Mobile Broadband measurements: representing results in a database-oriented fashion

ISMA 2013 AIMS-5

Džiugas Baltrūnas
Simula Research Laboratory

2013-02-07

Environment

- Few hundreds of geographically spread nodes equipped with 2+ 3G connections plus optional WiFi and LAN.
- Each node conducts multiple measurements of Mobile Broadband Networks to address resilience, robustness and performance.
- Different types of measurements: RTT, one-way delay, throughput, connectivity (ability to connect), etc.
- A matrix of different protocols (e.g. TCP, UDP, ICMP), inter-arrival times, periodicity, packet sizes.
- Useful metadata that is provided by the 3G USB modems (e.g. signal strength, RRC state, Cell ID, etc.).

Objectives

- Continuous RTT measurement of 4 MBB networks on a single node with IAT of 1 second produces 60 sec * 60 min * 24 h * 4 networks * 2 records = **691200** rows of data per day not counting the metadata.
- Different experiments measure different metrics and for each measurement there has to be a process to organize the data (write to a file, sync to a server, do post-processing actions and archive).
- Real-time web visualization frontend needs to access the fresh data from all nodes of all networks frequently and fast.
- Historical data needs to be easily accessed for post-processing (e.g. aggregation) and analysis.

Lessons learned

- Usually it takes about the same time to write a measurement script and to set up the logged data flow properly.
- Storing measurement results in text files quickly becomes a hassle from all perspectives.
- There are static and dynamic properties of a node and a network as well as inventory items that often needs to be displayed together with measurement results.
- Different data structures for different measurements makes the results hard to display, analyze, correlate and maintain.

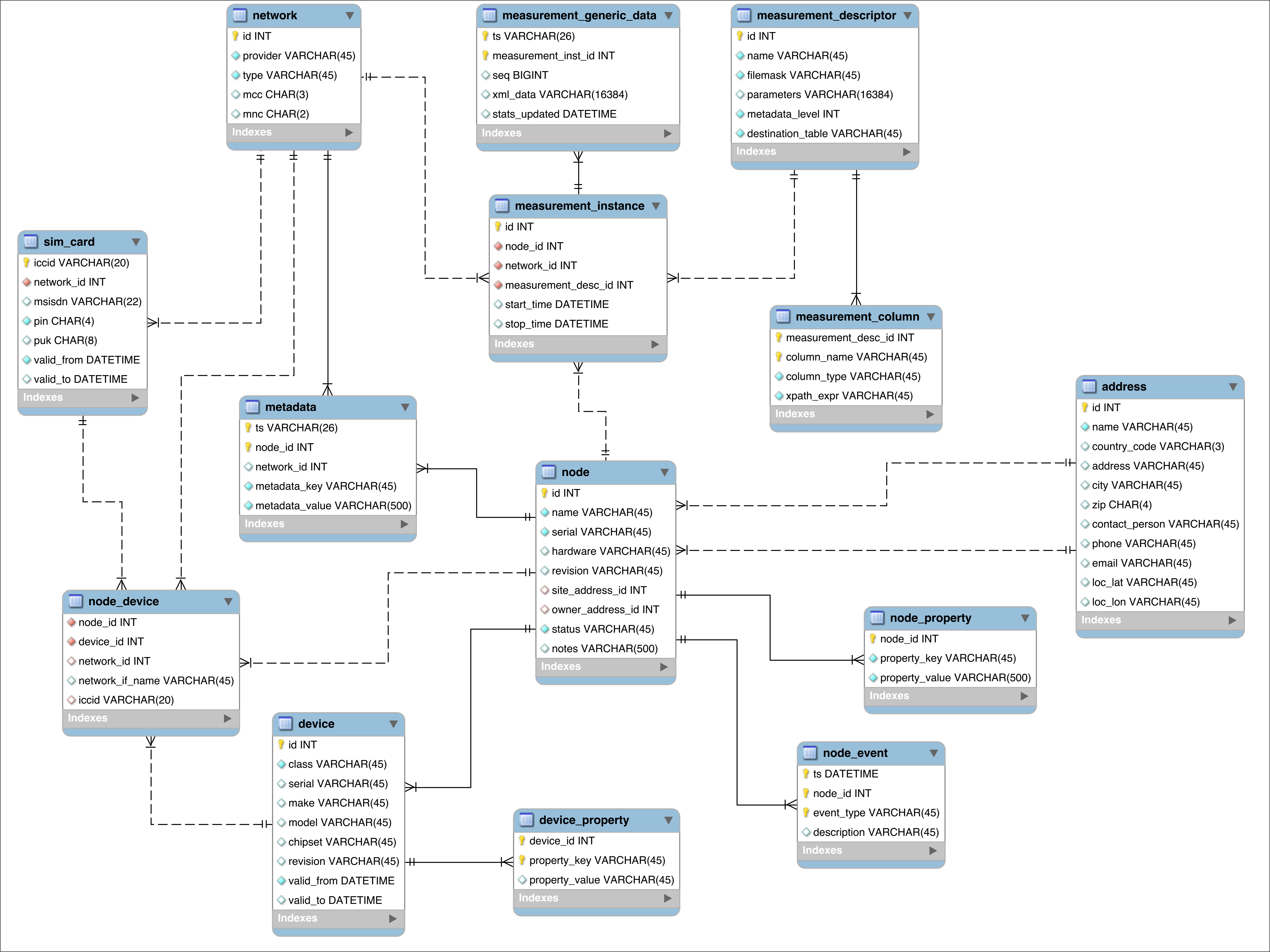
Our approach

- Organize the data in a relational database.
- Relate the inventory (nodes, devices, SIM cards) with the measurement results, but store it in separate tables.
- Abstract measurement data to a lowest common denominator and encode it in a unified format among different measurements.
- Set up a shared infrastructure for collecting measurement results from the nodes, parsing the data and importing it into the database, relaxing the measurement script from worrying about the logged data flow.
- If necessary, define a flexible configuration for a particular measurement so that the data can be imported into a measurement specific table for easier analysis.

Metadata

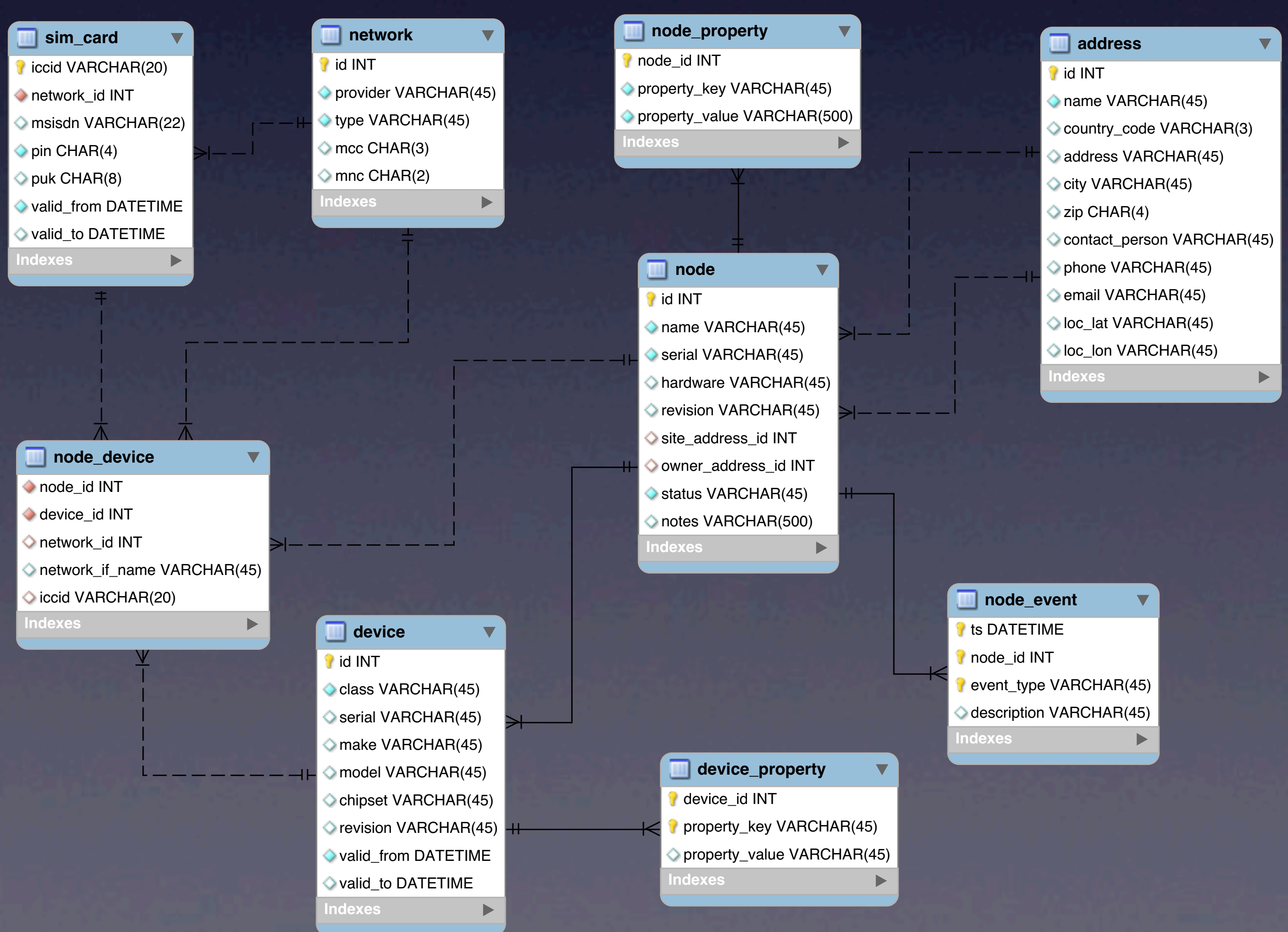
- Some of the 3G USB modems working as serial devices expose additional PCUI serial port which can be used to execute AT commands or receive unsolicited reports while the PPP connection is active. It provides information such as mode (e.g. GSM, WCDMA), submode (e.g. EDGE, HSDPA, HSPA+), location information (LAC and Cell ID), camping network (in case of national roaming), RSSI and other changes on demand.
- Modems with Qualcomm chipset provide Qualcomm Diagnostic Mode (QCDM) via yet another DIAG serial port. The protocol is proprietary, but there are some partial implementations such as *libqcdm*. Using QCDM, many different metadata, such as RRC state, can be extracted.
- Serial port's (e.g. /dev/ttyUSB0) is limited to one OS process, therefore metadata needs to be collected by the separate process which can broadcast it to other processes. We use ØMQ publisher-subscriber model.
- Real-time metadata can be used by both measurement scripts (e.g. to wait for a certain RRC state before sending a ping packet) or collected and imported into the database for further analysis.

The relational model


network
id INT
provider VARCHAR(45)
type VARCHAR(45)
mcc CHAR(3)
mnc CHAR(2)
Indexes
measurement_generic_data
ts VARCHAR(26)
measurement_inst_id INT
seq BIGINT
xml_data VARCHAR(16384)
stats_updated DATETIME
Indexes
measurement_descriptor
id INT
name VARCHAR(45)
filemask VARCHAR(45)
parameters VARCHAR(16384)
metadata_level INT
destination_table VARCHAR(45)
Indexes
sim_card
iccid VARCHAR(20)
network_id INT
msisdn VARCHAR(22)
pin CHAR(4)
puk CHAR(8)
valid_from DATETIME
valid_to DATETIME
Indexes
measurement_instance
id INT
node_id INT
network_id INT
measurement_desc_id INT
start_time DATETIME
stop_time DATETIME
Indexes
measurement_column
measurement_desc_id INT
column_name VARCHAR(45)
column_type VARCHAR(45)
xpath_expr VARCHAR(45)
Indexes
metadata
ts VARCHAR(26)
node_id INT
network_id INT
metadata_key VARCHAR(45)
metadata_value VARCHAR(500)
Indexes
address
id INT
name VARCHAR(45)
country_code VARCHAR(3)
address VARCHAR(45)
city VARCHAR(45)
zip CHAR(4)
contact_person VARCHAR(45)
phone VARCHAR(45)
email VARCHAR(45)
loc_lat VARCHAR(45)
loc_lon VARCHAR(45)
Indexes
node
id INT
name VARCHAR(45)
serial VARCHAR(45)
hardware VARCHAR(45)
revision VARCHAR(45)
site_address_id INT
owner_address_id INT
status VARCHAR(45)
notes VARCHAR(500)
Indexes
node_device
node_id INT
device_id INT
network_id INT
network_if_name VARCHAR(45)
iccid VARCHAR(20)
Indexes
node_property
node_id INT
property_key VARCHAR(45)
property_value VARCHAR(500)
Indexes
device
id INT
class VARCHAR(45)
serial VARCHAR(45)
make VARCHAR(45)
model VARCHAR(45)
chipset VARCHAR(45)
revision VARCHAR(45)
valid_from DATETIME
valid_to DATETIME
Indexes
node_event
ts DATETIME
node_id INT
event_type VARCHAR(45)
description VARCHAR(45)
Indexes
device_property
device_id INT
property_key VARCHAR(45)
property_value VARCHAR(45)
Indexes

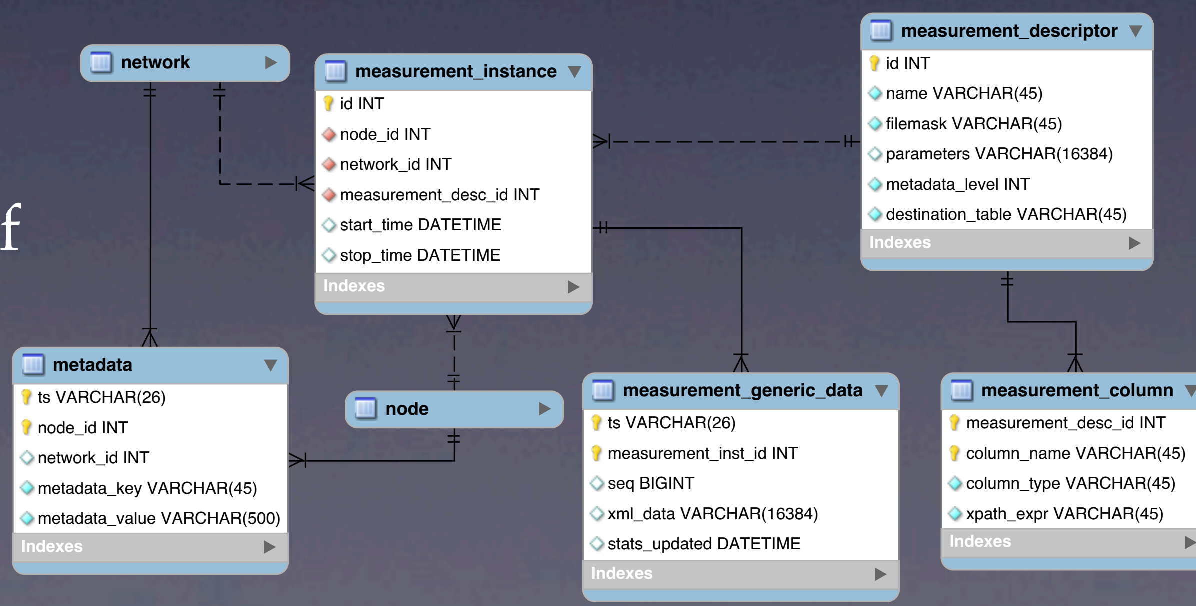
Inventory

- Inventory consists of nodes, SIM cards and devices.
- Each node has its mandatory and optional properties, address, event history and a list of devices attached to it.
- Each device has its mandatory and optional properties.
- A triplet of a node, device and an optional SIM card forms a network.
- Network can be one of 3GPP, 3GPP2, 802.3 or 802.11.



Measurements

- Measurement descriptor defines basic properties of the experiment.
- Each measurement is started by creating an instance for a particular node and network.
- Each measurement result contains a timestamp, measurement instance identifier, sequence number and logged data in XML format. It is either stored in the *metadata_generic_table* or measurement specific table (*destination_table*).
- Logged data can be decoded using XPath expressions that are defined for each measurement descriptor in *metadata_column* table.
- Optionally, measurement can define the level of metadata it want to be collected. It is then logged in the *metadata* table.



Example: RTT measurement

Define a measurement

- Create measurement descriptor

```
mysql> INSERT INTO measurement_descriptor(name, filemask) VALUES ('RTT measurement', '%NODE%-RTT-%TIMESTAMP%.dat');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM measurement_descriptor;
+----+-----------------+--------------------------+------------+----------------+-------------------------+
| id | name            | filemask                 | parameters | metadata_level | destination_table       |
+----+-----------------+--------------------------+------------+----------------+-------------------------+
|  1 | RTT measurement | %NODE%-RTT-%TIMESTAMP%.dat | NULL       |              0 | measurement_generic_data |
+----+-----------------+--------------------------+------------+----------------+-------------------------+
1 row in set (0.00 sec)
```

- Define measurement columns

```
mysql> INSERT INTO measurement_column(measurement_desc_id, column_name, column_type, xpath_expr) VALUES (1, 'rtt',
'DOUBLE', '/data/rtt');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM measurement_column;
+---------------------+-------------+-------------+------------+
| measurement_desc_id | column_name | column_type | xpath_expr |
+---------------------+-------------+-------------+------------+
|                   1 | rtt         | DOUBLE      | /data/rtt  |
+---------------------+-------------+-------------+------------+
1 row in set (0.00 sec)
```

- Create measurement instance

```
mysql> INSERT INTO measurement_instance(node_id, network_id, measurement_desc_id, start_time) VALUES(201, 2, 1,
NOW());
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM measurement_instance;
+----+---------+------------+---------------------+---------------------+-----------+
| id | node_id | network_id | measurement_desc_id | start_time          | stop_time |
+----+---------+------------+---------------------+---------------------+-----------+
|  1 |     201 |          2 |                   1 | 2013-01-24 20:01:05 | NULL      |
+----+---------+------------+---------------------+---------------------+-----------+
1 row in set (0.00 sec)
```

Collect and import the data

- Measurement script writes results to a text file and placed it under export folder

```
dziugas@festive:~$ cat /nne/logs/nne201-rtt-20130124200105.dat
2013-01-24 20:01:05.123456 1     1     <data><rtt>0.531</rtt></data>
2013-01-24 20:01:06.121021 1     2     <data><rtt>0.354</rtt></data>
2013-01-24 20:01:07.121020 1     3     <data><rtt>0.732</rtt></data>
2013-01-24 20:01:08.122013 1     4     <data><rtt>0.321</rtt></data>
2013-01-24 20:01:09.122045 1     5     <data><rtt>0.291</rtt></data>
2013-01-24 20:01:10.122056 1     6     <data><rtt>0.432</rtt></data>
2013-01-24 20:01:11.123001 1     7     <data><rtt>0.102</rtt></data>
2013-01-24 20:01:12.123404 1     8     <data><rtt>0.053</rtt></data>
2013-01-24 20:01:13.124531 1     9     <data><rtt>0.064</rtt></data>
2013-01-24 20:01:14.203953 1     10    <data><rtt>0.041</rtt></data>
```

- Data collector fetches the file from a remote node and imports it into the database

```
$ ln -s nne201-rtt-20130124200105.dat measurement_generic_data.dat
$ mysqlimport --local mbbm measurement_generic_data.dat
mbbm.measurement_generic_data: Records: 10  Deleted: 0  Skipped: 9  Warnings: 10

mysql> SELECT * FROM measurement_generic_data;
+----------------------------+---------------------+------+----------------------------------+---------------+
| ts                         | measurement_inst_id | seq  | xml_data                         | stats_updated |
+----------------------------+---------------------+------+----------------------------------+---------------+
| 2013-01-24 20:01:05.123456 |                   1 |    1 | <data><rtt>0.531</rtt></data>    | NULL          |
| 2013-01-24 20:01:06.121021 |                   1 |    2 | <data><rtt>0.354</rtt></data>    | NULL          |
| 2013-01-24 20:01:07.121020 |                   1 |    3 | <data><rtt>0.732</rtt></data>    | NULL          |
| 2013-01-24 20:01:08.122013 |                   1 |    4 | <data><rtt>0.321</rtt></data>    | NULL          |
| 2013-01-24 20:01:09.122045 |                   1 |    5 | <data><rtt>0.291</rtt></data>    | NULL          |
| 2013-01-24 20:01:10.122056 |                   1 |    6 | <data><rtt>0.432</rtt></data>    | NULL          |
| 2013-01-24 20:01:11.123001 |                   1 |    7 | <data><rtt>0.102</rtt></data>    | NULL          |
| 2013-01-24 20:01:12.123404 |                   1 |    8 | <data><rtt>0.053</rtt></data>    | NULL          |
| 2013-01-24 20:01:13.124531 |                   1 |    9 | <data><rtt>0.064</rtt></data>    | NULL          |
| 2013-01-24 20:01:14.203953 |                   1 |   10 | <data><rtt>0.041</rtt></data>    | NULL          |
+----------------------------+---------------------+------+----------------------------------+---------------+
10 rows in set (0.01 sec)
```

Extract the results

```
mysql> SELECT mdata.ts, n.name node_name, a.city, mdesc.name measurement_name,
    ->   nn.provider, mdata.seq, ExtractValue(mdata.xml_data, '/data/rtt') rtt
    -> FROM node n, network nn, address a, measurement_generic_data mdata,
    ->   measurement_instance mi, measurement_descriptor mdesc
    -> WHERE n.id = mi.node_id AND mi.network_id = nn.id AND mi.measurement_desc_id = mdesc.id
    ->   AND n.site_address_id = a.id AND mdata.measurement_inst_id = mi.id and mi.id = 1;
+----------------------------+-----------+------+-------------------+----------+-----+-------+
| ts                         | node_name | city | measurement_name  | provider | seq | rtt   |
+----------------------------+-----------+------+-------------------+----------+-----+-------+
| 2013-01-24 20:01:05.123456 | nne201    | Oslo | RTT measurement   | NetCom   |   1 | 0.531 |
| 2013-01-24 20:01:06.121021 | nne201    | Oslo | RTT measurement   | NetCom   |   2 | 0.354 |
| 2013-01-24 20:01:07.121020 | nne201    | Oslo | RTT measurement   | NetCom   |   3 | 0.732 |
| 2013-01-24 20:01:08.122013 | nne201    | Oslo | RTT measurement   | NetCom   |   4 | 0.321 |
| 2013-01-24 20:01:09.122045 | nne201    | Oslo | RTT measurement   | NetCom   |   5 | 0.291 |
| 2013-01-24 20:01:10.122056 | nne201    | Oslo | RTT measurement   | NetCom   |   6 | 0.432 |
| 2013-01-24 20:01:11.123001 | nne201    | Oslo | RTT measurement   | NetCom   |   7 | 0.102 |
| 2013-01-24 20:01:12.123404 | nne201    | Oslo | RTT measurement   | NetCom   |   8 | 0.053 |
| 2013-01-24 20:01:13.124531 | nne201    | Oslo | RTT measurement   | NetCom   |   9 | 0.064 |
| 2013-01-24 20:01:14.203953 | nne201    | Oslo | RTT measurement   | NetCom   |  10 | 0.041 |
+----------------------------+-----------+------+-------------------+----------+-----+-------+
10 rows in set (0.00 sec)
```

Correlating results with metadata

```
mysql> SELECT mdata.ts, mdata. seq, ExtractValue(mdata.xml_data, '/data/rtt') rtt,
    ->   m.ts mts, m.metadata_key mkey, m.metadata_value mvalue
    -> FROM measurement_generic_data mdata LEFT JOIN metadata m ON (
    ->   mdata.ts BETWEEN m.ts - INTERVAL 0.5 SECOND and m.ts + INTERVAL 0.1 SECOND
    -> );
+----------------------------+------+-------+----------------------------+------------------+-----------+
| ts                         | seq  | rtt   | mts                        | mkey             | mvalue    |
+----------------------------+------+-------+----------------------------+------------------+-----------+
| 2013-01-24 20:01:05.123456 |    1 | 0.531 | 2013-01-24 20:01:05.123456 | RRC_STATE_CHANGE | CELL_FACH |
| 2013-01-24 20:01:05.123456 |    1 | 0.531 | 2013-01-24 20:01:05.234592 | RRC_STATE_CHANGE | CELL_DCH  |
| 2013-01-24 20:01:06.121021 |    2 | 0.354 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:07.121020 |    3 | 0.732 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:08.122013 |    4 | 0.321 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:09.122045 |    5 | 0.291 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:10.122056 |    6 | 0.432 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:11.123001 |    7 | 0.102 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:12.123404 |    8 | 0.053 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:13.124531 |    9 | 0.064 | NULL                       | NULL             | NULL      |
| 2013-01-24 20:01:14.203953 |   10 | 0.041 | NULL                       | NULL             | NULL      |
+----------------------------+------+-------+----------------------------+------------------+-----------+
11 rows in set (0.00 sec)
```

Complex measurements

- *measurement_generic_data* table is suitable for simple measurements with a low level of post-processing analysis afterwards.
- Measurement specific table can be defined in a *measurement_descriptor* table as *destination_table* column.
- Custom table can still be populated dynamically from the measurement results using column definitions from *measurement_column* table.
- Additionally, we plan to extend *measurement_descriptor* table with configuration and rules for aggregated data (e.g. quarter, hourly, daily averages), so that those tables can be also populated dynamically.

Data partitioning

- Measurement data and metadata tables are organized into daily partitions: *PARTITION BY RANGE COLUMNS(ts)*.
- SELECT queries which need the data for a particular time interval, scans only partitions involved, therefore data lookup is significantly faster, effectively avoiding a full table scan.
- Partitions are rotated automatically, by pre-creating future ones and erasing (or optionally moving to a backup storage) older than a predefined interval.

```
mysql> EXPLAIN PARTITIONS SELECT * FROM measurement_generic_data;
+----+-------------+--------------------------+------------------------------------------------------+------+---------------+------+---------+------+------+-------+
| id | select_type | table                    | partitions                                           | type | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+--------------------------+------------------------------------------------------+------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | measurement_generic_data | p20130120,p20130121,p20130122,p20130123,p20130124 | ALL  | NULL          | NULL | NULL    | NULL |   21 |       |
+----+-------------+--------------------------+------------------------------------------------------+------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)

mysql> EXPLAIN PARTITIONS SELECT * FROM measurement_generic_data WHERE ts >= '2013-01-24' AND ts <= '2013-01-24 23:59:59';
+----+-------------+--------------------------+------------+-------+---------------+---------+---------+------+------+-------------+
| id | select_type | table                    | partitions | type  | possible_keys | key     | key_len | ref  | rows | Extra       |
+----+-------------+--------------------------+------------+-------+---------------+---------+---------+------+------+-------------+
|  1 | SIMPLE      | measurement_generic_data | p20130124  | range | PRIMARY       | PRIMARY | 80      | NULL |   10 | Using where |
+----+-------------+--------------------------+------------+-------+---------------+---------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

Summary

- Measurement results conducted daily by hundreds of nodes equipped with multiple modems produces few gigabytes of data which, if organized in text files, quickly becomes a mess.
- Each experiment is unique in terms of metrics it measures, but it can still be abstracted to a common format shared by all measurements.
- Inventory items and their properties relate to measurements and their results, therefore it is a natural choice to couple these data structures in a relational database where the power of relating the entities comes out of the box.
- The work is still in progress, therefore any ideas about the different types of experiments that might be performed on MBB networks as well as suggestions how the data structure can be optimized further are gladly welcome!

Thank you!

dziugas@simula.no
http://nornet-testbed.no
http://simula.no