SQLAlchemy - subquery in a WHERE clause

Ask Question

40

I've just recently started using SQLAlchemy and am still having trouble wrapping my head around some of the concepts.

Boiled down to the essential elements, I have two tables like this (this is through Flask-SQLAlchemy):

11

```
class User(db.Model):
    __tablename__ = 'users'
    user_id = db.Column(db.Integer, primary_ke

class Posts(db.Model):
    __tablename__ = 'posts'
    post_id = db.Column(db.Integer, primary_ke
    user_id = db.Column(db.Integer, db.Foreign
    post_time = db.Column(db.DateTime)

    user = db.relationship('User', backref='po
```

How would I go about querying for a list of users and their newest post (excluding users with no posts). If I was using SQL, I would do:

```
SELECT [whatever]
FROM posts AS p
    LEFT JOIN users AS u ON u.user_id = p.user
WHERE p.post_time = (SELECT MAX(post_time) FRO
```

So I know exactly the "desired" SQL to get the effect I want, but no idea how to express it "properly" in SQLAlchemy.

Edit: in case it's important, I'm on SQLAlchemy 0.6.6.

python sqlalchemy subquery

share improve this question

edited Dec 8 '12 at 0:01

kay
17.8k 9 68 110

asked Jun 1 '11 at 19:30

Chad Birch
59.4k 18 135 145

add a comment

2 Answers

active oldest votes

This should work (different SQL, same result):

42

```
t = Session.query(
    Posts.user_id,
    func.max(Posts.post_time).label('max_post_
).group_by(Posts.user_id).subquery('t')

query = Session.query(User, Posts).filter(and_
    User.user_id == Posts.user_id,
    User.user_id == t.c.user_id,
    Posts.post_time == t.c.max_post_time,
))

for user, post in query:
    print user.user_id, post.post_id
```

share
improve this answer

answered Jun 2 '11 at 7:21

sayap
4,335 1 29 35

6 What does the `c` represent in `t.c.user_id`? – Rus925 Jun 23 '14 at 8:49

6 `c` stands for 'columns' – 10flow Nov 10 '14 at 17:00

add a comment

54

the previous answer works, but also the exact sql you asked for is written much as the actual statement:

```
print s.query(Posts).\
    outerjoin(Posts.user).\
    filter(Posts.post_time==\
        s.query(
            func.max(Posts.post_time)
        ).
        filter(Posts.user_id==User.user_id).
        correlate(User).
        as_scalar()
    )
```

I guess the "concept" that isn't necessarily apparent is that as_scalar() is currently needed to establish a subquery as a "scalar" (it should probably assume that from the context against ==).

Edit: Confirmed, that's buggy behavior, completed ticket #2190. In the current tip or release 0.7.2, the as_scalar() is called automatically and the above query can be:

```
print s.query(Posts).\
    outerjoin(Posts.user).\
    filter(Posts.post_time==\
        s.query(
            func.max(Posts.post_time)
        ).
        filter(Posts.user_id==User.user_id).
        correlate(User)
    )
```