As String) As String

  'Luetaan URLin sisältö merkkijonona wininet.dll:n funktiokutsuilla

  Dim StrVar As String
  Dim hOpen As Long
  Dim hOpenUrl As Long
  Dim ReadBuffer As String * 32768
  Dim NumberOfBytesRead As Long

  StrVar = vbNullString

  'Avataan ensin internet-yhteys ja sen jälkeen URL
  hOpen = InternetOpen("", INTERNET_OPEN_TYPE_PRECONFIG, _
    vbNullString, vbNullString, 0)
```

```
  hOpenUrl = InternetOpenUrl(hOpen, URL, vbNullString, 0, _
    INTERNET_FLAG_RELOAD, 0)

  'Luetaan niin kauan kuin merkkejä on luettavissa
  Do
    Call InternetReadFile(hOpenUrl, ReadBuffer, _
      Len(ReadBuffer), NumberOfBytesRead)
    StrVar = StrVar & Left$(ReadBuffer, NumberOfBytesRead)
  Loop Until (NumberOfBytesRead = 0)

  'Suljetaan yhteys
  If hOpenUrl <> 0 Then InternetCloseHandle (hOpenUrl)
  If hOpen <> 0 Then InternetCloseHandle (hOpen)

  'Palautetaan luetut merkit yhtenä merkkijonona
  GetHTMLFromURL = StrVar

End Function
```

```
Public Function DownloadFile(SourceUrl As String, _
  LocalFile As String) As Boolean

  'URLDownloadToFile pelkistettynä (lisäparametrit oletusarvoina)
  DownloadFile = (URLDownloadToFile(0&, SourceUrl, LocalFile, _
    0&, 0&) = ERROR_SUCCESS)

End Function
```

```
Public Sub DownloadFiles(ByVal SourceUrl As String, ByVal LocalFile As _
  String, ByVal X_Start As Variant, ByVal X_End As Variant, _
  ByVal Y_Start As Variant, ByVal Y_End As Variant, ByVal LeadingZeros _
  As Boolean, ByVal MinimumSize As Long, ByRef FileCounter As Long, _
  ByVal MaxFiles As Long, ByVal ReplaceFiles As Boolean, _
  ByVal ExitInnerLoop As Boolean)

  'Web-sivujen massaluku

  Dim URL As String
  Dim LocalFileName As String
  Dim Counter1 As Long
  Dim Counter2 As Long
  Dim TotalCounter As Long
```

```
Dim FormatString1 As String
Dim FormatString2 As String
Dim X_Char As Boolean
Dim Y_Char As Boolean

On Error Resume Next

'Silmukkaparametrit voidaan antaa myös merkkeinä (esimerkiksi 'A'-'Z')
X_Char = Not (IsNumeric(X_Start))
Y_Char = Not (IsNumeric(Y_Start))

If X_Char Then
  X_Start = Asc(Trim(X_Start))
  X_End = Asc(Trim(X_End))
End If

If Y_Char Then
  Y_Start = Asc(Trim(Y_Start))
  Y_End = Asc(Trim(Y_End))
End If

'Käytetäänkö numeroparametreissa etunollia?
If LeadingZeros Then LeadingZeros = LeadingZeros And (X_Char = False) And (Y_Char = False)
FormatString1 = "0"
FormatString2 = "0"
If LeadingZeros Then
  FormatString1 = String(Len(Trim(X_End)), "0")
  FormatString2 = String(Len(Trim(Y_End)), "0")
End If

'Varmistetaan parametrien numeromuotoisuus
'(edellä muuttujan tyyppi piti vielä säilyä alkuperäisenä)
X_Start = Val(X_Start)
Y_Start = Val(Y_Start)
X_End = Val(X_End)
Y_End = Val(Y_End)

'Varmistetaan, ettei loppuarvo ole alkuarvoa pienempi
If X_End < X_Start Then X_End = X_Start
If Y_End < Y_Start Then Y_End = Y_Start

'Ladataan tiedosto n kertaa url-viittausta muuttaen
TotalCounter = 0
FileCounter = 0
```

```
  For Counter1 = X_Start To X_End
    For Counter2 = Y_Start To Y_End
      TotalCounter = TotalCounter + 1
      URL = SourceUrl
      LocalFileName = LocalFile
      '%N% paikallisen tiedoston nimessä generoi talletettavalle sivulle
      'juoksevan järjestysnumeron osana nimeä
      LocalFileName = Replace(LocalFileName, "%N%", Format(TotalCounter, _
        FormatString1), , , vbTextCompare)
      'Silmukkamuuttujat voivat olla A-Z tai vapaavalintainen numeroväli
      If X_Char Then
        URL = Replace(URL, "%X%", Chr(Counter1), , , vbTextCompare)
        LocalFileName = Replace(LocalFileName, "%X%", _
          Chr(Counter1), , , vbTextCompare)
      Else
        URL = Replace(URL, "%X%", Format(Counter1, FormatString1), , _
          , vbTextCompare)
        LocalFileName = Replace(LocalFileName, "%X%", Format(Counter1, _
          FormatString1), , , vbTextCompare)
      End If
      If Y_Char Then
        URL = Replace(URL, "%Y%", Chr(Counter2), , , vbTextCompare)
        LocalFileName = Replace(LocalFileName, "%Y%", _
          Chr(Counter2), , , vbTextCompare)
      Else
        URL = Replace(URL, "%Y%", Format(Counter2, FormatString2), , _
          , vbTextCompare)
        LocalFileName = Replace(LocalFileName, "%Y%", Format(Counter2, _
          FormatString2), , , vbTextCompare)
      End If
      'Parametrilla voidaan myös ohittaa jo aikaisemmin talletettu tiedosto
      If ReplaceFiles Or (Dir(LocalFileName) = "") Then
        Call DownloadFile(URL, LocalFileName)
      End If
      'Jos ladattu tiedosto alle minimikoon, tuhotaan se
      If (MinimumSize > 0) And (FileLen(LocalFileName) < MinimumSize) Then
        Kill LocalFileName
      Else
        FileCounter = FileCounter + 1
        'Parametrilla voidaan aina lopettaa silmukka tiedoston löytyessä
        If ExitInnerLoop Then Exit For
      End If
      'Jos maksimimäärä kelvollisia tiedostoja talletettu, poistutaan
      If (MaxFiles > 0) And (FileCounter >= MaxFiles) Then Exit Sub
```

```
    Next Counter2
  Next Counter1

End Sub
```

```
Public Function GetNumbers(ByVal StrValue As String) As Long

  'Poimitaan numerot merkkijonosta yhdeksi kokonaisluvuksi

  Dim Counter As Long
  Dim ReturnValue As String

  On Error Resume Next

  ReturnValue = vbNullString
  For Counter = 1 To Len(StrValue)
    If (Mid(StrValue, Counter, 1) >= "0") And (Mid(StrValue, Counter, 1) <= "9") Then
      ReturnValue = ReturnValue & Mid(StrValue, Counter, 1)
    End If
  Next Counter
  GetNumbers = Val(ReturnValue)

End Function
```

```
Public Function SendXML(ByVal XML As String, ByVal HTTPMethod, _
  ByVal URL As String) As String

   'Lähetetään parametrina saatu XML annettua metodia
   'käyttäen annettuun osoitteeseen
    'Jos metodi GET, XML voi olla tyhjä

   Dim X As Object
   Dim ErrorCode As Long

   On Error Resume Next

   'Luodaan ActiveX-objekti
   Set X = CreateObject("Msxml2.XMLHTTP")
   ErrorCode = Err
   If (ErrorCode <> 0) Then
     SendXML = Error(ErrorCode)
     Exit Function
```

```
   End If

   'Lähetetään mahdollinen XML-sanoma
   X.Open HTTPMethod, URL, False
   X.SetRequestHeader "Content-Type", "text/xml"
   X.Send Trim(XML)

   'Paluusanoma on annetun URLin sisältö GET-metodilla
   SendXML = X.ResponseText
   Set X = Nothing

End Function
```

```
Function GetHRefs(ByVal HTML As String, ByVal FindHRef As String, _
  ByVal TextLinksOnly As Boolean) As Variant

  'Funktio palauttaa HTML-sisällöstä annetulla merkkijonolla
  'alkavat linkit. TextLinksOnly ohittaa ne linkit, jotka avautuvat
  'esimerkiksi kuvaa klikkaamalla

  Dim Position1 As Long
  Dim Position2 As Long
  Dim Position3 As Long
  Dim HRefList As String
  Dim HRef As String

  On Error Resume Next

  GetHRefs = False
  HRefList = vbNullString

  'Etsitään href-alkuisen viittauksen ensimmäinen esiintymä
  Position1 = InStr(Position1 + 1, HTML, FindHRef, vbTextCompare)

  'Kerätään kaikki annetulla merkkijonolla alkavat linkit
  'ensin merkkijonoon (ASCII-koodi 255 erottaa merkkijonossa
  'eri osat/linkit)
  While (Position1 > 0)
    HRef = Mid(HTML, Position1, Len(HTML))
    Position2 = InStr(1, HRef, ">", vbTextCompare)
    If (Position2 > 0) Then
      'Linkkiviittaus päättyy lopulta <-merkkiin
      Position3 = InStr(1, HRef, "<", vbTextCompare)
```

```
      'Jos arvona ei-tyhjä (ei ><) tai otetaan kaikki linkit
      '(muuten vain ne, joilla teksti, jota klikkaamalla linkki avautuu)
      If (Position3 > Position2 + 1) Or (TextLinksOnly = False) Then
        HRefList = HRefList & Chr(255) & Left(HRef, Position3)
      End If
    End If
    Position1 = InStr(Position1 + 1, HTML, FindHRef, vbTextCompare)
  Wend

  'Lopuksi merkkijono puretaan taulukkomuuttujaksi
  If (HRefList <> vbNullString) Then
    HRefList = Mid(HRefList, 2, Len(HRefList))
    GetHRefs = Split(HRefList, Chr(255))
  End If

End Function
```

```
Function GetTableData(ByVal HTML As String, ByVal Tag As String) As Variant

  'Haetaan HTML:stä halutut "<tag " ja "</tag>"-lohkot

  Dim Position As Long
  Dim TableData As Variant
  Dim PrevPosition As Long

  GetTableData = False
  Position = InStr(1, HTML, "<" & Tag, vbTextCompare)

  'Palautetaan FALSE, jos etsittyä tagia ei löydy
  If (Position = 0) Then Exit Function

  'Pilkotaan koko HTML halutun tagin alkumerkinnän perusteella
  TableData = Split(Mid(HTML, Position + Len(Tag) + 1, Len(HTML)), "<" _
    & Tag, , vbTextCompare)

  'Varmistetaan, että taulukon viimeinen alkio päättyy viimeiseen löytyvään
  'loppumerkintään
  PrevPosition = 0
  'Position on valmiiksi >0 aikaisemman operaation jäljiltä
  While (Position > 0)
    Position = InStr(PrevPosition + 1, TableData(UBound(TableData)), "</" _
      & Tag, vbTextCompare)
    If (Position > 0) Then PrevPosition = Position
```

```
  Wend

  'Poistetaan viimeisen loppumerkinnän jälkeinen osuus
  If (PrevPosition > 0) Then
    TableData(UBound(TableData)) = Left(TableData(UBound(TableData)), _
      PrevPosition - 1)
  End If

  'Palautetaan 1-ulotteinen taulukko
  GetTableData = TableData

End Function
```

```
Function GetCellValue(ByVal td As String, ByVal Separator As String) _
  As Variant

  'Palauttaa taulukon solun arvon ohittaen muotoilut

  Dim TblData As Variant
  Dim Counter As Long
  Dim Position As Long
  Dim CellValue As Variant

  'Oletusarvoksi jää false, ellei solulle löydy arvoa
  CellValue = False

  'Jos arvo on jo valmiina (ei tageja), palautetaan merkkijono sellaisenaan
  If (InStr(1, td, "<", vbTextCompare) = 0) Then
    GetCellValue = td
    Exit Function
  End If

  'Pilkotaan merkkijono taulukon arvoa edeltävän >-merkin kohdalta
  TblData = Split(td, ">", , vbTextCompare)

  'Käydään kaikki löytyneet taulukon alkiot läpi
  'ensimmäistä lukuun ottamatta (ei voi sisältää arvoa)
  For Counter = LBound(TblData) + 1 To UBound(TblData)
    'Jos alkiotaulukon alkio alkaa <-merkillä, ohitetaan
    '(aloittaa muotoilun tms)
    Position = InStr(1, Trim(TblData(Counter)), "<", vbTextCompare)
    'Leikataan merkkijono seuraavaan <-merkkiin asti
    If (Position >= 1) Then
```

```
      TblData(Counter) = Left(TblData(Counter), _
        InStr(1, TblData(Counter), "<", vbTextCompare) - 1)
    End If
    If (Trim(TblData(Counter)) <> vbNullString) Then
      'Kootaan kelvolliset arvot yhdeksi
      If (VarType(CellValue) = vbBoolean) Then
        CellValue = TblData(Counter)
      Else
        CellValue = CellValue & Separator & TblData(Counter)
      End If
    End If
  Next Counter
  GetCellValue = CellValue

End Function
```

```
Function GetHTMLTable(ByVal HTML As String, ByVal TableIndex As Long, _
  ByVal ReLoad As Boolean, ByVal Header As Boolean, _
  ByVal Count As Boolean, ByVal ValueSeparator As String, _
  ByVal Links As Boolean) As Variant

  'Funktio palauttaa HTML-koodista halutun taulukon
  '(yksi kerrallaan)

  Dim RowCounter As Long
  Dim ColumnCounter As Long
  Static TableData As Variant
  Dim RowData As Variant
  Dim HeaderData As Variant
  Dim ColumnData As Variant
  Dim TableRowCounter As Long
  Dim TableColumnCounter As Long
  Dim MaxColumnCounter As Long
  Dim CellValue As Variant
  Dim Table() As Variant

  On Error Resume Next

  'Taulukkolista luetaan muistiin vain kerran
  If (Not (IsArray(TableData)) Or ReLoad) Then
    HTML = Replace(Replace(Replace(Replace(HTML, Chr(13), " "), _
      Chr(10), " "), Chr(9), " "), Chr(0), " ")
    HTML = Replace(HTML, " ", " ")
```

```
  TableData = GetTableData(HTML, "table")
End If

'Palautetaan FALSE, jos annettu indeksi suurempi kuin taulukoiden lkm
If (TableIndex > UBound(TableData)) Then
  GetHTMLTable = False
  Exit Function
End If

'Parametrilla Count palautetaan vain taulukoiden lukumäärä HTML-dokumentissa
If (Count) Then
  GetHTMLTable = (UBound(TableData) - LBound(TableData)) + 1
  Exit Function
End If

'Kootaan HTML-taulukon rivit erilliseen taulukkoon
RowData = GetTableData(TableData(TableIndex), "tr")
ReDim Table(LBound(RowData) To UBound(RowData), 255) As Variant
TableRowCounter = LBound(Table, 1) - 1
MaxColumnCounter = LBound(Table, 2) - 1
If (VarType(RowData) <> vbBoolean) Then
  For RowCounter = LBound(RowData) To UBound(RowData)
    'Haluttaessa taulukon otsikko otetaan mukaan yhdeksi riviksi
    If (Header) Then
      HeaderData = GetTableData(RowData(RowCounter), "th")
      Header = False
    End If
    'Otsikko käsitellään normaalina taulukon rivinä
    'muutoin taulukoidaan sarakkeiden arvot
    If (Header And (VarType(HeaderData) <> vbBoolean)) Then
      ColumnData = HeaderData
    Else
      ColumnData = GetTableData(RowData(RowCounter), "td")
    End If
    If (VarType(ColumnData) <> vbBoolean) Then
      TableRowCounter = TableRowCounter + 1
      TableColumnCounter = LBound(Table, 2) - 1
      For ColumnCounter = LBound(ColumnData) To UBound(ColumnData)
        'Oletusarvoisesti palautetaan solun arvo
        CellValue = GetCellValue(ColumnData(ColumnCounter), _
          ValueSeparator)
        'Mutta jos löytyy linkki, palautetaan se
        '(mahdollinen solun arvo mukana)
        If Links Then CellValue = GetHRefs(ColumnData(ColumnCounter), _
```

```
            "href=", False)(0)
          TableColumnCounter = TableColumnCounter + 1
          'Pidetään kirjaa suurimmasta arvon sisältävästä sarakenumerosta
          If (MaxColumnCounter < TableColumnCounter) Then _
            MaxColumnCounter = TableColumnCounter
          Table(TableRowCounter, TableColumnCounter) = CellValue
        Next ColumnCounter
      End If
    Next RowCounter
  End If

  'Supistetaan taulukko todellista sarakemäärää vastaavaksi
  ReDim Preserve Table(LBound(RowData) To UBound(RowData), MaxColumnCounter) As Variant
  GetHTMLTable = Table

End Function
```

```
Function ArrayLookup(ArrayVar As Variant, ByVal SearchValue As String, ByVal Column As Long) As Long

  'Funktio palauttaa halutun arvon indeksin 2-ulotteisen taulukon sarakkeessa

  Dim Counter As Long

  On Error Resume Next

  ArrayLookup = -1
  SearchValue = UCase(Trim(SearchValue))
  For Counter = LBound(ArrayVar, 1) To UBound(ArrayVar, 1)
    If (Trim(UCase(ArrayVar(Counter, Column))) = SearchValue) Then
      ArrayLookup = Counter
      Exit Function
    End If
  Next Counter

End Function
```

```
Function ArrayColumnOffset(ArrayVar As Variant, ByVal StartRow As Long, ByVal Column As Long, _
  ByVal CellCount As Long, ByVal StopWhenFalse As Boolean) As Variant

  'Funktio palauttaa yksiulotteisen matriisin alkaen halutulta riviltä ja sarakkeesta

  Dim Counter As Long
```

```
  Dim CellList As String

  On Error Resume Next

  CellList = vbNullString
  ArrayColumnOffset = False
  For Counter = StartRow To (StartRow + CellCount) - 1
    If (VarType(ArrayVar(Counter, Column - 1)) <> vbBoolean) Then
      CellList = CellList & Chr(255) & ArrayVar(Counter, Column - 1)
    Else
      If StopWhenFalse Then Exit For
    End If
  Next Counter
  If (Trim(CellList) <> vbNullString) Then ArrayColumnOffset = Split(Mid(CellList, 2, Len(CellList)), Chr(255))

End Function
```

```
Function Swapnames(ByVal NameStr As String) As String

  'Vaihdetaan etunimen ja sukunimen paikat
  'Esimerkiksi: Tarja Halonen -> Halonen Tarja

  Dim Position1 As Long
  Dim Position2 As Long
  Dim NewName As String

  On Error Resume Next

  NewName = NameStr
  Position1 = 0
  Position1 = Application.Match(NameStr, ThisWorkbook.Worksheets("Nimet").Range("A:A"), 0)
  If (IsError(Position1)) Then Position1 = 0
  If (Position1 > 0) Then
    NewName = ThisWorkbook.Worksheets("Nimet").Cells(Position1, 3).Value
  Else
    Position2 = InStr(1, NameStr, " ", vbTextCompare)
    If (Position2 > 0) Then
      NewName = Mid(NameStr, Position2 + 1, Len(NameStr)) & " " & Left(NameStr, Position2 - 1)
    End If
  End If
  Swapnames = Replace(NewName, "_", " ")

End Function
```

```
Sub VanhatOttelut(ByVal id As String, ByVal Kausi As String)

  'Vanhojen ottelutietosivustojen tallennus

  Dim Apu As String
  Dim Hrefs As Variant
  Dim Laskuri As Long
  Dim OtteluData As Ottelu
  Dim Laskin As Long
  Dim OttelulistaURL As String
  Dim OtteluHRef As String
  Dim URL As String
  Dim OtteluId As Long

  On Error Resume Next

  'Haetaan kauden ottelulista
  OttelulistaURL = PerusURL & "/sarjaohjelma.asp?sid=" & id & "&old=true"
  Application.StatusBar = OttelulistaURL
  Call DownloadFile(OttelulistaURL, Kansio & Kausi & ".htm")
  OtteluHRef = "href=""#"" onclick=""return popup('ohjelma','sarjaohjelma_historia.asp?id="

  'Luetaan koko lista muistiin ja kerätään linkit
  Apu = GetHTMLFromURL(OttelulistaURL)
  Hrefs = GetHRefs(Apu, OtteluHRef, True)

  'Eristetään linkeistä juokseva ottelunumero ja haetaan sen avulla
  'erikseen ottelutapahtumat ja otteluiden kokoonpanot
  For Laskuri = LBound(Hrefs) To UBound(Hrefs)
    OtteluId = GetNumbers(Mid(Hrefs(Laskuri), Len(OtteluHRef) - 1, 7))
    Application.StatusBar = Kausi & "/" & OtteluId
    URL = PerusURL & "/sarjaohjelma_historia.asp?id=" & OtteluId
    Call DownloadFile(URL, Kansio & Kausi & "_tapahtumat_" & OtteluId & ".htm")
    URL = PerusURL & "/sarjaohjelma_kokoonpano.asp?otteluid=" & OtteluId
    Call DownloadFile(URL, Kansio & Kausi & "_kokoonpanot_" & OtteluId & ".htm")
  Next Laskuri
  Application.StatusBar = False

End Sub
```

```
Sub TalletaVanhatOttelut()

  'Talletetaan vanhat tiedot kausi kerrallaan

  VanhatOttelut "1483", "2006-2007_runkosarja"
  VanhatOttelut "1484", "2006-2007_playoffs"

  VanhatOttelut "1465", "2005-2006_runkosarja"
  VanhatOttelut "1466", "2005-2006_playoffs"

  VanhatOttelut "1303", "2004-2005_runkosarja"
  VanhatOttelut "1304", "2004-2005_playoffs"

  VanhatOttelut "1124", "2003-2004_runkosarja"
  VanhatOttelut "1125", "2003-2004_playoffs"

  VanhatOttelut "938", "2002-2003_runkosarja"
  VanhatOttelut "939", "2002-2003_playoffs"

  VanhatOttelut "662", "2001-2002_runkosarja"
  VanhatOttelut "663", "2001-2002_playoffs"

  VanhatOttelut "468", "2000-2001_runkosarja"
  VanhatOttelut "469", "2000-2001_playoffs"

  VanhatOttelut "307", "1999-2000_runkosarja"
  VanhatOttelut "308", "1999-2000_playoffs"

End Sub
```

```
Sub PoimiOttelut()

  'Sovelluksen pääohjelma (talletettujen tietojen käsittely)

  Dim StartTime As Variant
  Dim Kausi As String

  'Samalla mitataan käytettyä aikaa
  StartTime = Now()

  On Error Resume Next

  'Kysytään käyttäjältä poimittava kausi
```

```
  Kausi = InputBox("Kausi:", "Kausi")

  'Tyhjennetään tietokantasivut
  With ThisWorkbook.Worksheets("Matches")
    .Range("2:" & .UsedRange.Rows.Count + 1).ClearContents
  End With

  With ThisWorkbook.Worksheets("Match_details")
    .Range("2:" & .UsedRange.Rows.Count + 1).ClearContents
  End With

  HaeOttelut Kansio, Kausi, "Runkosarja", "SM-liiga"

  MsgBox "Käytetty aika: " & Format(Now() - StartTime, " HH:MM:SS")

End Sub
```

```
Sub HaeOttelut(ByVal Kansio As String, ByVal Kausi As String, ByVal Sarja As String, ByVal Kilpailu As String)

  'Käydään läpi kauden ottelut

  Dim TableArray As Variant
  Dim Count As Long
  Dim HTML As String
  Dim Counter As Long
  Dim RowCounter As Long
  Dim CurrentRow As Long
  Dim OtteluPvm As String
  Dim Ottelu As String
  Dim OtteluData As Ottelu
  Dim Tulos As String
  Dim OtteluNro As String
  Dim Positio As Long
  Dim Apu As Variant
  Dim KokoonpanoHTML As String
  Dim TapahtumaHTML As String
  Dim TapahtumaRivit() As Tapahtuma
  Dim TapahtumaLaskuri As Long
  Dim OtteluLaskuri As Long

  On Error Resume Next
  Application.ScreenUpdating = False
```

```
'Avataan tallennettu ottelulista
HTML = GetTextFile(Kansio & Kausi & "_" & Sarja & ".htm")

'Ladataan taulukkodata muistiin ja lasketaan taulukkoalueiden lukumäärä
Count = GetHTMLTable(HTML, 0, True, False, True, "", False)

'Luetaan taulukkoalue kerrallaan ja poimitaan ottelutiedot
OtteluLaskuri = 0

For Counter = 0 To Count - 1

  'Jos kutsutaan monella tasolla, taulukkorakenne on ladattava ylimmällä
  'tasolla uudelleen, sillä alin taso päivittää funktion staattista tietoa
  TableArray = GetHTMLTable(HTML, Counter, (Counter > 0), True, False, ":", True)

  For RowCounter = LBound(TableArray, 1) To UBound(TableArray, 1)

    Ottelu = vbNullString
    Tulos = vbNullString
    OtteluNro = vbNullString

    'Jos löydetty taulukkorakenne ei ole tyhjä, poimitaan ottelun pvm ja tulos
    If (VarType(TableArray(RowCounter, 0)) <> vbBoolean) Then
      If (Mid(TableArray(RowCounter, 0), 6, 1) = ".") And (Mid(TableArray(RowCounter, 0), 9, 1) = ".") Then
        OtteluPvm = Trim(Mid(TableArray(RowCounter, 0), 4, 10))
      End If
     'Poimitaan joukkueet ja tulos taulukon sarakkeista
      Ottelu = GetCellValue(Trim(TableArray(RowCounter, 1)), "")
      Tulos = Trim(TableArray(RowCounter, 2))
    End If

    'Jatkotoimenpiteet vain jos taulukon rivi on aito ottelurivi
    If (InStr(1, Ottelu, " - ", vbTextCompare) > 0) And (InStr(1, Tulos, " - ", vbTextCompare) > 0) Then

      'Poimitaan ottelun linkistä id-numero, jolla löydetään tapahtumatiedosto
      Positio = InStr(1, TableArray(RowCounter, 1), "?id=", vbTextCompare)
      If (Positio > 0) Then
        OtteluNro = Mid(TableArray(RowCounter, 1), Positio + 4, 255)
        Positio = InStr(1, OtteluNro, "'", vbTextCompare)
        OtteluNro = Left(OtteluNro, Positio - 1)
      End If

      'Generoidaan ottelun tunniste (SMnnnnn, jossa nnnnn juokseva numero)
      OtteluLaskuri = OtteluLaskuri + 1
```

```
        OtteluData.Match_id = "SM" & Right(Kausi, 4) & Format(OtteluLaskuri, "00000")

        'Täytetään kannan vakiokentät
        OtteluData.Competition_id = Kilpailu
        OtteluData.Division_id = Sarja
        OtteluData.Season = Kausi
        OtteluData.Division_id = Sarja
        OtteluData.Match_date = OtteluPvm

        'Erotetaan ottelusta koti- ja vierasjoukkue
        Apu = Split(Ottelu, "-")
        OtteluData.Home_id = Trim(Apu(LBound(Apu)))
        OtteluData.Away_id = Trim(Apu(UBound(Apu)))

        'Erotetaan ottelusta koti- ja vierasjoukkueen maalit
        Apu = Split(Tulos, "-")
        OtteluData.Home_score = Val(Trim(Apu(LBound(Apu))))
        OtteluData.Away_score = Val(Trim(Apu(UBound(Apu))))

        'Tarkistuslistan alustus
        OtteluData.Check_list = vbNullString
        Application.StatusBar = OtteluData.Match_date & " " & OtteluData.Home_id & " - " & OtteluData.Away_id

        'Määritellään ottelun kokoonpanot ja tapahtumat sisältävät sivut
        KokoonpanoHTML = Kausi & "_" & Sarja & "_kokoonpanot_" & OtteluNro & ".htm"
        TapahtumaHTML = Kausi & "_" & Sarja & "_tapahtumat_" & OtteluNro & ".htm"

        'Kirjataan lähdetiedot muistiin
        OtteluData.Source = KokoonpanoHTML & "," & TapahtumaHTML

        'Määritellään tapahtumaluettelo aina uudelleen per ottelu
        ReDim TapahtumaRivit(1 To 1000) As Tapahtuma

        'Haetaan ottelun tapahtumat ja kokoonpanot ja kirjataan ne kantaan
        HaeOtteluTiedot Kansio & KokoonpanoHTML, Kansio & TapahtumaHTML, OtteluData, TapahtumaRivit
        PuraOtteluTiedot OtteluLaskuri, OtteluData, TapahtumaRivit

    End If

  Next RowCounter

Next Counter
Application.StatusBar = False
```

```
End Sub

Sub HaeOtteluTiedot(ByVal KokoonpanoHTML As String, ByVal TapahtumaHTML As String, ByRef OtteluData As Ottelu, _
  ByRef TapahtumaRivit() As Tapahtuma)

  'Haetaan yhden ottelun tapahtumat ja kokoonpanotiedot

  Dim TableArray As Variant
  Dim Count As Long
  Dim Laskuri As Long
  Dim Laskin As Long
  Dim Positio As Long
  Dim Apu As Variant
  Dim Apu2 As Variant
  Dim HTML As String
  Dim Tuomarit As Variant
  Dim Maalivahdit As Variant
  Dim Kentällinen As Variant
  Dim Tapahtumat As Variant
  Dim Tapahtumaluettelo As String
  Dim TapahtumaLaskuri As Long
  Dim Maalit As Long
  Dim Tulos As String
  Dim TapahtumaNro As Long
  Dim Yleisömäärä As Long
  Dim Erät As String
  Dim StopCounter As Long
  Dim TaulukkoLaskuri As Long
  Dim Pelaaja As String
  Dim Seura As String
  Dim Torjunnat As String

  On Error Resume Next
  Application.ScreenUpdating = False

  'Avataan kokoonpanot sisältävä sivu
  HTML = GetTextFile(KokoonpanoHTML)

  TapahtumaNro = LBound(TapahtumaRivit) - 1

  'Taulukkoalueella 2 kotijoukkueen tiedot, taulukkoalueella 4 vierasjoukkueen

  For TaulukkoLaskuri = 2 To 4 Step 2
```

```
  TableArray = GetHTMLTable(HTML, TaulukkoLaskuri, (TaulukkoLaskuri = 2), True, False, ":", False)

  'Poimitaan kokoonpanotiedoista maalivahdit
  Maalivahdit = ArrayColumnOffset(TableArray, 0, 2, 2, True)
  'Muotoillaan tapahtumat
  For TapahtumaLaskuri = LBound(Maalivahdit) To UBound(Maalivahdit)
    TapahtumaNro = TapahtumaNro + 1
    JäsennäTapahtumat Maalivahdit(TapahtumaLaskuri), "MV", OtteluData, TapahtumaNro, "1", _
      (TaulukkoLaskuri <> 2), TapahtumaRivit
  Next TapahtumaLaskuri

  'Maalivahteja on jääkiekossa ottelupöytäkirjassa aina 2 kpl
  If (UBound(Maalivahdit) <> 1) Then
    OtteluData.Check_list = OtteluData.Check_list & ",maalivahtien määrä virheellinen!"
  End If

  'Erotellaan kenttäkokoonpanot
  For Laskuri = 1 To 4
    'Etsitään taulukosta rivi, jossa esiintyy sana "kenttä"
    Apu = ArrayLookup(TableArray, Trim(Laskuri) & ". kenttä", 0)
    If (Apu < 0) Then Exit For
    'Kentällinen = 5 pelaajaa sarakkeessa 2
    Kentällinen = ArrayColumnOffset(TableArray, Apu + 1, 2, 5, False)
    'Muotoillaan tapahtumat
    For TapahtumaLaskuri = LBound(Kentällinen) To UBound(Kentällinen, 1)
      If (Kentällinen(TapahtumaLaskuri) > vbNullString) Then
        TapahtumaNro = TapahtumaNro + 1
        JäsennäTapahtumat Kentällinen(TapahtumaLaskuri), "KE", OtteluData, TapahtumaNro, Trim(Laskuri), _
          (TaulukkoLaskuri <> 2), TapahtumaRivit
      End If
    Next TapahtumaLaskuri
    'Tieto virhelistalle, jos kentällisessä eri määrä pelaajia kuin 5
    '(4. kentällisessä toisinaan on alle 5 pelaajaa)
    If (UBound(Kentällinen) <> 4) And (Laskuri < 4) Then
      OtteluData.Check_list = OtteluData.Check_list & ",kenttäpelaajien määrä virheellinen!"
    End If
  Next Laskuri

Next TaulukkoLaskuri

'Avataan ottelutapahtumat sisältävä sivu
HTML = GetTextFile(TapahtumaHTML)
Count = GetHTMLTable(HTML, 0, True, False, True, "", False)
```

```
'Poimitaan erätiedot ensimmäisestä taulukkolohkosta
TableArray = GetHTMLTable("", 0, False, True, False, ":", False)
Erät = TableArray(1, 1)
OtteluData.Total_time = Trim(Right(Erät, 6))
Positio = InStr(1, Erät, "(", vbTextCompare)
Erät = Mid(Erät, Positio + 1, InStr(1, Erät, ")", vbTextCompare) - Positio - 1)
Apu = Split(Erät, ",")
OtteluData.Periods = Erät

'Jatkoaika ja rangaistuslaukauskilpailu päätellään erien lukumäärästä
OtteluData.Extra_time = (UBound(Apu) >= 3)
OtteluData.Penalties = (UBound(Apu) = 4)

'Tarkistetaan erien maalimäärät (vastattava ottelun lopputulosta)
Maalit = 0
For Laskuri = LBound(Apu) To UBound(Apu)
  Apu2 = Split(Apu(Laskuri), "-")
  Maalit = Maalit + Val(Trim(Apu2(LBound(Apu2)))) + Val(Trim(Apu2(UBound(Apu2))))
Next Laskuri
If (Maalit <> OtteluData.Home_score + OtteluData.Away_score) Then
  OtteluData.Check_list = OtteluData.Check_list & ",erälukemat virheelliset!"
End If

'Eriä voi liigan runkosarjassa olla 3-5 (normaalien lisäksi
'jatkoaika ja rangaistuslaukauskilpailu)
If (UBound(Apu) < 2) Or (UBound(Apu) > 4) Then
  OtteluData.Check_list = OtteluData.Check_list & ",erien määrä virheellinen!"
End If

'Kerätään tapahtumarivit ensin listaan
Tapahtumaluettelo = vbNullString
StopCounter = Count
For Laskuri = 0 To Count - 1
  TableArray = GetHTMLTable("", Laskuri, False, True, False, ":", False)
  For Laskin = LBound(TableArray, 1) To UBound(TableArray, 1)
    'Samalla kirjataan muistiin rivi, jossa varsinaiset tapahtumat loppuvat
    If (UCase(Trim(TableArray(Laskin, 0))) = "YHTEENVETO") Then
      StopCounter = Laskuri
      Exit For
    End If
    'Tapahtumiksi lasketaan vain ne rivit, joilla kellonaika
    If (Right(Trim(TableArray(Laskin, 0)), 3) >= ".00") And _
      (Right(Trim(TableArray(Laskin, 0)), 3) <= ".59") Then
```

```
        Tapahtumaluettelo = Tapahtumaluettelo & Chr(255) & TableArray(Laskin, 0) & "/" & TableArray(Laskin, 1)
    End If
  Next Laskin
  If (Laskuri >= StopCounter) Then Exit For
Next Laskuri

'Muotoillaan tapahtumat
If (Tapahtumaluettelo <> vbNullString) Then
  Tapahtumat = Split(Mid(Tapahtumaluettelo, 2, Len(Tapahtumaluettelo)), Chr(255))
  For TapahtumaLaskuri = LBound(Tapahtumat) To UBound(Tapahtumat)
    TapahtumaNro = TapahtumaNro + 1
    JäsennäTapahtumat Tapahtumat(TapahtumaLaskuri), "TA", OtteluData, TapahtumaNro, "", False, TapahtumaRivit
  Next TapahtumaLaskuri
End If

'Maalivahtien torjunnat haetaan yhteenveto-osasta tapahtumien jälkeen
Apu = -1
Tapahtumaluettelo = vbNullString
For Laskuri = StopCounter - 1 To Count - 1
  TableArray = GetHTMLTable("", Laskuri, False, True, False, ":", False)
  If (Apu > -1) Then
    For TapahtumaLaskuri = LBound(TableArray, 1) To UBound(TableArray, 1)
      If (VarType(TableArray(TapahtumaLaskuri, 1)) <> vbBoolean) Then
        If (VarType(TableArray(TapahtumaLaskuri, 0)) <> vbBoolean) Then
          Seura = TableArray(TapahtumaLaskuri, 0)
          'Erotellaan pelaaja ja torjuntalukemat '+'-merkin perusteella
          Positio = InStr(1, TableArray(TapahtumaLaskuri, 1), "+", vbTextCompare)
          Pelaaja = Trim(Left(TableArray(TapahtumaLaskuri, 1), Positio - 3))
          'Vain torjuntojen yhteissumma huomioidaan
          Torjunnat = Trim(Mid(TableArray(TapahtumaLaskuri, 1), Positio - 2, _
            Len(TableArray(TapahtumaLaskuri, 1))))
          Positio = InStr(1, Torjunnat, "=", vbTextCompare)
          Tapahtumaluettelo = Tapahtumaluettelo & Chr(255) & Seura & "/" & Pelaaja & "/" & "S" & "/" & _
            Mid(Torjunnat, Positio + 1, Len(Torjunnat)) & "/" & Left(Torjunnat, Positio - 1)
        End If
        'Maalivahdin vaihto
        If (VarType(TableArray(TapahtumaLaskuri, 0)) = vbBoolean) Then
          If (UCase(Left(TableArray(TapahtumaLaskuri, 1), 7)) = "(VAIHTO") Then
            Tapahtumaluettelo = Tapahtumaluettelo & Chr(255) & Seura & "/" & Pelaaja & "/" & "L-" & "/" & _
              Replace(Mid(TableArray(TapahtumaLaskuri, 1), 9, 5), "/", "")
            Positio = InStr(1, TableArray(TapahtumaLaskuri + 1, 1), "+", vbTextCompare)
            Pelaaja = Trim(Left(TableArray(TapahtumaLaskuri + 1, 1), Positio - 2))
            Tapahtumaluettelo = Tapahtumaluettelo & Chr(255) & Seura & "/" & Pelaaja & "/" & "L+" & "/" & _
              Replace(Mid(TableArray(TapahtumaLaskuri, 1), 9, 5), "/", "")
```

```
            'Ristiintarkistus: kirjataan tarkistuslistalle sellainen teoreettinen
            'tilanne, että "vaihto"-merkinnän jäljessä on "pois"-merkintä
            If (UCase(Left(TableArray(TapahtumaLaskuri - 1, 1), 5)) = "(POIS") Or _
              (UCase(Left(TableArray(TapahtumaLaskuri + 1, 1), 5)) = "(POIS") Then
                OtteluData.Check_list = OtteluData.Check_list & ",Maalivahdeissa epäselvyyksiä!"
            End If
          End If
          'Poissaolotieto kirjataan eräkohtaisten torjuntalukemien lisäksi info-sarakkeeseen
          If (UCase(Left(TableArray(TapahtumaLaskuri, 1), 5)) = "(POIS") Then
            Tapahtumaluettelo = Tapahtumaluettelo & " " & TableArray(TapahtumaLaskuri, 1)
          End If
        End If
      Else
        Exit For
      End If
    Next TapahtumaLaskuri
  End If
  'Otsikko "Maalivahdit" on aina edellisen taulukkoalueen lopussa
  Apu = ArrayLookup(TableArray, "Maalivahdit", 0)
Next Laskuri

'Muotoillaan tapahtumat
If (Tapahtumaluettelo <> vbNullString) Then
  Tapahtumat = Split(Mid(Tapahtumaluettelo, 2, Len(Tapahtumaluettelo)), Chr(255))
  For TapahtumaLaskuri = LBound(Tapahtumat) To UBound(Tapahtumat)
    TapahtumaNro = TapahtumaNro + 1
    JäsennäTapahtumat Tapahtumat(TapahtumaLaskuri), "TO", OtteluData, TapahtumaNro, "", False, TapahtumaRivit
  Next TapahtumaLaskuri
End If

'Etsitään ottelun yleisömäärä sivun yhteenveto-osasta
Apu = -1
OtteluData.Attendance = 0
For Laskuri = StopCounter To Count - 1
  TableArray = GetHTMLTable("", Laskuri, False, True, False, ":", False)
  If (Apu > -1) Then
    OtteluData.Attendance = Val(Trim(TableArray(0, 1)))
    Exit For
  End If
  'Otsikko "Yleisöä" on aina edellisen taulukkoalueen lopussa
  Apu = ArrayLookup(TableArray, "Yleisöä", 0)
Next Laskuri

'Etsitään ottelun tuomaritiedot
```

```
Apu = -1
For Laskuri = StopCounter - 1 To Count - 1
  TableArray = GetHTMLTable("", Laskuri, False, True, False, ":", False)
  If (Apu > -1) Then
    Tuomarit = ArrayColumnOffset(TableArray, 0, 2, 4, False)
    OtteluData.Referee1 = vbNullString
    OtteluData.Referee2 = vbNullString
    OtteluData.Other_referees = vbNullString
    For Laskin = LBound(Tuomarit) To UBound(Tuomarit)
      If (VarType(Tuomarit(Laskin)) <> vbBoolean) Then
        Apu2 = Split(Tuomarit(Laskin), ":")
        If (InStr(1, Tuomarit(Laskin), "Päätuomari", vbTextCompare) > 0) Then
          'Käännetään tuomarien nimet sukunimi-etunimi -muotoon
          If (Trim(OtteluData.Referee1) = vbNullString) Then
            OtteluData.Referee1 = Swapnames(Apu2(LBound(Apu2)))
          Else
            OtteluData.Referee2 = Swapnames(Apu2(LBound(Apu2)))
          End If
        Else
          OtteluData.Other_referees = OtteluData.Other_referees & "," & Swapnames(Apu2(LBound(Apu2)))
        End If
      End If
    Next Laskin
    OtteluData.Other_referees = Mid(OtteluData.Other_referees, 2, Len(OtteluData.Other_referees))
    Exit For
  End If
  'Otsikko "Tuomarit" on aina edellisen taulukkoalueen lopussa
  Apu = ArrayLookup(TableArray, "Tuomarit", 0)
Next Laskuri

'Tarkistetaan vielä, että tilanneluvuista syntyy ottelun lopputulos
Tulos = "0-0"
Maalit = 0
For TapahtumaLaskuri = LBound(TapahtumaRivit) To UBound(TapahtumaRivit)
  If (TapahtumaRivit(TapahtumaLaskuri).Detail_type_id = "G") Then
    If (TapahtumaRivit(TapahtumaLaskuri).Score > vbNullString) Then
      Tulos = TapahtumaRivit(TapahtumaLaskuri).Score
      Maalit = Maalit + 1
      If ((TapahtumaRivit(TapahtumaLaskuri).Team_id) = vbNullString) Or _
        ((TapahtumaRivit(TapahtumaLaskuri).Person_id) = vbNullString) Then
          OtteluData.Check_list = OtteluData.Check_list & ",Tapahtuma virheellinen!"
      End If
    End If
  End If
```

```
  Next TapahtumaLaskuri
  If (Tulos <> Trim(OtteluData.Home_score) & "-" & Trim(OtteluData.Away_score)) Or _
    (Maalit <> OtteluData.Home_score + OtteluData.Away_score) Then
      OtteluData.Check_list = OtteluData.Check_list & ",Maalimäärä virheellinen!"
  End If

End Sub
```

```
Sub JäsennäTapahtumat(ByVal TapahtumaMerkintä As String, ByVal Tapahtumalaji As String, _
  ByRef OtteluData As Ottelu, ByRef Order_nr As Long, ByVal Info As String, ByVal AwayTeam As Boolean, _
  ByRef TapahtumaLista() As Tapahtuma)

  'Jäsennetään tapahtumarivi kerätyistä tietoalkioista kantaa vastaavaan muotoon
  'Lajikohtaista tapahtumien käsittelyä, kun kutsuva taso on esijäsentänyt
  'tapahtumarivin siten, että se sisältää tapahtumaan kuuluvat tiedot

  Dim Apu As Variant
  Dim Apu2 As String
  Dim Apu3 As Variant
  Dim Apu4 As Variant
  Dim Apu5 As Variant
  Dim Laskuri As Long
  Dim Laskin As Long
  Dim Positio As Long
  Static Result As String
  Dim Tulos As String
  Dim Kotimaali As Boolean
  Dim Pois As String

  On Error Resume Next
  Application.ScreenUpdating = False

  TapahtumaLista(Order_nr).Match_id = OtteluData.Match_id
  TapahtumaLista(Order_nr).Order_nr = Order_nr
  TapahtumaLista(Order_nr).Detail_subtype_id = vbNullString

  If (TapahtumaLista(Order_nr).Order_nr = 1) Then Result = "0-0"

  Select Case Tapahtumalaji
    'Maalivahti kokoonpanossa
    Case "MV": Apu = Split(TapahtumaMerkintä, ":")
               TapahtumaLista(Order_nr).Person_id = Swapnames(Trim(Apu(LBound(Apu))))
               TapahtumaLista(Order_nr).Detail_type_id = "L"
```

```
            If (UCase(Trim(Apu(UBound(Apu)))) = "AK") Then
              TapahtumaLista(Order_nr).Info = "MV"
              TapahtumaLista(Order_nr).Detail_subtype_id = "GK"
            Else
              TapahtumaLista(Order_nr).Info = "VMV"
            End If
            If (AwayTeam = False) Then
              TapahtumaLista(Order_nr).Team_id = OtteluData.Home_id
            Else
              TapahtumaLista(Order_nr).Team_id = OtteluData.Away_id
            End If
  'Kenttäpelaaja kokoonpanossa
  Case "KE": Apu = Split(TapahtumaMerkintä, ":")
            TapahtumaLista(Order_nr).Person_id = Swapnames(Apu(LBound(Apu)))
            TapahtumaLista(Order_nr).Detail_type_id = "L"
            Apu2 = Trim(Replace(Apu(LBound(Apu) + 1), ",", ""))
            TapahtumaLista(Order_nr).Info = Info
            If (AwayTeam = False) Then
              TapahtumaLista(Order_nr).Team_id = OtteluData.Home_id
            Else
              TapahtumaLista(Order_nr).Team_id = OtteluData.Away_id
            End If
            If (UCase(Trim(Left(Apu2, 5))) = "OIKEA") Then _
              TapahtumaLista(Order_nr).Info = TapahtumaLista(Order_nr).Info & "/" & "O"
            If (UCase(Trim(Left(Apu2, 5))) = "VASEN") Then _
              TapahtumaLista(Order_nr).Info = TapahtumaLista(Order_nr).Info & "/" & "V"
            If (UCase(Trim(Left(Apu2, 6))) = "KESKUS") Then _
              TapahtumaLista(Order_nr).Info = TapahtumaLista(Order_nr).Info & "/" & "K"
            If (UCase(Trim(Right(Apu2, 10))) = "PUOLUSTAJA") Then _
              TapahtumaLista(Order_nr).Info = TapahtumaLista(Order_nr).Info & "P"
            If (UCase(Trim(Right(Apu2, 9))) = "HYÖKKÄÄJÄ") Then
              Apu2 = UCase(Trim(Right(TapahtumaLista(Order_nr).Info, 2)))
              If (Apu2 = "/O") Or (Apu2 = "/V") Then
                TapahtumaLista(Order_nr).Info = TapahtumaLista(Order_nr).Info & "L"
              Else
                If (Apu2 = "/K") Then
                  TapahtumaLista(Order_nr).Info = TapahtumaLista(Order_nr).Info & "H"
                Else
                  TapahtumaLista(Order_nr).Info = TapahtumaLista(Order_nr).Info & "/H"
                End If
              End If
            End If
  'Varsinaiset tapahtumat
  Case "TA": Apu = Split(TapahtumaMerkintä, "/")
```

```
            TapahtumaLista(Order_nr).Time = Apu(LBound(Apu))
            Apu2 = Apu(LBound(Apu) + 1)
            Apu3 = Split(Apu2, ":")
            Tulos = Apu3(LBound(Apu3) + 1)
            Positio = InStr(1, Tulos, "-", vbTextCompare)
            If (Positio > 0) Then
              If Not ((IsNumeric(Mid(Tulos, Positio - 1, 1))) And _
                (IsNumeric(Mid(Tulos, Positio + 1, 1)))) Then Positio = 0
            End If
            If (Positio > 0) Then
              'Maali, jos tapahtumassa esiintyy tuloslukema
              TapahtumaLista(Order_nr).Detail_type_id = "G"
              TapahtumaLista(Order_nr).Person_id = Swapnames(Trim(Apu3(LBound(Apu3))))
              Apu4 = Split(Result, "-")
              Result = Trim(Mid(Tulos, Positio - 2, 5))
              Apu5 = Split(Result, "-")
              Kotimaali = (Apu4(LBound(Apu4)) <> Apu5(LBound(Apu5)))
              If (Kotimaali) Then
                TapahtumaLista(Order_nr).Team_id = OtteluData.Home_id
              Else
                TapahtumaLista(Order_nr).Team_id = OtteluData.Away_id
              End If
              TapahtumaLista(Order_nr).Score = Result
              If (OtteluData.Extra_time) Then OtteluData.Total_time = TapahtumaLista(Order_nr).Time
              Info = vbNullString
              Info = Trim(Mid(Tulos, Positio + 2, Len(Tulos)))
              If (Left(Info, 1) = ",") Then Info = Mid(Info, 2, Len(Info))
              TapahtumaLista(Order_nr).Info = Info
              If (InStr(1, Info, "RL", vbTextCompare) > 0) Then _
                TapahtumaLista(Order_nr).Detail_subtype_id = "PEN"
              If (OtteluData.Penalties) And (Result = Trim(OtteluData.Home_score) & "-" & _
                Trim(OtteluData.Away_score)) Then TapahtumaLista(Order_nr).Detail_subtype_id = "X"
              Positio = InStr(1, Apu2, "(", vbTextCompare)
              'Syöttö, jos maalitapahtumassa esiintyy sulut
              If (Positio > 0) Then
                Apu4 = Split(Replace(Mid(Apu2, Positio + 1, InStr(1, Apu2, ")", vbTextCompare) - _
                  Positio - 1), ":", ""), ",")
                Info = vbNullString
                Info = Trim(Mid(Apu2, InStr(1, Apu2, ")", vbTextCompare) + 1, Len(Apu2)))
                If (Left(Info, 1) = ",") Then Info = Mid(Info, 2, Len(Info))
                TapahtumaLista(Order_nr).Info = Info
                For Laskin = LBound(Apu4) To UBound(Apu4)
                  Order_nr = Order_nr + 1
                  TapahtumaLista(Order_nr).Match_id = OtteluData.Match_id
```

```
                    TapahtumaLista(Order_nr).Order_nr = Order_nr
                    TapahtumaLista(Order_nr).Detail_type_id = "A"
                    TapahtumaLista(Order_nr).Person_id = Swapnames(Trim(Apu4(Laskin)))
                    TapahtumaLista(Order_nr).Time = Apu(LBound(Apu))
                    TapahtumaLista(Order_nr).Info = Info
                    If (Kotimaali) Then
                      TapahtumaLista(Order_nr).Team_id = OtteluData.Home_id
                    Else
                      TapahtumaLista(Order_nr).Team_id = OtteluData.Away_id
                    End If
                    TapahtumaLista(Order_nr).Score = Result
                   Next Laskin
                End If
              Else
                'Muussa tapauksessa jäähy
                TapahtumaLista(Order_nr).Team_id = Trim(Apu3(LBound(Apu3)))
                TapahtumaLista(Order_nr).Person_id = Swapnames(Trim(Apu3(LBound(Apu3) + 1)))
                TapahtumaLista(Order_nr).Detail_type_id = "P"
                TapahtumaLista(Order_nr).Info = Trim(Apu3(LBound(Apu3) + 2))
                Apu2 = Right(TapahtumaLista(Order_nr).Info, 5)
                TapahtumaLista(Order_nr).Info = Trim(Left(TapahtumaLista(Order_nr).Info, _
                  Len(TapahtumaLista(Order_nr).Info) - 5))
                'Mahdollistetaan joukkuerangaistuksen erottelu muitsa jäähyistä
                If (UCase(Trim(TapahtumaLista(Order_nr).Info)) = "LIIAN MONTA PELAAJAA JÄÄLLÄ") Then
                  TapahtumaLista(Order_nr).Detail_subtype_id = "TP"
                End If
                TapahtumaLista(Order_nr).Duration = Trim(Left(Apu2, 2)) & ".00"
              End If
    'Maalivahtien torjunnat
    Case "TO": Apu = Split(TapahtumaMerkintä, "/")
              TapahtumaLista(Order_nr).Team_id = Trim(Apu(LBound(Apu)))
              TapahtumaLista(Order_nr).Person_id = Swapnames(Trim(Apu(LBound(Apu) + 1)))
              TapahtumaLista(Order_nr).Detail_type_id = Trim(Apu(LBound(Apu) + 2))
              If (TapahtumaLista(Order_nr).Detail_type_id = "S") Then
                TapahtumaLista(Order_nr).Amount = Val(Trim(Trim(Apu(LBound(Apu) + 3))))
                TapahtumaLista(Order_nr).Info = Trim(Trim(Apu(LBound(Apu) + 4)))
                TapahtumaLista(Order_nr).Time = OtteluData.Total_time
                Positio = InStr(1, TapahtumaLista(Order_nr).Info, "(pois", vbTextCompare)
                If (Positio > 0) Then
                  Pois = Replace(Trim(Mid(TapahtumaLista(Order_nr).Info, Positio + 5, _
                    Len(TapahtumaLista(Order_nr).Info))), ")", "")
                  Apu = Split(Pois, " ja ")
                  'Generoidaan maalivahdin poissaoloista erilliset
                  'tapahtumarivit (GK- ja GK+)
```

```
                  For Laskuri = LBound(Apu) To UBound(Apu)
                    Apu3 = Split(Apu(Laskuri), "-")
                    Order_nr = Order_nr + 1
                    TapahtumaLista(Order_nr).Order_nr = Order_nr
                    TapahtumaLista(Order_nr).Match_id = OtteluData.Match_id
                    TapahtumaLista(Order_nr).Detail_type_id = "GK-"
                    TapahtumaLista(Order_nr).Team_id = TapahtumaLista(Order_nr - 1).Team_id
                    TapahtumaLista(Order_nr).Person_id = TapahtumaLista(Order_nr - 1).Person_id
                    TapahtumaLista(Order_nr).Time = Apu3(LBound(Apu3))
                    If (Trim(Apu3(UBound(Apu3))) <> Trim(OtteluData.Total_time)) Then
                      Order_nr = Order_nr + 1
                      TapahtumaLista(Order_nr).Order_nr = Order_nr
                      TapahtumaLista(Order_nr).Match_id = OtteluData.Match_id
                      TapahtumaLista(Order_nr).Detail_type_id = "GK+"
                      TapahtumaLista(Order_nr).Team_id = TapahtumaLista(Order_nr - 1).Team_id
                      TapahtumaLista(Order_nr).Person_id = TapahtumaLista(Order_nr - 1).Person_id
                      TapahtumaLista(Order_nr).Time = Apu3(UBound(Apu3))
                    End If
                  Next Laskuri
                End If
              Else
                TapahtumaLista(Order_nr).Time = Trim(Apu(LBound(Apu) + 3))
              End If
    End Select

End Sub
```

```
Sub PuraOtteluTiedot(ByVal TilastoRivi As Long, OtteluData As Ottelu, ByRef TapahtumaRivit() As Tapahtuma)

  'Ottelun tietojen purku Excel-taulukkoon/tietokantaan
  '(tässä tietoa muokataan enää minimaalisesti)

  Dim TapahtumaLaskuri As Long
  Static TapahtumaLkm As Long

  On Error Resume Next
  Application.ScreenUpdating = False

  If (TilastoRivi <= 1) Then TapahtumaLkm = 1

  With ThisWorkbook.Worksheets("Matches")
    .Cells(TilastoRivi + 1, .Range("Match_id").Column).Value = OtteluData.Match_id
    .Cells(TilastoRivi + 1, .Range("Competition_id").Column).Value = OtteluData.Competition_id
```

```
    .Cells(TilastoRivi + 1, .Range("Division_id").Column).Value = OtteluData.Division_id
    .Cells(TilastoRivi + 1, .Range("Season").Column).Value = OtteluData.Season
    .Cells(TilastoRivi + 1, .Range("Match_date").Column).Value = DateValue(OtteluData.Match_date)
    .Cells(TilastoRivi + 1, .Range("Home_id").Column).Value = OtteluData.Home_id
    .Cells(TilastoRivi + 1, .Range("Away_id").Column).Value = OtteluData.Away_id
    .Cells(TilastoRivi + 1, .Range("Home_score").Column).Value = OtteluData.Home_score
    .Cells(TilastoRivi + 1, .Range("Away_score").Column).Value = OtteluData.Away_score
    If (OtteluData.Extra_time) Then .Cells(TilastoRivi + 1, .Range("Extra_time").Column).Value = "*"
    If (OtteluData.Penalties) Then .Cells(TilastoRivi + 1, .Range("Penalties").Column).Value = "*"
    .Cells(TilastoRivi + 1, .Range("Total_time").Column).Value = "'" & OtteluData.Total_time
    .Cells(TilastoRivi + 1, .Range("Periods").Column).Value = OtteluData.Periods
    .Cells(TilastoRivi + 1, .Range("Referee1").Column).Value = OtteluData.Referee1
    .Cells(TilastoRivi + 1, .Range("Referee2").Column).Value = OtteluData.Referee2
    .Cells(TilastoRivi + 1, .Range("Other_referees").Column).Value = OtteluData.Other_referees
    .Cells(TilastoRivi + 1, .Range("Attendance").Column).Value = OtteluData.Attendance
    .Cells(TilastoRivi + 1, .Range("Source").Column).Value = OtteluData.Source
    If (OtteluData.Check_list > vbNullString) Then
      .Cells(TilastoRivi + 1, .Range("Check_list").Column).Value = Mid(OtteluData.Check_list, 2, _
        Len(OtteluData.Check_list))
    End If
  End With

  With ThisWorkbook.Worksheets("Match_details")
    For TapahtumaLaskuri = LBound(TapahtumaRivit) To UBound(TapahtumaRivit)
      If (TapahtumaRivit(TapahtumaLaskuri).Match_id = vbNullString) Then Exit For
      TapahtumaLkm = TapahtumaLkm + 1
      .Cells(TapahtumaLkm, .Range("Match_id").Column).Value = TapahtumaRivit(TapahtumaLaskuri).Match_id
      .Cells(TapahtumaLkm, .Range("Person_id").Column).Value = TapahtumaRivit(TapahtumaLaskuri).Person_id
      .Cells(TapahtumaLkm, .Range("Team_id").Column).Value = TapahtumaRivit(TapahtumaLaskuri).Team_id
      .Cells(TapahtumaLkm, .Range("Order_nr").Column).Value = TapahtumaRivit(TapahtumaLaskuri).Order_nr
      .Cells(TapahtumaLkm, .Range("Detail_type_id").Column).Value = _
        TapahtumaRivit(TapahtumaLaskuri).Detail_type_id
      If (Trim(TapahtumaRivit(TapahtumaLaskuri).Time) > "") Then
        If (Len(Trim(TapahtumaRivit(TapahtumaLaskuri).Time)) < 5) Then _
          TapahtumaRivit(TapahtumaLaskuri).Time = "0" & TapahtumaRivit(TapahtumaLaskuri).Time
      End If
      .Cells(TapahtumaLkm, .Range("Time").Column).Value = "'" & TapahtumaRivit(TapahtumaLaskuri).Time
      If (TapahtumaRivit(TapahtumaLaskuri).Amount > 0) Or _
        (TapahtumaRivit(TapahtumaLaskuri).Detail_type_id = "S") Then
          .Cells(TapahtumaLkm, .Range("Amount").Column).Value = TapahtumaRivit(TapahtumaLaskuri).Amount
      End If
      If (Trim(TapahtumaRivit(TapahtumaLaskuri).Duration) > "") Then
        If (Len(Trim(TapahtumaRivit(TapahtumaLaskuri).Duration)) < 5) Then _
          TapahtumaRivit(TapahtumaLaskuri).Duration = "0" & TapahtumaRivit(TapahtumaLaskuri).Duration
```

```
      End If
      .Cells(TapahtumaLkm, .Range("Duration").Column).Value = "'" & TapahtumaRivit(TapahtumaLaskuri).Duration
      .Cells(TapahtumaLkm, .Range("Detail_subtype_id").Column).Value = _
        TapahtumaRivit(TapahtumaLaskuri).Detail_subtype_id
      .Cells(TapahtumaLkm, .Range("Score").Column).Value = TapahtumaRivit(TapahtumaLaskuri).Score
      .Cells(TapahtumaLkm, .Range("Info").Column).Value = TapahtumaRivit(TapahtumaLaskuri).Info
    Next TapahtumaLaskuri
  End With

End Sub
```