Django ORM crash test

@a_soldatenko

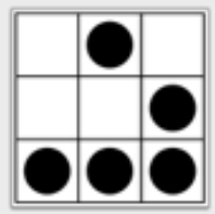
Django ORM crash test

Andrii Soldatenko
9 April 2017
Italy, Otto

@a_soldatenko

Andrii Soldatenko

- Senior Python Developer at toptal
- CTO at Persollo
- Co-organizer PyCon Belarus 2017 PyCon Belarus '17
- Speaker at many PyCons and open source contributor
- blogger at https://asoldatenko.com



@a_soldatenko

Table of contests:

- Object of testing
- SQL compare to Django ORM
- Crash tests and Django problems
- Conclusion

@a_soldatenko

Andrew Svetlov

asvetlov

Python core developer @a_soldatenko

Paweł Piotr Przeradowski Follow
Python programmer, Web developer, Open source lover, Linux user, amateur photographer
4 days ago · 6 min read

A million requests per second with Python

https://medium.freecodecamp.com/million-requests-per-second-with-python-95c137af319#.m9nhe1p2w

@a_soldatenko

Why?

@a_soldatenko

cat VERSION.md

 # Python 3.6.1

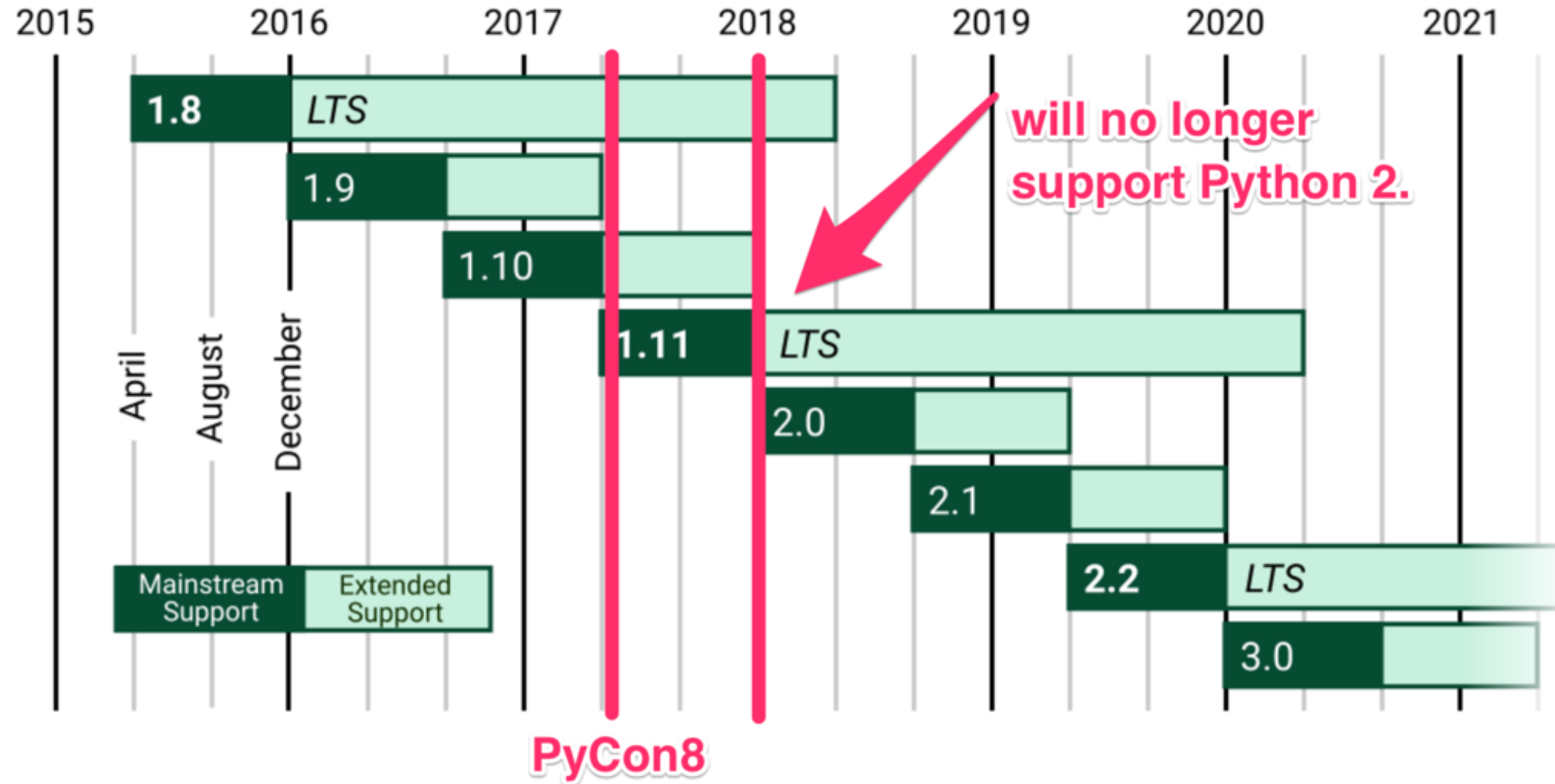
 # Django 1.11

 # PostgreSQL 9.6.2

 # Docker 1.13

@a_soldatenko

Django and Python 2



[django/django#7867]

@a_soldatenko

Remove Python 2 support

Refs #23919 -- Removed encoding preambles and #7867

Merged **claudep** merged 1 commit into `django:master` from `claudep:py3_preambles` 16 days ago

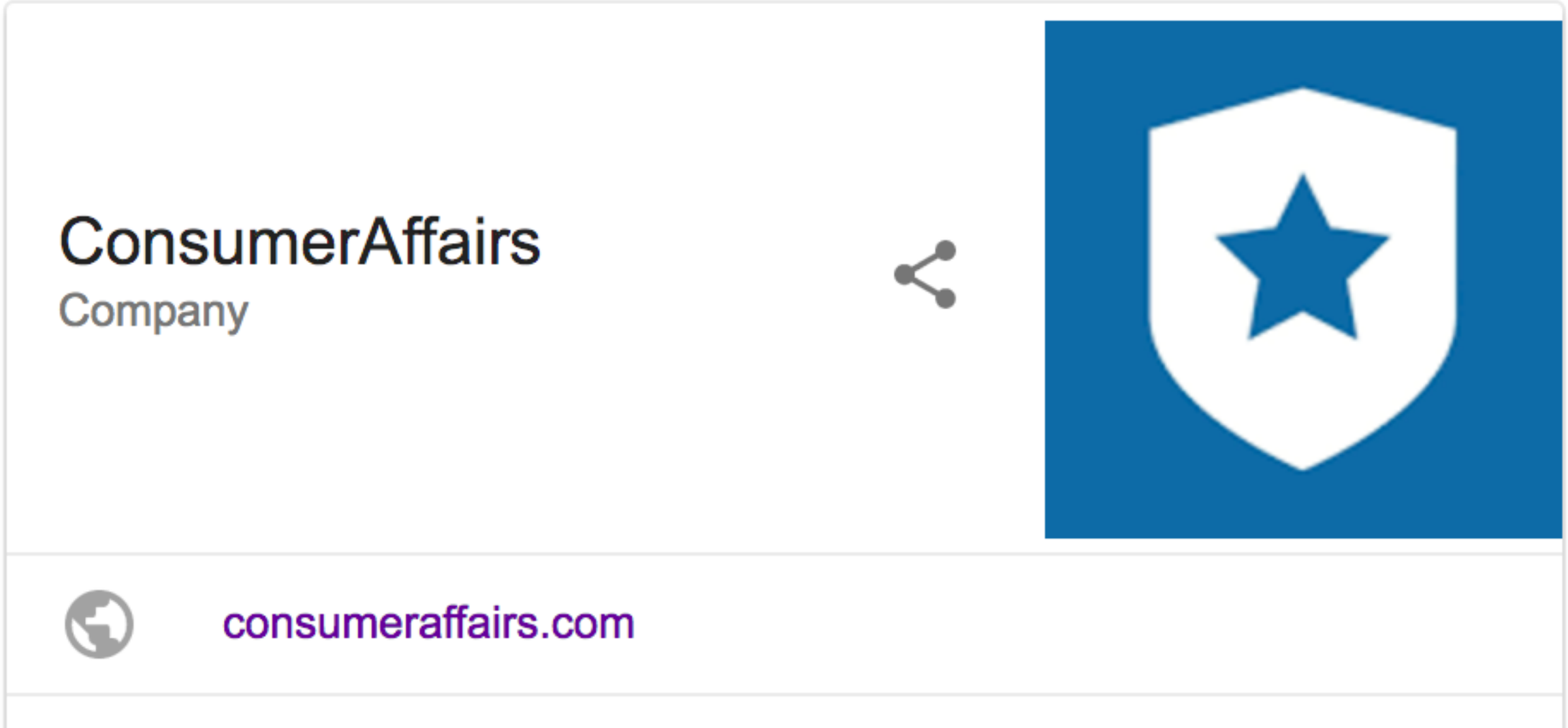
Conversation 3 Commits 1 Files changed 831

Changes from **all commits** 831 files +6 −2,066

https://github.com/django/django/pull/7867

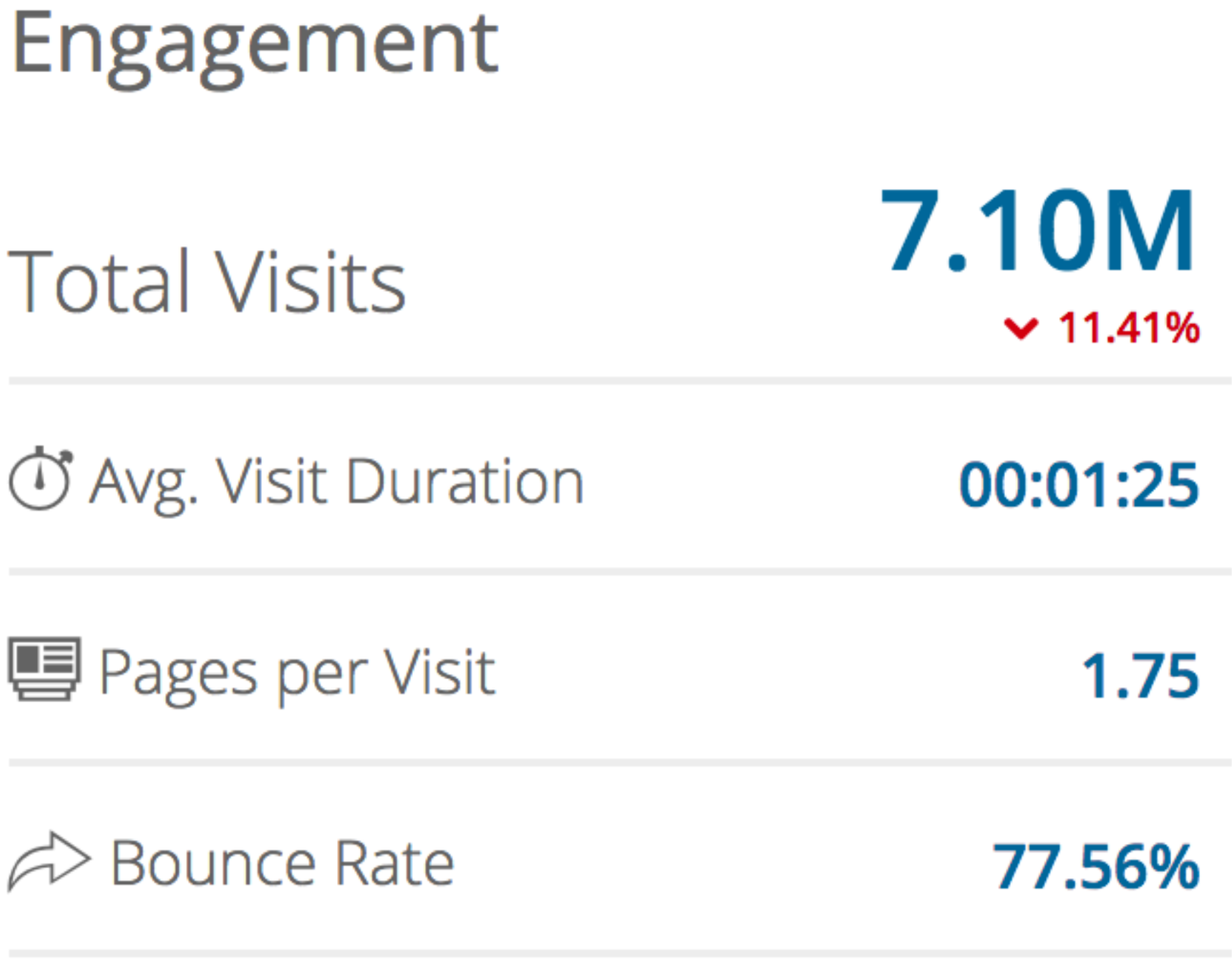
@a_soldatenko

Real world example



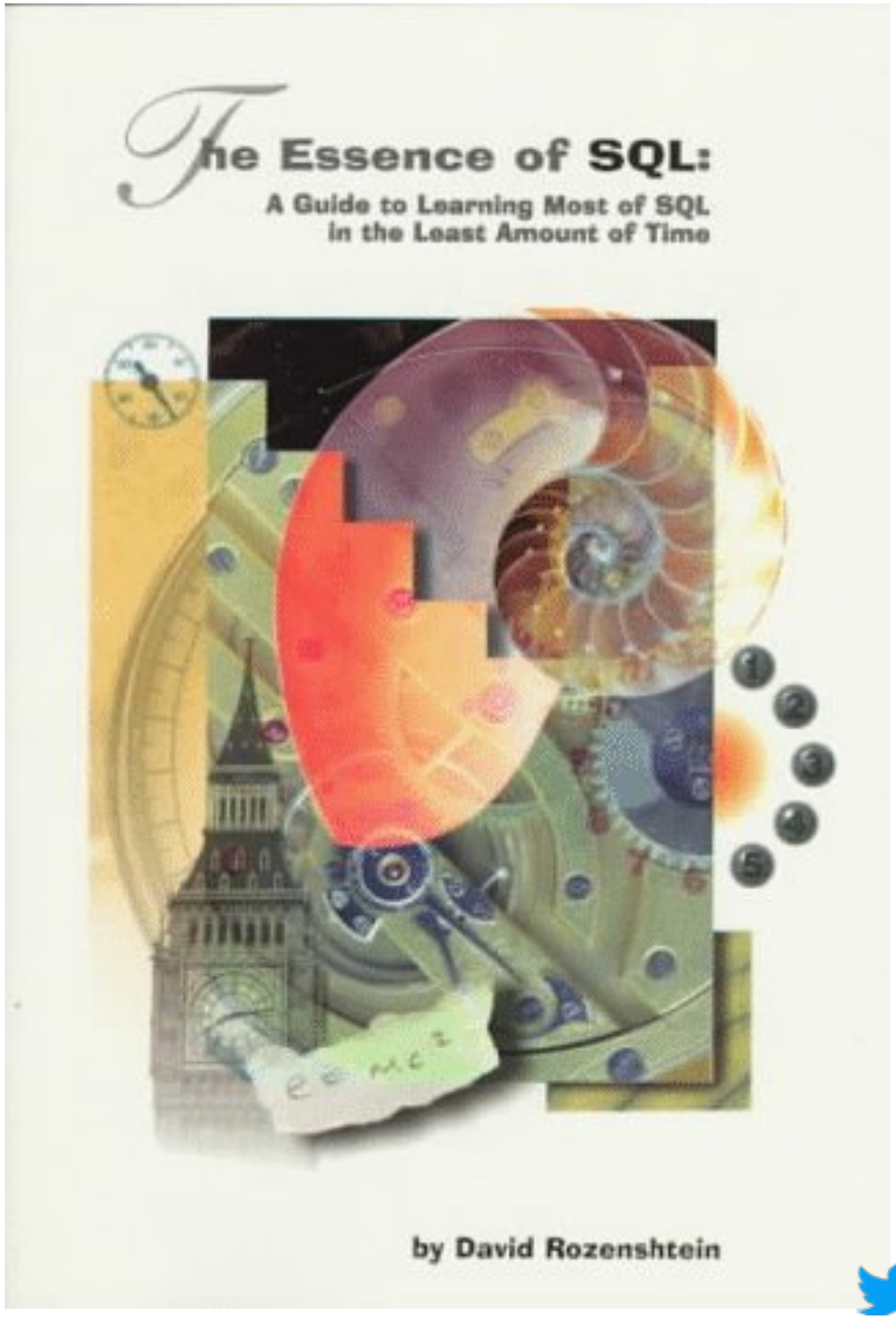
@a_soldatenko

Similar web of CA



@a_soldatenko

Rozenshtein tests



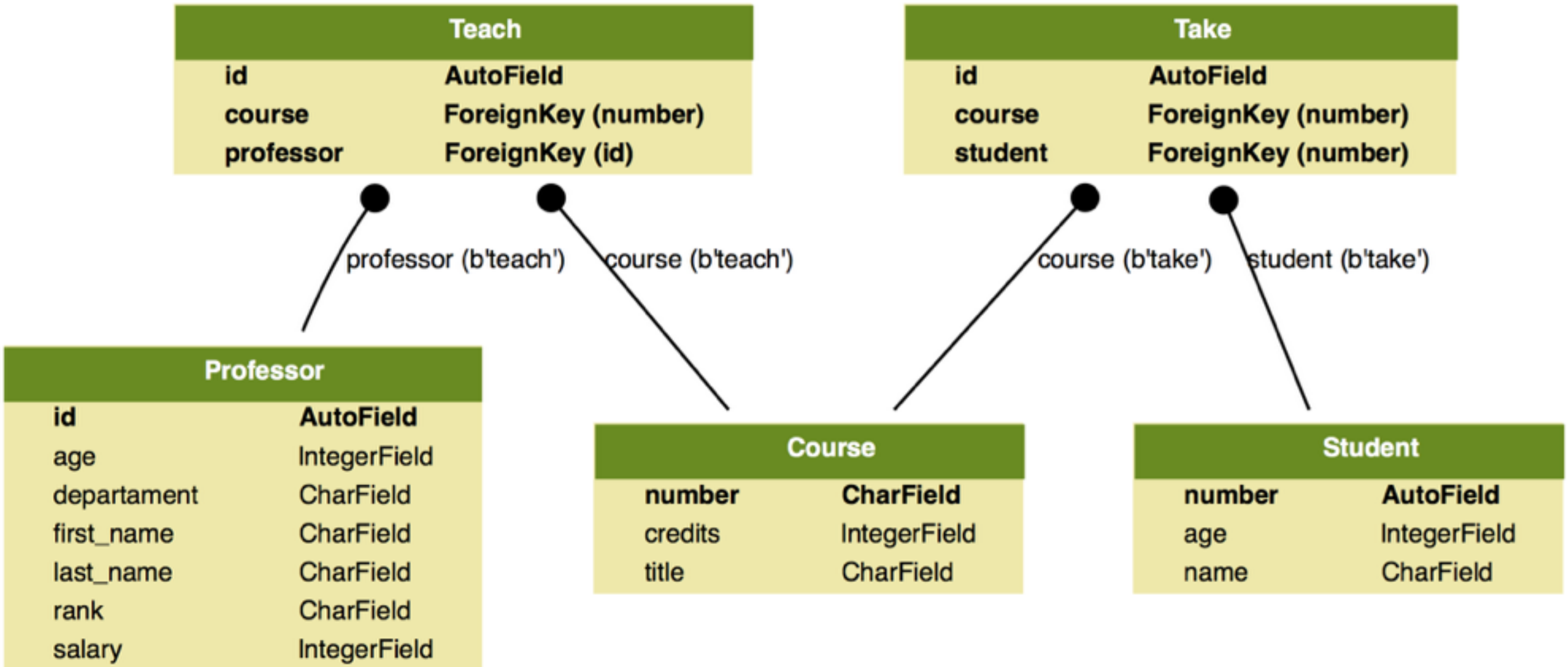
@a_soldatenko

Django crash tests project

https://github.com/andriisoldatenko/django-orm-crash-test

@a_soldatenko

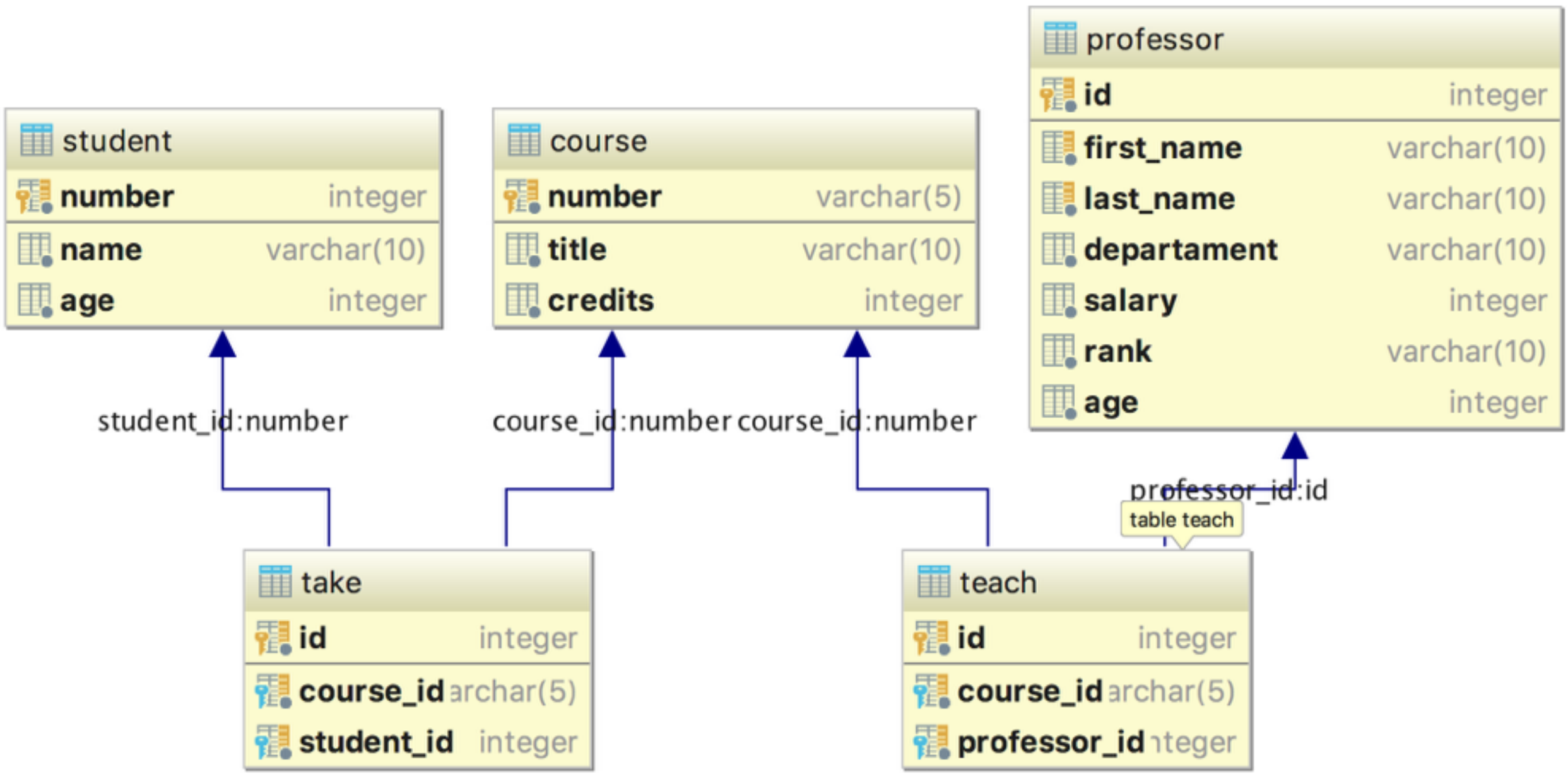
University Database



DjangoGraphviz

@a_soldatenko

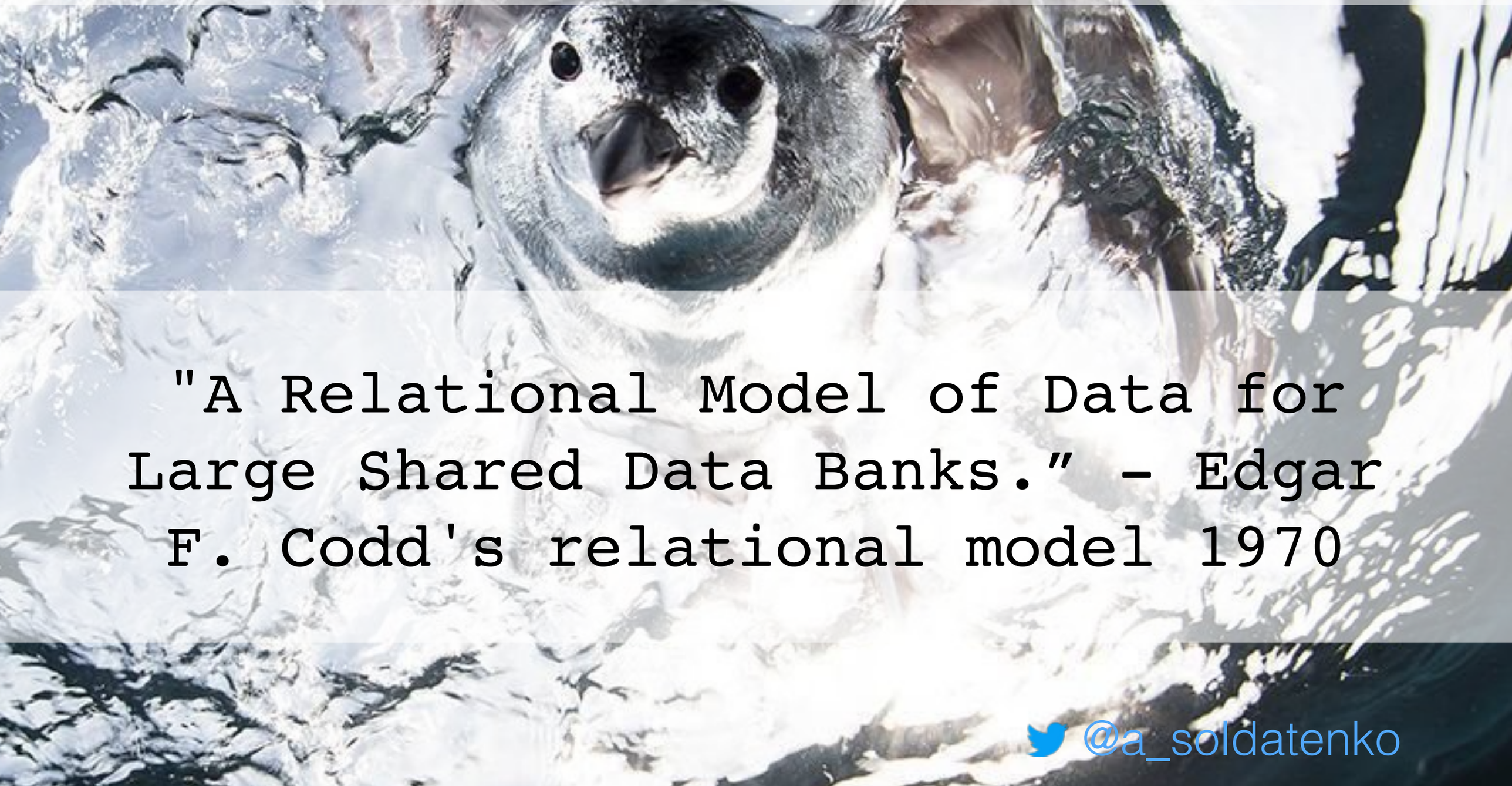
University Database



PyCharm

@a_soldatenko

A Bird's Eye View of SQL



"A Relational Model of Data for Large Shared Data Banks." - Edgar F. Codd's relational model 1970

@a_soldatenko

DDL

```
CREATE TABLE course (
    number varchar(5) NOT NULL PRIMARY KEY,
    title varchar(10) NOT NULL,
    credits integer NOT NULL
);
```

```
ALTER TABLE teach ADD CONSTRAINT
"teach_course_id_bebbbd64_fk_course_number"
FOREIGN KEY (course_id) REFERENCES course
("number")
DEFERRABLE INITIALLY DEFERRED;
```

@a_soldatenko

DML

```
SELECT ... FROM ... WHERE ...
INSERT INTO ... VALUES ...
UPDATE ... SET ... WHERE ...
DELETE FROM ... WHERE ...
```

@a_soldatenko

PostgreSQL Hackers

https://www.postgresql.org/list/pgsql-hackers/

https://wiki.postgresql.org/wiki/
So,_you_want_to_be_a_developer%3F

@a_soldatenko

Tom Lane

@a_soldatenko

ORM under the hood

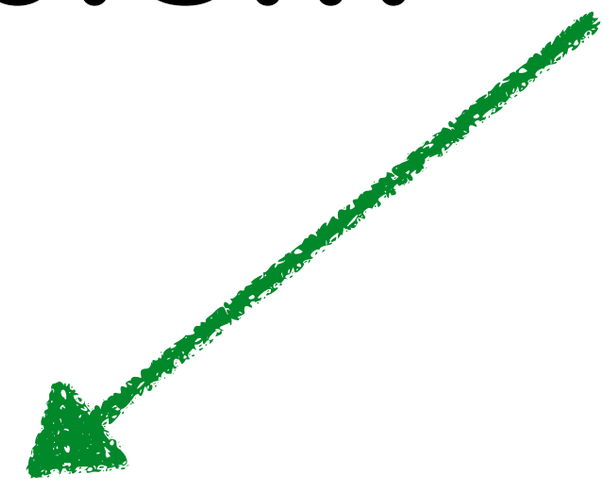
@a_soldatenko

Primary keys problem

```
class Student(models.Model):
    # id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=10)
    age = models.IntegerField()
```

@a_soldatenko

Primary keys problem



```
from django.db import models


class Student(models.Model):
    number = models.AutoField(primary_key=True)
    name = models.CharField(max_length=10)
    age = models.IntegerField()

    def __str__(self):
        return self.name

    class Meta:
        db_table = 'student'
```

@a_soldatenko

Multi-Column Primary key problem

```
class Student(models.Model):
    id = models.AutoField(primary_key=True)
    number = models.AutoField(primary_key=True)
    name = models.CharField(max_length=10)
    age = models.IntegerField()
```

```
    new_class.add_to_class(obj_name, obj)
  File "/Users/andrii/.pyenv/versions/django-orm-crash-test/lib/
python3.6/site-packages/django/db/models/base.py", line 328, in
add_to_class
    value.contribute_to_class(cls, name)
  File "/Users/andrii/.pyenv/versions/django-orm-crash-test/lib/
python3.6/site-packages/django/db/models/fields/__init__.py", line 965,
in contribute_to_class
    assert not cls._meta.auto_field, "A model can't have more than one
AutoField."
AssertionError: A model can't have more than one AutoField.
```

Multi-Column Primary keys problem PostgreSQL

```
CREATE TABLE student_w_two_primary (
    id INTEGER,
    number INTEGER,
    name VARCHAR(10) NOT NULL,
    age INTEGER NOT NULL,
    PRIMARY KEY (id, number)
);
```

@a_soldatenko

Multi-Column Primary keys major issues

 # obj._meta.pk

 # backward for (content_type_id, object_pk)

 # Admin URLs: ”/app_label/ module_name/pk/"

@a_soldatenko

Multi-Column Primary keys: Proposed Solutions

```
class Student(models.Model):
    id = models.BigIntegerField()
    number = models.AutoField(primary_key=True)
    name = models.CharField(max_length=10)
    age = models.IntegerField()

    class Meta:
        db_table = 'student'
        unique_together = ('id', 'number')
```

@a_soldatenko

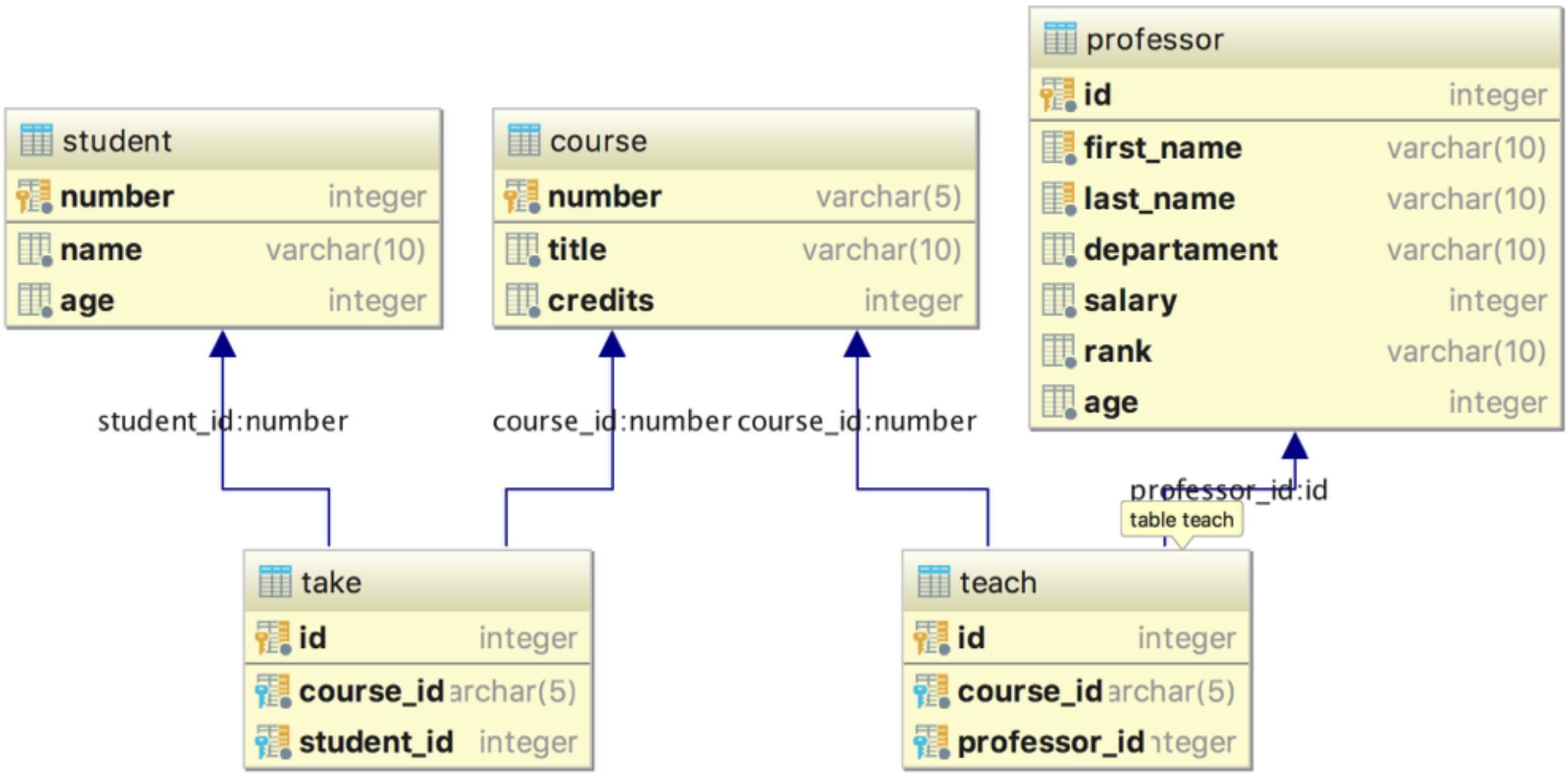
Multi-Column Primary keys: Proposed Solutions

```
ALTER TABLE student ADD CONSTRAINT
    "student_id_number_345a5f6e_uniq"
UNIQUE ("id", "number");
```

https://code.djangoproject.com/wiki/MultipleColumnPrimaryKeys

@a_soldatenko

University Database



@a_soldatenko

How takes the course CS112?

```
Take.objects.filter(
    course__number='CS112'
).values('student')
```

```
SELECT "take"."student_id"
FROM "take"
WHERE "take"."course_id" = 'CS112';
```

@a_soldatenko

What are student numbers and names of students who take CS112?

```
Take.objects.filter(
    course__number='CS112'
).values('student__name', 'number')
```

```
SELECT “student"."name", "take"."student_id"
FROM take
INNER JOIN student
ON ("take"."student_id" = "student"."number")
WHERE "take"."course_id" = 'CS112';
```

@a_soldatenko

Who takes the course CS112 or CS114?

```
Take.objects.filter(
    course__number__in=['CS112', 'CS114']
).values('student__name', 'number')
```

```
SELECT "student"."name"
FROM "take"
INNER JOIN "student" ON
    ("take"."student_id" = "student"."number")
WHERE "take"."course_id" IN ('CS112', 'CS114')
```

@a_soldatenko

Who takes the course CS112 or CS114?

```
from django.db.models import Q

Take.objects.filter(
    Q(course__number='CS112') |
    Q(course__number='CS114')
).values('student__name')
```

```
SELECT "student"."name"
FROM "take"
INNER JOIN "student" ON ("take"."student_id" =
"student"."number")
WHERE ("take"."course_id" = 'CS112'
    OR "take"."course_id" = 'CS114');
```

@a_soldatenko

Who takes both courses CS112 and CS114?

```
Take.objects.filter(
    course__number='CS112'
).filter(course__number='CS114'
    ).values('student__name')
```

```
SELECT "student"."name"
FROM "take"
INNER JOIN "student" ON ("take"."student_id" =
"student"."number")
WHERE ("take"."course_id" = 'CS112'
      AND "take"."course_id" = 'CS114');
```

@a_soldatenko

Table inheritance problem

@a_soldatenko

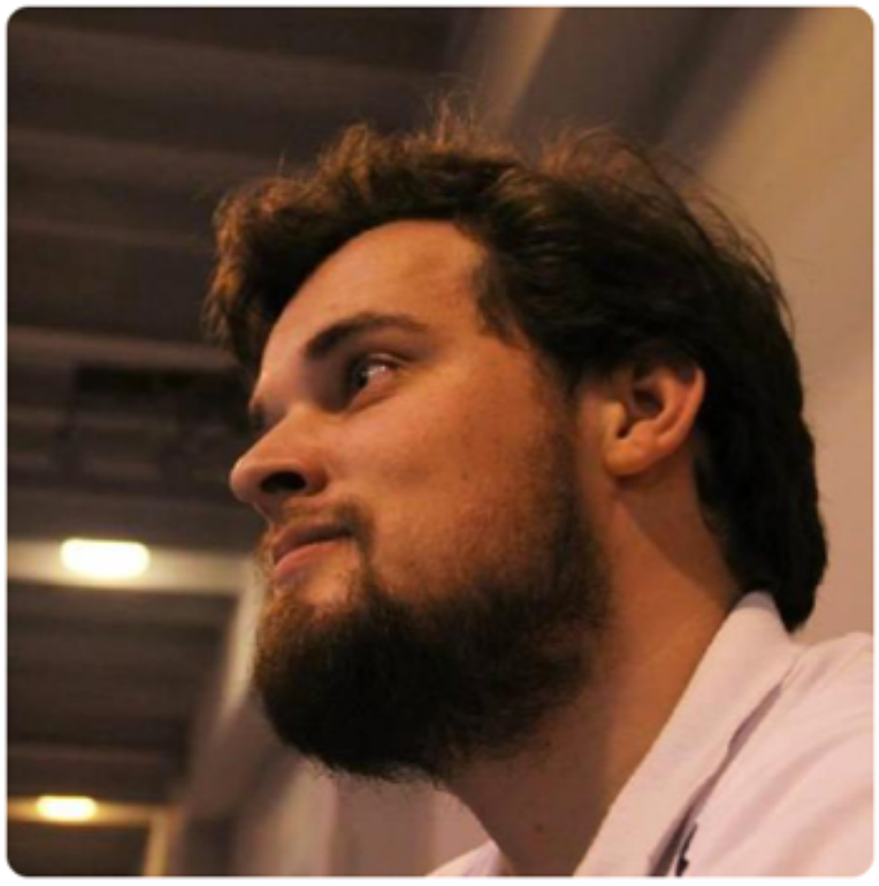
Table inheritance problem

```
CREATE TABLE student
(
   number INTEGER PRIMARY KEY NOT NULL,
   name VARCHAR(10) NOT NULL,
   age INTEGER NOT NULL
);

CREATE TABLE professor
(
   departament VARCHAR(10) NOT NULL,
   salary INTEGER NOT NULL,
   rank VARCHAR(10) NOT NULL,
) INHERITS (student);
```

@a_soldatenko

PostgreSQL table inheritance

…ultimately rejected it as I'm really not sure how we can handle it.



Marc Tamlyn
mjtamlyn

https://code.djangoproject.com/ticket/24632

Enable SQL logging.v0

raw SQL queries Django is running?

```
# variant 0
In [1]: from django.db import connection

In [2]: Student.objects.all()
Out[2]: <QuerySet [<Student: AARON>, <Student:
CHUCK>, <Student: DOUG>,...]>

In [3]: connection.queries
Out[3]:
[{'sql': 'SELECT "student"."id",
"student"."number", "student"."name",
"student"."age" FROM "student" LIMIT 21',
  'time': '0.003'}]
```

@a_soldatenko

Enable SQL logging.v1

raw SQL queries Django is running?

```
# variant 1
# pip install django-extensions
./manage.py shell_plus --print-sql
```

```
In [1]: student = Student.objects.first()
SELECT "student"."id", "student"."number",
"student"."name", "student"."age" FROM "student"
ORDER BY "student"."number" ASC LIMIT 1

Execution time: 0.002584s [Database: default]
```

@a_soldatenko

Enable SQL logging.v2

raw SQL queries Django is running?

```
# variant 3
# pip install django-debug-toolbar

./manage.py debugsqlshell
```

```
In [2]: from university.models import Student

In [3]: Student.objects.all()
Out[3]: SELECT "student"."id",
       "student"."number",
       "student"."name",
       "student"."age"
FROM "student" LIMIT 21 [3.10ms]
<QuerySet [<Student: AARON>, <Student: CHUCK>,]>
```

Enable SQL logging.v3

```
# variant 3
In [2]: from university.models import Student

In [4]: import logging
   ...: l =
logging.getLogger('django.db.backends')
   ...: l.setLevel(logging.DEBUG)
   ...: l.addHandler(logging.StreamHandler())

In [6]: Student.objects.first()
(0.003) SELECT "student"."id",
"student"."number", "student"."name",
"student"."age" FROM "student" ORDER BY
"student"."number" ASC LIMIT 1; args=()
Out[6]: <Student: AARON>
```

@a_soldatenko

Enable SQL logging.v4

```
In [1]: from django.test.utils import CaptureQueriesContext
In [6]:  with CaptureQueriesContext(connection) as context:
   ...:
print(Take.objects.filter(Q(course__number='CS112')).values(
'student__name'))
   ...:
<QuerySet [{'student__name': 'AARON'}, {'student__name':
'CHUCK'}, {'student__name': 'DOUG'}, {'student__name':
'MAGGIE'}]>
In [8]:  print(context.captured_queries)
[{'sql': 'SELECT "student"."name" FROM "take" INNER JOIN
"student" ON ("take"."student_id" = "student"."number")
WHERE "take"."course_id" = \'CS112\' LIMIT 21', 'time':
'0.005'}]
```

@a_soldatenko

Enable SQL logging.v5

Transaction trace

Track as Key Transaction

Delete this trace

Dec 9, '16 8:38 pm	4,890 ms	301 ms (6.17%)	77.3 ms (1.58%)
TRACE TIME	RESP. TIME	USER CPU BURN	SYSTEM CPU BURN

Summary | Trace details | **Database queries**

Show fast queries (< 5ms)

Total duration	Call count	Query
10 ms	7	`SELECT fc.*FROM field_config fcWHERE (fc.field_name = :db_condition_placeholder_?) AND (fc.active = :db_condition_placeholder_?) AND (fc.storage_active = :db_condition_placeholder_?) AND (fc.deleted = :db_condition_placeholder_?)`
98 ms	3	`DELETE FROM variable WHERE (name = :db_condition_placeholder_?)`
4,340 ms	1	`INSERT INTO semaphore (name, value, expire) VALUES (:db_insert_placeholder_?, :db_insert_placeholder_?, :db_insert_placeholder_?)`

Query details

Duration	4,344 ms

Stack trace

```
in PDOStatement::execute called at /mnt/www/html/newrelic/docroot/includes/database/database.inc (2171)
in DatabaseStatementBase::execute called at /mnt/www/html/newrelic/docroot/includes/database/database.inc (683)
```

@a_soldatenko

SQL string concatenation

```
Student.objects.all().update(age=F('age')+10)
```

```
UPDATE "student"
SET "age" = ("student"."age" + 10)
```

@a_soldatenko

SQL AS
column aliases

```
In [33]: Course.objects.extra(select={'id':
'number'}).values()
Out[33]: SELECT (number) AS "id",
         "course"."number",
         "course"."title",
         "course"."credits"
FROM "course" LIMIT 21

<QuerySet [{'id': 'CS112', 'number': 'CS112',
'title': 'PHYSICS', 'credits': 4}, {'id': 'CS113',
'number': 'CS113', 'title': 'CALCULUS', 'credits':
4}, {'id': 'CS114', 'number': 'CS114', 'title':
'HISTORY', 'credits': 4}]>
```

@a_soldatenko

SQL Functions

 # Concat

 # Greatest

 # and more

https://docs.djangoproject.com/en/1.10/ref/models/database-functions/

@a_soldatenko

Migration problem

```
✗ ./manage.py sqlmigrate university 0002
BEGIN;
--
-- Add field id to student
--
ALTER TABLE "student" ADD COLUMN "id" bigint DEFAULT 1 NOT
NULL;
ALTER TABLE "student" ALTER COLUMN "id" DROP DEFAULT;
--
-- Alter unique_together for student (1 constraint(s))
--
ALTER TABLE "student" ADD CONSTRAINT
"student_id_number_345a5f6e_uniq" UNIQUE ("id", "number");
COMMIT;
```

@a_soldatenko

Full text search problem

EuroPython 2016

July 21, 2016

What is the best full text search engine for python?

EUROPYTHON 2016 Bilbao, 17-24 July

@a_soldatenko

Full text search problem

New in Django 1.10.

But since

PostgreSQL 8.3

@a_soldatenko

django.contrib.postgres.search

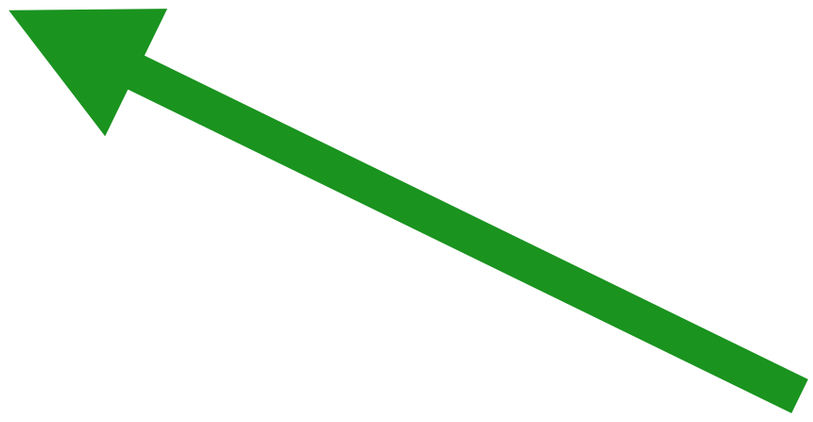
```
In [15]:
Course.objects.annotate(search=SearchVector('title
')).filter(search='PHYSIC')
Out[15]: SELECT "course"."number",
         "course"."title",
         "course"."credits",
         to_tsvector(COALESCE("course"."title", ''))
AS "search"
FROM "course"
WHERE to_tsvector(COALESCE("course"."title", ''))
@@ (plainto_tsquery('PHYSIC')) = true LIMIT 21

Execution time: 0.002756s [Database: default]

<QuerySet [<Course: CS112: PHYSICS>]>
```

Two nuances with FTS in Django

@a_soldatenko

You should add search_field create FTS field

```
class Course(models.Model):
    number = models.CharField(…)
    title = models.CharField(..)
    credits = models.IntegerField()
    search_vector = SearchVectorField()
```



You should manually create trigger to update search_vector

```
CREATE TRIGGER tsvectorupdate BEFORE
INSERT OR UPDATE
ON course FOR EACH ROW EXECUTE
PROCEDURE
tsvector_update_trigger(search_vector,
'pg_catalog.english', title);"
```

@a_soldatenko

If you want to use different PG search functions

```
class TsQuerySearchQuery(SearchQuery):
    def as_sql(self, compiler, connection):
        params = [self.value]
        if self.config:
            config_sql, config_params = compiler.compile(self.config)
            template = 'to_tsquery({}::regconfig, %s)'.format(config_sql)
            params = config_params + [self.value]
        else:
            template = 'to_tsquery(%s)'
        if self.invert:
            template = '!!({})'.format(template)
        return template, params
```

@a_soldatenko

And search query will looks like

```
Course.objects.filter(
    search=TsQuerySearchQuery('PHYSIC:A')
)
```

cd django
git checkout 1.11.x

 # Subquery

 # Exists

 # OuterRef

https://docs.djangoproject.com/en/1.11/ref/models/expressions/#django.db.models.Subquery

Async database problem

mike bayer
@zzzeek

Following

the reason I ask is bc, I've tried for 2 days to get asyncio *or* gevent to outperform threads on a real RDBMS speed test, and I can't.

RETWEETS	LIKES
4	4

1:44 AM - 13 Feb 2015

5 4 4

http://techspot.zzzeek.org/2015/02/15/asynchronous-python-and-databases/

Performance problem: pgbench

✗ docker exec -it postgresql sudo -u postgres pgbench -i -s 15 university_db
creating tables...
100000 of 1500000 tuples (6%) done (elapsed 0.77 s, remaining 10.79 s)
200000 of 1500000 tuples (13%) done (elapsed 2.37 s, remaining 15.39 s)
300000 of 1500000 tuples (20%) done (elapsed 4.13 s, remaining 16.51 s)

set primary keys...
done.

https://www.postgresql.org/docs/9.6/static/pgbench.html

Final Thoughts

- Thanks David Rozenshtein for his tests

- django.contrib.postgres

- New crash-test SQLALchemy

- github.com/andriisoldatenko/django-orm-crash-test

@a_soldatenko

Thank You

andrii.soldatenko@toptal.com

https://asoldatenko.com

@a_soldatenko

Questions

toptal

@a_soldatenko

We are hiring

https://www.toptal.com/#connect-fantastic-computer-engineers