MySQL Flexible Schema for IoT

Alexander Rubin, Percona

PERCONA

About Me

Alexander Rubin, Principal Architect, Percona

- Working with MySQL for 10-15 years
  - Started at MySQL AB, Sun Microsystems, Oracle (MySQL Consulting)
  - Joined Percona in 2013

PERCONA

IoT and MySQL

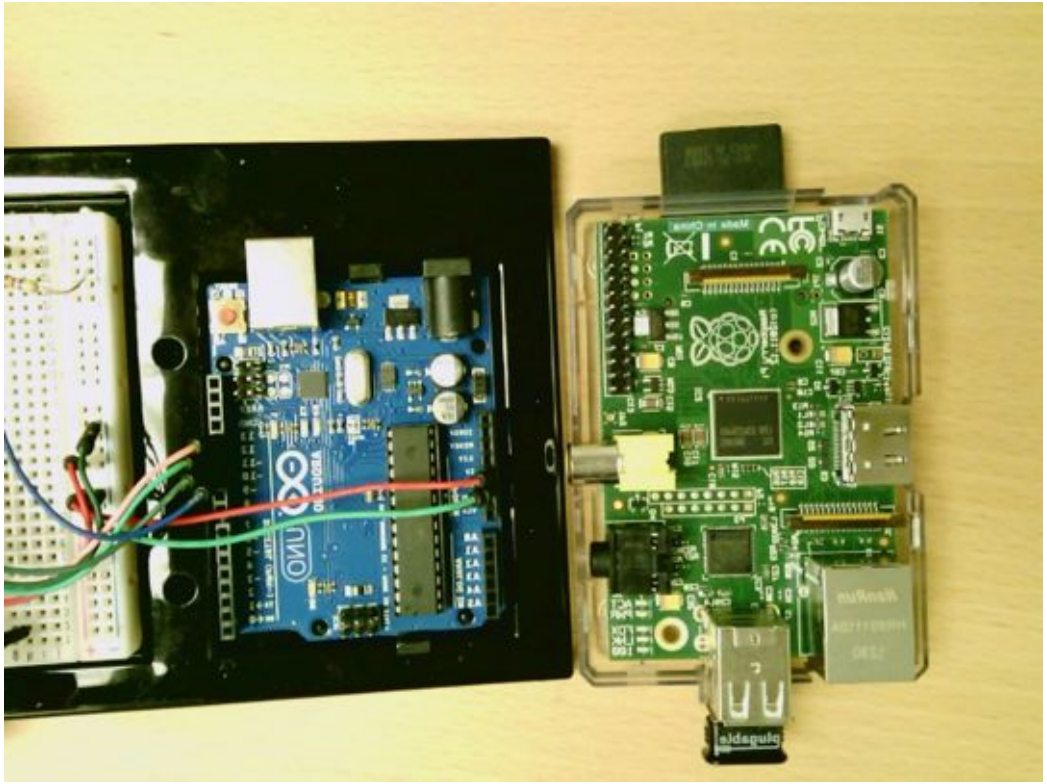
- Part 1: IoT demo
- Part 2: MySQL, JSON and Flexible storage

Part 1: IoT devices

Measuring Level of Light

PERCONA

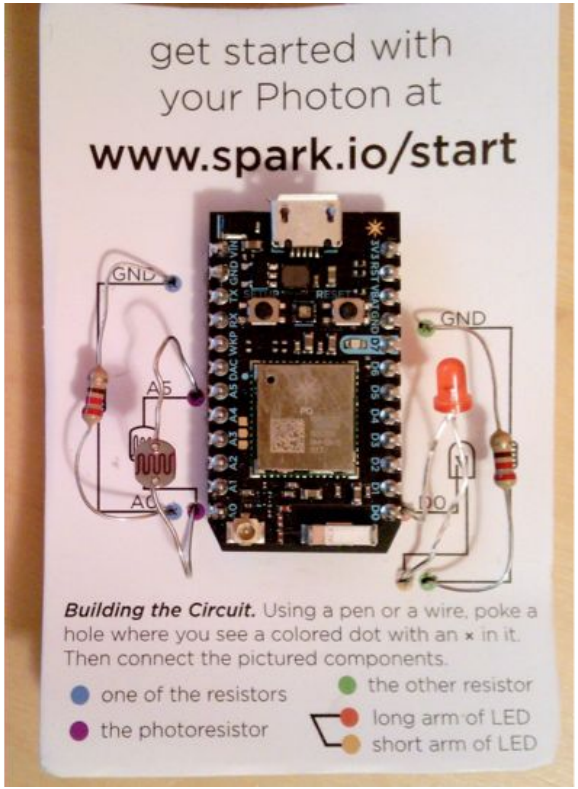
IoT devices



Tons of devices
to use for IoT…

PERCONA

Particle Photon

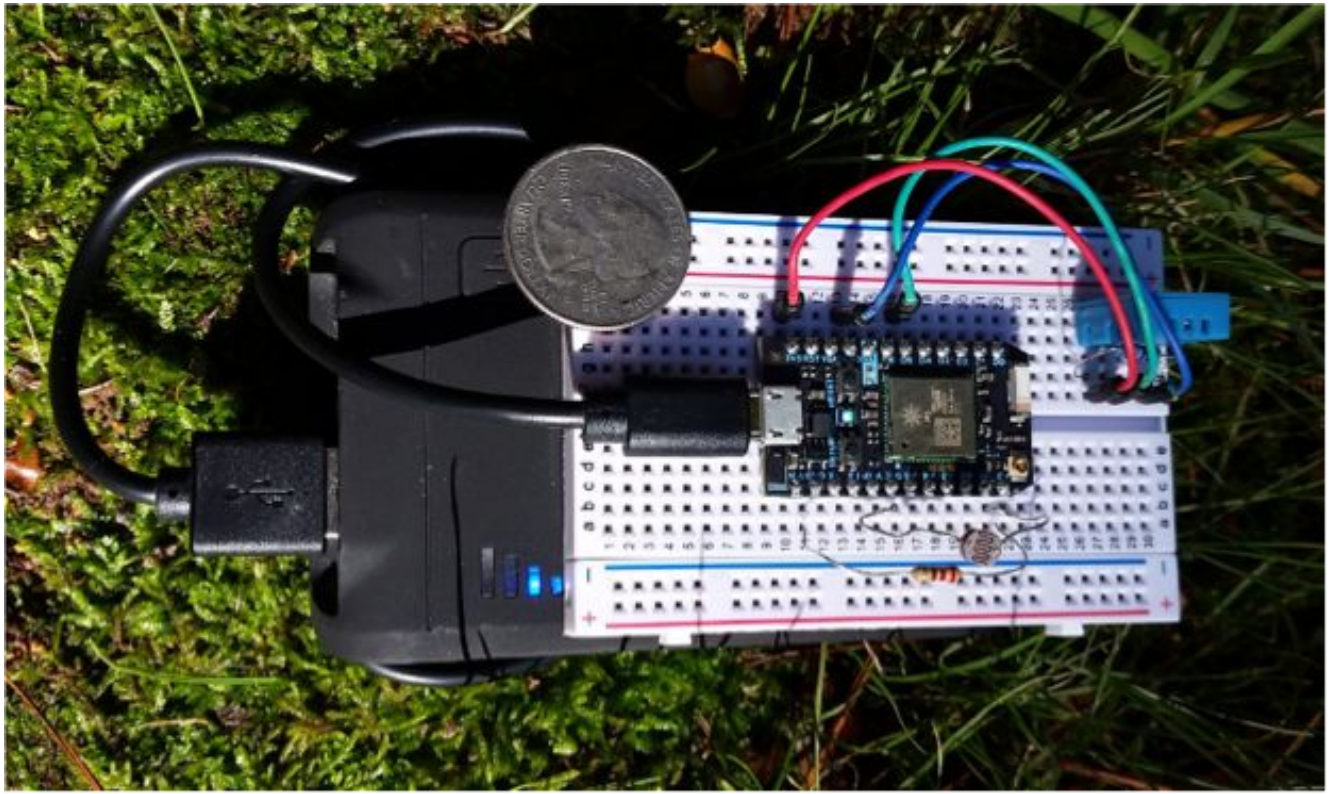


Particle Photon $19

https://store.particle.io

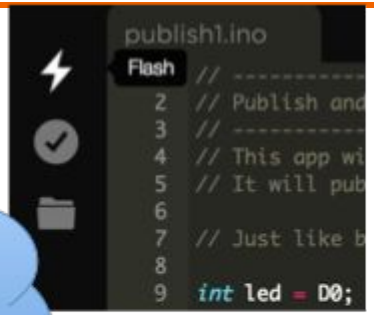
PERCONA

Particle Photon: Demo



PERCONA

Demo Recap… Cloud




```
$ node particle_mysql_all.js
Starting...
INSERT INTO cloud_data_json (name, data) values ('particle',
'{\"data\":\"null\",\"ttl\":60,\"published_at\":\"2017-09-28T19:40:49.869Z\",\"coreid\":\"1f003900094734333737373 8
\",\"name\":\"Server Error\"}')
...
```

PERCONA

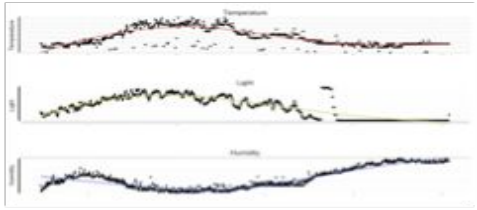
Get Data and Insert it into in MySQL

```
console.log("Starting...");
var Particle = require('particle-api-js');
var particle = new Particle();

fs = require('fs')
var token = fs.readFileSync('.token', 'ascii').replace(/\n$/, '');

//Get your devices events
// MySQL Connection
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'photon',
  password : 'photon',
  database : 'particle'
});
```

Get Data and Insert it into in MySQL

```
...
particle.getEventStream({deviceId: 'mine', auth: token}).then(function(stream) {
  stream.on('event', function(data) {
        var query = connection.query(' INSERT INTO cloud_data (name, data) values
        (?, ?)', [data.name, data.data], function(err, result) {
        if (err) {
                console.log('Error in ' + query.sql + err);
        }
        });
  console.log(query.sql);
  });
});
```

Demo Recap… Pipeline



Wi-Fi

MQTT

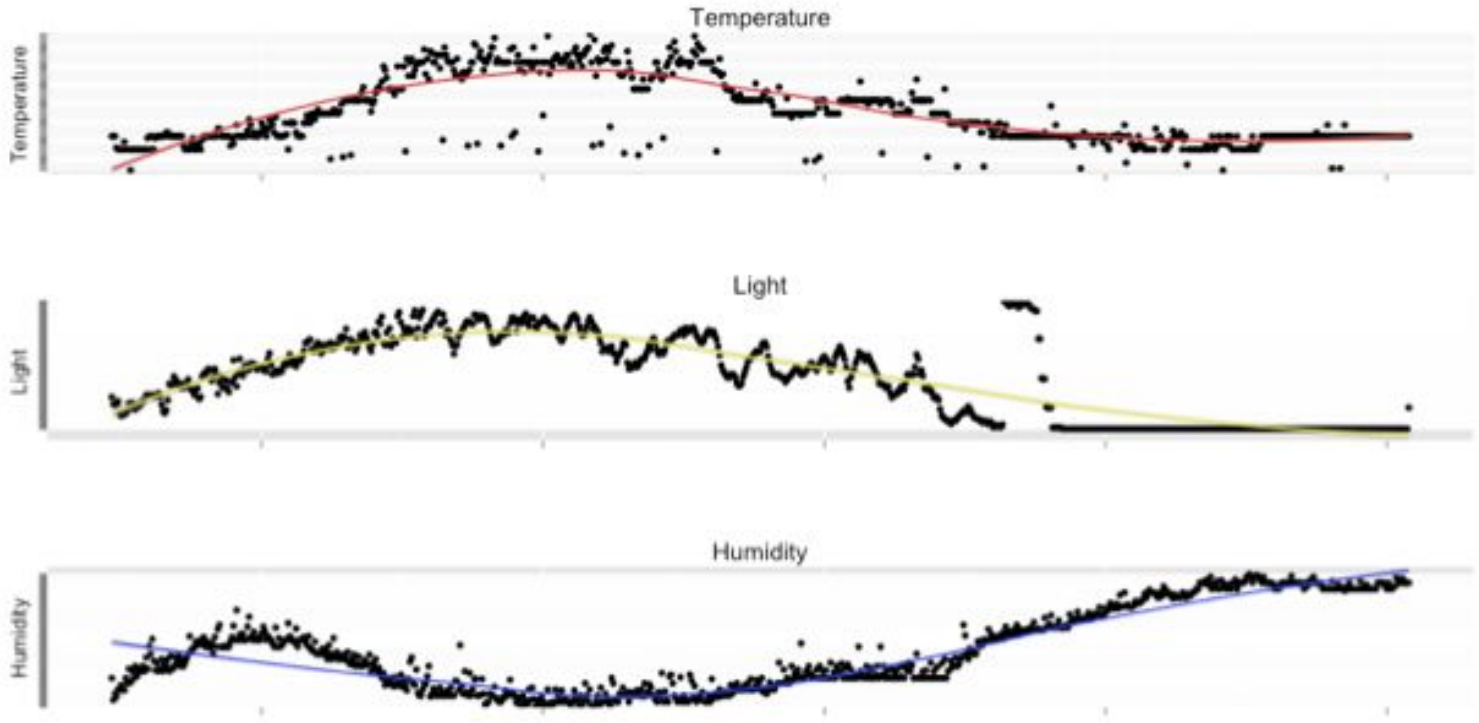
Particle

console.particle.io

Node/JS +
Particle API

PERCONA

Measurement Results



PERCONA

Part 2: MySQL

Storing data in MySQL …

```
$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.3-rc MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Storing data in MySQL, wide

```
CREATE TABLE `sensor_wide` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `light` int(11) DEFAULT NULL,
  `temp` double DEFAULT NULL,
  `humidity` double DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB
```

```
alter table sensor_wide
add water_level double ...;
```

+ Storage is good

- Alter table is a hard, not flexible

Storing data in MySQL: key/value

```
CREATE TABLE `cloud_data` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `data` text DEFAULT NULL,
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
  PRIMARY KEY (`id`)
) ENGINE=InnoDB
```

+ More Flexible
- High overhead on storage

PERCONA

Enter JSON

Everyone knows what JSON is, right?

Storing data in MySQL: JSON

```
CREATE TABLE `cloud_data_json` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `data` JSON,
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
PRIMARY KEY (`id`)
) ENGINE=InnoDB;
```

+ Most Flexible + Indexes

- Overhead on storage

Storing data in MySQL: JSON

```
...
  stream.on('event', function(data) {
     var query = connection.query(
       'INSERT INTO cloud_data_json (client_name, data)
        VALUES (?, ?)',
        ['particle', JSON.stringify(data)]
)
...
(demo)
```

PERCONA

Document Store Version

```
root@ip-172-30-2-254:~# mysqlsh -u user -h 127.0.0.1 -P 33060 --user=root --password=
mysqlx: [Warning] Using a password on the command line interface can be insecure.
Creating a session to 'root@127.0.0.1:33060'
Your MySQL connection id is 25 (X protocol)
Server version: 8.0.3-rc MySQL Community Server (GPL)
No default schema selected; type \use <schema> to set one.
MySQL Shell 8.0.3-dmr

Copyright (c) 2016, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type '\help' or '\?' for help; '\quit' to exit.

 MySQL  127.0.0.1:33060+ ssl  JS  \use particle
Schema `particle` accessible through db.
 MySQL  127.0.0.1:33060+ ssl  particle  JS  db.createCollection("cloud_data_docstore");
<Collection:cloud_data_docstore>
```

Document Store


```
const mysqlx = require('@mysql/xdevapi');
// MySQL Connection
var mySession =
mysqlx.getSession({
    host: 'localhost', port: 33060,  dbUser: 'photon'
});
...
session.getSchema("particle").getCollection("cloud_data_docstore")
                        .add(  data  )
                        .execute(function (row) {
                        }).catch(err => {
                              console.log(err);
                        })
                        .then( function (notices) {
                              console.log("Wrote to MySQL")
                        });
```

… https://dev.mysql.com/doc/dev/connector-nodejs/

Storing data in MySQL: JSON + Indexes

```
select data->>'$.name' as data_name,
       data->>'$.data' as data,
       data->>'$.published_at' as published
from cloud_data_json
order by data->'$.published_at' desc
limit 10;
```

Storing data in MySQL: JSON + Indexes

```
EXPLAIN select data->>'$.name' as data_name …
order by data->>'$.published_at' desc limit 10

  select_type: SIMPLE
        table: cloud_data_json
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 101589
     filtered: 100.00
        Extra: Using filesort
```

PERCONA

Storing data in MySQL: JSON + Indexes

```
mysql> ALTER TABLE cloud_data_json
    -> ADD published_at DATETIME(6)
    -> GENERATED ALWAYS AS
   (STR_TO_DATE(data->>'$.published_at',"%Y-%m-%dT%T.%fZ")) VIRTUAL;

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE cloud_data_json
    -> ADD data_name VARCHAR(255)
    -> GENERATED ALWAYS AS (data->>'$.name') VIRTUAL;

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Storing data in MySQL: JSON + Indexes

```
mysql> alter table cloud_data_json add key (published_at);
Query OK, 0 rows affected (0.31 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select data_name, published_at, data->>'$.data' as data  from
cloud_data_json   order by published_at desc  limit 10\G

           table: cloud_data_json
      partitions: NULL
            type: index
possible_keys: NULL
             key: published_at
         key_len: 9
             ref: NULL
            rows: 10
        filtered: 100.00
           Extra: Backward index scan
```

PERCONA

Storing data in MySQL: JSON + Indexes

```
mysql> explain select data_name, published_at, data->>'$.data' as data  from
cloud_data_json order by published_at desc, data_name asc limit 10\G

  select_type: SIMPLE
        table: cloud_data_json
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 101589
     filtered: 100.00
        Extra: Using filesort
```

PERCONA

Storing data in MySQL: JSON + Indexes

New in MySQL 8.0

```
mysql> alter table cloud_data_json
       add key published_at_data_name
       (published_at desc, data_name asc);
Query OK, 0 rows affected (0.44 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Storing data in MySQL: JSON + Indexes

```
mysql> explain select data_name, published_at, data->>'$.data' as data  from
cloud_data_json   order by published_at desc  limit 10\G

  select_type: SIMPLE
        table: cloud_data_json
   partitions: NULL
         type: index
possible_keys: NULL
          key: published_at_data_name
      key_len: 267
          ref: NULL
         rows: 10
     filtered: 100.00
        Extra: NULL
```

New JSON features in MySQL 8.0

JSON field type was introduced in 5.7, improved in 8.0

- Partial updates
  - in-place update of a JSON column instead of removing the old document and writing the new document in its entirety to the column
  - Only functions are supported: `JSON_SET()`, `JSON_REPLACE()`, or `JSON_REMOVE()`

Use `JSON_STORAGE_FREE(json_val)` to see how much storage space was freed in its binary representation after it was updated in place

PERCONA

MySQL 8.0: Pretty please print the JSON...

```
mysql> select json_pretty(data) from cloud_data_json
where data->>'$.data' like '%beer%' limit 1\G
...
json_pretty(data): {
  "ttl": 60,
  "data": "FvGav,tagkey=beer-store spFridge=7.00,pvFridge=7.44",
  "name": "LOG_DATA_DEBUG",
  "coreid": "3600....",
  "published_at": "2017-09-28T18:21:16.517Z"
}
```

Full Text Search inside JSON Documents

```
mysql> alter table cloud_data_json_indexes add fulltext key (data);
ERROR 3152 (42000): JSON column 'data' supports indexing only via generated columns on
a specified JSON path.

mysql>  ALTER TABLE cloud_data_json_indexes
    ->  ADD data_data VARCHAR(255)
    ->  GENERATED ALWAYS AS (data->>'$.data') VIRTUAL;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table cloud_data_json_indexes add fulltext key ft_json(data_name,
data_data);
ERROR 3106 (HY000): 'Fulltext index on virtual generated column' is not supported for
generated columns.
```

Full Text Search inside JSON Documents

```
mysql>  ALTER TABLE cloud_data_json_indexes
    ->  ADD data_name VARCHAR(255) CHARACTER SET UTF8MB4
    ->  GENERATED ALWAYS AS (data->>'$.name') STORED;
Query OK, 123518 rows affected (1.75 sec)
Records: 123518  Duplicates: 0  Warnings: 0

mysql> alter table cloud_data_json_indexes add fulltext key ft_json(data_name);
Query OK, 0 rows affected, 1 warning (3.78 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> show warnings;
+---------+------+-----------------------------------------------------+
| Level   | Code | Message                                             |
+---------+------+-----------------------------------------------------+
| Warning |  124 | InnoDB rebuilding table to add column FTS_DOC_ID |
+---------+------+-----------------------------------------------------+
```

PERCONA

Full Text Search inside JSON Documents

```
mysql>  ALTER TABLE cloud_data_json_indexes
    ->  ADD data_data TEXT CHARACTER SET UTF8MB4
    ->  GENERATED ALWAYS AS ( CONVERT(data->>'$.data' USING UTF8MB4) ) STORED;
Query OK, 123518 rows affected (3.14 sec)
Records: 123518  Duplicates: 0  Warnings: 0

mysql> alter table cloud_data_json_indexes drop key ft_json,
       add fulltext key ft_json(data_name, data_data);
Query OK, 0 rows affected (1.85 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

New JSON features in MySQL 8.0

- JSON aggregation functions
  - JSON_ARRAYAGG()
  - JSON_OBJECTAGG()

```
mysql> SELECT JSON ARRAYAGG(`key`) AS `keys` FROM t1;
+--------------------------+
| keys                     |
+--------------------------+
| [ "key1",
    "key2",
    "key3" ]
|
+--------------------------+
1 row in set (0,00 sec)
```

PERCONA

Not only IoT

Other uses for JSON and flexible storage

- Custom fields (CMS)
- Complex structures
- Etc

Thank you!

Alexander Rubin

https://www.linkedin.com/in/alexanderrubin

PERCONA