数据库管理系统
DBMS

Iydon Liang, 11711217

2020 年 3 月 29 日

目录

1 介绍

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2 实验设计

2.1 实验数据和环境

原先数据集采用 `Python` 爬取 bilibili 知名或有争议的 UP 主（具体见附录 A）全部视频下的全部评论，用以做水军等检测，因此数据库课程的 Project 可以拿来直接使用。但是个人觉得数据集不够完整，因此在数据库建模部分增加用户及视频表格，并通过外键将三张表关联起来。前期下载数据量有些大，用户数量约五百万，如果将全部用户信息爬下来时间较长，因此放弃。同时期间有被永久封禁的用户，现在已经找不到其个人信息，所以数据库建模时用户信息省略，仅保存其 uid。当然为了以后研究方便，并未取消用户表格，所以数据库中共有三张表格：`User`、`Video`、`Comment`。数据库建模可视化如图 1，同时利用数据库对象关系映射[1] 进行数据库建模，但是为进行跨文件系统的查询，因此有不符合范式的地方，数据库采用 `PostgreSQL`、`MySQL` 与 `SQLite`，具体代码见附录 B。

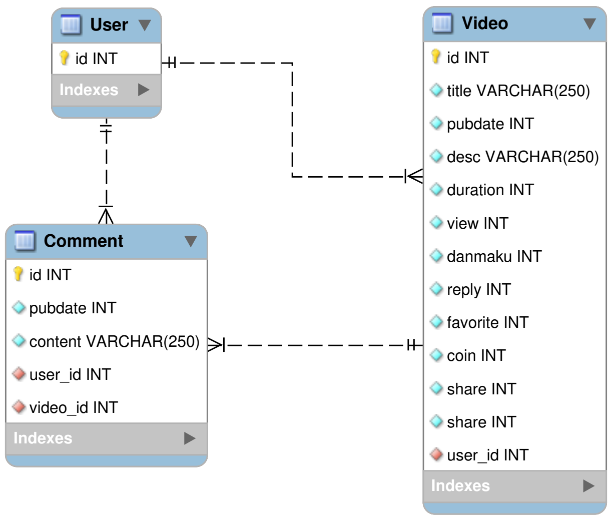

图 1: Workbench 数据库模型

所有数据分为三个文件夹，分别存储用户信息、视频信息以及评论信息，用户信息文件夹仅保存 UP 主的信息及出现在评论区全部用户的 uid，例如：

```
users/546195.json
```

```
{"name": " 老番茄", "sex": " 男", "face":
↪  "http://i2.hdslb.com/bfs/face/bc5ca101313d4db223c395d64779e76eb3482d60.jpg",
↪  "sign": " 新浪微博: _ 老番茄 _", "level": 6, "birthday": "08-13",
↪  "archive_view": 909097528, "article_view": 0, "likes": 37261387,
↪  "following": 1, "follower": 9548766}
```

视频信息文件夹保存全部视频的信息，例如：

```
videos/av84887919.json
```

```
{"pic":
↪  "http://i2.hdslb.com/bfs/archive/202bc40ecf4991d21df91f52a2d112eddb977de1.jpg",
↪  "title": " 最强自夸王!!!!! ", "pubdate": 1579924812, "desc": " 歌
↪  曲:《自夸小队》\n作词/演唱: 某幻君, 老番茄, 中国 boy 超级大猩猩, 花少
↪  北\nHOOK: 茶理理理子\n编曲: Lglywww \n混音: Blue coat\n摄影: 藤井旋风, 大
↪  饼\n后期: 藤井旋风", "duration": 231, "owner": 1577804, "view":
↪  11436611, "danmaku": 224768, "reply": 26880, "favorite": 530197,
↪  "coin": 989930, "share": 130342, "like": 956804}
```

评论信息文件夹按照视频通过嵌套字典保存保存全部评论信息，第一层的键为用户的 uid，第二册的键为时间戳，第二层的值为评论内容，后期更新代码可以保存评论的全部信息，但是本次 Project 仅考虑用户及时间戳，例如：

```
comments/av3051327.json
```

```
{"585481": {"1444662778": " 作为天天刷太平洋的刷子团, 今天在运送车队这个准备
↪  任务上栽跟头了, 来看看妹子的抢劫放松一下"}, "10917828": {"1444665830": "
↪  这期有点恶意卖萌    不过还好  反正我看了"}, "8872386": {"1444671846": "
↪  么么哒 (`  ·  ·´)"}, "342211686": {"1581531133": " 考古 [呲牙][呲牙]"},
↪  "29407696": {"1530278286": " 考古"}, "3999038": {"1444689968": " 来顶
↪  (づ    ) づ"}, "7266479": {"1444661483": "44444444 先投币点赞在说"},
↪  "495569": {"1444661357": " 第三 (`  ·  ·´)"}, "5713034": {"1444659325":
↪  " 第二 ^_^"}, "2812699": {"1444658015": " 第一 (  )"}, "6306456":
↪  {"1497997152": " ( -    - ) "}, "15514305": {"1493999905": " 考古 (=·  ·
↪  =) 感觉每天就指紫雨视频活了"}, "10954013": {"1547426274": " 🄵  🄵
↪   "}, "74234761": {"1535518362": " 考古"}}
```

数据库中的用户表格共有 5890919 行数据，视频表格共有 24742 行数据表格，评论共有 34097426 行数据。全部的程序见 GitHub，至于 bilibili 工具类例如兼容 bv 索引的爬虫见 GitHub，预计 2020 年 4 月 1 日开源。

2.2 实验细节

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.3 实验结果

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

3 结论

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

4 附加内容

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

5　参考文献

[1] AUTHORS S, Contributors. The Python SQL Toolkit and Object Relational Mapper[Z]. https://www.sqlalchemy.org/. Accessed March 11, 2020. 2020.

[2] WATT A. Chapter 3 Characteristics and Benefits of a Database[Z]. https://opentextbc.ca/dbdesign01/chapter/chapter-3-characteristics-and-benefits-of-a-database/. Accessed August 24, 2014. 2014.

[3] CASTRO K. Advantages of Database Management System[Z]. https://www.tutorialspoint.com/Advantages-of-Database-Management-System. Accessed July 25, 2018. 2018.

[4] Vianzhang. BTree 和 B+Tree 详解[Z]. https://www.cnblogs.com/vianzhang/p/7922426.html. Accessed November 29, 2017. 2017.

[5] REES G. Python’s underlying hash data structure for dictionaries[Z]. https://stackoverflow.com/questions/4279358/pythons-underlying-hash-data-structure-for-dictionaries. Accessed November 25, 2010. 2010.

[6] CENTER O H. Transaction Management[Z]. https://docs.oracle.com/cd/B19306_01/server.102/b14220/transact.htm. Accessed March 29, 2020. 2020.

A 配置文件

Pipfile

```
[[source]]
name = "pypi"
url = "https://pypi.mirrors.ustc.edu.cn/simple"
verify_ssl = true

[dev-packages]

[packages]
sqlalchemy = "*"
ipython = "*"
requests = "*"
psycopg2 = "*"
faker = "*"
fire = "*"
tqdm = "*"
retrying = "*"
pymysql = "*"

[requires]
python_version = "3.8"
```

config.py

```
#!/usr/bin/python3
import getpass


_prompt = 'Which database (1: PostgreSQL, 2: MySQL, 3: SQLite) >> '
choice = {
    '1': ('PostgreSQL', 'postgresql', 'iydon', '5432'),
    '2': ('MySQL', 'mysql+pymysql', 'root', '3306'),
}.get(input(_prompt), None)
if choice:
    database_password = getpass.getpass('({} Password) >>>
    ↪  '.format(choice[0]))
    database_path =
    ↪  f'{{}}://{{}}:{database_password}@localhost:{{}}/bilibili' \
        .format(*choice[1:])
else:
    database_path = 'sqlite:////data/bilibili.sqlite'
string_max_len = 250

user_ids = 25876945, 121225592, 10909041, 9824766, 36825256, 2375158,
↪   164139557, 437316738, 436000615, 472747194, 17819768, 1654470,
↪   10330740, 32786875, 258150656, 496085430, 378885845, 350905839,
↪   19577966, 32820037, 1791101, 486287787, 546195, 278761367, 96081167,
↪   37663924, 80304, 2381918, 17873487, 163637592, 279583114, 1577804,
↪   562197, 2206456, 318223, 481393564, 63231, 254726274, 124735327,
↪   14110780, 8047632, 2374194, 265059948, 8366990, 116683, 375375,
↪   297242063, 40433405, 333849444, 7792521, 67141499, 1754707, 888465,
↪   22500342, 125526, 360739161, 5294454, 37199377, 178135921, 5687194,
↪   39298350, 21448599, 89595, 52982448, 246370149, 438880209, 442184180,
↪   478904588, 51896064, 18690024, 27880221, 168064909, 203581440,
↪   1918296, 250681504, 585267, 290738125, 9064879, 433351, 7349, 7150454,
↪   223958603, 1532165, 467942, 30625977, 26463792, 123938419, 28266043,
↪   8261480, 16529853, 11984956, 10243458, 4766804, 225558766, 346353480,
↪   434376696, 406595597, 7788379, 52250, 513811800, 83228795, 4898834,
↪   168064909, 471300508, 510856133, 3530725, 320491072, 389988163,
↪   291939565, 883968, 483935679, 168552156, 119801456, 54992199,
↪   14583962, 479409514, 312177468

comment_path = '/data/bilibili/comments'
user_path = '/data/bilibili/users'
video_path = '/data/bilibili/videos'
```

B 数据库建模

database.py

```
#!/usr/bin/python3
__all__ = ('session', 'User', 'Video', 'Comment')


from sqlalchemy import Column, String, Integer, ForeignKey
from sqlalchemy import create_engine
from sqlalchemy.orm import relationship, sessionmaker
from sqlalchemy.ext.declarative import declarative_base

from config import database_path, string_max_len
```

```python


Base = declarative_base()


String = String(string_max_len)


class User(Base):
    __tablename__ = 'user'

    id = Column(Integer, primary_key=True)
    videos = relationship('Video', back_populates='user')
    comments = relationship('Comment', back_populates='user')

    # 前期下载数据量有些大，所以这里暂时注释掉
    # name = Column(String, nullable=True)
    # sex = Column(String, nullable=True)
    # sign = Column(String, nullable=True)
    # level = Column(Integer, nullable=True)
    # archive_view = Column(Integer, nullable=True)
    # article_view = Column(Integer, nullable=True)
    # likes = Column(Integer, nullable=True)
    # following = Column(Integer, nullable=True)
    # follower = Column(Integer, nullable=True)


class Video(Base):
    __tablename__ = 'video'

    id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey('user.id'))
    user = relationship('User', back_populates='videos')
    comments = relationship('Comment', back_populates='video')

    title = Column(String, nullable=False)
    pubdate = Column(Integer, nullable=False)
    desc = Column(String, nullable=False)
    duration = Column(Integer, nullable=False)
    view = Column(Integer, nullable=False)
    danmaku = Column(Integer, nullable=False)
    reply = Column(Integer, nullable=False)
    favorite = Column(Integer, nullable=False)
    coin = Column(Integer, nullable=False)
    share = Column(Integer, nullable=False)
    like = Column(Integer, nullable=False)


class Comment(Base):
    __tablename__ = 'comment'

    id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey('user.id'))
    user = relationship('User', back_populates='comments')
    video_id = Column(Integer, ForeignKey('video.id'))
    video = relationship('Video', back_populates='comments')

    pubdate = Column(Integer, nullable=False)
    content = Column(String, nullable=False)


engine = create_engine(database_path)
Base.metadata.create_all(engine)
DBSession = sessionmaker(bind=engine, autoflush=False)
session = DBSession()


if __name__ == '__main__':
    '''
    session.add(...)
    session.commit()
    session.close()

    print(session.query(...).all())
    print(session.query(...).filter(...).first())
    '''
    u = User(
        id=8888, name='XXXX', sex='保密', sign='hello world',
        level=5, archive_view=100, article_view=100,
        likes=100, following=100, follower=100,
    )
    v = Video(
        id=1024, user_id=8888, title='测试标题', pubdate=1496979918,
        desc='懒', duration=233, view=100, danmaku=100, reply=100,
        favorite=100, coin=100, share=100, like=100,
    )
    c = Comment(
        user_id=8888, video_id=1024, pubdate=1496979888, content='前排',
    )
```