Core режим

Сначала необходимо определить, как данные хранятся в таблице. Варианты определения:

- Объект Table
- Декларативный класс
- Получение структуры из базы данных

Сопоставление типов

SQLAlchemy	Python	SQL
BigInteger	int	BIGINT
Boolean	bool	BOOLEAN or SMALLINT
Date	datetime.date	DATE (SQLite: STRING)
DateTime	datetime.datetime	DATETIME (SQLite: STRING)
Time	datetime.time	DATETIME
Enum	str	ENUM or VARCHAR
Float	float or Decimal	FLOAT or REAL
Integer	int	INTEGER
Interval	datetime.timedelta	INTERVAL or DATE from epoch
LargeBinary	byte	BLOB or BYTEA
Numeric	decimal.Decimal	NUMERIC or DECIMAL
Unicode	unicode	UNICODE or VARCHAR
Text	str	CLOB or TEXT

Metadata

Каталог объектов Table с опциональной информацией о engine и соединении.

```
from sqlalchemy import MetaData
metadata = MetaData()
```

Создание таблицы

```
metadata.create_all(engine)
```

Метод ...create_all не пересоздает таблицы.

Объект таблицы состоит из названия, переменной метаданных и столбцов.

```
from sqlalchemy import Table, Column, Integer, Numeric, String, ForeignKey
from datetime import datetime
from sqlalchemy import DateTime

cookies = Table('cookies', metadata,
    Column('cookie_id', Integer(), primary_key=True),
    Column('cookie_name', String(50), index=True),
    Column('cookie_recipe_url', String(255)),
    Column('cookie_sku', String(55)),
    Column('quantity', Integer()),
    Column('unit_cost', Numeric(12, 2))
)

users = Table('users', metadata,
    Column('user_id', Integer(), primary_key=True),
    Column('username', String(15), nullable=False, unique=True),
    Column('email_address', String(255), nullable=False),
    Column('phone', String(20), nullable=False),
    Column('password', String(25), nullable=False),
    Column('created_on', DateTime(), default=datetime.now),
    Column('updated_on', DateTime(), default=datetime.now, onupdate=datetime.now)
)
```

Класс Column

- название столбца
  - тип данных
    - в String обязательно указывать длину
    - Numeric(11,2) означает 11
  - доп. параметры

```
primary_key=True
index=True
nullable=False
unique=True
default=datetime.now
onupdate=datetime.now
```

Ключи, ограничения и индексы

Могут быть определены в конструкторе столбца (primary_key=True) или позже в конструкторе таблицы.

```
from sqlalchemy import PrimaryKeyConstraint, UniqueConstraint, CheckConstraint


users = Table(...
            PrimaryKeyConstraint('user_id', name='user_pk'),
            UniqueConstraint('username', name='uix_username'),
            CheckConstraint('unit_cost >= 0.00', name='unit_cost_positive'),
            ...)
```

Множественные ключи перечисляются через запятую.

```
from sqlalchemy import Index


Index('ix_cookies_cookie_name', 'cookie_name')
Index('ix_test', mytable.c.cookie_sku, mytable.c.cookie_name)
```

Внешние связи

```
from sqlalchemy import ForeignKey
orders = Table('orders', metadata,
    Column('order_id', Integer(), primary_key=True),
    Column('user_id', ForeignKey('users.user_id')),
    Column('shipped', Boolean(), default=False)
)
line_items = Table('line_items', metadata,
    Column('line_items_id', Integer(), primary_key=True),
    Column('order_id', ForeignKey('orders.order_id')),
    Column('cookie_id', ForeignKey('cookies.cookie_id')),
    Column('quantity', Integer()),
    Column('extended_cost', Numeric(12, 2))
```

```
)
```

Связь для поля order_id:

```
Column('user_id', ForeignKey('users.user_id'))
#При желании - ограничение
ForeignKeyConstraint(['order_id'], ['orders.order_id'])
```

Добавление данных

```
from sqlalchemy import insert
перем = таблица.insert().values()
```

Лучше (?) вариант

```
from sqlalchemy import insert
перем = insert(таблица).values()
```

Строковое представление запроса

```
str(перем)
```

Компиляция запроса

```
перем.compile()
```

```
перем.compile().params
```

Примеры

```
with engine.connect() as connection:
    metadata = ...
    cookies = Table...
    ins = insert(cookies).values(...)
    res = connection.execute(ins)
    #res.inserted_primary_key - какой в будущем будет ключ (сейчас фактически в БД нет данных)
    connection.commit()
```

```
ins = cookies.insert()
inventory_list = [
    {
```

```
    'cookie_name': 'peanut butter',
    'cookie_recipe_url': 'http://some.aweso.me/cookie/peanut.html',
  },
  {
    'cookie_name': 'oatmeal raisin',
    'cookie_recipe_url': 'http://some.okay.me/cookie/raisin.html',
  }
]
result = connection.execute(ins, inventory_list)
```

```
from sqlalchemy import create_engine
from sqlalchemy import MetaData
from sqlalchemy import Table, Column, Integer, Numeric, String, ForeignKey

metadata = MetaData()
cookies = Table('cookies', metadata,
  Column('cookie_id', Integer(), primary_key=True),
  Column('cookie_name', String(50), index=True),
  Column('cookie_recipe_url', String(255))
)
engine = create_engine('sqlite:///:memory:')
connection = engine.connect()
metadata.create_all(engine)

from sqlalchemy import insert

ins = cookies.insert().values(
  cookie_name="chocolate chip",
  cookie_recipe_url="http://some.aweso.me/cookie/recipe.html"
  )
print(str(ins))
```

Получение данных

```
from sqlalchemy.sql import select
s = select(cookies)
rp = connection.execute(s)
```

Список столбцов

```
rp.keys()
```

Получение результата

```
results = rp.fetchall()
```

fetchall()	Все записи
first()	Возвращает одну запись если она единственная и закрывает соединение
fetchone()	Возвращает одну запись и оставляет соединение. Аккуратно!
scalar()	Возвращает одно значение если результат запроса одна строка с одним столбцом

Доступ возможен по:

first_row = results[0]	по индексу результата
first_row[1]	по номеру столбца в результате
first_row.cookie_name	по имени столбца
first_row[cookies.c.cookie_name]	через объект таблицы

Сортировка

```
s = select(cookies.c.cookie_name, cookies.c.quantity)
s = s.order_by(cookies.c.quantity)
s = s.order_by(desc(cookies.c.quantity))
```

Ограничения количества

```
s = s.limit(2)
```

Встроенные функции

```
from sqlalchemy.sql import func
s = select([func.sum(cookies.c.quantity)])
rp = connection.execute(s)
print(rp.scalar())
```

Фильтрация

```
s = select([cookies]).where(cookies.c.cookie_name == 'chocolate chip')
```

Оператор	Описание
==	Точное равенство
like('%chocolate%')	Вхождение элемента (регистрозависимый)
ilike(string)	Вхождение элемента
between(cleft, cright)	Элемент между значениями
concat(column_two)	Объединение столбцов
distinct()	Только уникальные значения столбца
in_([list])	Значения столбца в списке
is_(None)	Значение в столбце None
contains(string)	Содержит в себе строку (регистрозависимый)
endswith(string)	Заканчивается строкой (регистрозависимый)
startswith(string)	Начинается строкой (регистрозависимый)
notin_()	Отрицание
isnot()	Исключение

Внутри where можно использовать and_, or_, not_

```
from sqlalchemy import and_, or_, not_
s = select([cookies]).where(
    and_(
        cookies.c.quantity > 23,
        cookies.c.unit_cost < 0.40
    )
)
```

Join

```
columns = [orders.c.order_id, users.c.username, users.c.phone,
        cookies.c.cookie_name, line_items.c.quantity, line_items.c.extended_cost]
cookiemon_orders = select(*columns)
cookiemon_orders =
cookiemon_orders.select_from(orders.join(users).join(line_items).join(cookies)).where(users.c.username ==
```

```
'cookiemon')
result = connection.execute(cookiemon_orders).fetchall()
for row in result:
    print(row)
```

Для outerjoin: join -> outerjoin

Алиасы

```
manager = employee_table.alias('mgr')
```

Grouping:

```
columns = [users.c.username, func.count(orders.c.order_id)]
all_orders = select(columns)
all_orders = all_orders.select_from(users.outerjoin(orders))
all_orders = all_orders.group_by(users.c.username)
```

Обновление данных:

```
from sqlalchemy import update
u = update(cookies).where(cookies.c.cookie_name == "chocolate chip")
u = u.values(quantity=(cookies.c.quantity + 120))
result = connection.execute(u)
```

Удаление данных:

```
from sqlalchemy import delete
u = delete(cookies).where(cookies.c.cookie_name == "dark chocolate chip")
result = connection.execute(u)
```

Сырые запросы (raw)

```
result = connection.execute("select * from orders").fetchall()
```

Обработка исключений

AttributeError - ошибка набора данных
IntegrityError - ошибка ограничений
Стандартная обработка подходит если исполняется один независимый запрос.

```
try:
    result = connection.execute(ins)
except IntegrityError as error:
    print(error.orig.message, error.params)
```

В случае нескольких взаимозависимых запросов необходимо использовать транзакции.

```
transaction = connection.begin()
try:
    ...
    transaction.commit()
except IntegrityError as error:
    transaction.rollback()
```

Пример:

```
transaction = connection.begin()
cookies_to_ship = connection.execute(s).fetchall()
try:
    for cookie in cookies_to_ship:
        u = update(cookies).where(cookies.c.cookie_id == cookie.cookie_id)
        u = u.values(quantity = cookies.c.quantity-cookie.quantity)
        result = connection.execute(u)
    u = update(orders).where(orders.c.order_id == order_id)
    u = u.values(shipped=True)
    result = connection.execute(u)
    print("Shipped order ID: {}".format(order_id))
    transaction.commit()
except IntegrityError as error:
    transaction.rollback()
```

Revision #8
Created 23 July 2024 09:33:27 by Admin
Updated 30 July 2024 17:41:13 by Admin