Pony ORM

Dominik Kozaczko, PyWaw #73

Trochę o sobie

- w Pythonie od 2005 roku; w Django od 2006
- wraz z uczniami przetłumaczyłem “Byte of Python”
- uczyłem Pythona w liceum zanim to stało się modne ;)
- zainicjowałem akcję “Python na maturze” (kudos dla CEO)
- uczę Pythona i Django na YouTube - youtube.com/dekoza

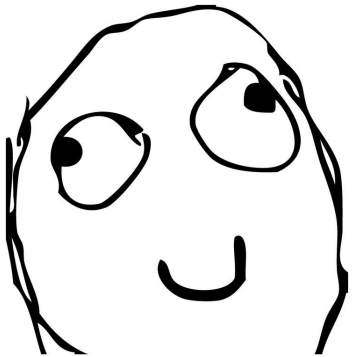
Plan prezentacji

1. Zady i walety ORM.
2. Co nam daje PonyORM?
3. Ograniczenia
4. Przykłady
5. Jak to działa pod maską?
6. Podsumowanie

Zady i walety ORM

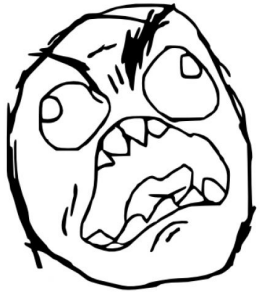
Fajniusie:

- obiektowa manipulacja wpisami w bazie
- wygodne portowanie struktury bazy oraz zapytań



Mniej fajniusie:

- własna składnia zapytań
- rakowacenie złożonych kwerend




See this pony?
This is an awesome pony!
Umm...thanks?
new hub

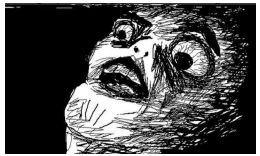
Co nam daje Pony ORM?

- obiektowa manipulacja wpisami w bazie
- wygodne portowanie struktury bazy oraz zapytań
- **pythonowa składnia zapytań**
- **wysoka optymalizacja nawet złożonych kwerend**

Ej! Na pewno ma jakieś wady!

- dłuższe kwerendy robią się nieczytelne i trzeba rozbijać
- ciężko się przestawić z “tradycyjnych” ORMów
- jeszcze nie wszystkie funkcje są zaimplementowane

Przykład - deklaracja modeli

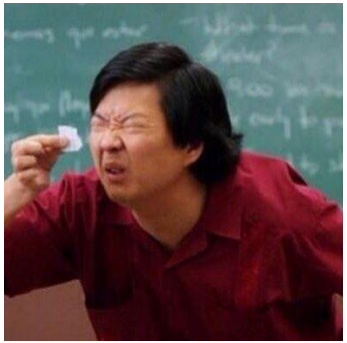


```python
from pony.orm import *

db = Database("sqlite", "estore.sqlite", create_db=True)

class Customer(db.Entity):
    email = Required(str, unique=True)
    password = Required(str)
    name = Required(str)
    country = Required(str)
    address = Required(str)
    cart_items = Set("CartItem")
    orders = Set("Order")
```

```python
class Product(db.Entity):
    id = PrimaryKey(int, auto=True)
    name = Required(str)
    categories = Set("Category")
    description = Optional(str)
    picture = Optional(buffer)
    price = Required(Decimal)
    quantity = Required(int)
    cart_items = Set("CartItem")
    order_items = Set("OrderItem")
```

etc…

Przykład - zapytania



```
result = select(p for p in Product).order_by(
   lambda p: desc(
      sum(p.order_items.quantity)
   )
).first()
```

```
SELECT "p"."id"
FROM "Product" "p"
  LEFT JOIN "OrderItem" "orderitem"
    ON "p"."id" = "orderitem"."product"
GROUP BY "p"."id"
ORDER BY coalesce(SUM("orderitem"."quantity"), 0) DESC
LIMIT 1
```

```
result = select((customer, category.name)
                  for customer in Customer
                  for product in customer.orders.items.product
                  for category in product.categories
                  if count(product) > 1)[:]
```

```
SELECT DISTINCT "customer"."id", "category"."name"
FROM "Customer" "customer", "Order" "order", "OrderItem" "orderitem",
"Category_Product" "t-1", "Category" "category"
WHERE "customer"."id" = "order"."customer"
  AND "order"."id" = "orderitem"."order"
  AND "orderitem"."product" = "t-1"."product"
  AND "t-1"."category" = "category"."id"
GROUP BY "customer"."id", "category"."name"
HAVING COUNT(DISTINCT "orderitem"."product") > 1
```

Jak to działa pod maską?

1. Dekompilacja generatora (i lambd) i przebudowanie go do postaci Abstract Source Tree
2. Przetłumaczenie AST na “abstrakcyjny SQL”
3. Przetłumaczenie abstrakcyjnego SQLa do konkretnego dialektu

Źródło: https://stackoverflow.com/questions/16115713/how-pony-orm-does-its-tricks

Jak to działa pod maską?

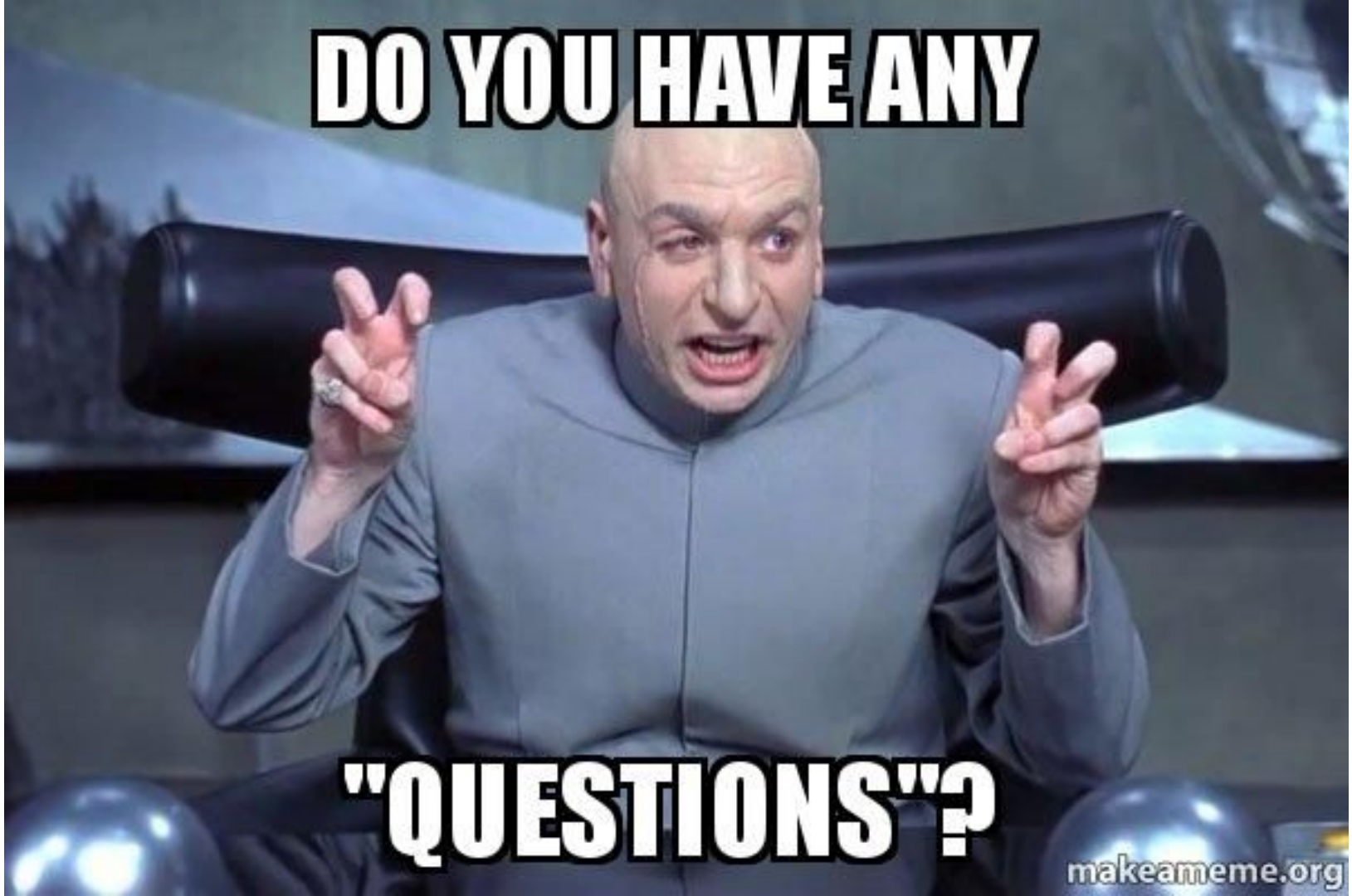
```
>>> gen = (c for c in Customer if c.country == 'USA')
>>> import dis
>>> dis.dis(gen.gi_frame.f_code)
 1           0 LOAD_FAST                0 (.0)
     >>      3 FOR_ITER                26 (to 32)
             6 STORE_FAST               1 (c)
             9 LOAD_FAST                1 (c)
            12 LOAD_ATTR                0 (country)
            15 LOAD_CONST               0 ('USA')
            18 COMPARE_OP               2 (==)
            21 POP_JUMP_IF_FALSE        3
            24 LOAD_FAST                1 (c)
            27 YIELD_VALUE
            28 POP_TOP
            29 JUMP_ABSOLUTE            3
     >>     32 LOAD_CONST               1 (None)
            35 RETURN_VALUE
```

```
>>> from pony.orm.decompiling import decompile
>>> ast, external_names = decompile(gen)
>>> ast
GenExpr(GenExprInner(Name('c'),
[GenExprFor(AssName('c', 'OP_ASSIGN'), Name('.0'),
[GenExprIf(Compare(Getattr(Name('c'), 'country'), [('==',
Const('USA'))])])]))
```

Źródło: https://stackoverflow.com/questions/16115713/how-pony-orm-does-its-tricks

Ciekawe konsekwencje

- generowanie optymalnego kodu SQL
- możliwość keszowania fragmentów oraz całych zapytań
    - tworzenie zapytania jest szybsze 1,5x - 3x niż w Django czy SQLAlchemy


DO YOU HAVE ANY
"QUESTIONS"?
makeameme.org

DZIEKUJĘ!

Live long and may the Force be with you (͡° ͜ʖ ͡°)