Django-MySQL

Adam Johnson - me@adamj.eu

April 2015

Database Share

- Who's using PostgreSQL?
- Who's using MySQL?

Database Share

- Worldwide:
  - MySQL 56%
  - MariaDB 18% (MySQL fork!)
  - PostgreSQL 13%
- (scalebase.com)

Motivation

- Django 1.8 comes with “django.contrib.postgres”
- Hey, MySQL should have something too!

Mascot

+

Mascot

It can't be unseen.

1. Database Functions

- New in Django 1.8!
- "The" place to find MySQL-only functions

```
>>> Author.objects.filter(
...     fewest_sales=Least('sales_eu', 'sales_us')
... )
>>> Author.objects.annotate(
...     full_name=ConcatWS('first_name',
...                        'last_name',
...                        separator=' ')
.. )
```

2. Named Locks

- Easy way to limit access to a resource
- E.g. API with connection limit

```
try:
    with Lock('my_unique_name', acquire_timeout=2.0):
        mutually_exclusive_process()
except TimeoutError:
    print "Could not get the lock"
```

3. Approximate Count

- `Model.objects.count()` = `SELECT COUNT(*)` which requires a table scan - slow!
- Various snippets and libraries exist, but this is (hopefully) *the best*^{TM}.
- Easy to hook into external code such as admin

```
>>> Author.objects.count()  # slow
509741
>>> Author.objects.approx_count()  # fast, some error
531140
```

4. List and Set Fields

- Cousins to `django.contrib.postgres`'s `ArrayField`
- Store values in comma-separated strings, lookups and F implementations using MySQL functions.

```
# models
class Person(Model):
    post_nominals = ListTextField(
        base_field=CharField(max_length=32)
    )

# shell
>>> Person.objects.create(
...     name='Horatio',
...     post_nominals=['PhD', 'Esq.', 'III']
... )
>>> Person.objects.filter(post_nominals__contains='PhD')
[<Person: Horatio>]
```

5. Smart Iteration

- Iterate and modify "big data" without fear, in primary-key-based slices
- Dynamically modifies slice size and checks MySQL status to avoid long-running outage-causing operations

```
# Turn this...
min_id, max_id = 0, 1000
max_author_id = Author.objects.order_by('-id')[0].id
while True:
    author_slice = Author.objects.filter(
        address="Nowhere",
        id__gte=min_id,
        id__lte=  # BLA BLA BLA
    # YOU GET THE IDEA IT'S A LOT OF CODE

# ...into this:
bad_authors = Author.objects.filter(address="Nowhere")
for author in bad_authors.iter_smart():
    author.send_apology_email()
```

Coming Soon...

- Fast DatabaseCache implementation using upserts
- Dynamic Columns - MariaDB's answer to HStore/json but with extra types
- Migration operations for e.g. loading extensions, changing table storage engine

- github.com/adamchainz/django-mysql
- me@adamj.eu