# El piano, el camión y los vochos

## o, NoSQL - unos qués, por qués y por qué nos

(Sin qué pedos.)

Michael Wolf <maw@pobox.com>

# El reto: mover este piano

# desde aquí

# hasta aquí.

# ¿Cómo lo hacemos?

Muy sencillo. Llamamos al sitio de taxis:

—*¿Dónde está usted?*

—*El Auditorio Nacional.*

—*Y, ¿adónde va?*

—*Bellas Artes.*

—*Muy bien. Enseguida mandamos la unidad 1234556 por usted.*

# ¿Cómo lo hacemos?

Pero tenemos un problema.

¿?

(El piano y el vocho mostrados
a sus tamaños
relativos verdaderos.)

# ¿Cómo lo hacemos?

—Es que... esteee... vengo con... aaaa... un grupo bastante grande. ¿Podrá mandar... esteee... quince coches?

...

—¿Qué?

—Quince coches.

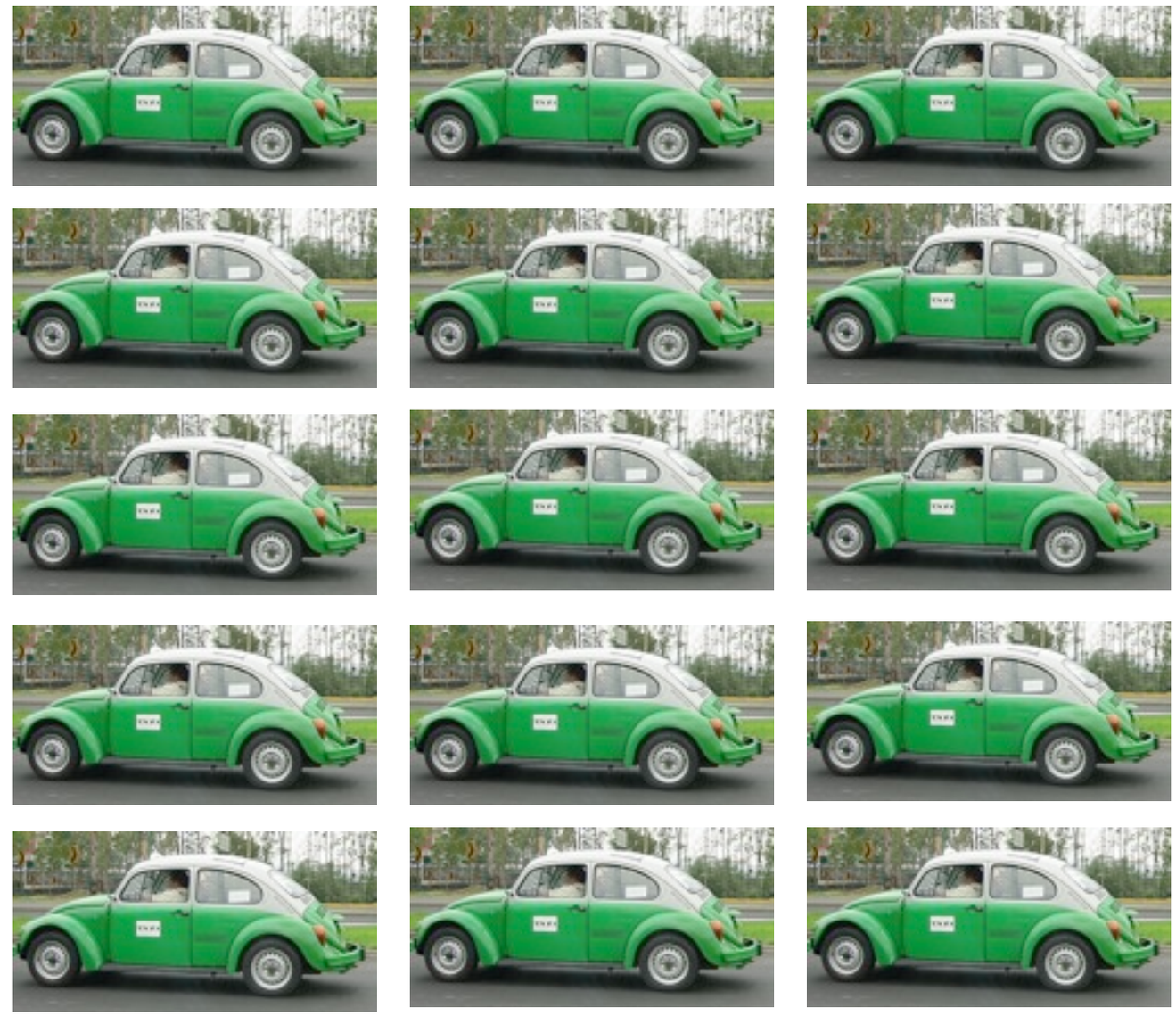
—Aaaa, ok, muy bien. En seguida.

# ¿Cómo lo hacemos?

*¡Rápido, equipo! Cortemos el piano en pedazos.*

# ¿Cómo lo hacemos?

Ya vienen los taxis. Una parte del piano en cada taxi.



# ¿Cómo lo hacemos?

En Bellas Artes armaremos el piano de nuevo.

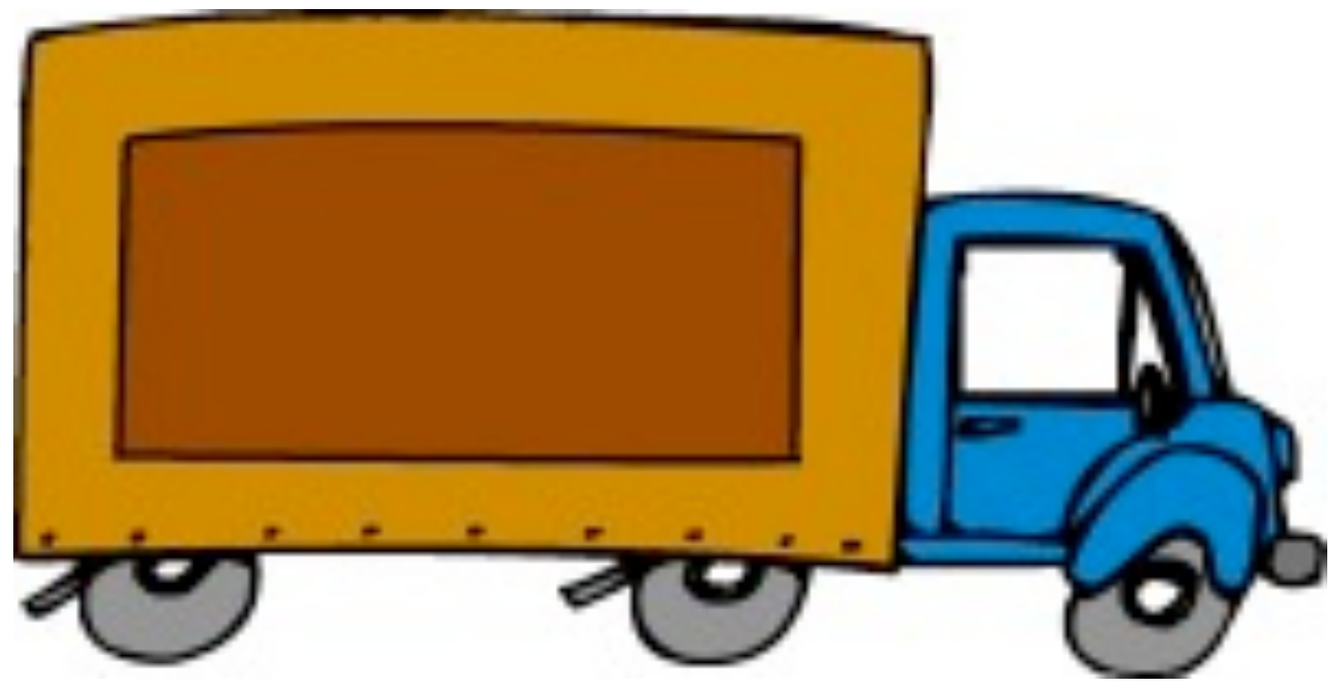
(Qué hueva.)

# ¿Hay alternativas?

Sí. Podemos usar un camión. Así nos saltamos los pasos de desarmar y rearmar el piano.

# ¡O-jo!

¡!



SQL: Una flota de vochos.
NoSQL: Un camión.

¿Exagero?
(¿Yo? Nunca.)

# Un ejemplo: definamos las estructura de datos de software übersencillo de un blog.

Definamos las estructuras de datos de software übersencillo de un blog.

¿Se puede leer? Arriba está algo en python. Abajo está lo equivalente en SQL. La esquema de Wordpress es parecido a ésta. Sí, está más completo y flexible, pero su esencia o estructura fundamental es lo mismo.

schema.sql – emacs running on fountain.local

```python
import uuid

entries = []

class entry(object):
    def __init__(self, body="empty"):
        self.__id = uuid.uuid4().hex
        self.__body = body
        self.__tags = []
        self.__comments = []

    def add_tag(self, tag):
        self.__tags.append(tag)

    def add_comment(self, comment):
        self.__comments.append(comment)
```

--:--- blog.py Top (3,12) (Python)--Wed 3 Nov 10:52 0.62----------

```sql
CREATE TABLE entries (entryid VARCHAR, body VARCHAR, PRIMARY KEY (entryid));

CREATE TABLE tags (tagid VARCHAR, txt VARCHAR, PRIMARY KEY (tagid));
CREATE TABLE tags_entries (entryid VARCHAR REFERENCES entries (entryid),
                           tagid INTEGER REFERENCES tags (tagid));

CREATE TABLE comments (commentid VARCHAR, body VARCHAR,
                       PRIMARY KEY (commentid));
CREATE TABLE comments_entries (entryid VARCHAR REFERENCES entries (entryid),
                               commentid VARCHAR REFERENCES comments (commentid));
```

--:--- schema.sql All (11,0) (SQL[ansi])--Wed 3 Nov 10:52 0.62----------

# Un ejemplo: definamos las estructura de datos de software übersencillo de un blog.

[Cambio al navegador por un momento.]

# ¿Qué es NoSQL?

- Perdemos las garantías ACID
- ¿Qué ganamos?
  - Les acabo de mostrar lo que me interesa
  - Hay otras razones también.

# Dos estrategias para la escalabilidad (de bases de datos)

# Vertical

Añades más memoria, mejor(es) procesador(es), más disco &c.

# Vertical

Pero a fin de cuentas—y después de cierto punto—, los costos suben más que la velocidad.

# Vertical

Y, ¿qué pasa si tu máquina de £1000000 muere?

# Horizontal

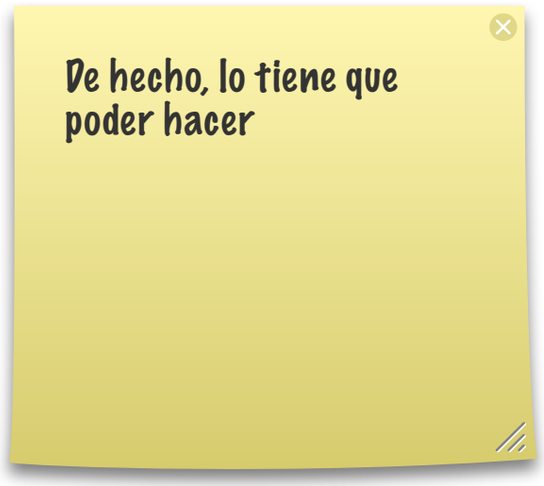
- Añades más máquinas
- Tienes que *distribuir* el trabajo que antes hacía una sola máquina—¿Cómo?
  - Particiones: algunos datos en una máquina; otros en otra
  - Denormalización, porque es difícil hacer *join*s sobre varias máquinas.

# Horizontal

- Sistemas NoSQL son diseñados para:
  - *hash*es distribuidos
  - replicación automática
  - resolución de conflictos
  - http://github.com/cloudant/bigcouch

# Horizontal

- Alta disponibilidad
  - Un nodo puede seguir funcionando de manera independiente
  - w00t
  - phw0ar



# Otras

- couchdb
  - mi favorita
- membase
  - memcached + persistencia
- cassandra
  - del libro de face
  - clave/valor y semi-tabular
- redis
  - servidor de estructuras de datos
- bigtable
  - magia de Google

# Unas palabras sobre CouchDB

- Todo se hace vía HTTP
  - curl
  - Interfaz web
  - Bibliotecas para varias lenguajes (fáciles de escribir porque el protocolo es HTTP)
  - Vamos al navegador para ver unos ejemplos

**Todo esto es SQL vs NoSQL.**

Si sigues usando SQL eres una *mala persona*.

SQL is for the **obsessive child**; the infantile boys preffer SQL because the aspect of the NoSQL is more stetic, more beautifull, and SQL is horrendous, then, SQL is the more different in relation to MS Access.

Sólo un `$algo` usaría SQL.

**¿Verdad?**


NOOOOOOOOOOOOOOOOOOOO

La gente busca la polémica. Y la crea donde no se encuentra.

Yo también.

Pero, hablando en serio, ahora tenemos un chingo de opciones. Hace poco tiempo, cuando alguien necesitaba persistencia, o usaba una base de datos de SQL o mantenía archivos en algún formato ridículo.  Todos (casi) lo hacíamos así.

¿Active record?  Me cago en active record.

# ¿SíSQL o NoSQL? Consideraciones.

- Las bases de datos de NoSQL todavía son nuevas
- Algunas estructuras de datos caben bien con una esquema tabular
- Tu jefe insiste en algo «comprobado»
- Tienes más experiencia con MySQoracLgres
- Las mismas razones que siempre

# Pero... Espérate. ¿No se puede hacer todo ésto con SQL?

# Pero... Espérate. ¿No se puede hacer todo ésto con SQL?

Sí, se puede:

```
$ sqlite mynosqlite.db
sqlite> CREATE TABLE prueba (key VARCHAR, value VARCHAR);
sqlite> INSERT INTO prueba VALUES (‘test’, ‘{“json”:”object”}’);
sqlite> SELECT value FROM prueba WHERE key = ‘test’;
{“json”:”object”}
sqlite>
```

# Pero... Espérate. ¿No se puede hacer todo ésto con SQL?

http://github.com/maw/mynosqlitegres. Es el resultado de unas noches cuando no me pude dormir. **No lo uses.** Es un ***juguete***, con todo lo que se implica: no útil pero bastante divertido para mí.

«Voy a atacar ese pinche molino, güey.»

# ¿Preguntas?

# ¿Preguntas?

Ándale.

# Gracias por asistir.

Michael Wolf <maw@pobox.com>