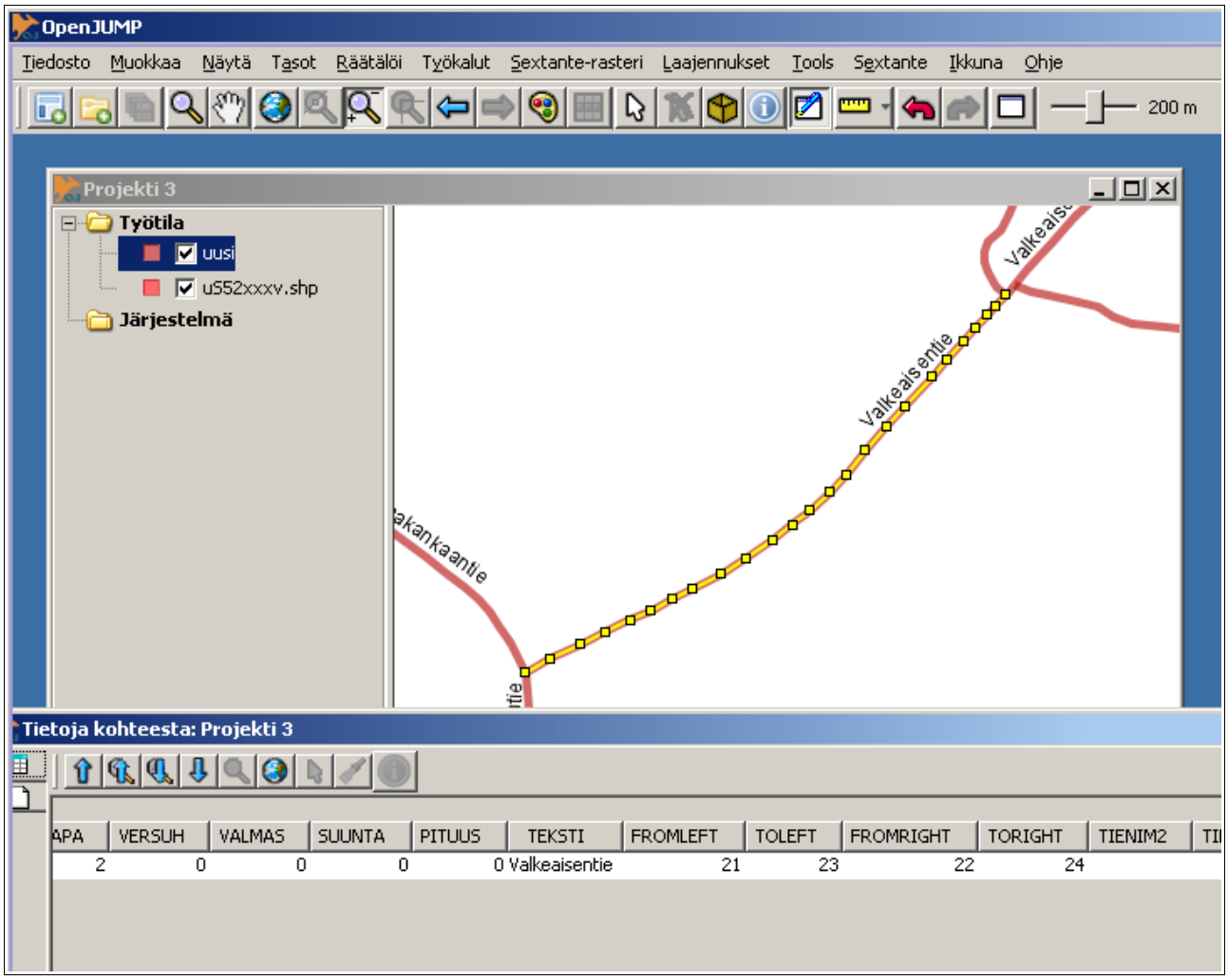
Maastotietokannan tiestö osoitteilla

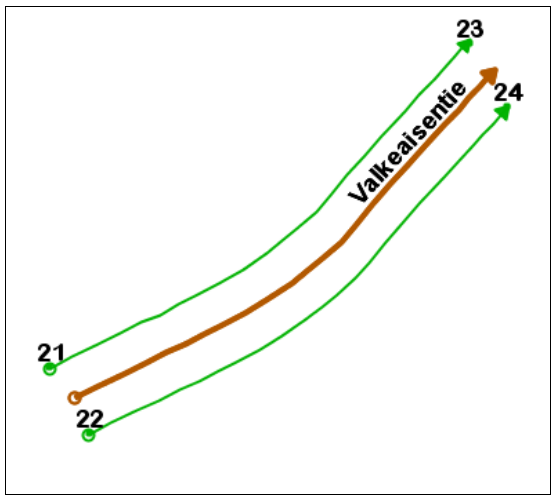
Osoitteiden löytäminen tieaineistossa ja niiden irrottaminen omaksi pisteaineistokseen

Jukka Rahkonen, http://latuviitta.org
Viimeksi muutettu 5. heinäkuuta 2012

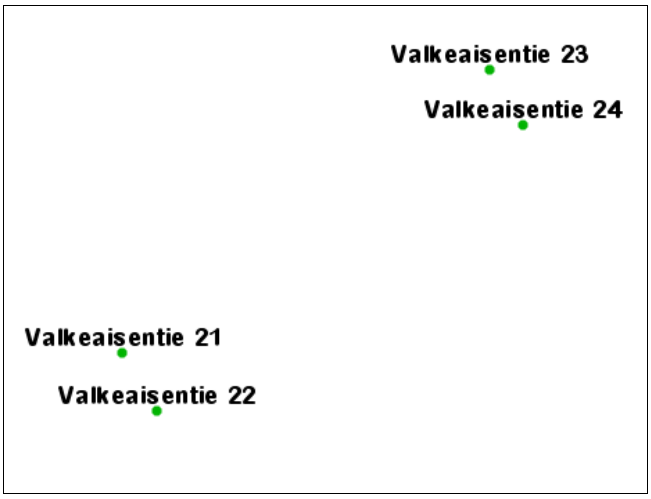
Osoitteiden löytäminen aineistosta



”Maastotietokannan tiestö osoitteilla” on viivamainen aineisto, jossa osoitteet on annettu tieosuuksien ominaisuustietoihin upotettuina. Esimerkiksi ylhäällä olevassa kuvassa valitun tien nimi on Valkeaisentie, ja sen osoitenumerot löytyvät kentistä ”fromleft”, ”fromright”, ”toleft” ja ”toright”. Kenttien arvot merkitsevät tieosuuden alku- ja loppupisteiden vasemmalla ja oikealla puolella olevia osoitenumeroita. Seuraava kuva havainnollistaa tilannetta.



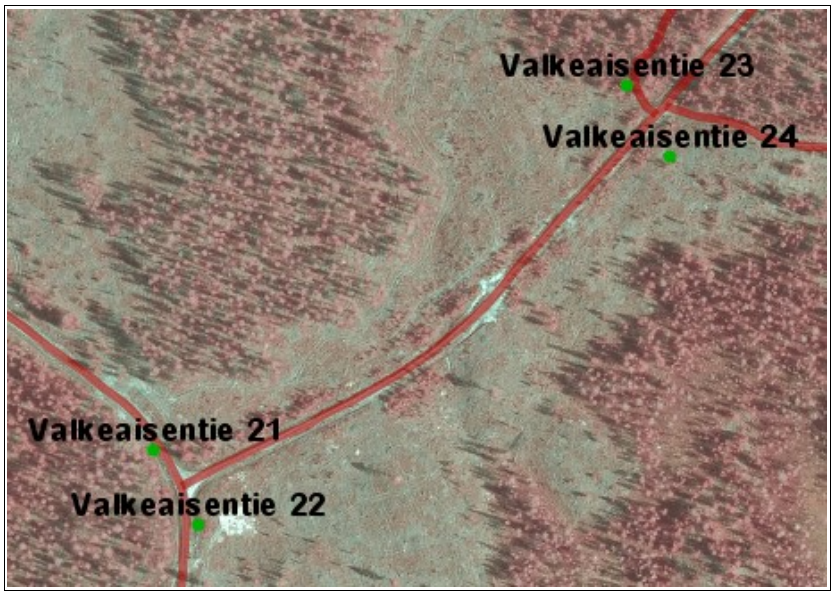
Nuolet osoittavan tieosuuden kulkusuunnan aineistossa. Osuuden alkupisteen vasemmalla puolella on ”fromleft”-osoite ja loppupisteen vasemmalla puolella ”toleft”-osoite. ”Fromright”- ja ”toright”-osoitteet löytyvät vastaavasti tieosuuden oikealta puolelta.



Osoitteiden sitomisessa viivan alku- ja loppupisteisiin on tiettyjä etuja, kuten esimerkiksi mahdollisuus selvittää väliosoitteet laskemalla. Jos alkupisteen oikean puolen osoitenumero on 2 ja loppupisteen osoitenumero 10, niin osoitenumero 6 löytyy tieosuuden puolesta välistä. Osoitteiden etsimiseen tieosuuksien alku- ja päätepisteistä tarvitaan kuitenkin jokin etevä paikkatieto-ohjelma. Väliosoitteiden laskemiseksi ohjelman olisi lisäksi hallittava niin sanottu lineaarinen referointi eli etäisyyksien mittaaminen viivaa seuraten. Tällaiset ohjelmat ovat harvinaisia, ja usein osoitetietoja olisi helpompi käyttää, jos ne olisivat saatavilla yksinkertaisina osoitepisteinä kuten yllä olevassa kuvassa. Pisteaineiston perusteella olisi yksinkertaista etsiä vastaus kysymyksiin:

- Missä on Valkeaisentie 23?
- Mikä on kartalta osoitettua kohtaa tai numeroina annettuja koordinaatteja lähimpänä oleva osoite?

Erillistä pistemäistä aineistoa voitaisiin myös helposti käyttää muiden kartta-aineistojen höystämiseen, kuten vaikkapa osoitetietojen lisäämiseen ortoilmakuville.



Pisteaineiston luominen GDAL- ja Spatialite-ohjelmilla

Vaihe 1: Aineiston vieminen Spatialite-tietokantaan GDAL-apuohjelmalla ”ogr2ogr”

```
REM komentojono, jolla viedään shapefilet yksi kerrallaan Spatialiteen
REM shapefilet käyttävät (ainakin melkein) merkistökoodausta ISO-8859-1
REM seuraavan ympäristömuuttujan asettaminen jättää vähemmän sattuman varaan

SET SHAPE_ENCODING=ISO-8859-1
ogr2ogr -f SQLite -dsco spatialite=yes -dsco init_with_epsg=yes -a_srs epsg:3067
mtk_tos.sqlite  -nlt linestring -nln mtk_tos_viiva -gt 20000  uK34xxxv.shp

ogr2ogr -append -f SQLite  -a_srs epsg:3067 mtk_tos.sqlite  -nlt linestring -nln
mtk_tos_viiva -gt 20000  uK42xxxv.shp

ogr2ogr -append -f SQLite  -a_srs epsg:3067 mtk_tos.sqlite  -nlt linestring -nln
mtk_tos_viiva -gt 20000  uL23xxxv.shp
......
```

Huomautuksia muutamista parametreistä:

SHAPE_ENCODING-muuttuja kannattaa asettaa. Saamenkielisiä nimiä ei ole koodattu aivan ISO-8859-1 mukaan, mutta suomen- ja ruotsinkieliset nimet siirtyvät tällä asetuksella oikein.

-a_srs epsg:3067

> Käsketään ogr2ogr nimenomaan kertoa Spatialitelle, että aineiston koordinaattijärjestelmä on EPSG:3067

-gt 20000

> Kannattaa ehdottomasti antaa, koska se tekee muunnoksen paljon nopeammaksi. Oletuksena tietokantaan tallennetaan tietoa vain 200 tietueen suuruisina erinä.

Vaihe 2: Vasemman- ja oikeanpuoleisten kopioiden tekeminen osoitteellisista tieviivoista

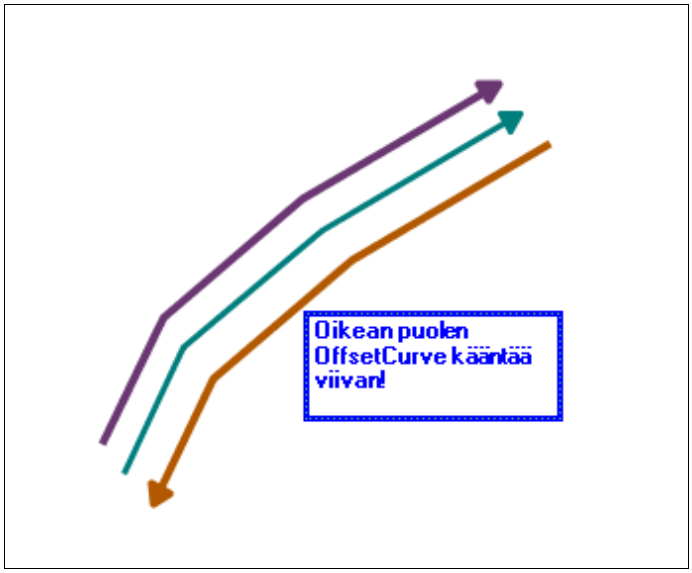
Spatialite-funktiolla **ST_OffsetCurve** voidaan luoda tieviivoista kopiot. ST_OffsetCurve(GEOMETRY,15,1) tekee kopion 15 metriä alkupäisen viivan vasemmalle puolelle, ST_OffsetCurve(GEOMETRY,15,0) vastaavasti oikealle puolelle. Koska meitä kiinnostaa pelkästään osoitteet, niin seuraavalla kyselyllä tehdään kopiot vain niistä viivoista, joiden siirtymän puolen aloituspisteellä on osoitetietoja. Komennot voidaan antaa Spatialite-GUI -ohjelman SQL-ikkunassa.

```
create table tos_left as
SELECT "OGC_FID" AS "OGC_FID", ST_OffsetCurve(GEOMETRY,15,1) AS "GEOMETRY",
    "teksti" AS "katuos_fi", "fromleft" AS "fromleft", "toleft" AS "toleft",
    "fromright" AS "fromright", "toright" AS "toright", "tienim2" as
"katuos_sv", "tienim3" as "katuos_se"
FROM "mtk_tos_viiva"
where fromleft is not null
```

```
create table tos_right as
SELECT "OGC_FID" AS "OGC_FID", ST_OffsetCurve(GEOMETRY,15,0) AS "GEOMETRY",
   "teksti" AS "katuos_fi",  "fromleft" AS "fromleft", "toleft" AS "toleft",
    "fromright" AS "fromright", "toright" AS "toright","tienim2" as "katuos_sv",
"tienim3" as "katuos_se"
FROM "mtk_tos_viiva"
where fromright is not null
```

Vaihe 3: Viivojen alku- ja loppupisteiden irrottaminen osoitepisteiksi

Viimeistään tässä vaiheessa on hyvä tietää, että ST_OffsetCurve kääntää viivan kulkusuunnan silloin, kun käytetään oikeanpuoleista siirtoarvoa. Jos tätä ei huomaa, niin jatkossa saadaan kummallisia tuloksia, kun liitetään osoitenumeroita ominaisuustiedoista viivojen alku- ja loppupisteisiin.



Aloitetaan osoitepisteiden luominen irrottamalla alku- ja loppupisteet edellisessä vaiheessa luoduista vasemmalle ja oikealle siirretyistä osoitteellisista tieosuuksista. Tämä onnistuu Spatialite-funktioilla **StartPoint** ja **EndPoint**. Samalla SQL-kyselyllä voidaan lisäksi poimia pisteille sopiva määrä myöhemmin tarvittavia ominaisuustietoja.

```
create table tos_left_start as
SELECT "OGC_FID" AS "OGC_FID", StartPoint(GEOMETRY) AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "fromleft" AS "fromleft",
    "toleft" AS "toleft", "fromright" AS "fromright",
    "toright" AS "toright", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se"
FROM "tos_left";
```

```
create table tos_left_end as
SELECT "OGC_FID" AS "OGC_FID", EndPoint(GEOMETRY) AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "fromleft" AS "fromleft",
    "toleft" AS "toleft", "fromright" AS "fromright",
    "toright" AS "toright", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se"
FROM "tos_left";
```

```
create table tos_right_start as
SELECT "OGC_FID" AS "OGC_FID", StartPoint(GEOMETRY) AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "fromleft" AS "fromleft",
    "toleft" AS "toleft", "fromright" AS "fromright",
    "toright" AS "toright", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se"
FROM "tos_right";
```

```
create table tos_right_end as
SELECT "OGC_FID" AS "OGC_FID", EndPoint(GEOMETRY) AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "fromleft" AS "fromleft",
    "toleft" AS "toleft", "fromright" AS "fromright",
    "toright" AS "toright", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se"
FROM "tos_right";
```

ST_OffsetCurve ei tuota kaikissa tapauksissa uutta viivaa, vaan tuloksena on tyhjä geometria. Koska tyhjistä geometrioista ei ole mahdollista löytää alku- ja loppupisteitä, niin uusiin pistetauluihin tulee rivejä, joilla ei ole geometriatietoa. Niitä ei ole kovin paljon, eikä virheiden metsästys ole tämän harjoituksen pääasia, joten voimme kylmästi tuhota kaikki rivit, joilta geometriatieto puuttuu.

```
delete from tos_left_start where GeometryType("GEOMETRY") is null;
delete from tos_left_end where GeometryType("GEOMETRY") is null;
delete from tos_right_start where GeometryType("GEOMETRY") is null;
delete from tos_right_end where GeometryType("GEOMETRY") is null;
```

Näin on saatu aikaan osoitteellisia pisteitä. Ne ovat perineet emoviivoiltaan neljä eri osoitetta: fromleft, toleft, fromright ja toright. Pisteen sijainnista riippuen yksi näistä arvoista on pisteen oikea osoite. Seuraavassa yksinkertaistetaan tilannetta ja luodaan uusi ominaisuustieto ”osoite”, johon poimitaan kullekin pisteelle sille kuuluva osoitenumero. Tämä voidaan tehdä luomalla tietokantaan muutamia näkymiä. Samalla kertaa voidaan näppärästi hoitaa oikealle puolelle poikkeutettujen viivojen kulkusuunnan kääntymisestä koituvat ongelmat.

```
CREATE VIEW "l_start_view" AS
SELECT "GEOMETRY" AS "GEOMETRY",
"katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv", "katuos_se" AS
"katuos_se", CAST(fromleft AS INTEGER) AS "osoite", 'fromleft' AS "os_tieto"
FROM "tos_left_start";
```

```
CREATE VIEW "l_end_view" AS
SELECT "GEOMETRY" AS "GEOMETRY",
"katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv", "katuos_se" AS
"katuos_se", CAST(toleft AS INTEGER) AS "osoite", 'toleft' AS "os_tieto"
FROM "tos_left_end";
```

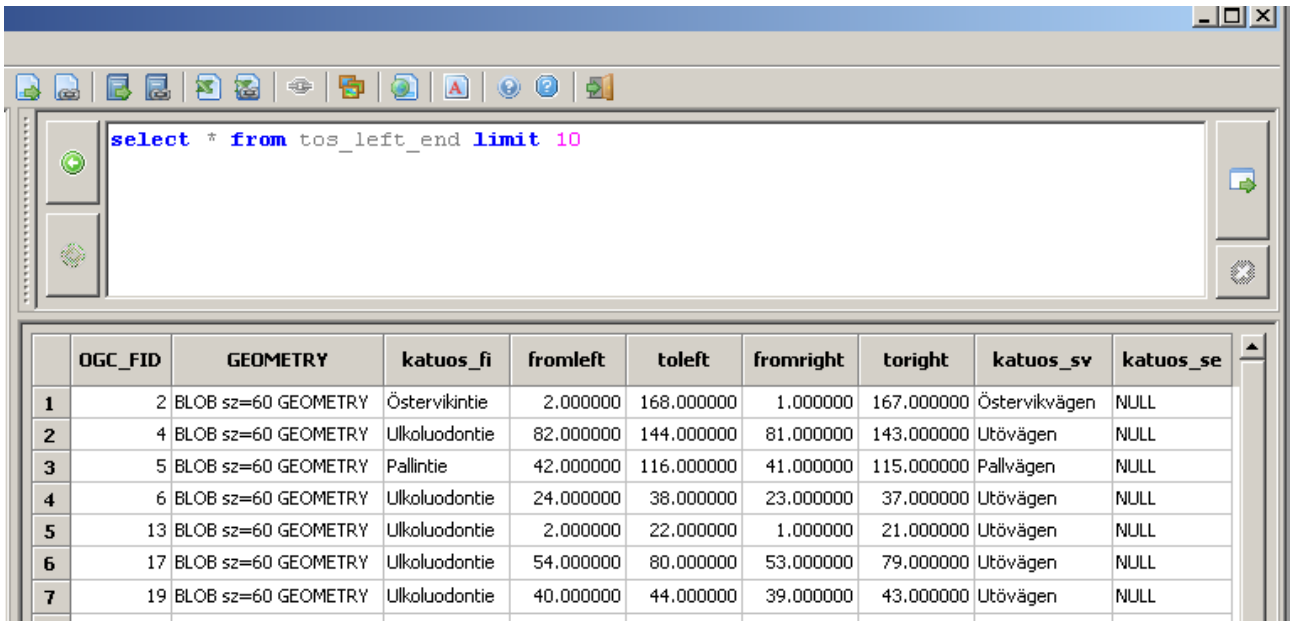
```
CREATE VIEW "r_start_view" AS
SELECT "GEOMETRY" AS "GEOMETRY",
"katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv", "katuos_se" AS
"katuos_se", CAST(fromright AS INTEGER) AS "osoite", 'fromright' AS "os_tieto"
FROM "tos_right_end";
```

```
CREATE VIEW "r_end_view" AS
SELECT "GEOMETRY" AS "GEOMETRY",
"katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv", "katuos_se" AS
"katuos_se", CAST(toright AS INTEGER) AS "osoite", 'toright' AS "os_tieto"
FROM "tos_right_start";
```

Oikean puolen viivojen kääntymisen takia otetaan alkupistenäkymään (r_start_view) pisteet loppupisteiden taulusta (tos_right_end) ja loppupistenäkymään alkupisteiden taulusta.

Mahdollista myöhempää tarvetta varten otetaan myös talteen tieto siitä, mistä alkuperäisen aineiston ominaisuustietokentästä osoitenumerot oikein ovat peräisin. Tämä tieto tallennetaan kenttään nimeltä ”os_tieto”. Hölmöä lyhennettä puolestaan käytetään siitä syystä, että joku kenties haluaa muuntaa aineiston myöhemmin shapefile-muotoon, jossa kenttien nimet voivat olla korkeintaan 10 merkin mittaisia. Esimerkiksi ”osoite_tieto” katkeaisi muunnoksessa muotoon ”osoite_tie”, joka olisi vielä hölmömpi nimi kuin ”os_tieto”.

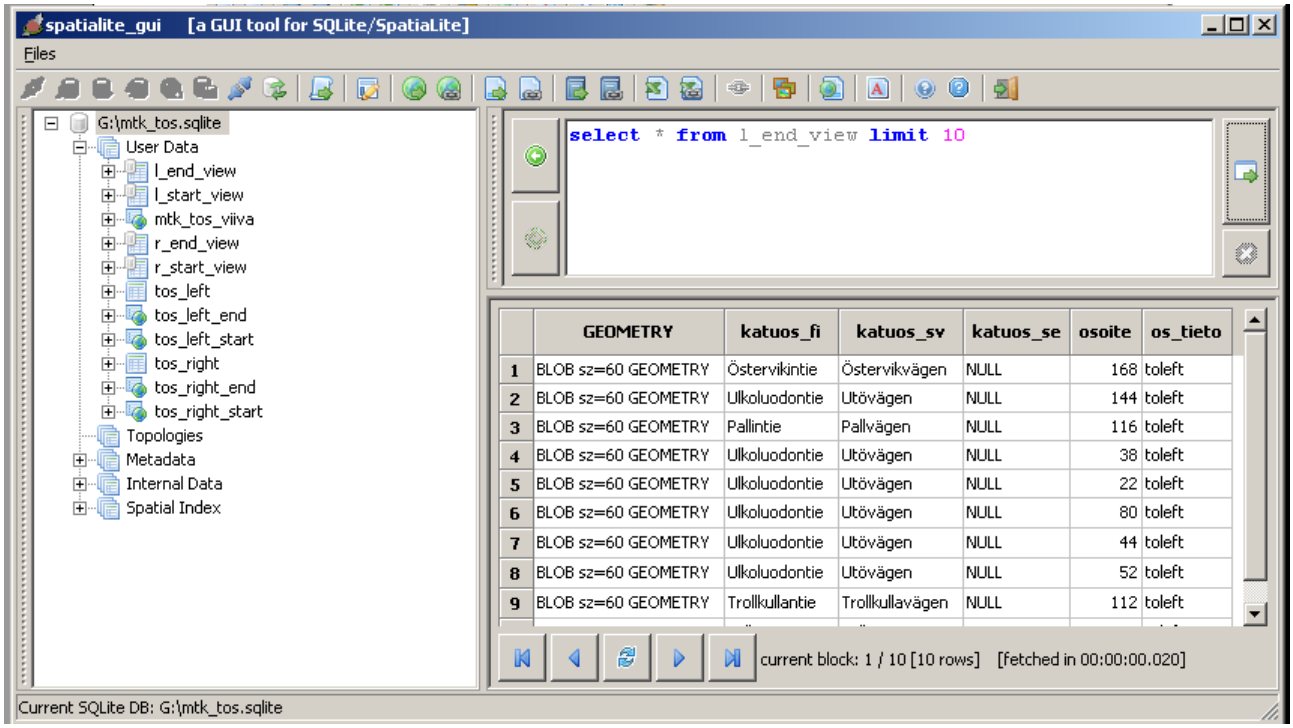
Osoitenumeroiden poimiseen käytetään muotoa `CAST(fromleft AS INTEGER) AS "osoite"`. Tämä tehdään lopputuloksen ulkoasun siistimiseksi. Spatialite, tai oikeammin SQLite, jonka päälle Spatialite on rakennettu, ei suuremmin välitä tietotyypeistä, ja jossain aikaisemmassa vaiheessa lähtöaineistossa kokonaislukuina olevat osoitteet ovat muuttuneet desimaaliluvuiksi, minkä näkee alla olevasta kuvasta.



```
select * from tos_left_end limit 10
```

	OGC_FID	GEOMETRY	katuos_fi	fromleft	toleft	fromright	toright	katuos_sv	katuos_se
1	2	BLOB sz=60 GEOMETRY	Östervikintie	2.000000	168.000000	1.000000	167.000000	Östervikvägen	NULL
2	4	BLOB sz=60 GEOMETRY	Ulkoluodontie	82.000000	144.000000	81.000000	143.000000	Utövägen	NULL
3	5	BLOB sz=60 GEOMETRY	Pallintie	42.000000	116.000000	41.000000	115.000000	Pallvägen	NULL
4	6	BLOB sz=60 GEOMETRY	Ulkoluodontie	24.000000	38.000000	23.000000	37.000000	Utövägen	NULL
5	13	BLOB sz=60 GEOMETRY	Ulkoluodontie	2.000000	22.000000	1.000000	21.000000	Utövägen	NULL
6	17	BLOB sz=60 GEOMETRY	Ulkoluodontie	54.000000	80.000000	53.000000	79.000000	Utövägen	NULL
7	19	BLOB sz=60 GEOMETRY	Ulkoluodontie	40.000000	44.000000	39.000000	43.000000	Utövägen	NULL

Seuraavasta kuvasta puolestaan nähdään, että edellä tehdyissä näkymissä osoitteet näkyvät siististi kokonaislukuina ja että myös osoitetiedon alkuperä on näkyvissä. Ääkkösten näkyminen on syytä myös tarkistaa, ja tässä ne näyttävät olevan kunnossa.



	GEOMETRY	katuos_fi	katuos_sv	katuos_se	osoite	os_tieto
1	BLOB sz=60 GEOMETRY	Östervikintie	Östervikvägen	NULL	168	toleft
2	BLOB sz=60 GEOMETRY	Ulkoluodontie	Utövägen	NULL	144	toleft
3	BLOB sz=60 GEOMETRY	Pallintie	Pallvägen	NULL	116	toleft
4	BLOB sz=60 GEOMETRY	Ulkoluodontie	Utövägen	NULL	38	toleft
5	BLOB sz=60 GEOMETRY	Ulkoluodontie	Utövägen	NULL	22	toleft
6	BLOB sz=60 GEOMETRY	Ulkoluodontie	Utövägen	NULL	80	toleft
7	BLOB sz=60 GEOMETRY	Ulkoluodontie	Utövägen	NULL	44	toleft
8	BLOB sz=60 GEOMETRY	Ulkoluodontie	Utövägen	NULL	52	toleft
9	BLOB sz=60 GEOMETRY	Trollkullantie	Trollkullavägen	NULL	112	toleft

Osoitepisteiden irrottaminen on nyt periaatteessa valmis. Osoitteet ovat kuitenkin vielä tietokannassa neljässä eri näkymässä. Ne olisi mahdollista yhdistää työn alla olevaan tietokantaan, mutta koska tavoitteena on pelkät osoitepisteet sisältävä kanta, niin nyt on hyvä aika jättää Maastotietokannan tiestöviivat ja niistä tehdyt tielinjan sivuun poikkeutetut kopiot taakse ja jatkaa pelkkien pisteiden kanssa. Yksi vaihtoehto on koota osoitetiedot näistä neljästä näkymästä ja tallentaa ne kokonaan uuteen Spatialite-tietokantaan omaksi yhteiseksi taulukseen. Tämä onnistuu esimerkiksi ogr2ogr-ohjelmalla käyttämällä seuraavia komentoja:

```
ogr2ogr -f SQLite -dsco init_with_epsg=yes -dsco spatialite=yes
mtk_osoitteet_2012_pohja.sqlite mtk_tos.sqlite -sql "select * from l_start_view"
-nlt point -nln mtk_osoitteet_2012 -a_srs epsg:3067  -gt 50000
```

```
ogr2ogr -f SQLite -append mtk_osoitteet_2012_pohja.sqlite mtk_tos.sqlite -sql
"select * from l_end_view" -nlt point -nln mtk_osoitteet_2012 -a_srs epsg:3067
-gt 50000
```

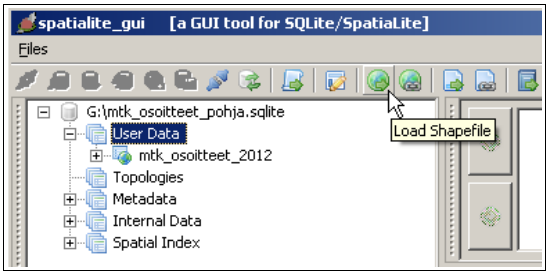
```
ogr2ogr -f SQLite -append mtk_osoitteet_2012_pohja.sqlite mtk_tos.sqlite -sql
"select * from r_start_view" -nlt point -nln mtk_osoitteet_2012 -a_srs epsg:3067
-gt 50000
```

```
ogr2ogr -f SQLite -append mtk_osoitteet_2012_pohja.sqlite mtk_tos.sqlite -sql
"select * from r_end_view" -nlt point -nln mtk_osoitteet_2012 -a_srs epsg:3067
-gt 50000
```

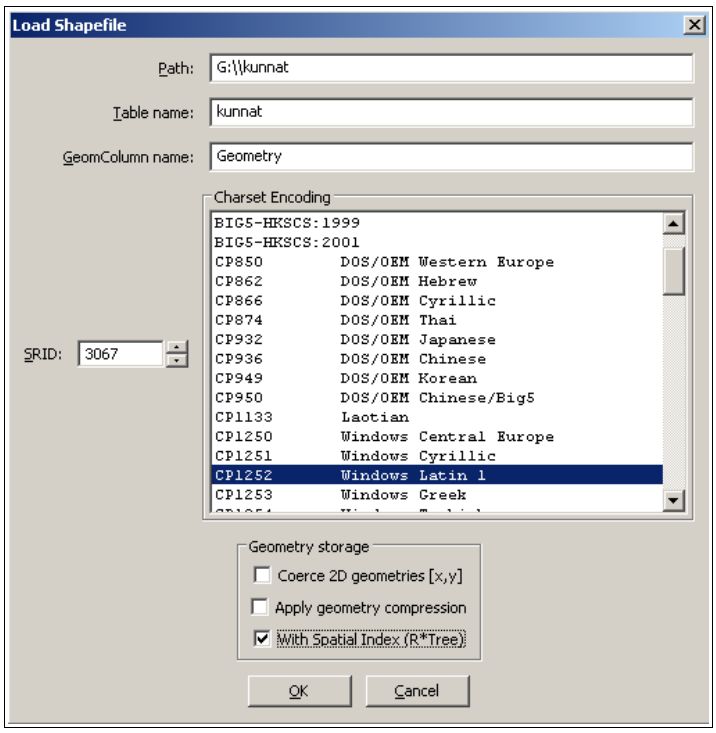
Vaihe 4: Osoitetietojen rikastuttaminen muista paikkatietoaineistosta liitettävillä ominaisuustiedoilla; esimerkkinä tieto sijaintikunnasta

Nyt kun osoitetiedot on saatu siirretyksi pisteisiin, niin huomataan karu tosiasia: meiltä puuttuu tieto siitä, missä kunnassa mikäkin osoite sijaitsee. Tämä tieto on itse asiassa mukana lähtöaineistossa, jossa kenttään ”kunta_nro” on tallennettu tieosuuden sijaintikunnan kuntakoodi. Tämä unohdus olisi helppo korjata muokkaamalla kaikkia SQL-kyselyitä vaiheessa 2 ja sen jälkeen niin, että myös ”kunta_nro” tulisi poimituksi mukaan. Koska olisi hauska saada osoitteet liitytyiksi myös kuntien nimiin eikä vain kuntakoodeihin, niin tehdään kuitenkin kuntatietojen yhdistäminen spatiaalisella liitoksella eli yhdistämällä tietoja toisesta aineistoista kohteiden sijainnin perusteella. Tässä tapauksessa tutkitaan osoitepiste kerrallaan minkä kunnan alueen sisäpuolelle piste osuu. Tätä menetelmää voidaan käyttää ominaisuustietojen siirtämiseen minkä tahansa piste- ja alueaineiston välillä, joten se on joka tapauksessa kiva osata mahdollista myöhempää tarvetta varten.

Mallisuorituksessa ominaisuustietojen yhdistäminen aloitetaan tuomalla aluemaisina kohteina olevat kuntarajat samaan Spatialite-kantaan, jossa osoitepisteet ovat. Tässä esimerkissä kuntarajat ovat shapefile-muodossa, ja niiden tuominen tehdään Spatialite-gui -ohjelmalla, jossa on tätä tarkoitusta varten oma nappulansa ”Load Shapefile”.



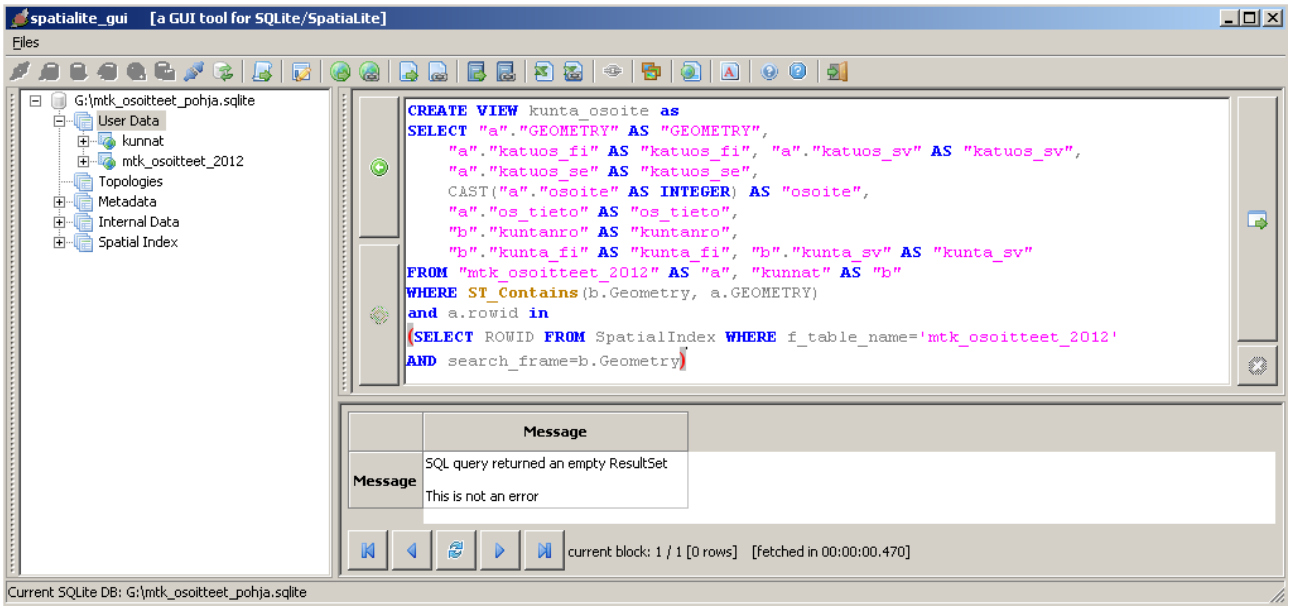
Ladataan kuntakartta ”kunnat.shp” -nimisestä tiedostosta. Esimerkissä käytetty aineisto on Maanmittauslaitoksen 1:100000 mittakaavan kuntarajat alueina. Huomaa projektiokoodin EPSG:3067 ja shapefilessä käytetyn merkistökoodauksen antaminen, samoin kuin käsky luoda spatiaali-indeksi syntyvällä taululle.



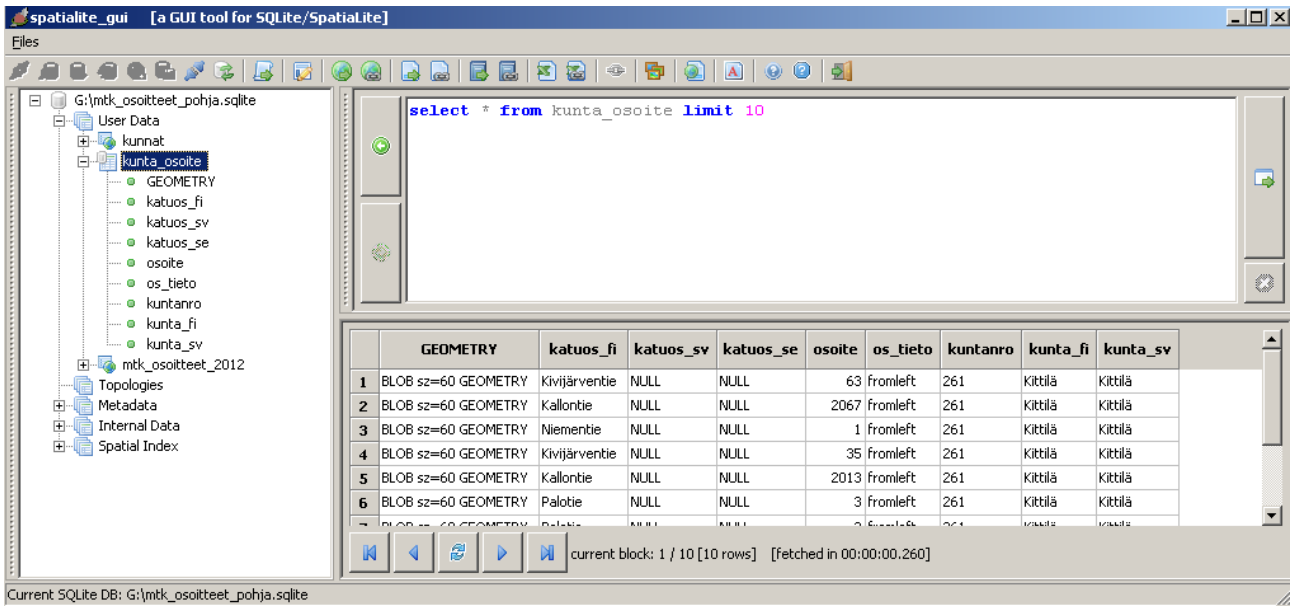
Seuraavaksi luodaan näkymä, joka yhdistää osoitepisteeseen muutaman ominaisuustiedon siitä kuntapolygonista, jonka sisäpuolella se sijaitsee. Paikkaan perustuvan vertailun toteuttaa rivi ”WHERE ST_Contains(b.Geometry, a.GEOMETRY)”, ja se merkitsee siis ”kunnan alue sisältää osoitepisteen”.

SQL-kyselyn viimeisillä kolmella rivillä hyödynnetään Spatialiten hieman erikoista tapaa toteuttaa spatiaali-indeksi eli kohteiden sijaintiin perustuva hakemisto. Ilman tätä alikyselyä Spatialite sovittaisi jokaista 4,5 miljoonaa osoitepistettä jokaista 336 kuntapolygonia vasten eli tekisi puolitoista miljardia sijaintien vertailua. Tämä kysely tulisi kestämään ehkä kaksi kuukautta, kun spatiaali-indeksiä käytettäessä tietojen yhdistäminen kestää noin viisi tuntia, joten indeksin käyttäminen kannattaa (vertailukoneessa 1 x 3 Ghz Xeon CPU vuodelta 2004).

```
CREATE VIEW kunta_osoite as
SELECT "a"."GEOMETRY" AS "GEOMETRY",
    "a"."katuos_fi" AS "katuos_fi", "a"."katuos_sv" AS "katuos_sv",
    "a"."katuos_se" AS "katuos_se", CAST("a"."osoite" AS INTEGER) AS "osoite",
    ”a”.”os_tieto” AS ”os_tieto”,
    "b"."kuntanro" AS "kuntanro",
    "b"."kunta_fi" AS "kunta_fi", "b"."kunta_sv" AS "kunta_sv"
FROM "mtk_osoitteet_2012" AS "a", "kunnat" AS "b"
WHERE ST_Contains(b.Geometry, a.GEOMETRY)
and a.rowid in
(SELECT ROWID FROM SpatialIndex WHERE f_table_name='mtk_osoitteet_2012'
AND search_frame=b.Geometry);
```



Kunnat ja osoitteet yhdistävän näkyman luominen Spatialite-gui-ohjelmalla



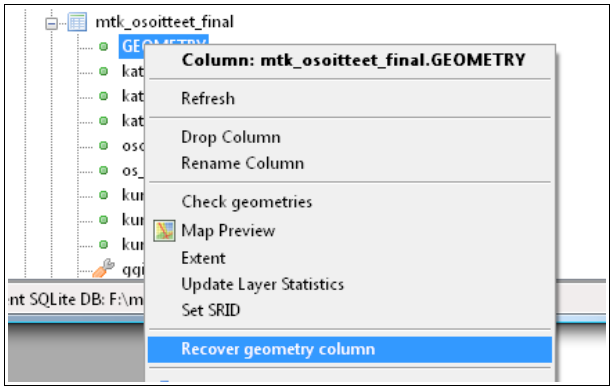
Kymmenen ensimmäistä riviä näkymästä palauttava kysely, joka osoittaa, että osoitepisteillä on nyt myös tieto sijaintikunnan kuntakoodista sekä kunnan suomen- ja ruotsinkielisestä nimestä.

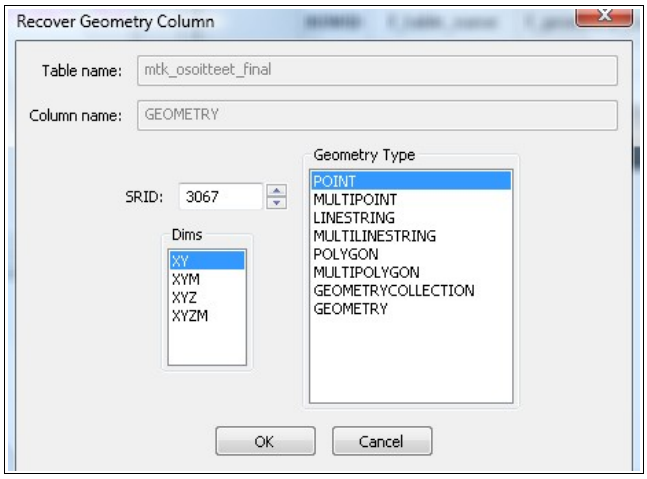
Vaihe 5: Lisää vauhtia ja pientä hifistelyä

Edellisessä vaiheessa tehty kunta_osoite -näkymä on sellaisenaan käyttökelpoinen, mutta kyselyyn kuuluva ST_Contains -osa on aika hidas. Vauhtia saadaa lisää, jos luodaan näkymän perusteella uusi taulu, ja kysellään jatkossa tietoja tästä taulusta.

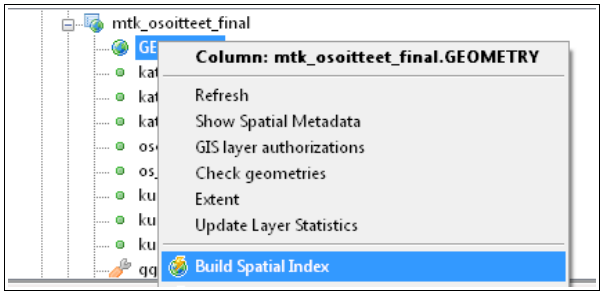
```
create table osoitteet_final as
select * from kunta_osoite;
```

Kun uusi taulu on luotu, niin sille on syytä tehdä ensin ”Recover geometry column” -toiminto, joka tehdään kuten alla olevista kuvista käy selville. Valikot saadaan näkyville valitsemalla ensin GEOMETRY-kenttä ja painamalla sitten hiiren oikeaa näppäintä.





Kun uuden taulun geometriakenttä on rekisteröity, niin taululle voidaan tehdä spatiaali-indeksi. Tämäkin toiminto löytyy painamalla hiiren oikeaa näppäintä, mutta vasta sitten, kun ”Recover geometry column” on tehty.



Pelkät osoitetiedot sisältävä taulu (esimerkissä mtk_osoitteet_2012) voidaan nyt hävittää turhana. Uusi, myös kuntanimet sisältävä taulu (mtk_osoitteet_final) voidaan haluttaessa nimetä uudestaan, niin, että siitä tulee mtk_osoitteet_2012. Nämä ovat normaaleja Spatialite-toimintoja eikä niiden tekemistä neuvota tässä.

Uudesta taulusta halutaan kenties tehdä hakuja myös kadun tai tien nimen perusteella tai osoitteen sijaintikunnan mukaan. Näitä kyselyjä saadaan nopeutetuksi tekemällä muutamia tavallisia indeksejä seuraavilla SQL-lauseilla.

```
CREATE INDEX mtk_osoitteet_2012_katuos_fi_idx ON "mtk_osoitteet_2012"(katuos_fi);
CREATE INDEX mtk_osoitteet_2012_katuos_sv_idx ON "mtk_osoitteet_2012"(katuos_sv);
CREATE INDEX mtk_osoitteet_2012_katuos_se_idx ON "mtk_osoitteet_2012"(katuos_se);
CREATE INDEX mtk_osoitteet_2012_os_tieto_idx ON "mtk_osoitteet_2012"(os_tieto);
CREATE INDEX mtk_osoitteet_2012_kuntanro_idx ON "mtk_osoitteet_2012"(kuntanro);
CREATE INDEX mtk_osoitteet_2012_kunta_fi_idx ON "mtk_osoitteet_2012"(kunta_fi);
CREATE INDEX mtk_osoitteet_2012_kunta_sv_idx ON "mtk_osoitteet_2012"(kunta_sv);
```

Alkuperäisessä maastotietokannan aineistossa ensimmäisen viivan loppupiste on sama kuin seuraavan viivan alkupiste. Tästä syystä osoitepisteet ovat päällekkäin. Jos ne nyt lisätään karttaohjelmaan ja pyydetään ohjelmaa piirtämää osoitenumerot, niin kartalle tulee kaksi numeroa päällekkäin, eikä niistä kummastakaan saa selvää. Kömpelö ratkaisu, jolla tätä ongelmaa voi yrittää kiertää, on erottaa osoitepisteen niiden alkuperän mukaa omiksi karttatasoikseen. Alkupisteiden osoitteille voidaan sitten karttaohjelmassa piirtää osoitenumerot esimerkiksi suoraan pisteen päälle ja loppupisteiden osoitteet voidaan piirtää vaikkapa 20 pikseliä pisteestä vasemmalle. Esimerkiksi Quantum GIS -ohjelmassa tämä onnistuu antamalla tunnuksen paikalle poikkeutuksen suuruus offset-parametrillä tason piirtoasetuksissa. Todennäköisesti tulos ei ole erityisen hieno, mutta kuitenkin parempi kuin ilman erottelua ja poikkeutusta. Kokeilu ei ainakaan käy kalliiksi, ja se on helppo tehdä luomalla tietokantaan muutamia spatiaalisia näkymiä:

```
CREATE VIEW "fromleft_view" AS
SELECT "ROWID" AS "ROWID", "GEOMETRY" AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se", "osoite" AS "osoite",
    "os_tieto" AS "os_tieto", "kuntanro" AS "kuntanro",
    "kunta_fi" AS "kunta_fi", "kunta_sv" AS "kunta_sv",
    '0' AS "offset"
FROM "mtk_osoitteet_2012"
WHERE "os_tieto" = 'fromleft';
```

```
CREATE VIEW "toleft_view" AS
SELECT "ROWID" AS "ROWID", "GEOMETRY" AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se", "osoite" AS "osoite",
    "os_tieto" AS "os_tieto", "kuntanro" AS "kuntanro",
    "kunta_fi" AS "kunta_fi", "kunta_sv" AS "kunta_sv",
    '-10' AS "offset"
FROM "mtk_osoitteet_2012"
WHERE "os_tieto" = 'toleft';
```

```
CREATE VIEW "fromright_view" AS
SELECT "ROWID" AS "ROWID", "GEOMETRY" AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se", "osoite" AS "osoite",
    "os_tieto" AS "os_tieto", "kuntanro" AS "kuntanro",
    "kunta_fi" AS "kunta_fi", "kunta_sv" AS "kunta_sv",
    '0' AS "offset"
FROM "mtk_osoitteet_2012"
WHERE "os_tieto" = 'fromright';
```

```
CREATE VIEW "toright_view" AS
SELECT "ROWID" AS "ROWID", "GEOMETRY" AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se", "osoite" AS "osoite",
    "os_tieto" AS "os_tieto", "kuntanro" AS "kuntanro",
    "kunta_fi" AS "kunta_fi", "kunta_sv" AS "kunta_sv",
    '10' AS "offset"
FROM "mtk_osoitteet_2012"
WHERE "os_tieto" = 'toright';
```

```
CREATE VIEW "alku_view" AS
SELECT "ROWID" AS "ROWID", "GEOMETRY" AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se", "osoite" AS "osoite",
    "os_tieto" AS "os_tieto", "kuntanro" AS "kuntanro",
    "kunta_fi" AS "kunta_fi", "kunta_sv" AS "kunta_sv",
    '0' AS "offset"
FROM "mtk_osoitteet_2012"
WHERE "os_tieto" = 'fromleft'
OR "os_tieto" = 'fromright';
```

```
CREATE VIEW "loppu_view" AS
SELECT "ROWID" AS "ROWID", "GEOMETRY" AS "GEOMETRY",
    "katuos_fi" AS "katuos_fi", "katuos_sv" AS "katuos_sv",
    "katuos_se" AS "katuos_se", "osoite" AS "osoite",
    "os_tieto" AS "os_tieto", "kuntanro" AS "kuntanro",
    "kunta_fi" AS "kunta_fi", "kunta_sv" AS "kunta_sv",
    '20' AS "offset"
FROM "mtk_osoitteet_2012"
WHERE "os_tieto" = 'toleft'
OR "os_tieto" = 'toright';
```

Näkymille täytyy tehdä rivit ”views_geometry_columns” -tauluun, ennen kuin niitä voidaan käyttää suoraan Quantum GIS -ohjelmassa. Rivit voidaan tehdä yksi kerrallaan käyttöliittymästä tai suoraan SQL:llä:

```
insert into views_geometry_columns values
("fromleft_view","GEOMETRY","ROWID","mtk_osoitteet_2012","GEOMETRY");

insert into views_geometry_columns values
("toleft_view","GEOMETRY","ROWID","mtk_osoitteet_2012","GEOMETRY");

insert into views_geometry_columns values
("fromright_view","GEOMETRY","ROWID","mtk_osoitteet_2012","GEOMETRY");

insert into views_geometry_columns values
("toright_view","GEOMETRY","ROWID","mtk_osoitteet_2012","GEOMETRY");

insert into views_geometry_columns values
("alku_view","GEOMETRY","ROWID","mtk_osoitteet_2012","GEOMETRY");

insert into views_geometry_columns values
("loppu_view","GEOMETRY","ROWID","mtk_osoitteet_2012","GEOMETRY");
```

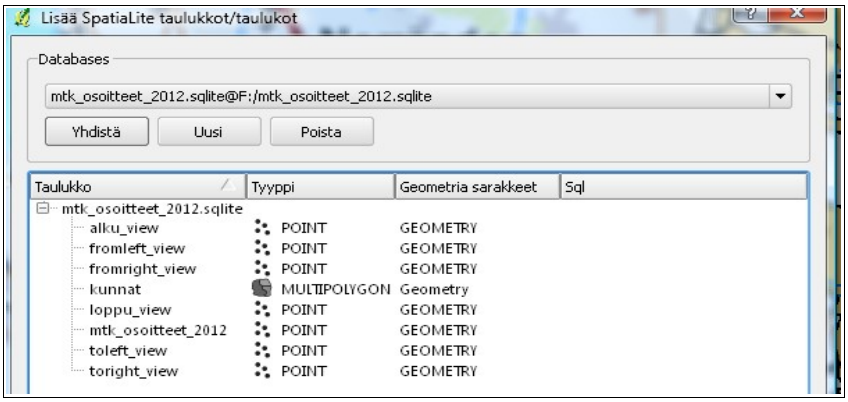
Koordinaattijärjestelmien muunnosparametrit sisältävä taulu on myös hyvä päivittää, jos joku jostain kumman syystä tahtoisi muuntaa koordinaatteja suomalaiseen KKJ/YKJ-järjestelmään. Ilman päivitystä muunnos johtaa reilun sadan metrin sijaintivirheeseen.

```
UPDATE spatial_ref_sys set proj4text='+proj=tmerc +lat_0=0 +lon_0=21 +k=1
+x_0=1500000 +y_0=0 +ellps=intl +towgs84=-96.0617,-82.4278,-
121.7435,4.80107,0.34543,-1.37646,1.4964 +units=m +no_defs ' WHERE srid=2391;

UPDATE spatial_ref_sys set proj4text='+proj=tmerc +lat_0=0 +lon_0=24 +k=1
+x_0=2500000 +y_0=0 +ellps=intl +towgs84=-96.0617,-82.4278,-
121.7435,4.80107,0.34543,-1.37646,1.4964 +units=m +no_defs ' WHERE srid=2392;

UPDATE spatial_ref_sys set proj4text='+proj=tmerc +lat_0=0 +lon_0=27 +k=1
+x_0=3500000 +y_0=0 +ellps=intl +towgs84=-96.0617,-82.4278,-
121.7435,4.80107,0.34543,-1.37646,1.4964 +units=m +no_defs ' WHERE srid=2393;
```

```
UPDATE spatial_ref_sys set proj4text='+proj=tmerc +lat_0=0 +lon_0=30 +k=1
+x_0=4500000 +y_0=0 +ellps=intl +towgs84=-96.0617,-82.4278,-
121.7435,4.80107,0.34543,-1.37646,1.4964 +units=m +no_defs ' WHERE srid=2394;

UPDATE spatial_ref_sys set proj4text='+proj=tmerc +lat_0=0 +lon_0=18 +k=1
+x_0=500000 +y_0=0 +ellps=intl +towgs84=-96.0617,-82.4278,-
121.7435,4.80107,0.34543,-1.37646,1.4964 +units=m +no_defs ' WHERE srid=3386;

UPDATE spatial_ref_sys set proj4text='+proj=tmerc +lat_0=0 +lon_0=33 +k=1
+x_0=5500000 +y_0=0 +ellps=intl +towgs84=-96.0617,-82.4278,-
121.7435,4.80107,0.34543,-1.37646,1.4964 +units=m +no_defs ' WHERE srid=3387;

UPDATE spatial_ref_sys set proj4text='+proj=utm +zone=35 +ellps=GRS80
+towgs84=0,0,0,0,0,0,0 +units=m +no_defs ' WHERE srid=3067;
```

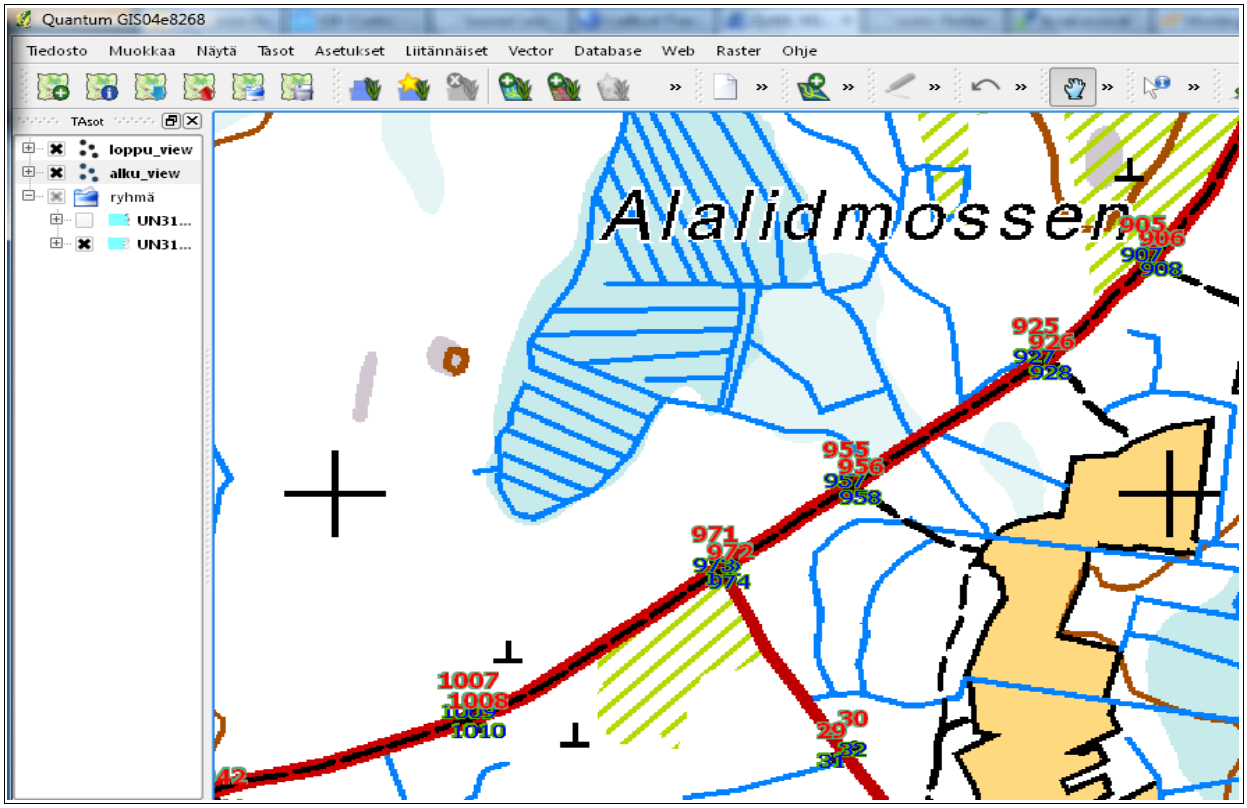
Edelleen, koska tietokannasta on poistettu yksi suuri taulu, niin tietokannan pakkaamiseksi ja tiedostokoon pienentämiseksi on hyvä ajaa VACUUM-toiminto, vaikka sen suorittaminen kestääkin kauan.

Lopputuloksena Quantum GIS -ohjelma näyttää seuraavat tasot suoraan valikossa:



Käytännössä on tuskin kovin järkevää valita uuteen Qgis-projektiin alkajaisiksi osoitetasoa Spatialite-kannasta ja avata sitä. Tämä johtaisi siihen, että Qgis lukisi tietokannasta kaikki 4,5 miljoonaa osoitepistettä ja piirtäisi ne kartalle. Normaalisti kannattaa ensin avata kartalle jokin muu aineisto ja kohdistaa karttaikkuna pienemmälle, esimerkiksi muutaman kunnan suuruiselle alueelle. Alueen rajaaminen on varsin yksinkertaista, koska tietokannassa on edelleen mukana myön kuntien rajat. Kannattaa siis ensin avata tietokannasta kuntarajat ja kohdistaa kartta haluttuun kuntaan, ja vasta sen jälkeen lukea osoitteet kartalle. Qgis käyttää Spatialite-tietokantaa dynaamisesti, eli se hakee kannasta vain karttaikkunaan osuvat kohteet. Toiminto on automaattinen niin, että kartta päivittyy aina kun karttaikkunan koko tai paikka muuttuu. Kun karttainäkymä pidetään suhteellisen pienenä, niin tämän ikkuna läpi voidaan kiikaroida varsin nopeasti ja tehokkaasti koko maan osoiteaineistoa.

Seuraavassa kuvassa näkyy peruskartta, jonka päälle on poimittu viivan alkupisteiden ja loppupisteiden osoitteet omiksi karttatasoikseen Spatialite-tietokannan näkymistä ”alku_view” ja ”loppu_view”. Tunnusten paikkaa kartalla on poikkeutettu Quantum GIS:in piirto-ominaisuuksien valinnoilla, eivätkä numerot enää peitä toisiaan.



Huomautus lähtöaineistosta

Aineistossa ”Maastotietokannan tiestö osoitteilla” on yli 200 tuhatta sellaista tieosuutta, jonka lähtöpisteen ja loppupisteen osoitteet ovat samat. Tämä vaikuttaa oudolta, mutta siihen on varmaan jokin järkevä syy.

Loppusanat: Nopeammin , paremmin, halvemmalla

Tässä raportissa on esitetty suhteellisen yksinkertainen mallisuoritus. Sen ei ole tarkoitus olla täydellinen, vaan tavoitteena on laittaa rima aloituskorkeuteen ja tarjota jotain mitattavissa olevia asioita, joiden perusteella on mahdollista pyrkiä parempaan.. Latuviitan mittarit ovat yksinkertaiset ja olympiahenkiset: nopeus, laatu ja hinta. Jos jokin muu vaihtoehdo on mallisuoritusta hitaampi, tuottaa huonompaa laatua ja on kalliimpi, niin on myös mahdollista, että se vain yksinkertaisesti on huonompi menetelmä. Käytännössä paremmuuden arviointi ei ole helppoa, koska kyseessä on arvostelulaji. Millä perusteilla arvioidaan lopputuloksen laatu, ja mikä on ilmaisten ohjelmistojen hinta, kun niitä käyttämään tarvitaan kuitenkin ihminen, jolle kenties maksetaan myös palkkaa, ja joka on kenties työajalla hankkinut tarvittavan osaamisen? Edes nopeus ei ole aina yksiselitteinen asia, jollain muulla menetelmällä voidaan kenties vastaava lopputulos saada aivan erilaisia polkuja etenemällä.

Galileo Galilei sanoi vuonna 1610 ”Meidän täytyy mitata se mikä voidaan mitata, ja tehdä mitattavaksi se, mitä ei voida mitata”. Nykyisin sanonta laitetaan useimmin W. Edwards Demingin suuhun ja vuodelle 1981 muodossa ”Mitä et voi mitata, sitä et voi hallita”, vaikkakin Deming itse

asiassa kirjoittaa, että monia asioita ei voi mitata, mutta silti on tehtävä päätöksiä. Demingin väitetään myös sanoneen: ”Me luotamme jumalaan; muilta me vaadimme dataa”. Avointen paikkatietoaineistojen ansiosta nykyisin on ennennäkemättömän helppoa järjestää kontrolloituja testejä, joissa käytetään todellisia ja realistisen kokoisia aineistoja, joten dataa on helppo toimittaa, jos joku sitä pyytää. Avoimen lähdekoodin ohjelmistojen ansiosta datan vastaanottaja voi myös tehdä vertailuja, ellei katso vastapuolen olevan sellaisessa asemassa, että luottamus riittää.

Seuraavassa Latuviitan itsearviointi tästä mallisuorituksesta

Nopeammin

Suoritusaikoja testikoneella, jonka tärkeimmät ominaisuudet ovat

- Kevyt Windows-kannettava vuodelta 2008
- Intel Core2 Duo U7600 1,2 Ghz
- 2 Gt keskusmuistia
- Levytoiminnot testin aikana ulkoisella USB2-kovalevyllä

toiminto	aika	huomautuksia
Tiestön shp-tiedostojen vienti Spatialite-kantaan	30 minuuttia	Shp-tiedostot 4,6 gigatavua kohteita 3451629 kappaletta
Spatialiten ST_OffsetCurve	70 minuuttia/puoli	Noin 1,14 miljoonaa osoitteellista viivaa/puoli
StartPoint ja EndPoint	1 minuutti/suoritus	Neljä suoritusta, vasen/oikea ja alku/loppupisteet
Osoite_kunta -näkymän muuttaminen tauluksi	3 tuntia 10 minuuttia	
Spatiaali-indeksin teko uudelle taululle	30 minuuttia	Karkea arvio, ei mitattu kellolla

Lisäksi mallisuorituksessa tehtiin pisteiden siirto uuteen Spatialite-kantaan ogr2ogr-ohjelmalla (noin tunti) ja VACUUM-valmiille tietokannalle (noin tunti).

Vertailun vuoksi, hitaimman yksittäisen vaiheen eli kunta- ja osoitetietojen yhdistämisen pystyi samalla testikoneella tekemään PostGIS-tietokannassa noin 10 minuutissa ja OpenJUMP-ohjelmalla 70 minuutissa (jaettuna 400000 osoitteen eriin muistin loppumisen takia). Pari vuotta vanhalla keskiverto kotikoneella (2 x 3 GHZ AMD, 6 Gt muistia, SATA-kovalevy) Spatialite selviytyi samasta työstä 70 minuutissa eli noin kolme kertaa nopeammin kuin testikoneella.

Johtopäätös: Spatialite on laskentaa vaativissa tehtävissä hitaanpuoleinen ja on selvää, että PostGIS selviäisi tehtävästä nopeammin. Todennäköisesti sama pätee Oracleen ja muihin ”oikeisiin” tietokantaohjelmiin. Toisaalta Spatialiten asentamiseen ei tarvita käytännössä ollenkaan työtä, ja tulokseksi saatava Spatialite-osoitetietokanta on sellaisenaan valmis monistettavaksi ja jaettavaksi käyttäjille ilman muunnosta esimerkiksi GML- tai shapefile-muotoon. Nopeusarvosana riippuu siis siitä, mitä kaikkea otetaan kokonaisajan mittauksessa huomioon.

Paremmin

Parempaan lopputulokseen pääsemiseksi voisi kiinnittää huomiota seuraaviin asioihin:

- Selvittää, miksi osa viivoista kadotti geometriansa ST_OffsetCurve-toiminnossa.
- Miettiä, mitä tehdä tapauksissa, joissa viivan alku- ja loppupisteillä on sama osoitenumero. Pitäisikö esimerkiksi loppupiste hävittää kokonaan?
- Tutkia mahdollisuutta erotella alku- ja loppupisteiden osoitteet hallitusti niin, että ne voisi piirtää nätisti kartalle ilman päällekkäin menemistä. Ehkä ST_Line_Interpolate_Point olisi tähän sopiva työkalu http://www.gaia-gis.it/spatialite-3.0.0-BETA/GEOS-advanced.pdf
- Suurin haaste osoitteiston tihentäminen laskemalla viivan välipisteille osoitteet interpoloimalla. Yksinkertainen tihentäminen voisi onnistua edellisellä ST_Interpolate_Poin-funktiolla. Logiikka voisi olla vaikkapa tällainen: Laske viivan loppu- ja alkupisteiden osoitteiden erotus tietokannasta (toleft-fromleft ja toright-fromright). Jos se ylittää valitus raja-arvon, sijoita viivan puoliväliin uusi piste ST_Interpolate_Point-funktiolla ja anna sille osoitenumero alku- ja loppuosoitteiden puolivälistä.
- Myös aivan uusi Spatialite-funktio ST_Line_Interpolate_Equidistant_Points voi olla käyttökelpoinen välipisteiden interpoloinnissa. Ilmoitus uudesta funktiosta https://groups.google.com/forum/#!msg/spatialite-users/I2_raqivuu0/MMMKoFshd0IJ
- Uusimmassa Spatialitessä on myös viivan kääntämiseen tarkoitettu ST_Reverse-funktio, jolla voi ratkaista ST_OffsetCurve:n aiheuttaman loogisen sotkun.
- Muutamia saamenkielen kirjaimia ei muunnettu UTF-8 -muotoon oikein.

Halvemmalla

Harrastajan silmin katsottuna tässä selostetun menetelmän käyttäminen ei maksa mitään, koska ohjelmat ovat ilmaisia eikä harrastukseen käytetty aika ole kustannus. Ammatikseen ohjelmistoja käyttävät ja myyvät näkevät kustannukset toisella tavalla ja kehittävät kokonaiskulujen arvioimiseen omia yhtälöitään. Toivottavasti niitäkin julkaistaan parametreineen ja painoarvoineen. Latuviitta.org on harrastussivusto ja katsoo, että jos harrastukseen ei kulu rahaa, niin sitten se on ilmaista.