Migrating to PostgreSQL, the new story

FOSDEM PGDAY, 2015

Dimitri Fontaine dimitri@2ndQuadrant.fr
@tapoueh

January 30, 2015

Dimitri Fontaine

Principal Consultant at 2ndQuadrant

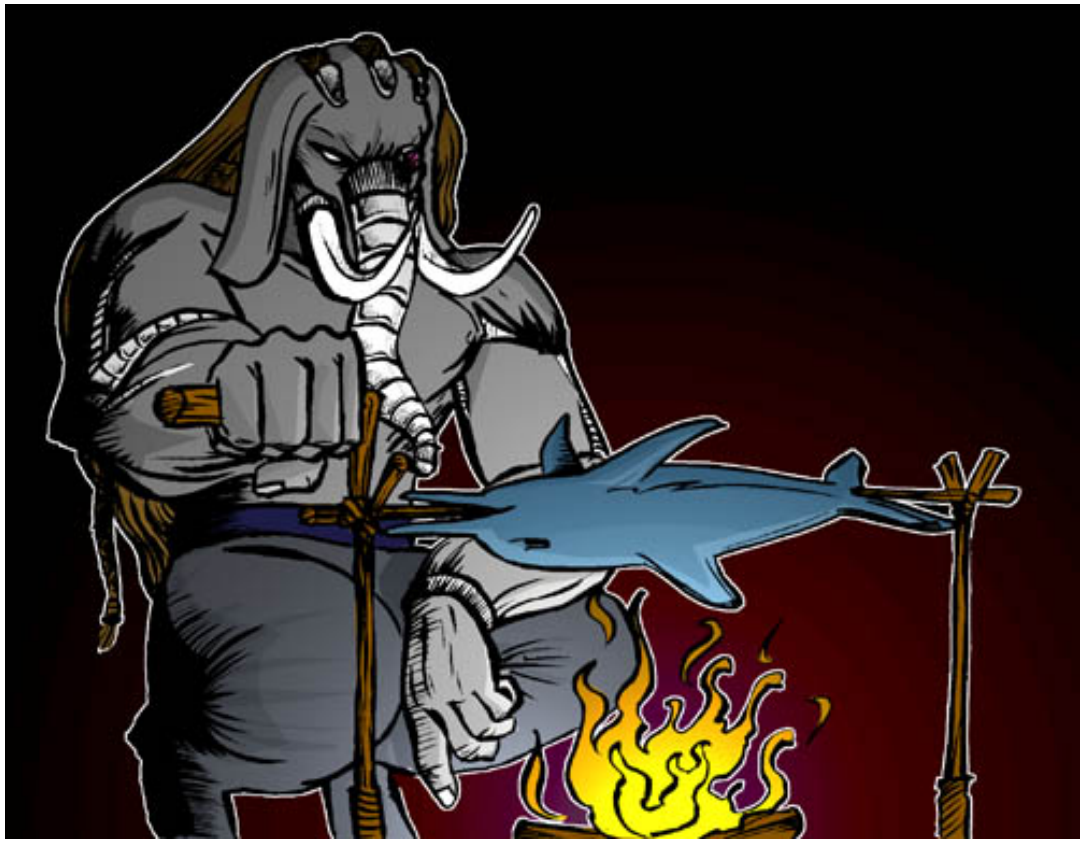
PostgreSQL Major Contributor

- `pgloader`
- `prefix`, `skytools`
- `apt.postgresql.org`
- `CREATE EXTENSION`
- `CREATE EVENT TRIGGER`
- *Bi-Directional Réplication*
- `pginstall`

PostgreSQL is YeSQL!

Let's talk about MySQL for a minute

Just in the context of migrating from it, of course



Why Migrating from MySQL to PostgreSQL?

MySQL

- Storage Engine
- Single Application
- Data Loss with Replication
- Weak Data Types Validation
- Either transactions or
- Lack of

PostgreSQL

- Data Access Service
- Application Suite
- Durability and Availability
- Consistency
- Full Text Search, PostGIS
- Proper Documentation

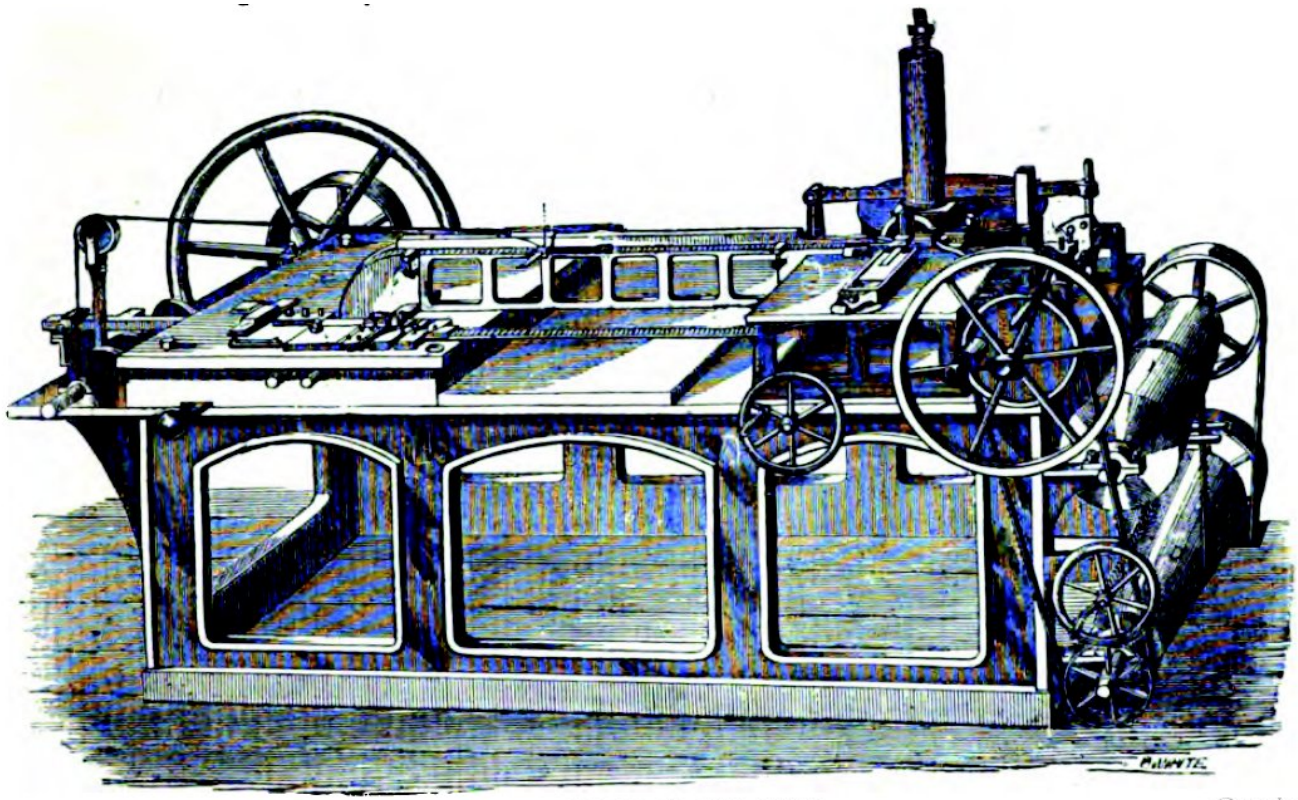
The migration budget

What are the costs?

- Migrating the Data
- Migrating the Code
- Quality Assurance
- Opportunity Cost

The boring parts really are

MySQL used not to be so serious about data consistency... still is



JOHNSON AND ATKINSON'S TYPE CASTER.

Difficulties when migrating MySQL data

Dates and The *Gregorian* Calendar

```
MariaDB [talk]> create table dates(d datetime);
MariaDB [talk]> insert into dates
    values('0000-00-00'), ('0000-10-31'), ('2013-10-00');

MariaDB [talk]> select * from dates;
+---------------------+
| d                   |
+---------------------+
| 0000-00-00 00:00:00 |
| 0000-10-31 00:00:00 |
| 2013-10-00 00:00:00 |
+---------------------+
3 rows in set (0.00 sec)
```

The God algorithm

The God algorithm

What you want to do instead

```
$ createdb pagila
$ pgloader mysql://user@localhost/sakila
           pgsql:///pagila
```

pgloader mysql://root@localhost/sakila pgsql:///pagila

```
./build/bin/pgloader mysql://root@localhost/sakila pgsql:///pagila
2015-01-30T12:49:09.030000+01:00 LOG Main logs in '/private/tmp/pgloader/pgloader.log'
2015-01-30T12:49:09.034000+01:00 LOG Data errors in '/private/tmp/pgloader/'
                  table name       read   imported     errors            time
------------------------------  ---------  ---------  ---------  --------------
                fetch meta data         43         43          0          0.358s
                   create, drop          0         36          0          0.263s
------------------------------  ---------  ---------  ---------  --------------
                         actor        200        200          0          0.151s
                       address        603        603          0          0.067s
                      category         16         16          0          0.025s
                          city        600        600          0          0.036s
                       country        109        109          0          0.019s
                      customer        599        599          0          0.040s
                          film       1000       1000          0          0.106s
                     film_actor       5462       5462          0          0.171s
                  film_category       1000       1000          0          0.068s
                      film_text       1000       1000          0          0.070s
                     inventory       4581       4581          0          0.172s
                      language          6          6          0          0.078s
                       payment      16049      16049          0          0.470s
                        rental      16044      16044          0          0.458s
                         staff          2          2          0          0.103s
                         store          2          2          0          0.035s
       Index Build Completion          0          0          0          0.001s
------------------------------  ---------  ---------  ---------  --------------
                Create Indexes         41         41          0          0.291s
               Reset Sequences          0         13          0          0.045s
                  Primary Keys         16         16          0          0.014s
                  Foreign Keys         22         22          0          0.079s
                      Comments          0          0          0          0.000s
------------------------------  ---------  ---------  ---------  --------------
            Total import time      47273      47273          0          2.829s
```

pgloader: load data into PostgreSQL

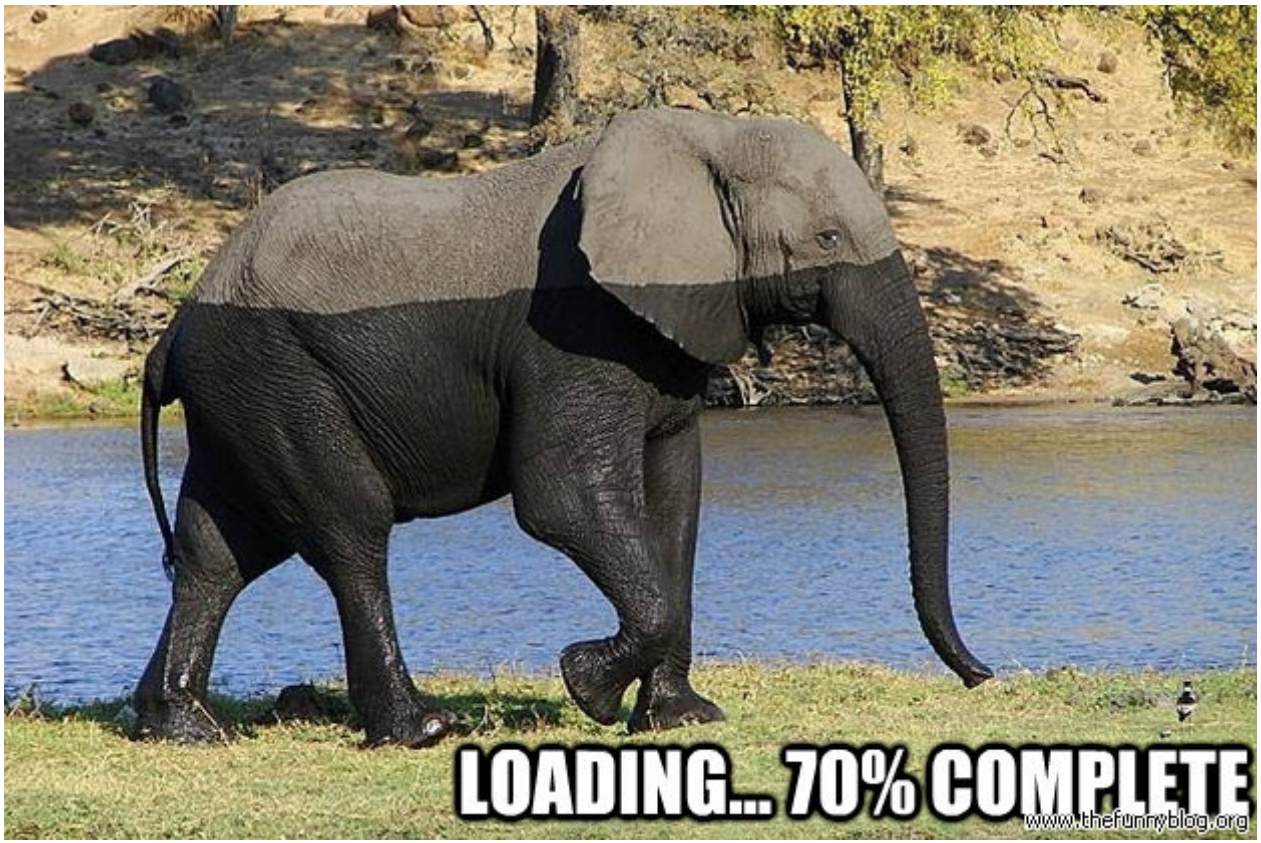
http://pgloader.io/

pgloader: Open Source, github

https://github.com/dimitri/pgloader

pgloader: what about loading data?

http://pgloader.io/



pgloader main features

pgloader is built around copy

- Error handling and reject files
- On the fly data transformations
- Very simple command line for simple use cases
- Advanced command language for advanced use cases
- Parallelism to benefit from async IO
- Lots of input formats

CSV

http://pgloader.io/howto/csv.html

COPY

http://pgloader.io/howto/quickstart.html

dBase III

http://pgloader.io/howto/dBase.html

SQLite

http://pgloader.io/howto/sqlite.html

MySQL

http://pgloader.io/howto/mysql.html

MS SQL Server

http://pgloader.io/

And more to come

File formats with on-the-fly normalisation

Other database systems

pgloader database migration process

The data migration process, step by step

1. Fetch metadata from source database catalogs
2. Prepare PostgreSQL database
3. COPY data
4. Complete PostgreSQL database
5. Display summary (human readable, json, csv, copy)

Fetching Metadata

Currently supported metadata

- Schemas
- Tables
- *Columns*
- Indexes
- Constraints
- Comments (mysql only)
- *Materializing* Views (mysql only)

Prepare PostgreSQL database

Prepare PostgreSQL for receiving the data

- Schemas
- Tables
- Columns
- Rename indexes with table oids in memory

Copy Data

Copy the data from the source to the target

- For each table, COPY data in
- Two threads work in parallel (reader/writer)
- Then for each table, create all indexes in parallel
- Currently no cap on the number of threads

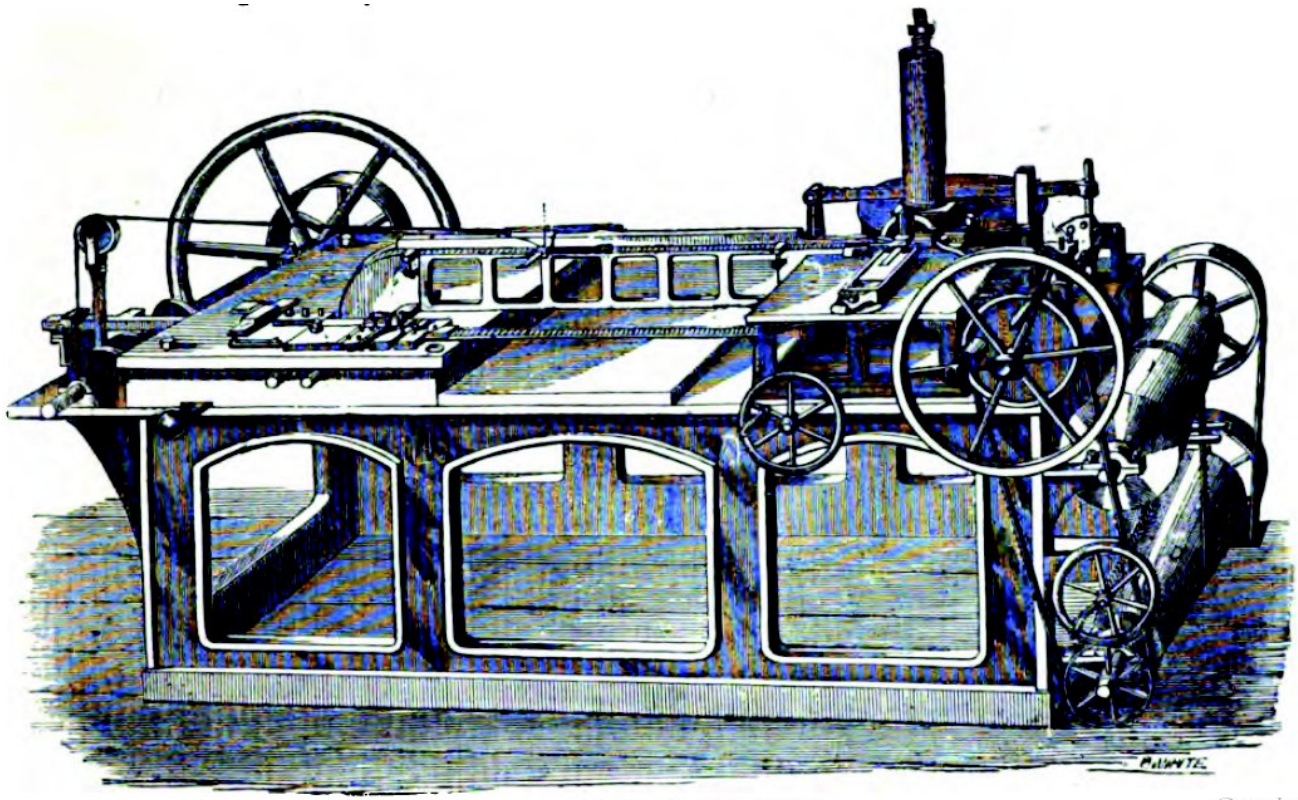
Complete PostgreSQL Database

Install constraints

- Reset **Sequences**
- Upgrade unique indexes into **Primary Keys** where required
- **Foreign Keys**
- **Comments**

Migrating columns

What do you mean **columns**?



JOHNSON AND ATKINSON'S TYPE CASTER.

A simple use case

Remember that example? we'll see a more detailed one...

```
$ createdb pagila
$ pgloader mysql://user@localhost/sakila
           pgsql:///pagila

$ pgloader migration.load
```

An avdanced use case 1/3

```
LOAD DATABASE
     FROM      mysql://root@unix:/tmp/mysql.sock:/goeuro
     INTO postgresql://dim@unix:/tmp:/godollar

 -- INCLUDING ONLY TABLE NAMES MATCHING ~/sizes/
 EXCLUDING TABLE NAMES MATCHING ~/encoding/

 DECODING TABLE NAMES
            MATCHING ~/messed/,
                     ~/encoding/
                  AS utf-8

 SET maintenance_work_mem to '1 GB'
```

An avdanced use case 2/3

```
CAST type datetime to timestamptz
                 drop default drop not null
                using zero-dates-to-null,

     column bools.a to boolean drop typemod
                 using tinyint-to-boolean,

     type char when (= precision 1)
       to char keep typemod,

     column ascii.s using byte-vector-to-bytea,

     column enumerate.foo
      using empty-string-to-null
```

An avdanced use case 3/3

```
WITH downcase identifiers,          -- quote idenfifiers
     truncate,
     -- data only,
     -- schema only,
     create tables,                 -- create no tables
     include drop,                  -- include no drop
     create indexes,                -- create no indexes
     reset sequences,               -- reset no sequences
     foreign keys;                  -- no foreign keys
```

pgloader: load data into PostgreSQL

http://pgloader.io/

Questions?

Now is the time to ask!