Pääkaupunkiseudun liikennemeluvyöhykkeiden haltuunotto

Jukka Rahkonen, http://latuviitta.org
Viimeksi muutettu 15. joulukuuta 2012

Tiivistelmä

Tässä ohjeessa muunnetaan pääkaupunkiseudun liikennemeluvyöhykkeistä julkaistut paikkatietoaineistot sellaiseen muotoon, joka soveltuu alkuperäistä jakelumuotoa paremmin ”piste alueen sisällä” -kyselyiden tekoon. Muunnoksen ansiosta nämä kyselyt nopeutuvat jopa yli satakertaisesti.

Ohjeessa esitetään lisäksi, kuinka muunnettua aineistoa käyttävästä WFS-palvelusta voidaan tehdä ”Mikä on melutaso tässä paikassa?” -kyselyitä ilman että tarvittaisiin mitään erityistä juuri tätä tarkoitusta varten tehtyä melu-API:a. Tällainen WFS-taso on myös ainakin toistaiseksi käytettävissä Latuviitan WFS-palvelussa http://hip.latuviitta.org.

Katsaus aineistoon

Aineistot ovat ladattavissa Helsinki Region Infoshare –sivustolta alueittain.

http://www.hri.fi/fi/data/espoon-ja-kauniaisten-liikennemeluvyohykkeet/
http://www.hri.fi/fi/data/helsingin-liikennemeluvyohykkeet/
http://www.hri.fi/fi/data/espoon-ja-kauniaisten-liikennemeluvyohykkeet/

Jokaisessa zip-arkistossa on 12 kartta-aineistoa, koska meluvyöhykkeet on laskettu neljällä eri laskentamallilla (L_{aeq} päivä, L_{aeq} yö, L_{den} ja L_{yo}) ja kolmelle eri liikennelajille (tieliikenne, maantiet ja rautatieliikenne). Koska alueitakin on kolme, niin yhteensä meluvyöhykeaineistoon kuuluu 36 karttatasoa shapefile-muodossa.

Nopea tapa tutustua aineistoon yhtenä kokonaisuutena on vielä kaikki 36 karttatasoa Spatialite-tietokantaan ogr2ogr-ohjelmalla, joka pystyy ”vsizip”-menetelmän avulla löytämään kaikki zip-tiedoston sisältä löytyvät aineistot. Ne voidaan muuntaa edelleen muihin tiedostomuotoihin joko yksitellen antamalla komennossa zipin sisältä löytyvän halutun karttatason nimi, tai kaikki tasot yhdellä kerralla. Tämä onnistuu yksinkertaisesti jättämällä tason nimi antamatta, kuten seuraavissa esimerkkikomennoissa tehdään. Huomaa, että ensimmäinen komento luo tietokannan, ja toiset kaksi lisäävät olemassa olevaan tietokantaan, mistä syystä komennoissa on pieni ero.

```
ogr2ogr -f SQLite -dsco Spatialite=yes -a_srs epsg:3067 melu_raaka.sqlite
/vsizip/Helsinki_meluvyohykkeet.zip -nlt PROMOTE_TO_MULTI -gt 2000
```

```
ogr2ogr -f SQLite -append -a_srs epsg:3067 melu_raaka.sqlite
/vsizip/espoo_kauniainen_meluvyohykkeet.zip -nlt PROMOTE_TO_MULTI -gt 2000

ogr2ogr -f SQLite -append -a_srs epsg:3067 melu_raaka.sqlite
/vsizip/vantaa_meluvyohykkeet.zip -nlt PROMOTE_TO_MULTI -gt 2000
```

Lopputulosta voidaan tarkastella esimerkiksi ”ogrinfo” -ohjelmalla

```
ogrinfo melu_raaka.sqlite
INFO: Open of `melu_raaka.sqlite'
      using driver `SQLite' successful.
1: helsinki_rautatiet_lden (Multi Polygon)
2: helsinki_tieliikenne_l_den (Multi Polygon)
3: helsinki_tieliikenne_l_yo (Multi Polygon)
4: helsinki_tieliikenne_laeq_paiva (Multi Polygon)
5: helsinki_tieliikenne_laeq_yo (Multi Polygon)
6: helsinki_maantiet_l_aeq_paiva (Multi Polygon)
7: helsinki_maantiet_l_aeq_yo (Multi Polygon)
8: helsinki_maantiet_l_den (Multi Polygon)
9: helsinki_maantiet_l_yo (Multi Polygon)
10: helsinki_raideliikenne_l_aeq_paiva (Multi Polygon)
11: helsinki_raideliikenne_l_aeq_yo (Multi Polygon)
12: helsinki_rautatiet_l_yo (Multi Polygon)
13: espoo_kauniainen_maantiet_lden (Multi Polygon)
14: espoo_kauniainen_maantiet_lyo (Multi Polygon)
15: espoo_kauniainen_rautatieliikenne_laeq_paiva (Multi Polygon)
16: espoo_kauniainen_rautatieliikenne_laeq_yo (Multi Polygon)
17: espoo_kauniainen_rautatieliikenne_lden (Multi Polygon)
18: espoo_kauniainen_rautatieliikenne_lyo (Multi Polygon)
19: espoo_kauniainen_tieliikenne_laeq_paiva (Multi Polygon)
20: espoo_kauniainen_tieliikenne_laeq_yo (Multi Polygon)
21: espoo_kauniainen_tieliikenne_lden (Multi Polygon)
22: espoo_kauniainen_tieliikenne_lyo (Multi Polygon)
23: espoo_kauniainen_maantiet_laeq_paiva (Multi Polygon)
24: espoo_kauniainen_maantiet_laeq_yo (Multi Polygon)
25: vantaa_rautatieliikenne_laeq_paiva (Multi Polygon)
26: vantaa_rautatieliikenne_laeq_yo (Multi Polygon)
27: vantaa_rautatieliikenne_lden (Multi Polygon)
28: vantaa_rautatiet_l_yo (Multi Polygon)
29: vantaa_tieliikenne_laeq_paiva (Multi Polygon)
30: vantaa_tieliikenne_laeq_yo (Multi Polygon)
31: vantaa_tieliikenne_lden (Multi Polygon)
32: vantaa_tieliikenne_lyo (Multi Polygon)
33: vantaa_maantiet_laeq_paiva (Multi Polygon)
34: vantaa_maantiet_laeq_yo (Multi Polygon)
35: vantaa_maantiet_lden (Multi Polygon)
36: vantaa_maantiet_lyo (Multi Polygon)
```

Ogr2ogr-ohjelman karttatasolistauksesta huomataan, että Spatialite-tietokanta todellakin sisältää nyt 12 karttatasoa joka alueelta, ja että tasojen nimeämisessä on alueiden välillä pientä horjuntaa, esimerkiksi Helsingin aineistossa käytetään nimiä ”rautatiet” ja ”raideliikenne”, kun taas muissa käytetään samasta aiheesta nimeä ”rautatieliikenne”.

Yksinkertaistus 1: karttatasojen vähentäminen

HRI:n toimitustavassa ei ole mitään vikaa, mutta aineiston loppukäyttäjä saattaa olla kiinnostunut esimerkiksi L_{den}-mallin mukaan lasketuista melutasoista koko pääkaupunkiseudulla ja kaikkien liikennemuotojen aiheuttamasta melusta. Tällainen tarkastelu saattaa olla helpompaa, jos kolmen alueen kolmen eri liikennemuodon aineistot eli yhteensä yhdeksän shapefileä yhdistetään yhdeksi karttatasoksi. **Liitteessä 1** on esitetty ogr2ogr-komennot, joiden avulla voidaan rakentaa alkuperäisistä 36 karttatasosta neljä melulaskentamenetelmän mukaan yhdistettyä aineistoa: Laeq päivä, Laeq yo, Lden ja Lyo. Komennot kirjoittavat lähtöaineistojen alueet ja liikennemeluluokat uusiin ominaisuustietokenttiin, joten yhdistely ei hävitä tätäkään tietoa. Samalla kertaa voidaan yhtenäistää tasojen nimeäminen ja välttää samoja vyöhykkeitä tarkoittavista erilaisista nimistä mahdollisesti koituvat hankaluudet.

Yhdistellyn aineiston tarkastelu

Meluaineiston avaaminen karttaohjelmalla paljastaa, että meluvyöhykkeiden geometriat ovat erittäin suuria ja mutkikkaita. Alkuperäiset aineistot on muodostettu siten, että yhdelle melunvoimakkuusvälille (esimerkiksi 40-45 dB) osuvat alueet esitetään yhdenä ainoana multipolygonina. Suurimmassa aineistoon kuuluvassa multipolygoneissa on 4711 osaa, 5727 saareketta ja 328034 taitepistettä. ”Yksi meluvyöhyke – yksi geometria” –ajattelu sopii hyvin aineiston jakeluun, jättiläismäiset geometriat voivat tehdä kohteiden sijaintiin perustuvat analyysit hitaiksi ja raskaiksi. Esimerkkinä voisi olla vaikkapa vastauksen etsiminen kysymykseen ”Minkä meluvyöhykkeen sisällä on piste, jonka koordinaatit ovat x,y” tai sama kysymys sovitettuna elevään elämään: ”Haluan ostaa äänille herkän puolisoni kanssa asunnon tästä talosta, jonka pihassa nyt seison älypuhelimeni kanssa. Kuinkahan korkea melutaso täällä on päivisin? Jos laskennassa on käsiteltävä 328000 taitepistettä ja tuhansia erillisiä osia ja saarekkeita, niin voi arvata, että tuloksen saamiseen menee enemmän aikaa kuin jos geometriat olisivat yksinkertaisempia. Tosin yksinkertaisin tapaus tässä esimerkissä on asunnonostaja, joka aikoo luottaa enemmän älypuhelinsovellukseensa kuin omiin korviinsa, mutta se on aivan toinen ongelma.

Yksinkertaistus 2: multipolygonien purkaminen

Helppo tapa yksinkertaistaa meluvyöhykkeiden geometrioita on purkaa moniosaiset multipolygonit yksiosaisiksi. Tämän tekemiseksi tarvitsee vain antaa ogr2ogr-ohjelmalle yksi parametri lisää: ”`-explodecollections`”. Seuraavat taulukot antavat tuntumaa tämän toimenpiteen vaikutuksesta.

Seuraavissa taulukoissa esitetään pääkaupunkiseudun ”Laeq päivä” -aineiston tilastotiedot ennen purkamista ja purkamisen jälkeen.

Taso: select * from pks_melu_laeq_paiva: alkuperäinen

 # Kohteita: 85

	Min	Max	keskiarvo	yhteensä
Pisteitä	5	328034	68488	5821500
Saarekkeita	0	5727	954	81167
Osia	1	4711	850	72332

Taso: select * from pks_melu_laeq_paiva: purettu

 # Kohteita: 72332

	Min	Max	keskiarvo	Total
Pisteitä	4	123473	80.48	5821500
Saarekkeita	0	2680	1.12	81167
Osia	1	1	1.0	72332

Aineistoon jäi edelleen jäljelle hurjan mutkikkaita geometrioita (123473 taitepistettä ja 2680 saareketta yhdessä polygonissa), mutta keskimäärin geometriat ovat jo varsin yksinkertaisia (80 taitepistettä ja vähän yli yksi saareke polygonia kohti). Numeroiden perusteella tuntuu todennäköiseltä, että kaikkein suurimmat polygonit kannattaisi vielä pilkkoa pienemmiksi paloiksi, ja tähän tullaankin palaamaan myöhemmin tässä ohjeessa.

Aineiston soveltuvuus ”piste alueen sisällä” kyselyihin

Edellä luonnosteltu asunnonostaja saa olinpaikkansa melutason selville tekemällä kyselyn, joka sanallisesti kuuluu ”Valitse meluvyöhykkeet, joiden sisällä seison”. SQL-kielelle käännettynä kysely on

```
select * from pks_melu_laeq_paiva m
where contains(m.GEOMETRY, GeomFromText('POINT (388837 6683611)'))
```

Jos halutaan hyödyntää Spatialite-tietokannan spatiaali-indeksiä, niin kysely on tehtävä vähän mutkikkaamalla tavalla.

```
select * from pks_melu_laeq_paiva m
where contains(m.GEOMETRY, GeomFromText('POINT (388837 6683611)'))
AND ROWID IN (
SELECT ROWID
FROM SpatialIndex
WHERE f_table_name = 'pks_melu_laeq_paiva'
AND search_frame = MakePoint(388837,6683611));
```

Tässä vaiheessa meillä on kaksi tietokantaa, jotka sisältävät melutiedot hieman erilaisina geometrioina. Toisessa tietokannassa on 85 alkuperäistä hurjan isoa multipolygonia, toisessa 72332 näisä purettua yksinkertaisempaa polygonia. Alla olevassa taulukossa esitetään kyselyiden suoritusajat eräällä tietokoneella mitattuina.

	Hakuaika ilman indeksiä (sek.)	Hakuaika indeksillä (sek.)
Alkuperäiset multipolygonit (85 kpl)	21,1	20,9
Osiin puretut polygonit (72332 kpl)	4,68	2,96

Välijohtopäätökset

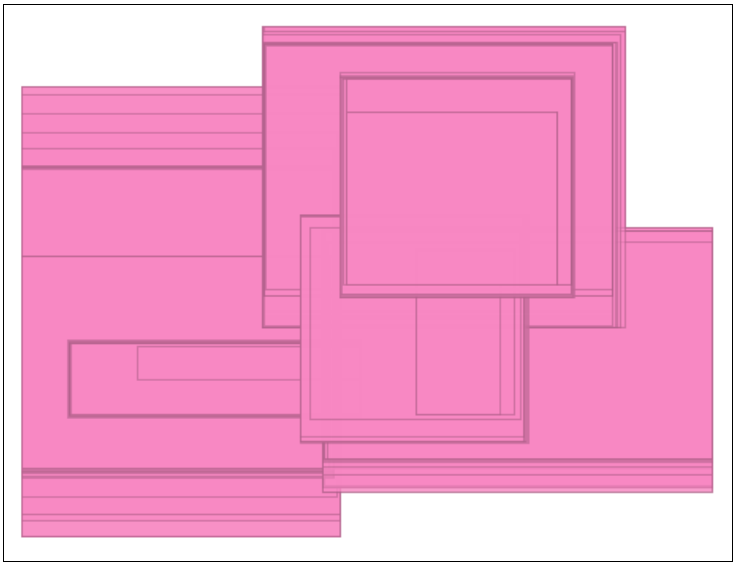
Jättiläismäiset geometriat tekevät selvästikin paikkaan perustuvat haut varsin hitaiksi. Spatiaali-indeksistäkään ei näytä tässä tapauksessa olevan mitään iloa. Jättimultipolygonien purkaminen osiin parantaa tilannetta selvästi, ja spatiaali-indeksilläkin alkaa olla vaikutusta. Silti 3-5 sekunnin suoritusaika tuntuu vasta siedettävältä, mutta ei mitenkään henkeäsalpaavan hyvältä.

Kurkistus spatiaaali-indekseihin

Spatialiten spatiaali-indeksi perustuu siihen, että jokaisen kohteen ympärille piirretään pienin mahdollinen suorakaiteen muotoinen laatikko, joka tallennetaan tietokantaan indeksitauluun. Tämä saattaa tuoda hakuihin lisää nopeutta, sillä neljästä nurkkapisteestä tehdyille laatikoille on nopeampi tehdä paikkaan perustuvia analyysejä kuin monimutkaisille geometrioille. Tarkoituksena on käyttää laatikoita apuna esivalinnassa, jonka perusteella voidaan nopeasti hylätä ne kohteet, jotka eivät varmasti tule mukaan tulosjoukkoon. Ohjelma päättelee “Jos tämän kohteen pakkauslaatikko on hakualueen ulkopuolella, niin kyllä sitten itse kohdekin on ulkopuolella”. Jäljelle jäävissä pakkauslaatikoissa on sitten kohteet, jotka mahdollisesti kuuluvat tulosjoukkoon. Tämä ei kuitenkaan ole varmaa, sillä yleensä kohteet ovat enemmän tai vähemmin väljästi laatikkonsa sisällä. Esimerkiksi laatikon nurkasta nurkkaan menevä joki täyttää vain hyvin pienen osan laatikostaan, ja siitä syystä esivalinnassa seulaan jääneille ehdokkaille on tehtävä tarkemmat testit, joissa käytetään kohteen oikeaa geometriaa.

Alla olevat kuvat esittävät edellisen esimerkin Laeq_paiva -tason indeksilaatikoita, jotka on luotu Spatialite-tietokannassa kyselyllä

```
create table bbox as
select rowid, Envelope(geometry) as geometry
from pks_melu_laeq_paiva
```



Kuva 1. Pääkaupunkiseudun meluaineistojen purkamattomien multipolygonien indeksilaatikot

Kuva 2. Pääkaupunkiseudun meluaineistoista purettujen polygonien indeksilaatikot

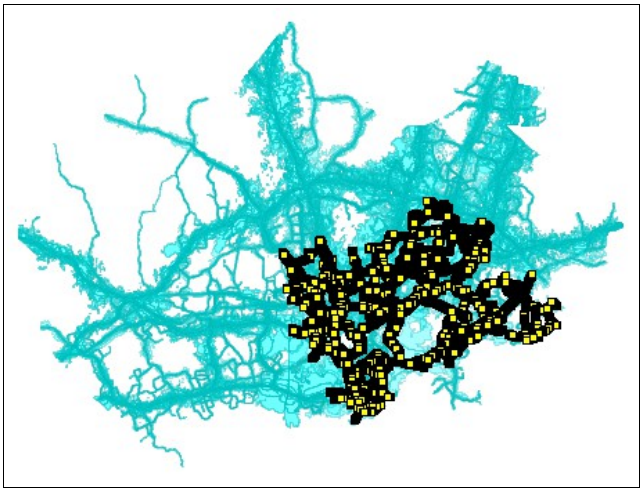
Kuvista voidaan päätellä, että purkamattomien geometrioiden tapauksessa indeksin käyttö on melkoisen turhaa, koska sillä tavalla ei voida sulkea pois edes kaikkia naapurikaupunkien alueella olevia kohteita, koska laatikot menevät limittäin. Puretuilla polygoneilla sen sijaan on melko paljon pieniä indeksilaatikoita, jotka voidaan usein hylätä pienelle alueelle kohdistuvassa indeksihaussa. Ikävä kyllä jäljellä on myös hyvin suuria laatikoita, jotka tahtovat väkisinkin jäädä jäljelle indeksihaussa, ja joiden sisällä on juuri ne kaikkein suurimmat, mutkikkaimmat ja hitaimmin laskennassa käsiteltävät geometriat.

Johtopäätös: Tämä aineisto ei ole vieläkään oikein hyvä nopeaan “piste alueen sisällä” -analyysiin.

Geometrioiden yleistäminen ja miksi emme käytä sitä

Geometrioiden käsittelyssä jokainen taitepiste on otettava laskennassa huomioon, ja siitä syystä taitepisteiden vähentäminen nopeuttaa toimintoja. Yleistämisellä puolestaan tarkoitetaan menetelmiä, joilla taitepisteitä pyritään vähentämään niin, ettei kohteiden muoto muutu liian paljon. Jos esimerkiksi ajatellaan, ettei meluvyöhykkeiden kohdalla ole niin tarkkaa, onko vyöhykkeen raja viisi metriä tuonnempana tai tännempänä, niin ohjelmaa voidaan käskeä poistamaan taitepiste, jos reunaviiva ei siitä syystä siirry yli viittä metriä alkuperäiseltä paikaltaan. Koska taitepisteiden suuri määrä on ainakin osasyy meluaineiston raskauteen, niin mieleen tulee kokeilla yleistämistä. Kokeillaan ja katsotaan, mitä tapahtuu.

Kuvassa 3 on vielä moniosaisten multipolygonien purkamisen jälkeenkin jäljelle jäänyt monsteri, joka peittää yli puolet Helsingistä ja sisältää 123473 taitepistettä ja 2018 reikää.



Kuva 3. Laeq_paiva tason mutkikkain polygoni “Helsinki – tieliikenne – 65-70 dB”

Seuraava kuva osoittaa, mitä sivuvaikutuksia voi olla lähes jokaisesta paikkatieto-ohjelmasta löytyvällä yleistystoiminnolla. Yleistäminen viiden metrin toleranssilla kyllä poistaa 70 prosenttia taitepisteistä ja siten keventää geometrioita huomattavasti, mutta ikävä kyllä toimenpide poraa karttatasolle reikiä. Tämä johtuu siitä, että topologian säilyttävä yleistys (SimplifyPreserveTopology) säilyttää kyllä yksittäisten polygonien topologian eli ei poista niistä yhtäkään reikää, mutta se ei säilytä välttämättä koko karttatason topologiaa estämällä aukkojen ja päällekkäisyyksien syntymisen. Ja aukkojahan me emme voi tässä esimerkissä hyväksyä missään tapauksessa, koska silloin asunnonostaja voisi saada kännykkäänsä vastauksen, ettei kyseisessä kohdassa ole ääntä lainkaan.

```
create table simple as
sele SimplifyPreserveTopology(geometry) as geometry
from pks_melu_laeq_paiva
```

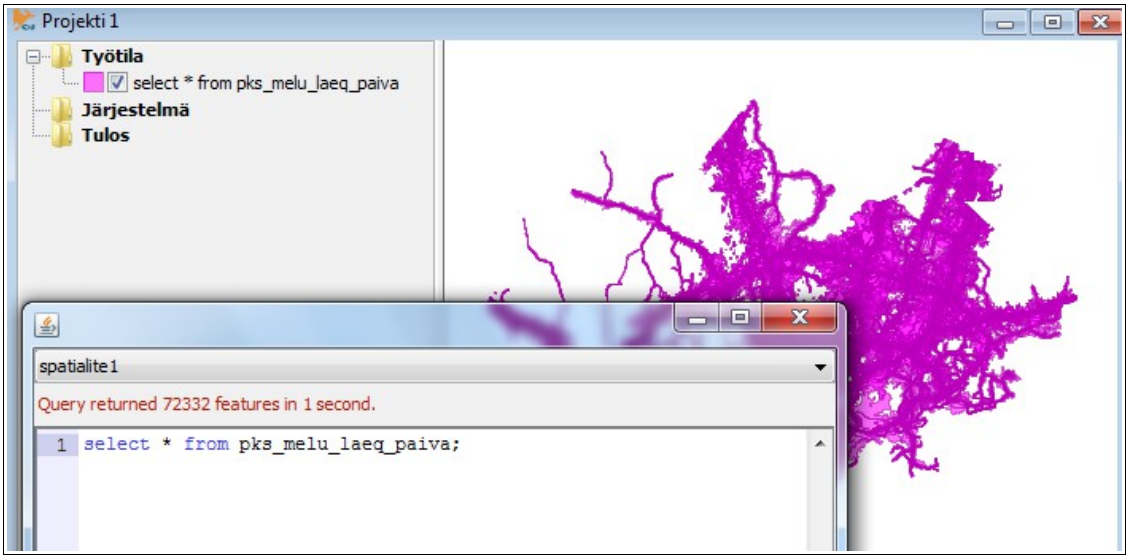
Kuva 4. Yleistetyt geometriat eivät enää peitä koko aluetta. Sinisillä alueilla ei enää ole tietoa melutasosta.

Mainittakoon, että paikkatieto-ohjelmista, myös avoimen lähdekoodin ohjelmistoista, löytyy aluetopologian säilyttäviäkin yleistysmenetelmiä. Tässä tapauksessa emme vaivaudu tutkimaan niitä, koska tehokkuusongelmamme johtuu myös maantieteellisesti liian suurista geometriosta, jotka tuhoavat spatilaali-indeksin tehon, eikä yleistäminen poista tätä ongelmaa.

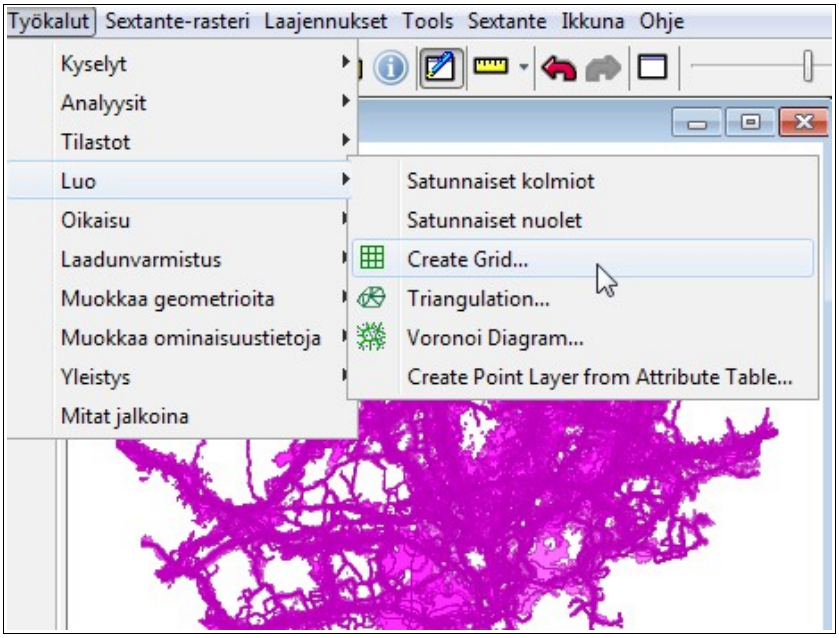
Yksinkertaistus 3: geometrioiden pienentäminen

Seuraavat kuvaruutukaappaukset esittävät, kuinka geometriat voidaan pilkkoa OpenJUMP-ohjelmalla. Työvaiheet ja niiden suorittamiseen mittausten mukaan kulunut ovat seuraavat:

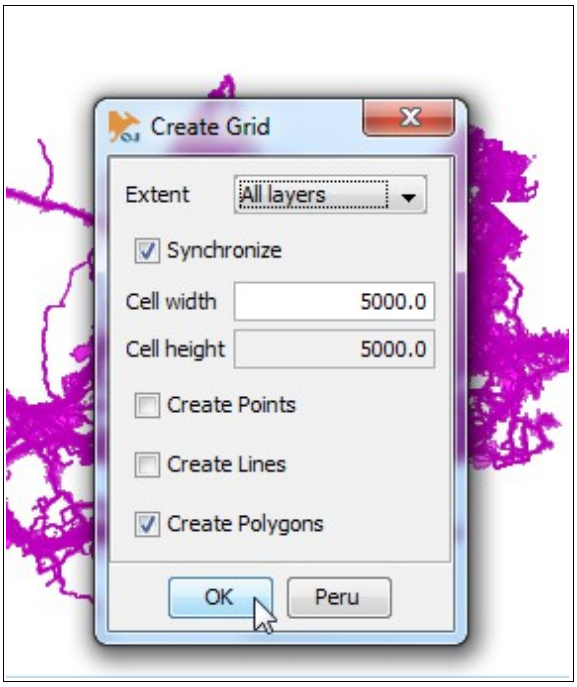
- Avataan lähtöaineisto, tässä tapauksessa Spatialite-tietokannasta (1 sekunti).
- Luodaan pikkomista varten hila, tässä esimerkissä 5 x 5 km ruutukoolla (muutama sekunti).
- Pilkotaan geometriat siten, että hilan viivat katkaisevat ne kaikista leikkauskohdista (2 minuuttia 4 sekuntia Laeq_paiva -tasolle, muilla tasoilla vähemmän).
- Tallennetaan shapefile-muotoon ja tuodaan shapefile Spatialite-tiedokantaan Spatialite-gui -ohjelmalla, koska OpenJUMP ei valitettavasti pysty kirjoittamaan Spatialiteen (tallennus 5 sekuntia, Spatialiteen tuonti 60 sekuntia).



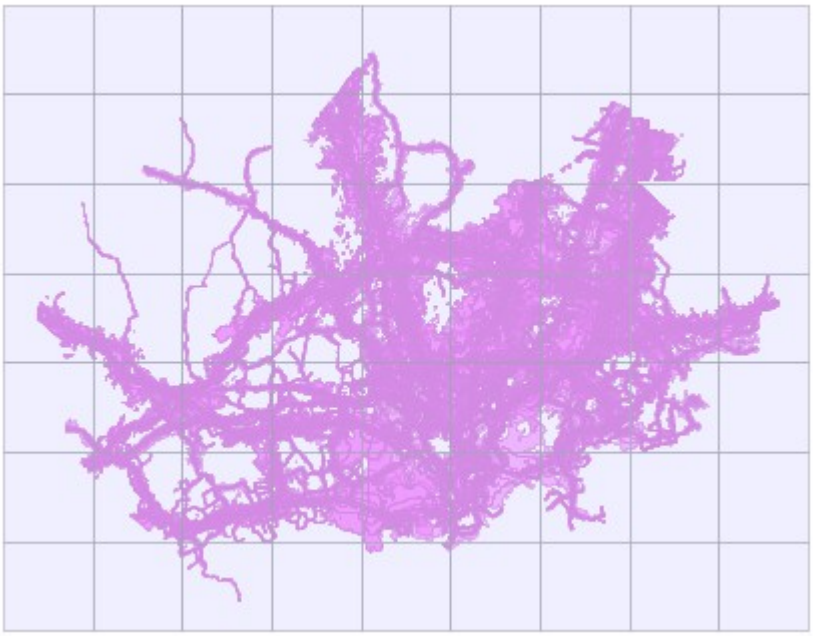
Kuva 5. OpenJUMP:in DB Query -laajennos lukee Spatialite-tietokantaa, eikä vauhti ole ollenkaan huono.



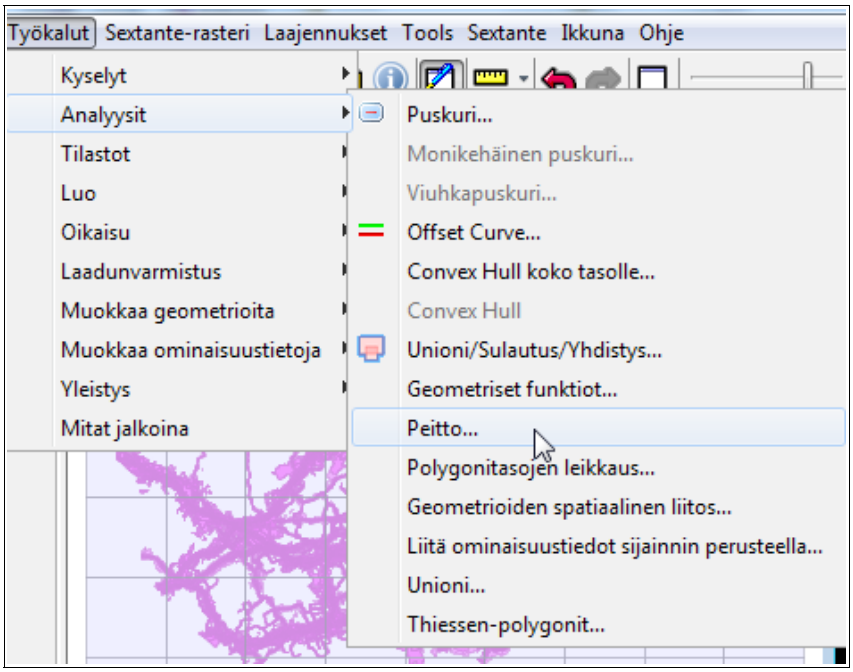
Kuva 6. OpenJUMP Plus -versiossa on työkalu hilojen luomista varten.



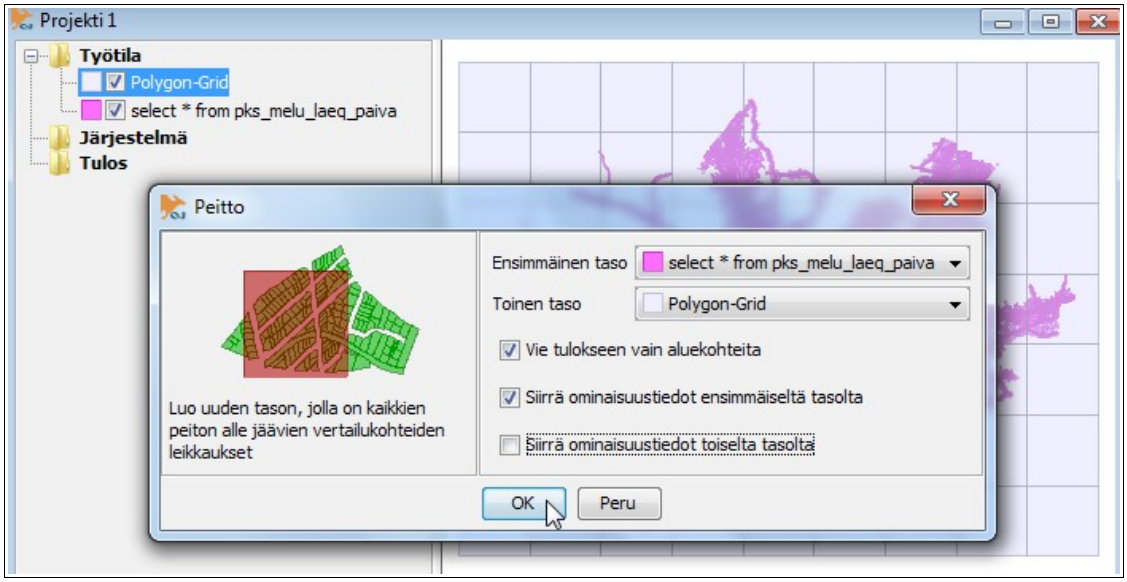
Kuva 7. Kaikki karttatasot kattavat 5 x 5 km ruudukon luominen OpenJUMP:lla.



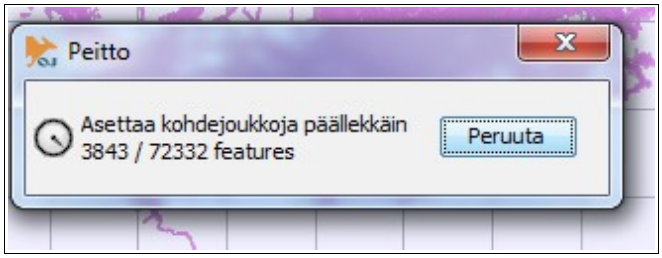
Kuva 8. Valmis hila.



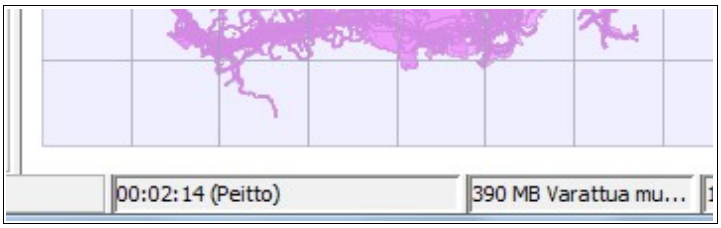
Kuva 9. Geometrioiden silppuamiseen käytetään Peitto-nimistä funktiota.



Kuva 10. Sopivat silppuamisasetukset tätä mallisuoritusta varten.



Kuva 11. Silppuaminen menossa.



Kuva 12. Silppuaminen suoritettu kohtuullisessa ajassa.

Kuva 13. Pilkotulta tasolta valittu kohde näyttää selvästi kuinka se on katkaistu hilaruudukon mukaan.

Pilkotun aineiston tarkastelu

Tarkastellaan pilkottua aineistoa ensin tason tilastotietojen avulla. Niistä huomataan, että suurimmassa polygonissa on enää 22492 taitepistettä ja 513 saareketta. Tämäkään ei tunnu kovin vähältä, mutta on se sentään selvästi vähemmän kuin ennen.

Taso: select * from pks_melu_laeq_paiva: pilkottu

 # Kohteita: 76370

	Min	Max	Keskiarvo	Yhteensä
Pisteitä	4	22492	76.40	5835126
Saarekkeita	0	513	41000	79705
Osia	1	1	1.0	76370

Leikatun tason indeksilaatikoissa ei leikkauksen tekotavan vuoksi ole mukana yhtäkään yli 5 x 5 kilometrin suuruista laatikkoa.

Kuva 14. Pilkotun polygoniaineiston indeksilaatikkojen jakauma.

Teoriassa aineiston on nyt nopeampi, mutta entä käytännössä?

Alla olevaan taulukossa on vertailun vuoksi mukana myös aikeisempien mittausten tulokset.

	Hakuaika ilman indeksiä (sek.)	Hakuaika indeksillä (sek.)
Alkuperäiset multipolygonit (85 kpl)	21,1	20,9
Osiin puretut polygonit (72332 kpl)	4,68	2,96
Pienennetyt polygonit (76370 kpl)	2,18	0,16

Johtopäätös: Hakuaika on alle 1 prosentti alkuperäisestä . Lopputulos täyttää Latuviitta.org:in korkeat laatuvaatimukset.

Melutiedot WFS-palvelusta

Seuraavassa esitetään vielä, kuinka asunnonostajan tiedonhalu voidaan tyydyttää WFS-kyselyllä. Tämän ohjeen osan tarkoituksena on herättää mielenkiintoa WFS:n käyttämiseksi ja osoittaa, että tällainen kyselypalvelu voidaan aivan hyvin toteuttaa käyttämällä standardia WFS-palvelua ilman mitään erityisesti tätä tarkoitusta varten tehtävää melutietorajapintaa. Kaikki tehtävän suorittamiseen tarvittavat WFS-pyynnöt esitetään kevyesti kommentoituina, mutta mikään varsinainen WFS-opas tämä ei ole, vaan perustiedot on opiskeltava muualta mielenkiinnon heräämisen jälkeen.

Aikaisemmin kuvatulla pilkkomismenetelmällä luodut meluvyöhykeaineistot on viety Latuviitan WFS-palveluun http://hip.latuviitta.org. Aineistot ovat palvelussa nimillä

- lv:pks_melu_laeq_paiva
- lv:pks_melu_laeq_paiva
- lv:pks_melu_den
- lv:pks_melu_yo

WFS-tasojen rakenne saadaan selville pyynnöllä DescribeFeatureType. Pyyntö voidaan lähettää vaikkapa selaimella.

```
http://hip.latuviitta.org/cgi-bin/tinyows?
service=WFS&version=1.1.0&request=DescribeFeatureType&typeName=lv:pks_melu_laeq_
paiva
```

Palvelu lähettää vastaukseksi XML-skeeman, josta selviää, mitä ominaisuustietoja kyseiselle WFS-tasolla on.

```
<xs:schema targetNamespace="http://latuviitta.fi/"
elementFormDefault="qualified" version="1.1">
<xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
<xs:element name="pks_melu_laeq_paiva" type="lv:pks_melu_laeq_paivaType"
substitutionGroup="gml:_Feature"/>
<xs:complexType name="pks_melu_laeq_paivaType">
<xs:complexContent>
<xs:extension base="gml:AbstractFeatureType">
<xs:sequence>
<xs:element name="wkb_geometry" type="gml:SurfacePropertyType" nillable="true"
minOccurs="0" maxOccurs="1"/>
<xs:element name="db_lo" type="double" nillable="true" minOccurs="0"
maxOccurs="1"/>
<xs:element name="db_hi" type="double" nillable="true" minOccurs="0"
maxOccurs="1"/>
<xs:element name="alue" type="string" nillable="true" minOccurs="0"
maxOccurs="1"/>
<xs:element name="melutyyppi" type="string" nillable="true" minOccurs="0"
maxOccurs="1"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:schema>
```

Käyttämällä näitä tietoja yhdessä yleisen WFS-tietämyksen kanssa voidaan pyytää palvelua lähettämään annetun paikan meluvyöhykkeen melun alaraja ”db_lo” ja melulajin nimi ”melutyyppi”. Paikkan perustuva kysely tehdään käyttämällä Filter-elementtiä ja siinä Contains-funktiota. Selvällä suomen kielellä tämän suodattimen merkitys on ”Valitse kohteet, jotka sisältävät pisteen, jonka koordinaatiti ovat 388837,6683611”. Koordinaattijärjestelmänä tässä kyselyssä on suomalainen ETRS-TM35FIN, jonka EPSG-koodi on 3067.

```
<wfs:GetFeature xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml" xmlns:wfs="http://www.opengis.net/wfs"
service="WFS" version="1.1.0" maxFeatures="1000"
outputFormat="text/xml; subtype=gml/3.1.1">
<wfs:Query xmlns:lv="http://latuviitta.fi/" srsName="urn:ogc:def:crs:EPSG::3067"
typeName="lv:pks_melu_laeq_paiva">
<wfs:PropertyName>lv:db_lo</wfs:PropertyName>
<wfs:PropertyName>lv:alue</wfs:PropertyName>
<ogc:Filter>
<ogc:Contains>
<ogc:PropertyName>lv:wkb_geometry</ogc:PropertyName>
<gml:Point srsName="urn:ogc:def:crs:EPSG::3067">
<gml:pos srsDimension="2">388837 6683611</gml:pos>
</gml:Point>
</ogc:Contains>
</ogc:Filter></wfs:Query></wfs:GetFeature>
```

Vastaukseksi melukyselyyn saadaan jälleen XML:ää. Sitä tulee pitkät pätkät, mutta jostain sieltä kyllä löytyy vastaus siihen varsinaiseen kysymykseenkin: kyseisellä paikalla on maantie- ja tieliikennemelua reilut 45 desibeliä kumpaakin.

```
<?xml version='1.0' encoding='UTF-8'?>
<wfs:FeatureCollection
 xmlns:tows='http://www.tinyows.org/'
 xmlns:lv='http://latuviitta.fi/'
 xmlns:wfs='http://www.opengis.net/wfs'
 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
 xmlns:gml='http://www.opengis.net/gml'
 xmlns:xsd='http://www.w3.org/2001/XMLSchema'
 xmlns:ogc='http://www.opengis.net/ogc'
 xmlns:xlink='http://www.w3.org/1999/xlink'
 xmlns:ows='http://www.opengis.net/ows'
 xsi:schemaLocation='http://www.tinyows.org/
    http://188.64.1.61/cgi-bin/tinyows?
service=WFS&amp;version=1.1.0&amp;request=DescribeFeatureType&amp;Typename=lv:pk
s_melu_laeq_paiva   http://www.opengis.net/wfs
   http://schemas.opengis.net/wfs/1.1.0/wfs.xsd
   http://www.opengis.net/gml
   http://schemas.opengis.net/gml/3.1.1/base/gml.xsd'
>
<gml:boundedBy>
  <gml:Envelope
srsName="urn:ogc:def:crs:EPSG::3067"><gml:lowerCorner>6680158.000000
388106.750000</gml:lowerCorner><gml:upperCorner>6684395.500000
390000.000000</gml:upperCorner></gml:Envelope>
</gml:boundedBy>
  <gml:featureMember>
   <lv:pks_melu_laeq_paiva gml:id="pks_melu_laeq_paiva.12107">
   <lv:db_lo>45</lv:db_lo>
   <lv:alue>Helsinki</lv:alue>
   <lv:melutyyppi>tieliikenne</lv:melutyyppi>
   </lv:pks_melu_laeq_paiva>
  </gml:featureMember>
  <gml:featureMember>
   <lv:pks_melu_laeq_paiva gml:id="pks_melu_laeq_paiva.53069">
   <lv:db_lo>45</lv:db_lo>
   <lv:alue>Helsinki</lv:alue>
   <lv:melutyyppi>maantiet</lv:melutyyppi>
   </lv:pks_melu_laeq_paiva>
  </gml:featureMember>
</wfs:FeatureCollection>
```

Nopeammin, paremmin ja halvemmalla eli tulosten tarkastelu

Nopeammin

Nopeustarkastelu voidaan tällä kertaa tehdä kahdelle asialle: aineiston muokkaamiseen kuluneelle ajalle ja vasteajalle, kun muokatusta aineistosta etsitään melutasoa ”piste alueen sisällä” -kyselyllä.

Aineiston muokkaus tehtiin kannettavalla tietokoneella, jossa on 64-bittinen Windows 7 -käyttöjärjestelmä ja Intel i7 -prosessori. Kaikki tiedostojen käsittely tehtiin ulkoisella USB 2 -kovalevyllä. Käytetyt ohjelmistoversiot olivat: GDAL 1.10 (kehitysversio 10. joulukuuta 2012) , OpenJUMP 1.6 (kehitysversio 10. joulukuuta 2012), 32-bittinen Java 1.6.0_37 , Spatialite-gui 1.6.

- Zip-tiedostoihin kuuluvien 36 shapefilen vieminen Spatialite-tietokantaan ogr2og-ohjelmalla: **46 sekuntia**
- Karttatasojen yhdistely yhdestä Spatialitekannasta toiseen ogr2ogr-ohjelmalla purkamatta moniosaisia multipolygoneja: **27 sekuntia**
- Karttatasojen yhdistely yhdestä Spatialitekannasta toiseen ogr2ogr-ohjelmalla, samalla puretaan moniosaiset multipolygonit: **8 minuuttia 30 sekuntia**
- Suurten polygonien pilkkominen OpenJUMP-ohjelmalla: noin **10 minuuttia**
- Pilkottujen polygonien lukeminen uuteen Spatialite-kantaan shapefile-muodosta: noin **4 minuuttia**
- Kaikki työvaiheet yhteensä: **24 minuuttia**

Hitaimpia työvaiheita näyttävät siis olevan moniosaisten multipolygonien purkaminen osiinsa (yli 8 minuuttia) sekä purettujen polygonien pilkkominen (10 minuuttia). Vertailua muihin ohjelmistoihin ei tehty, joten jää toisten asiaksi selvittää, onko tämän mallisuorituksen menetelmä nopea vai hidas. Näin pienen aineiston yhteydessä ajankäytöllä ei ole juuri merkitystä, koska muunnos tarvitsee tehdä vain kerran, ja vaikka se tapahtuisi silmänräpäyksessä, niin aikaa säästyisi vain 24 minuuttia. Tilanne on toinen, jos aineisto olisi vaikka tuhat kertaa suurempi, koska tasaisen vauhdin taulukon mukaan muunnos kestäisi silloin noin 17 vuorokautta.

Muunnoksen jälkeen ”piste alueen sisällä” -kysely Spatialite-tietokannassa kesti 0,16 sekuntia. Nopeutta ei verrattu mihinkään muihin vaihtoehtoihin, joten tämä tulos asettaa riman aloituskorkeuteen muita varten. Karkeana yleistyksenä voidaan varmaan sanoa, että jos vastaavan kyselyn teko liikennemeluaineistosta kestää yli puoli sekuntia, niin systeemi ei toimi parhaalla mahdollisella tavalla siinä tarkoituksessa. Tämän ei tarvitse tarkoittaa, että koko systeemi olisi susi, koska kaikkia asioita ei voi optimoida yhtä aikaa, eikä hakunopeus ole kaikkein tärkein tarve.

Paremmin

Menetelmän parantamisessa suurimmat mahdollisuudet lienevät käyttömukavuudessa. Mallisuorituksessa käytettiin kolmea eri ohjelmaa, mikä vaatii toisaalta – ohjeista huolimatta – jonkinlaista osaamista niistä kaikista tai sitten hieman uteliaisuutta ja seikkailuhenkeä. Toisaalta siirtyminen ohjelmasta toiseen vie aikaa ja saattaa johtaa virheisiin.

Lopputuloksen laadussa ei huomattu mitään moittimista, vaan laatu vastaa alkuperäistä aineistoa.

Halvemmalla

Kuten muissakin Latuviitan mallisuorituksissa, lähtöaineistot ovat avoimia ja käytetyt ohjelmistot ovat vapaasti saatavia avoimen lähdekoodin ohjelmistoja, joten rahapussia ei tarvitse avata. Koska harrastukseen käytettyä aikaa ei ole tapana laskea, niin amatöörille kokonaissuoritus on ilmainen. Mutta jos jonkun mieleen tulisi tehdä jotain vastaavaa muutenkin kuin omaksi huvikseen, niin kustannukset kertyvät epäsuorasti edellä kerrotuista asioista, eli työajan käytöstä ja siitä, että usean ohjelmiston käytön opettelemiseen menee siihenkin aikaa.

Harjoituksen varsinainen opetus: Etsi todelliset pullonkaulat

Muutama sivu sitten huomattiin, kuinka aineistoa muokkaamalla saatiin tiettyyn kyselyyn yli sata kertaa enemmän vauhtia ja tämän parannuksen aikaansaamiseksi tarvittiin 14 minuuttia prosessointiaikaa eikä mitään muuta. Laitteiston päivittäminen järeämmäksi tai lisänopeuden etsiminen ohjelmistoja vaihtamalla olisi tässä tapauksessa ollut aivan varmasti ajan ja rahan haaskausta.

Kiitokset

Tässä ohjeessa selostettu mutkikkaiden polygonien pilkkomismenetelmä on sovellettu Alessandro Furierin artikkelista http://www.gaia-gis.it/spatialite-3.0.0-BETA1/WorldBorders.pdf

Kuva 15. Melukyselyn koordinaatit osuivat Helsingin Siltamäkeen.

Liite 1.

Komentojono, joka muuntaa ohjeen sivulla 1 luodun tietokannan ”melu_raaka.sqlite” 36 taulua neljäksi yhdistelmätauluksi tietokantaan ”pks_melu_2.sqlite” ja purkaa samalla moniosaiset multipolygonit yksinkertaisiksi. Komennon suoritusaika vuoden 2012 mittapuun mukaan hyvällä kannettavalla tietokoneella on noin 9 minuuttia.

```
ogr2ogr -f SQlite -dsco spatialite=yes pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'tieliikenne' as melutyyppi
from espoo_kauniainen_tieliikenne_laeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'tieliikenne' as melutyyppi from
helsinki_tieliikenne_laeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'tieliikenne' as melutyyppi from
vantaa_tieliikenne_laeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'maantiet' as melutyyppi
from espoo_kauniainen_maantiet_laeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'maantiet' as melutyyppi from
helsinki_maantiet_l_aeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'maantiet' as melutyyppi from
vantaa_maantiet_laeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'rautatieliikenne' as
melutyyppi from espoo_kauniainen_rautatieliikenne_laeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'rautatieliikenne' as melutyyppi
from helsinki_raideliikenne_l_aeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_paiva -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'rautatieliikenne' as melutyyppi from
vantaa_rautatieliikenne_laeq_paiva"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'tieliikenne' as melutyyppi
from espoo_kauniainen_tieliikenne_laeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'tieliikenne' as melutyyppi from
helsinki_tieliikenne_laeq_yo"
```

```
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'tieliikenne' as melutyyppi from
vantaa_tieliikenne_laeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'maantiet' as melutyyppi
from espoo_kauniainen_maantiet_laeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'maantiet' as melutyyppi from
helsinki_maantiet_l_aeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'maantiet' as melutyyppi from
vantaa_maantiet_laeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'rautatieliikenne' as
melutyyppi from espoo_kauniainen_rautatieliikenne_laeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'rautatieliikenne' as melutyyppi
from helsinki_raideliikenne_l_aeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_laeq_yo -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'rautatieliikenne' as melutyyppi from
vantaa_rautatieliikenne_laeq_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'tieliikenne' as melutyyppi
from espoo_kauniainen_tieliikenne_lden"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'tieliikenne' as melutyyppi from
helsinki_tieliikenne_l_den"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'tieliikenne' as melutyyppi from
vantaa_tieliikenne_lden"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'maantiet' as melutyyppi
from espoo_kauniainen_maantiet_lden"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'maantiet' as melutyyppi from
helsinki_maantiet_l_den"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'maantiet' as melutyyppi from
vantaa_maantiet_lden"
```

```
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'rautatieliikenne' as
melutyyppi from espoo_kauniainen_rautatieliikenne_lden"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'rautatieliikenne' as melutyyppi
from helsinki_rautatiet_lden"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lden -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'rautatieliikenne' as melutyyppi from
vantaa_rautatieliikenne_lden"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'tieliikenne' as melutyyppi
from espoo_kauniainen_tieliikenne_lyo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'tieliikenne' as melutyyppi from
helsinki_tieliikenne_l_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'tieliikenne' as melutyyppi from
vantaa_tieliikenne_lyo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'maantiet' as melutyyppi
from espoo_kauniainen_maantiet_lyo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'maantiet' as melutyyppi from
helsinki_maantiet_l_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'maantiet' as melutyyppi from
vantaa_maantiet_lyo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Espoo-Kauniainen' as alue, 'rautatieliikenne' as
melutyyppi from espoo_kauniainen_rautatieliikenne_lyo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Helsinki' as alue, 'rautatieliikenne' as melutyyppi
from helsinki_rautatiet_l_yo"
ogr2ogr -f SQlite -append -update pks_melu_3.sqlite melu_raaka.sqlite
-explodecollections -gt 5000 -nlt POLYGON -nln pks_melu_lyo -sql "select
geometry, db_lo, db_hi, 'Vantaa' as alue, 'rautatieliikenne' as melutyyppi from
vantaa_rautatiet_l_yo"
```