Building a Multi-Purpose Platform For Bulk Data Using SqlAlchemy

Introducing a way of building data processing applications that can be used in many business domains

Christian Trebing (@ctrebing)

blueyonder

Europython 2015

SQLAlchemy

From SQLAlchemy website:

> *SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.*
>
> *SQL databases behave less like object collections the more size and performance start to matter; object collections behave less like tables and rows the more abstraction starts to matter. SQLAlchemy aims to accommodate both of these principles.*

Let's Build... a Multi Domain Platform

Business Requirements

- Load bulk data via csv
- Verify/clean data
- Make clean data available to machine learning
- Different business domains: Retail, Tourism, ...

Let's Build... a Multi Domain Platform

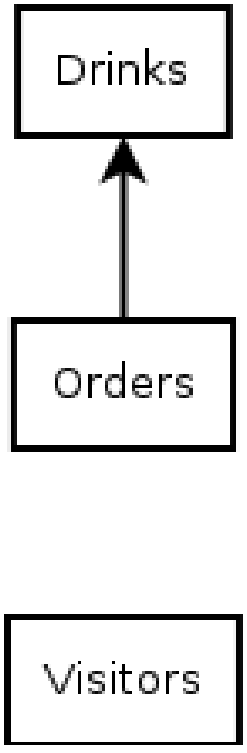
Technical Todos

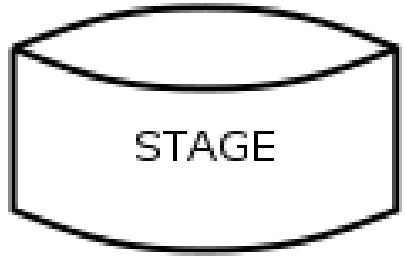
- Create database schema
- Parse csv, save parsed csv to database
- Validate data
  - Check required fields
  - Check references between data records
  - ...
- Give feedback to customer about processing status
- Separate clean, validated data from raw input

First Customer: A Pub

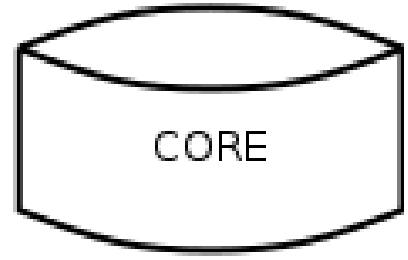
source: wikimedia commons

Data Model for Pub





ST_DRINKS
ST_ORDERS
ST_VISITORS



DRINKS
ORDERS
VISITORS

A CSV Delivery

```
--Drinks--
ExternalCode,Description,Alcohol
BEER,        Beer,       4.9
WHISKY,      Whisky,     40
COKE,        Coca Cola,

--Orders--
ExternalCode,Drinks, Count
2015-07-10,  BEER,   10
2015-07-10,  COKE,   8
2015-07-11,  BEER,   15
2015-07-11,  WHISKY, 2
2015-07-12,  BEER,   13
2015-07-12,  WHISKEY,1
```

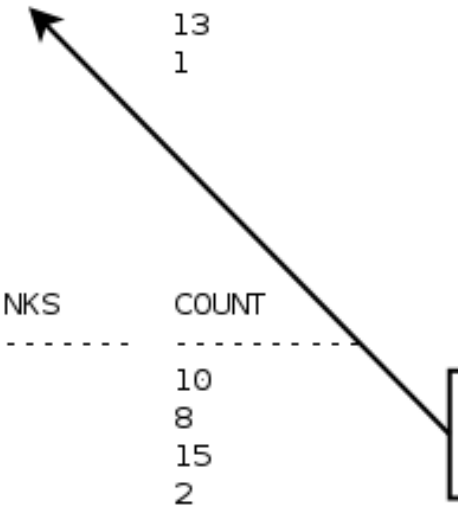
One Task: Find References Between Objects

--Orders(stage)--

EXTERNALCODE	DRINKS	DRINKS_REF	COUNT
2015-07-10	BEER		10
2015-07-10	COKE		8
2015-07-11	BEER		15
2015-07-11	WHISKY		2
2015-07-12	BEER		13
2015-07-12	WHISKEY		1

--Orders(core)--

ID	EXTERNALCODE	DRINKS	COUNT
1	2015-07-10	1	10
2	2015-07-10	3	8
3	2015-07-11	1	15
4	2015-07-11	2	2
5	2015-07-12	1	13



--Drinks(core)--

ID	EXTERNALCODE	DESCRIPTI(
1	BEER	Beer
2	WHISKY	Whisky
3	COKE	Coca Cola

How To Implement Reference Finding? Core / ORM

SQL:

```
UPDATE ST_ORDERS SET DRINKS_REF=(
        SELECT ID FROM DRINKS
        WHERE EXTERNALCODE=ST_ORDERS.DRINKS);
```

Core:

```
stmt = orders_stage.update().values(drinks_ref=(
    select([drinks_core.c.id]).\\
    where(drinks_core.c.externalcode==\\
        orders_stage.c.drinks)))
```

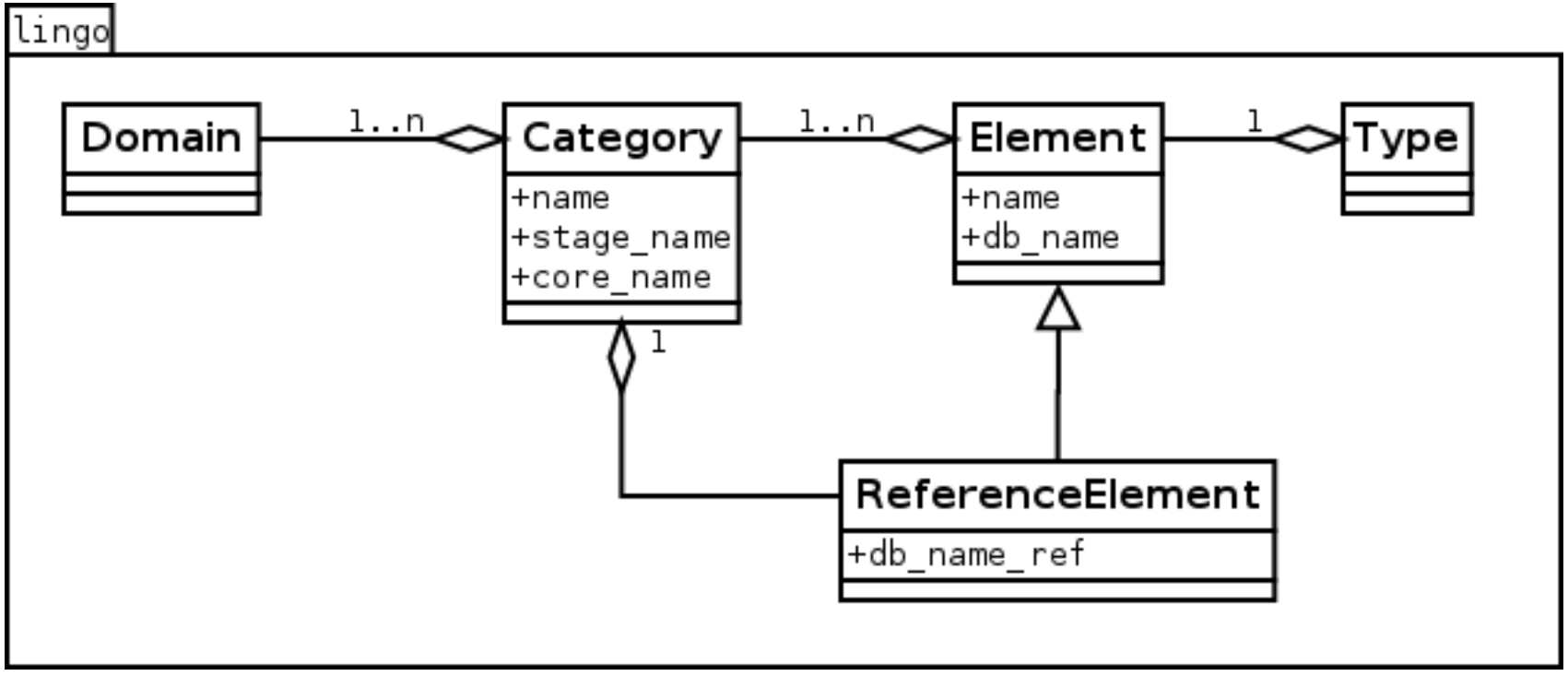
ORM:

```
for order in session.query(Orders_stage).all():
    id = session.query(Drinks_core.id).\\
         filter(Drinks_core.externalcode==\\
            order.drinks).one()
    order.drinks_ref = id
```

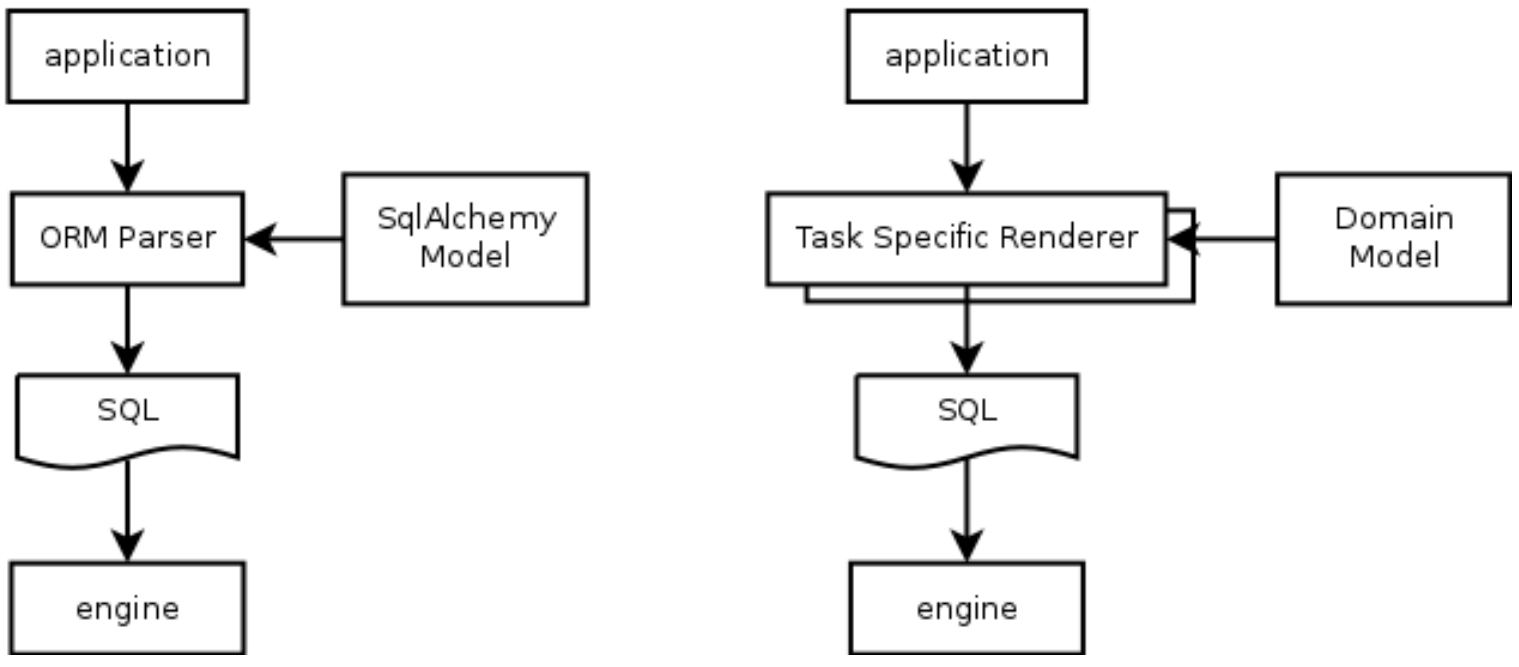
Next Customer: A Brewery

source: wikimedia commons

Domain Model: The Lingo Package



Task Specific Model Renderers



Code Sample: Domain for Pub

```
from lingo import Domain, Category, elements as e
class Pub(Domain):
    def __init__(self):
        super(Pub, self).__init__()
        drinks = Category(
            'Drinks',
            e.ExternalCode(),
            e.Description(),
            e.Numeric('Alcohol', 3, 2),
            e.String('Handling', 200)
        )

        visitors = Category(
            'Visitors',
            e.ExternalCode(),
            e.Numeric('Count', 6, 0)
        )

        orders = Category(
            'Orders',
            e.ExternalCode(),
            e.Reference(drinks),
            e.Numeric('Count', 6, 0)
        )
```

Code Sample: Find References

```
1  references = [e for e in category.elements if
       isinstance(e, ReferenceElement)]
2  for element in references:
3      # stage/core_tables are SqlAlchemy Metadata
4      stage = stage_tables[element.category.stage_name]
5      ref_core = core_tables[
6          element.ref_category.core_name]
7      update_dict = {
8          stage.c[element.db_name_ref]: \\
9          (select([ref_core.c.ID]).where(
10             ref_core.c.EXTERNALCODE == \\
11             stage.c[element.db_name]
12         ))
13     }
14     statement = stage.update().values(update_dict)
15     print str(statement)
16     self.engine.execute(statement)
```

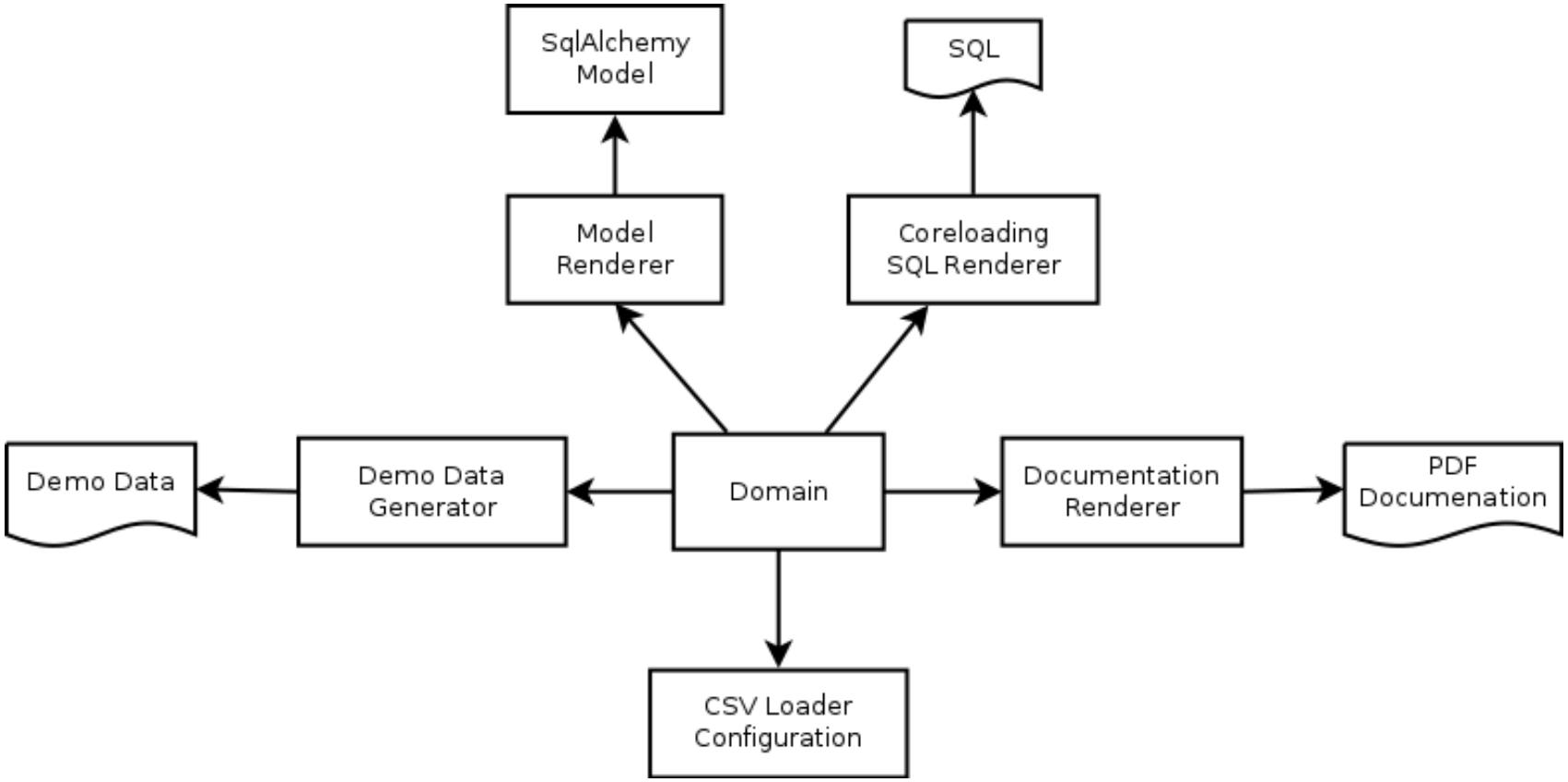
Code Sample: Domain for Brewery

```
1  from lingo import Domain, Category, elements as e
2  class Brewery(Domain):
3      def __init__(self):
4          super(Brewery, self).__init__()
5          machines = Category(
6              'Machines',
7              e.ExternalCode(),
8              e.Description()
9          )
10         sensors = Category(
11             'Sensors',
12             e.ExternalCode(),
13             e.Description(),
14             e.Reference(machines)
15         )
16         measurements = Category(
17             'Measurements',
18             e.ExternalCode(),
19             e.Numeric('Value', 10, 3),
20             e.Reference(sensors)
21         )
```

Discussion of Domain Model and Task Specific Renderers

- Optimized for high throughput of large amounts of data
  - Good fit for analytical models
  - May not fit that well for transactional models
- Comparison to SqlAlchemy model
  - SqlAlchemy model is focused on database description
  - Domain model can contain more information
    - for example: time dependent references, value checks
  - SqlAlchemy model can be generated out of domain model

Applications Of Domain Model




We're passionate about
Big Data. You too?
Then join us >>
www.blue-yonder.com
blueyonder