sqlalchemy-migrate 学习笔记

1、什么是 sqlalchemy-migrate

sqlalchemy-migrate 是 sqlalchemy 框架的数据库 schema 移植模块，支持不同版本之间的数据库迁移，包括：

Dialect support

Operation / Dialect	*sqlite*	*postgres*	*mysql*	*oracle*
ALTER TABLE RENAME TABLE	yes	yes	yes	yes
ALTER TABLE RENAME COLUMN	yes (workaround) [5]	yes	yes	yes
ALTER TABLE ADD COLUMN	yes (workaround) [6]	yes	yes	yes
ALTER TABLE DROP COLUMN	yes (workaround) [5]	yes	yes	yes
ALTER TABLE ALTER COLUMN	yes (workaround) [5]	yes	yes	yes (with limitation
ALTER TABLE ADD CONSTRAINT	partial (workaround) [5]	yes	yes	yes
ALTER TABLE DROP CONSTRAINT	partial (workaround) [5]	yes	yes	yes
RENAME INDEX	no	yes	no	yes

2、安装 migrate

$ pip install sqlalchemy-migrate

```
(mentalgames)➜  server git:(master) ✗ pip list
You are using pip version 7.1.0, however version 7.1.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
blinker (1.4)
click (5.1)
decorator (4.0.4)
dominate (2.1.12)
enum (0.4.4)
Flask (0.10.1)
flask-appconfig (0.11.0)
Flask-Bootstrap (3.3.5.6)
Flask-Login (0.2.11)
Flask-Mail (0.9.1)
Flask-SQLAlchemy (2.0)
Flask-WTF (0.12)
itsdangerous (0.24)
Jinja2 (2.8)
MarkupSafe (0.23)
pbr (1.8.0)
Pillow (2.9.0)
pip (7.1.0)
psycopg2 (2.6.1)
qrcode (5.1)
requests (2.7.0)
setuptools (18.0.1)
six (1.9.0)
SQLAlchemy (1.0.8)
sqlalchemy-migrate (0.10.0)
sqlparse (0.1.16)
Tempita (0.5.2)
Werkzeug (0.10.4)
wheel (0.24.0)
WTForms (2.0.2)
```

3、基本命令

1） 为工程创建数据库仓库

$ migrate create db_repo “izyou”

#db_repo=database_repository_path　izyou=project_name

2） 将数据库加入版本控制

$ python db_repo/manage.py version_control

postgresql+psycopg2://hema:123456@localhost/izyou db_repo

#红色部分为数据库的 URL 蓝色部分为仓库 path

3） 查看数据库的当前版本

$ python db_repo/manage.py db_version

postgresql+psycopg2://hema:123456@localhost/izyou db_repo

4） 设置默认的数据库和仓库

$ migrate manage db_repo/manage.py --repository=db_repo

--url=postgresql+psycopg2://hema:123456@localhost/izyou

5） 版本升级和回退

5.1 创建一个脚本

$ python db_repo/manage.py script “test-script”

在数据库仓库的 versions 目录下会出现一个{$version}_test-script.py，其中{$version}是数据库的当前版本号+1，脚本中定义了 upgrade 和 downgrade 两个回调，用于实现数据库的升级和回退。

```
from sqlalchemy import *
from migrate import *


def upgrade(migrate_engine):
    # Upgrade operations go here. Don't create your own engine; bind
    # migrate_engine to your metadata
    pass


def downgrade(migrate_engine):
    # Operations to reverse the above upgrade go here.
    pass
~
```

5.2 编辑脚本

```
from sqlalchemy import Table, Column, Integer, String, MetaData

meta = MetaData()

account = Table(
    'account', meta,
    Column('id', Integer, primary_key=True),
    Column('login', String(40)),
    Column('passwd', String(40)),
)


def upgrade(migrate_engine):
    meta.bind = migrate_engine
    account.create()


def downgrade(migrate_engine):
    meta.bind = migrate_engine
    account.drop()
```

5.3 测试脚本

$ python db_repo/manage.py test

5.4 升级数据库

$ python db_repo/manage.py upgrade

5.5 回退数据库

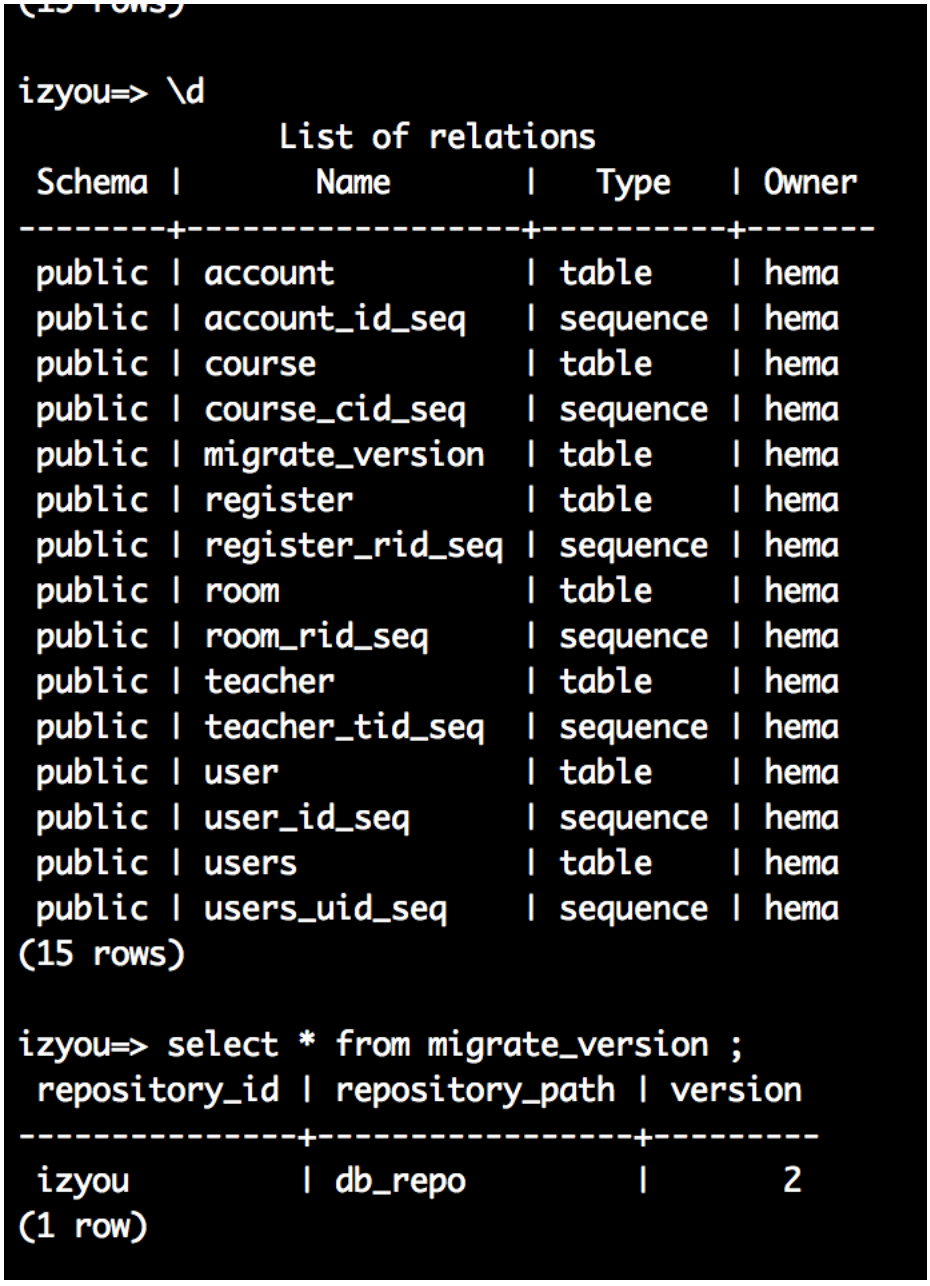
$ python db_repo/manage.py downgrade 0

红色部分为想要回退到的目标版本号

```
(mentalgames)➜  server git:(master) ✗ python db_repo/manage.py script "test-script"
(mentalgames)➜  server git:(master) ✗ vi db_repo/versions/002_test-script.py
(mentalgames)➜  server git:(master) ✗ python db_repo/manage.py test
Upgrading...
done
Downgrading...
done
Success
(mentalgames)➜  server git:(master) ✗ python db_repo/manage.py upgrade
1 -> 2...
done
(mentalgames)➜  server git:(master) ✗ python db_repo/manage.py downgrade 0
2 -> 1...
done
1 -> 0...
done
(mentalgames)➜  server git:(master) ✗ python db_repo/manage.py upgrade
0 -> 1...
done
1 -> 2...
done
(mentalgames)➜  server git:(master) ✗ python db_repo/manage.py downgrade 1
2 -> 1...
done
(mentalgames)➜  server git:(master) ✗ python db_repo/manage.py upgrade
1 -> 2...
done
```

6） 数据库的版本管理

当我们对一个数据库进行版本控制时，该数据库中会生成一张表，其中记录的是当前的版本号、仓库 id、仓库路径。



注意：仓库 id、仓库 path 必须与 migrate.cfg 中一致

```
[db_settings]
# Used to identify which repository this database is versioned under.
# You can use the name of your project.
repository_id=izyou

# The name of the database table used to track the schema version.
# This name shouldn't already be used by your project.
# If this is changed once a database is under version control, you'll need to
# change the table name in each database too.
version_table=migrate_version

# When committing a change script, Migrate will attempt to generate the
# sql for all supported databases; normally, if one of them fails - probably
# because you don't have that database installed - it is ignored and the
# commit continues, perhaps ending successfully.
# Databases in this list MUST compile successfully during a commit, or the
# entire commit will fail. List the databases your application will actually
# be using to ensure your updates to that database work properly.
# This must be a list; example: ['postgres','sqlite']
required_dbs=['postgres']

# When creating new change scripts, Migrate will stamp the new script with
# a version number. By default this is latest_version + 1. You can set this
# to 'true' to tell Migrate to use the UTC timestamp instead.
use_timestamp_numbering=False
```

7） 其它迁移工作

https://sqlalchemy-migrate.readthedocs.org/en/latest/versioning.html