OSM data in MariaDB/MySQL
All the world in a few large tables

Well, almost ...

Hartmut Holzgraefe

SkySQL AB *hartmut@skysql.com*

February 1, 2014

Overview

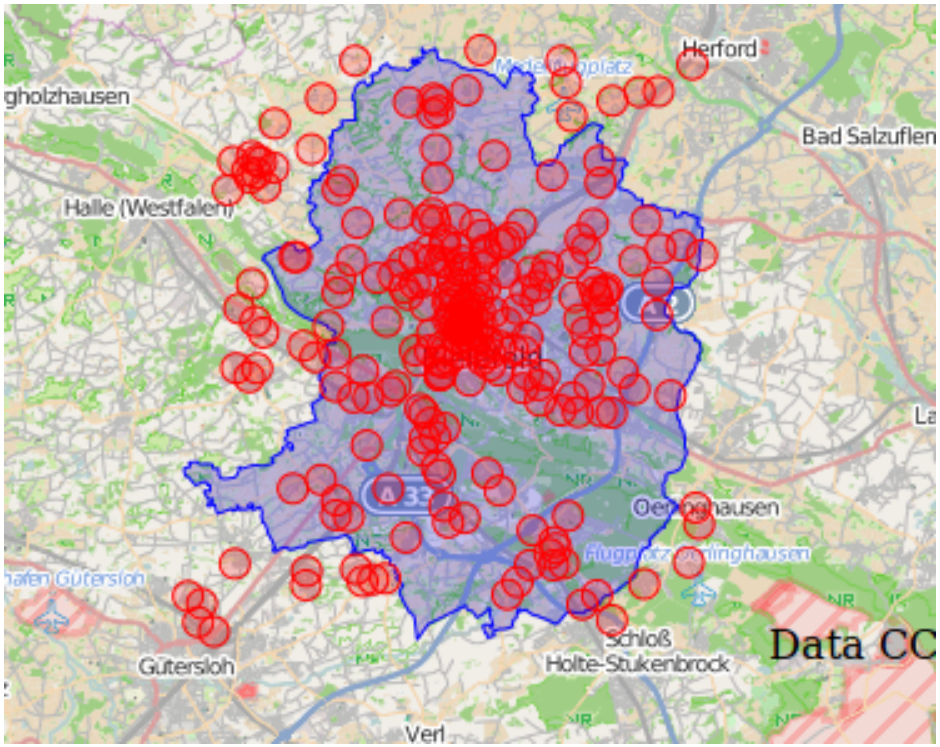
MySQL, MariaDB and GIS

History

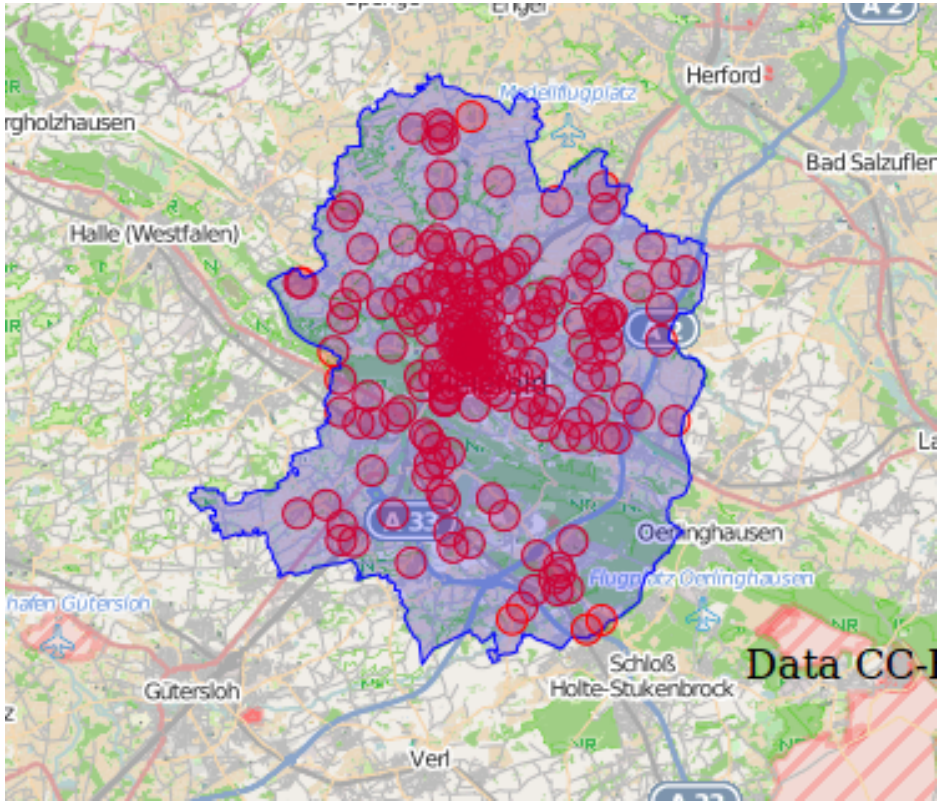
- First appeard in MySQL 4.1 (2004)
- ... with MBR relations only
- Lab Release adds true spatial relations (200?)
- MariaDB 5.3 GA with true spatial relations (2011)
- MySQL 5.6 GA with true spatial relations (2013)
- ... to be continued ...

MBR is not enough

MBR CONTAINS()



True ST_CONTAINS()



Current Status

- Spatial relations work
- ... but the world is still flat (no projections)
- OK for many use cases
- ... but be aware of gotchas like DISTANCE()
- GEOMETRY types in all storage engines
- ... but only MyISAM has SPATIAL indexes

MariaDB 10.1 Roadmap

- System Tables (`spatial_ref_sys`, `geometry_columns`, ...)
- Precision math calculations and storage
- Coordinate transformations / projections
- 3rd coordinate (e.g. for altitude)
- all spatial functions required by OGC
- spatial aware optimizer

... and beyond

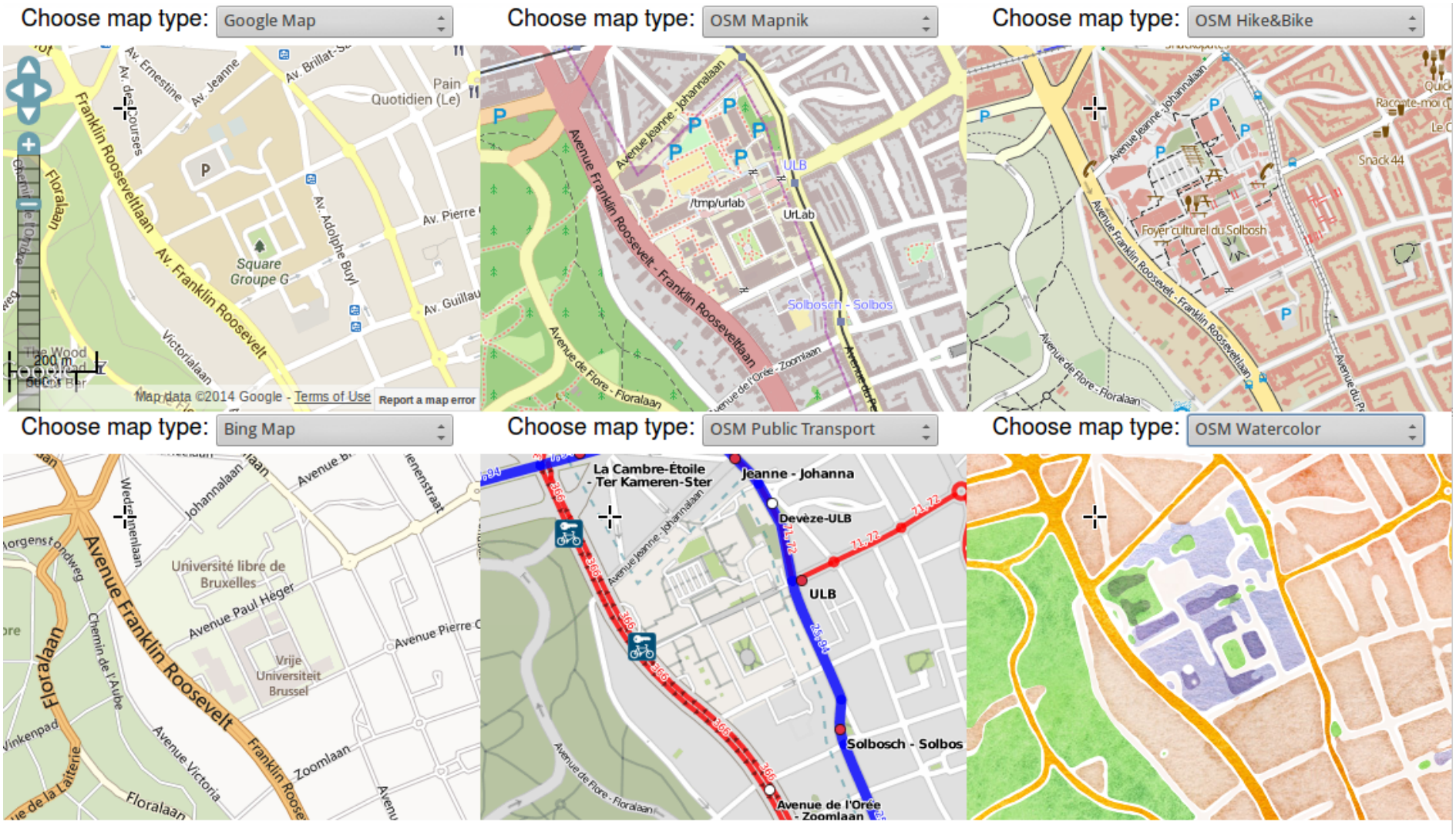
- SPATIAL indexes in other storage engines
- 3D calculations
- client side support for GIS transformations

Openstreetmap

OpenStreetMap History

- founded in 2004 by Steve Coast
- data under open license (CC-BY-SA first, now ODBL)
- 1.5 million contributors
- 2 billion map nodes
- 200 million ways
- 2 million relations
- almost 4 billion GPX points

Pretty Tiles



... and raw data

Raw map data can be used for other things, too:

- for routing
- for coverage checks
- for flight simulators
- for science

Core Data Model

Just three simple things

- Nodes (Points)
- Ways
- Relations

Nodes

Nodes describe a single point at a specific location using:

- A numeric ID
- Object version, Timestamp of last change, User
- Node coordinates
- Node attributes as key/value pairs

Ways

Ways form an open or closed line by connecting nodes, using:

- A numeric ID
- Object version, Timestamp of last change, User
- An ordered list of node IDs
- Way attributes as key/value pairs

Relations

Relations bundle objects to describe more complex relations, using:

- A numeric ID
- Object version, Timestamp of last change, User
- Ordered lists of member nodes, ways and sub-relations
- Optional member roles
- Attributes as key/value pairs

Data Access

- The main database is not exposed directly
- Only one central instance, accessible via “The API”
- API meant for editor applications only

- Full data export once a week (“the planet”)
- Plus daily, hourly, minutely diffs
- Two file formats for planets:
- .osm XML based, usually bz2 compressed (32GB packed, 400GB unpacked)
- .pbf compact binary format based on Google ProtoBuf (23GB)
- Regional extracts available by 3rd parties, e.g. GeoFabrik.de

Data Import

- The raw data is not really suitable for most purposes, esp. rendering
- Several import/preprocessing tools provide more convenient schemas, e.g. by
- ... only extracting certain attributes
- ... making a difference beween ways and areas
- ... resolving relations into simpler objects

Besides `osm2pgsql` that I'm about to talk about in a minute there are also `impOsm`, ...

osm2pgsql

osm2pgsql

osm2pgsql is a tool

- written in C
- with a small C++ part now
- reads OSM data
- preprocesses it
- stores results in relational tables
- originally in PostGIS only

Block Diagram

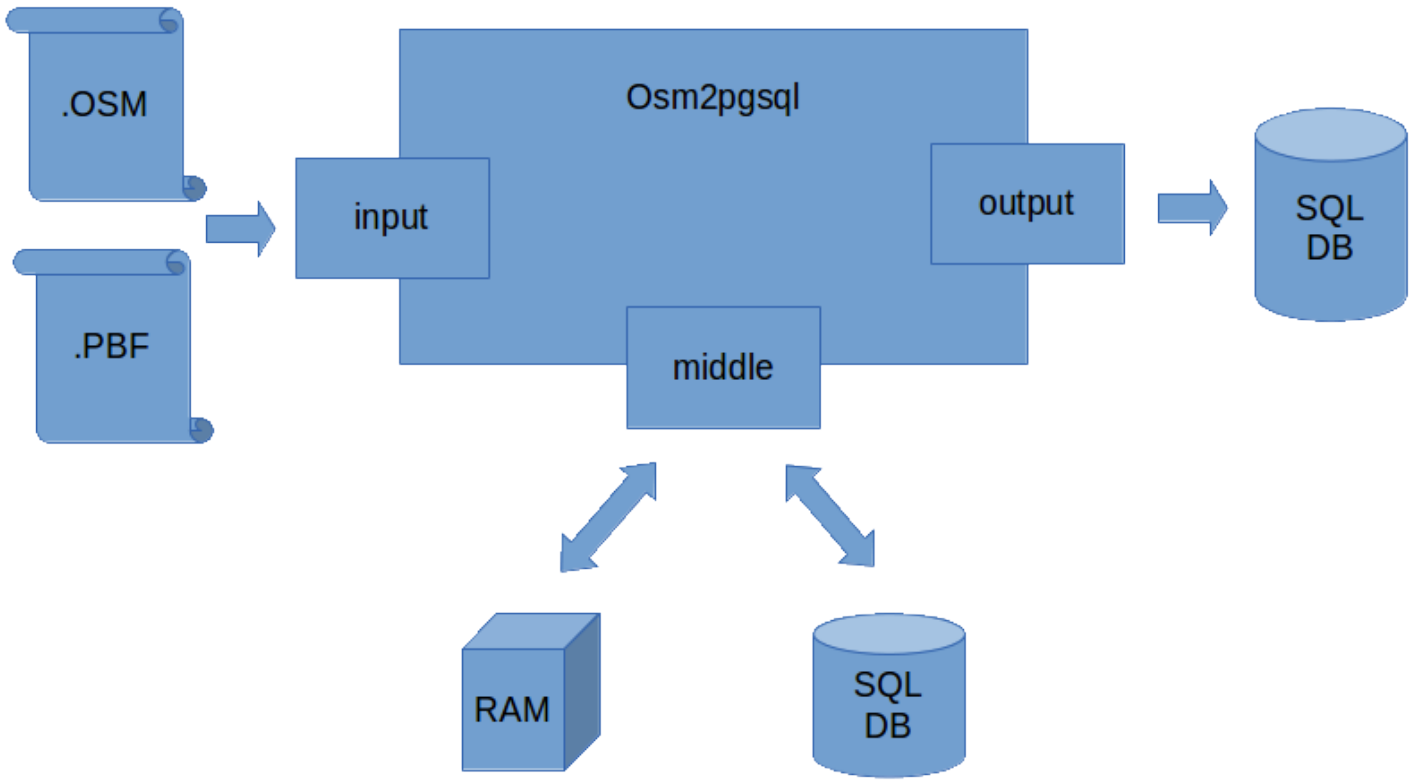


Figure : osm2pgsql block diagram

Data Model Again

- *prefix*_point for single node POIs
- *prefix*_line for linear 2D objects like roads, rivers, power lines ...
- *prefix*_roads a subset of the above, optimizer for rendering
- *prefix*_polygon objects covering an area: buildings, landuse, administrative borders ...

Adding MySQL Support

- turned out to be more tricky than thought
- some core parts directly called PostgreSQL functions
- a lot of general functionality was hidden in PostgreSQL specific modules

- MySQL output module works, could be faster though
- MySQL middle layer is “code complete” ...
- ... but crashes while processing relations :(
- so for now imports are limited by RAM size

Import performance

- imports currently take about 4-5 times as long
- ... as we have no direct equivalent to COPY
- `osm2pgsql` at less than 50% CPU only
- ... so switching to async API would be a 2x win already
- ... with multi-insert even more so
- index building is faster ...
- ... but may not be once we get I/O bound
- tables on disk are of similar size

Query performance

```
select count(*)
  from nrw_point n
  join nrw_polygon p
    on st_contains(p.way,n.way)
 where p.name = 'Bielefeld'
   and n.amenity='post_box';
```

Data Set	MySQL 5.5 (MBR)	MariaDB 5.5	PostGIS
Germany	15.8s	16.5s	?
Northrhine-Westfalia	2.7s	3.1s	6.1s
same with better indexes	0.2s	0.2s	.04s

Examples

The End ...

References

Contact	hartmut@skysql.com
MariaDB GIS	https://mariadb.com/kb/en/gis-functionality/
MySQL GIS	http://dev.mysql.com/doc/refman/5.6/en/spatial-extensions.html
OpenStreetMap	http://openstreetmap.org/
MapCompare	http://mc.bbbike.org/mc/
RiverMap	http://www.kompf.de/gps/rivermap.html
osm2pgsql	https://wiki.openstreetmap.org/wiki/Osm2pgsql
My Code	https://github.com/hholzgra/osm2pgsql/tree/devel-mysql
Table Files	http://php-groupies.de/gis-data/ (soon)

Questions!

The End?

Or just the beginning?