Loading Data In Postgresql, Fast. Any Data.

PostgreSQL Conference Europe, 2014

Dimitri Fontaine dimitri@2ndQuadrant.fr
@tapoueh

Oct. 23, 2014

Dimitri Fontaine

2ndQuadrant France

PostgreSQL Major Contributor

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

pgloader : Load Data Into PostgreSQL

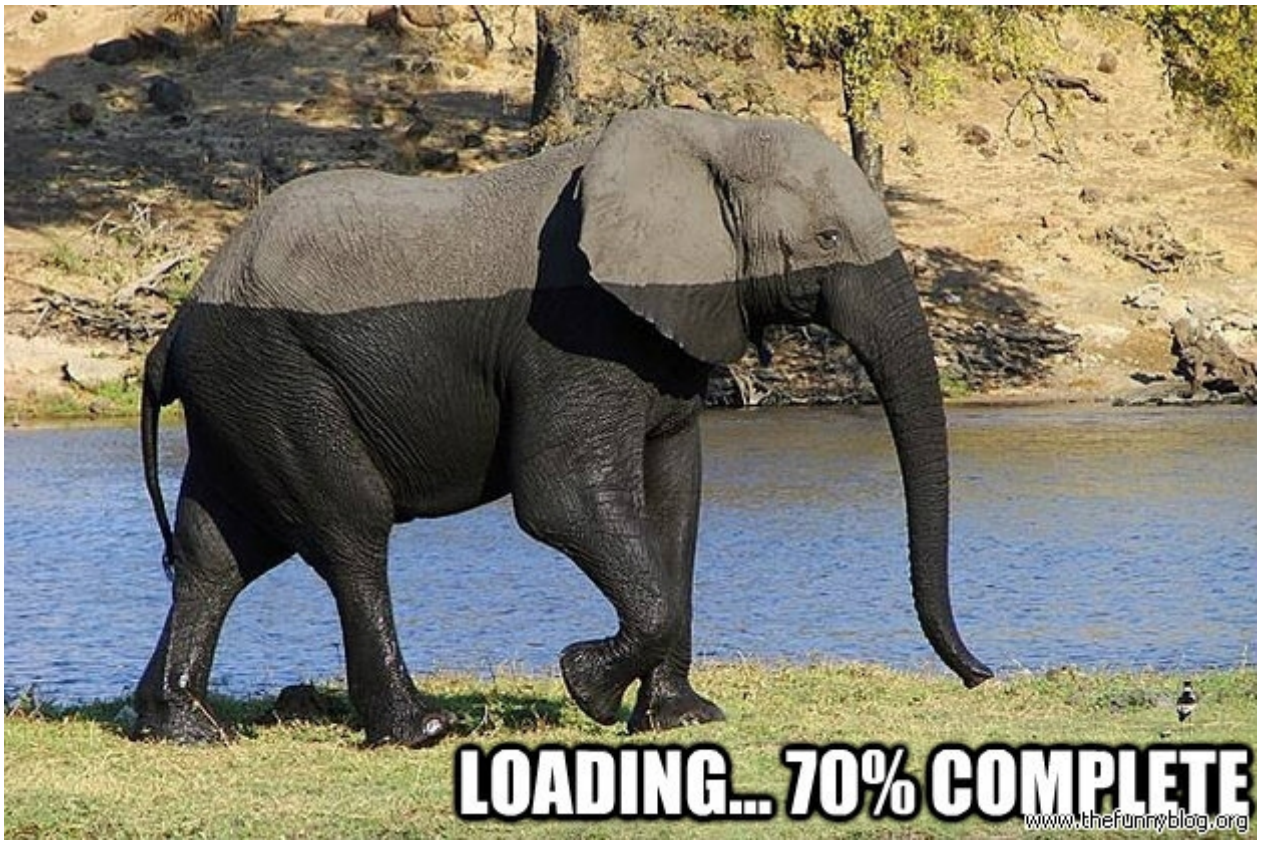
http://pgloader.io/

pgloader : Open Source, github

https://github.com/dimitri/pgloader

pgloader : Load Data

http://pgloader.io/



From CSV Files

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

From dBase III Files

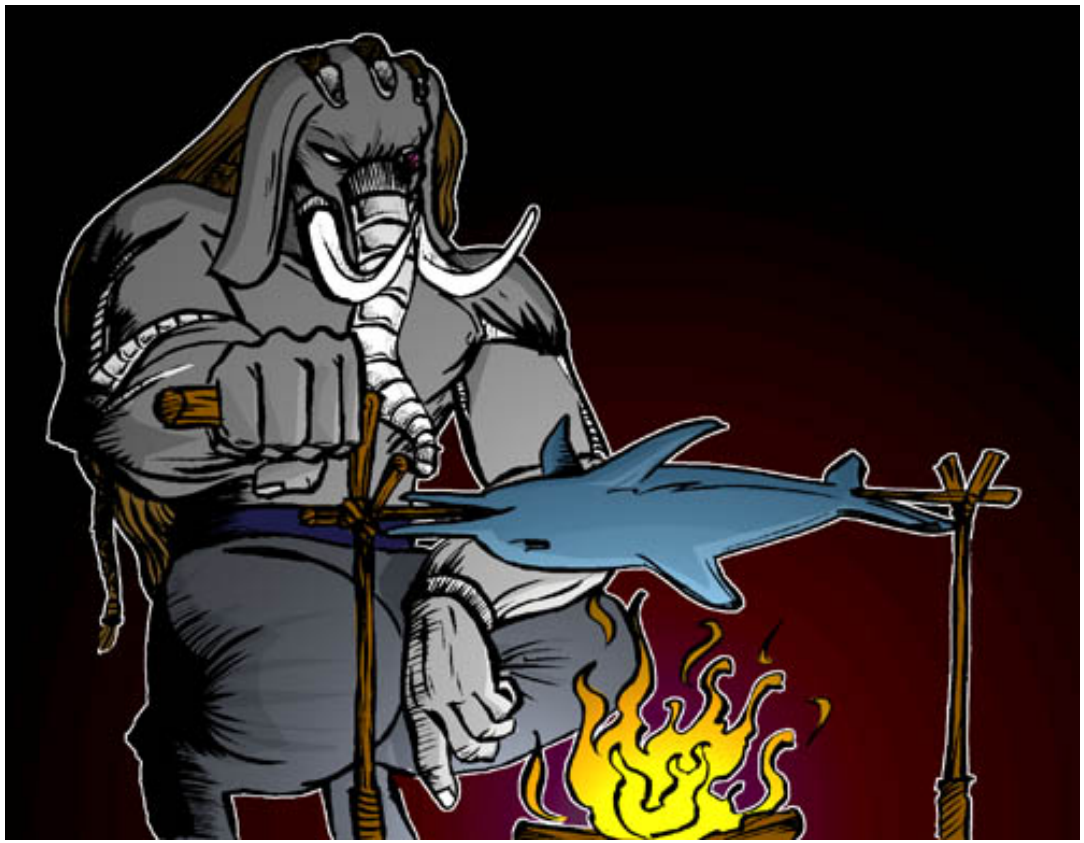
http://pgloader.io/howto/dBase.html

From SQLite

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

From a MySQL connection string

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

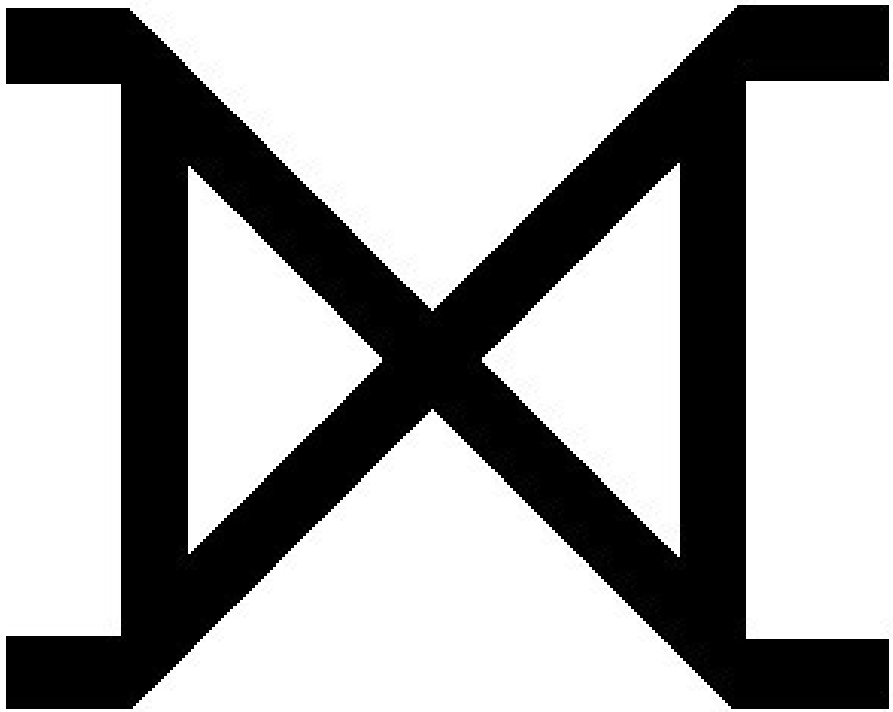


From a MySQL connection string

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

pgloader: Transforming data on the fly

http://pgloader.io/



pgloader version 3.1.0 is now available

Version 1 **TCL**, version 2 **Python**, version 3 **Common Lisp**

pgloader: loading data **fast**

User feedback comparing versions 2 and 3, loading CSV files

```
 select rows, v2, v3,
        round((  extract(epoch from v2)
        / extract(epoch from v3))::numeric, 2)
        as speedup
   from timing;
```

```
  rows   |         v2          |        v3         | speedup
---------+---------------------+-------------------+---------
 4768765 | @ 37 mins 10.878    | @ 1 min 26.917    |   25.67
 3115880 | @ 36 mins 5.881     | @ 1 min 10.994    |   30.51
 3865750 | @ 33 mins 40.233    | @ 1 min 15.33     |   26.82
 3994483 | @ 29 mins 30.028    | @ 1 min 18.484    |   22.55
(4 rows)
```

COPY

The copy command will always be faster than pgloader, but has very limited error handling capability.

pgloader main features

pgloader is not just copy

- Error handling with reject files
- Transforming data on the fly
- A full command language to specify the data loading
- Parallel processing architecture allowing async IO
- Large number of input file formats, and growing!

Setup used to look like an INI file

```
[pgsql]
base = pgloader
client_encoding = 'latin1'
pg_option_standard_conforming_strings = on
null          = ""
empty_string = "\ "

[csv]
table             = csv
format            = csv
filename          = csv/csv.data
field_size_limit = 512kB
field_sep         = ,
quotechar         = "
columns           = x, y, a, b, d:6, c:5
only_cols         = 3-6
skip_head_lines   = 1
```

Now the setup is a *command*

```
LOAD CSV
     FROM inline (x, y, a, b, c, d)
     INTO postgresql:///pgloader?csv (a, b, d, c)

     WITH truncate,
          skip header = 1,
          fields optionally enclosed by '"',
          fields escaped by double-quote,
          fields terminated by ','

      SET client_encoding to 'latin1',
          work_mem to '12MB',
          standard_conforming_strings to 'on'
```

and the command continues

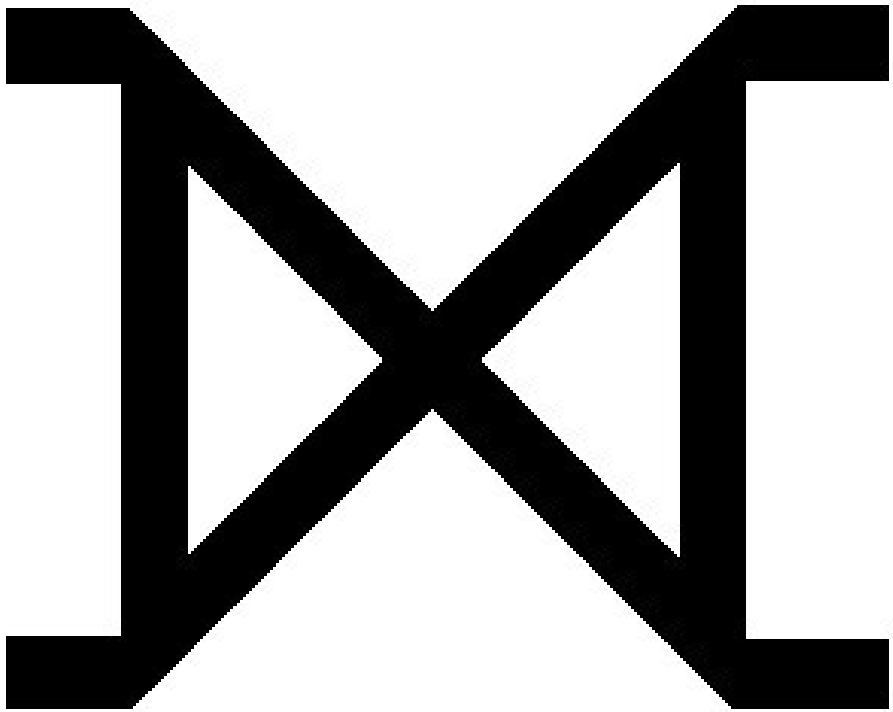
```
BEFORE LOAD DO
 $$ drop table if exists csv; $$,
 $$ create table csv (
     a bigint,
     b bigint,
     c char(2),
     d text
    );
 $$;
```

Examples of supported data sources

```
FROM stdin
FROM inline (a, b, c)
FROM data/2013_Gaz_113CDs_national.txt

FROM FILENAME MATCHING ~/GeoLiteCity-Location.csv/
FROM ALL FILENAMES MATCHING ~/F[A-Z]{4}1[45]|OZ20/

FROM http://www.census.gov/geo/maps-data/
           data/docs/gazetteer/places2k.zip

FROM http://www.insee.fr/fr/methodes/nomenclatures/
           cog/telechargement/2013/dbf/historiq2013.zip

FROM 'sqlite/sqlite.db'
FROM mysql://root@localhost/sakila
FROM mysql://root@unix:/tmp/mysql.sock:/goeuro
```

pgloader : Transforming data on the fly

http://pgloader.io/



Transforms

```
FROM FILENAME MATCHING ~/GeoLiteCity-Blocks.csv/
     WITH ENCODING iso-8859-1
     (
        startIpNum, endIpNum, locId
     )
INTO postgresql:///ip4r?geolite.blocks
     (
        iprange ip4r using (ip-range startIpNum endIpNum),
        locId
     )
```

Transforms

```
FROM FILENAME MATCHING ~/GeoLiteCity-Location.csv/
     (
        locId, country,
        region     null if blanks,
        city       null if blanks,
        postalCode null if blanks,
        latitude, longitude,
        metroCode  null if blanks,
        areaCode   null if blanks
     )
INTO postgresql:///ip4r?geolite.location
     (
        locid, country, region, city, postalCode,
        location point
           using (format nil "(~a,~a)" longitude latitude),
        metroCode, areaCode
     )
```

Full MySQL to PostgreSQL migration in one command

http://www.galaxya.fr/

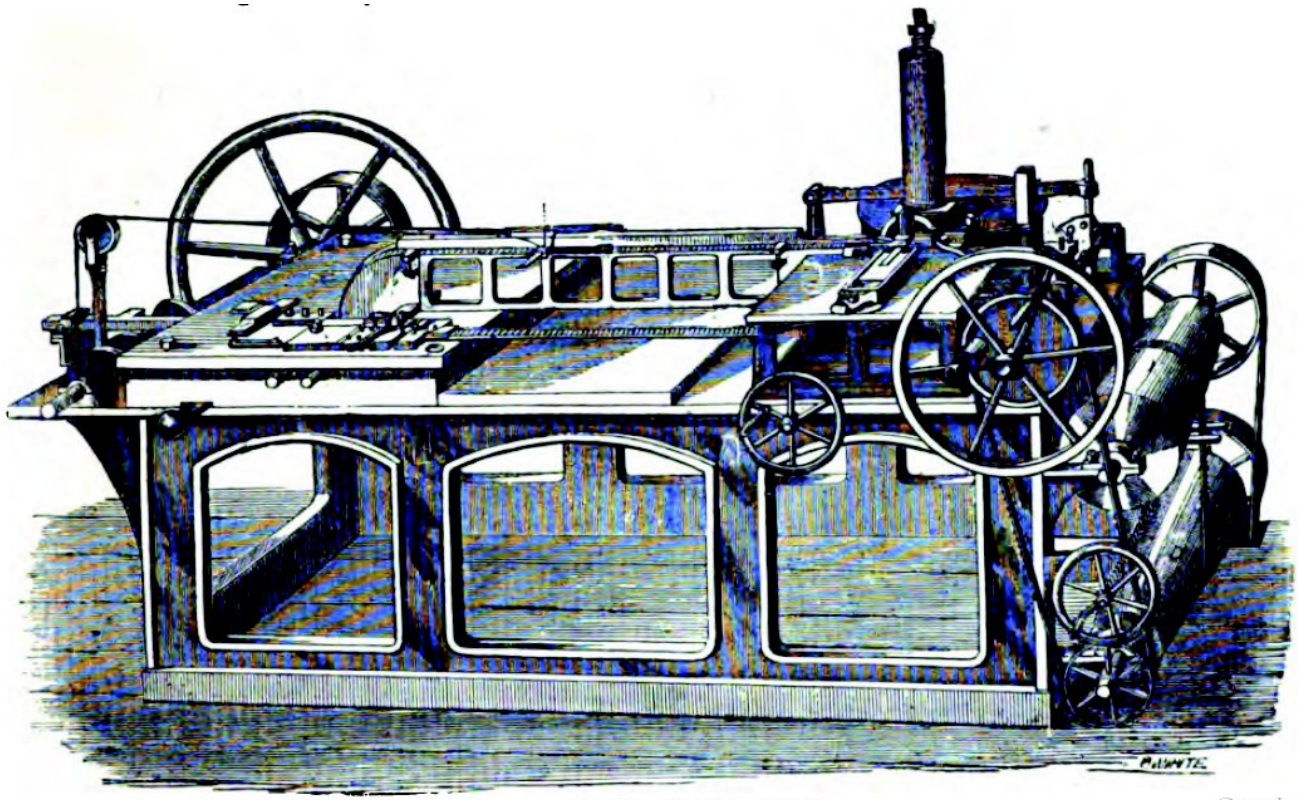
Working with Galaxya, used previous toolset

None of them are solving the actual (data) problems...

- `mysql2pgsql`, then manually edit the schema
- `SELECT INTO OUTFILE` server side, then `COPY`
- MySQL client pretends to be able to output CSV...
- There's always that last `awk` or `sed` step
- Some ruby and python scripts exist too

MySQL vision of data types

Empty string and `NULL`, default values, zero dates `0000-00-00`, `int(11)`, `float(20,2)`, `tinyint` rather than `boolean`, `sets`, encoding, ...



JOHNSON AND ATKINSON'S TYPE CASTER.

MySQL: CAST rules

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

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

MySQL: MATERIALIZE VIEWS

So that it's possible to change the Schema (DDL) while migrating

```
MATERIALIZE VIEWS foo,
  d as $$
      select cast(d as date) as d, count(*) as n
        from plop
       where d > '2013-10-02'
    group by cast(d as date);
  $$
```

pgloader limits

Still quite some work ahead

- Full support for views (it's another SQL dialect entirely)
- Triggers
- Stores Procedures
- Some data types (geometric)
- `ON UPDATE CURRENT_TIMESTAMP`

Other full database source types

Future

New data sources

Normalizing data while loading might be possible

Supporting new connection types


FREE
OUR
DATA

Questions?

Now is the time to ask!