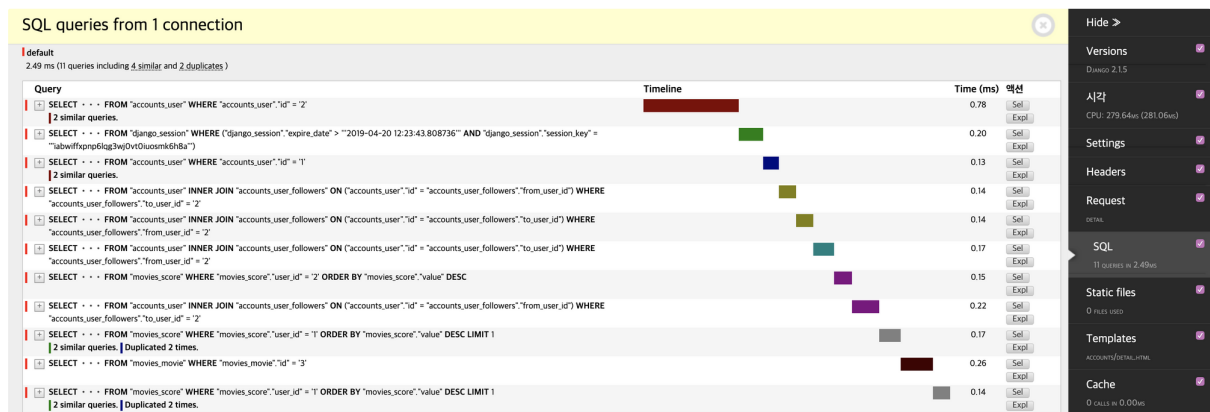




Improve your query

1. 기본 사항(11 Query, 4 similar and 2 duplicates)



```
def detail(request, user_pk): User = get_user_model() user =  
get_object_or_404(User, pk=user_pk) context = {'user_info': user}  
return render(request, 'accounts/detail.html', context) def  
follow(request, user_pk): User = get_user_model() target_user =  
get_object_or_404(User, pk=user_pk) if request.user in  
target_user.followers.all():  
target_user.followers.remove(request.user) else:  
target_user.followers.add(request.user) return  
redirect('accounts:detail', user_pk)
```

Python ▾

```
{% extends 'base.html' %} {% block body %} <p>팔로잉 : {{
user_info.followings.all.count }}</p> <p>팔로워 : {{
user_info.followers.all.count }}</p> {% if user != user_info %} {% if
user in user_info.followers.all %} <a href="{% url 'accounts:follow'
user_info.pk %}">언팔!</a> {% else %} <a href="{% url
'accounts:follow' user_info.pk %}">팔로우!</a> {% endif %} {% endif %}
<hr> <p>내가 작성한 리뷰 목록</p> {% for score in user_info.score_set.all
%} <p>영화명 : {{ score.movie.title }}</p> <p> 평점 : {{ score.value }}
</p> <hr> {% endfor %} <hr> {% for follow in
user_info.followings.all %} <p>{{ follow.username }}의 최고 평점</p>
<p>{{ follow.score_set.first.movie.title }}</p> <p>{{
follow.score_set.first.value }}</p> <hr> {% endfor %} {% endblock %}
```

HTML ▾

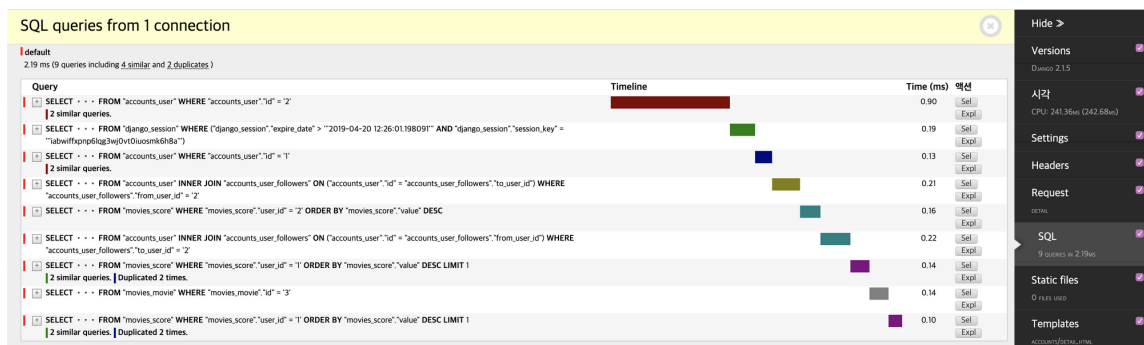
2-1. annotate

- ▶ 단순 `count`, `max` 값을 위해서라면 DB에서 계산해서 가져오자.

```
from django.db.models import Count
def detail(request, user_pk):
    User = get_user_model()
    user = get_object_or_404(
        User.objects.annotate(
            followers_count=Count('followers'),
            followings_count=Count('followings')
        ), pk=user_pk)
    context = {'user_info': user}
    return render(request, 'accounts/detail.html', context)
```

Python ▾

▼ 결과



- 여기까지는 중복을 제거하지 않고 단순히 쿼리 갯수만 날린거! 근데 이 것보다 더 큰 문제는 **반복문을 도는 상황**에서의 1:N, M:N 호출 관계에서이다.

2-2. prefetch_related

- ▼ 현재의 코드에서 중복되는 내용은 아래의 코드이다.

```
{% for follow in followings %} <p>{{ follow.username }}의 최고 평점
</p> <p>{{ follow.score_set.first.movie.title }}</p> <p>{{
follow.score_set.first.value }}</p> <hr> {% endfor %}
```

HTML ▾

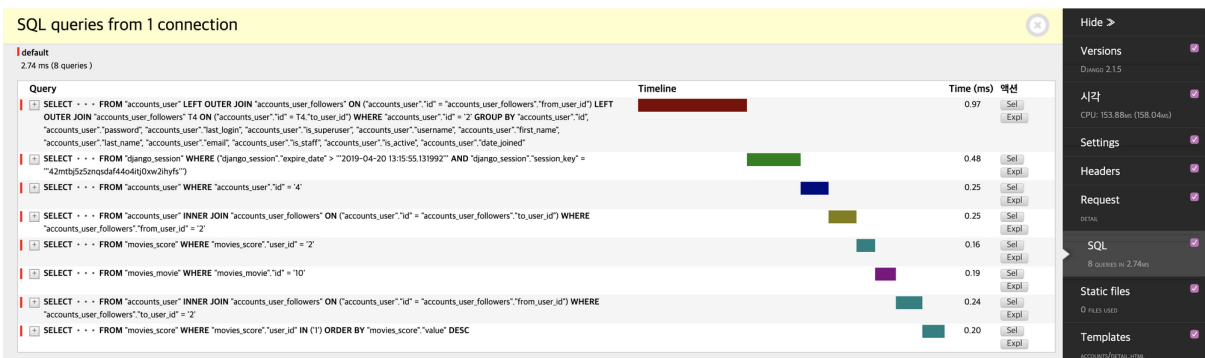
- 지금 follow 오브젝트가 정확히 동일한 쿼리인 `score_set` 을 두번 던지게 되는 것이 문제이다.
- 현재의 상황은 내가 팔로우 한 인원이 한명이어서 망정이지.. 사람이 한 명 늘어날 때마다 쿼리가 계속 추가될 것이다.

```
from django.db.models import Count, Prefetch
def detail(request, user_pk):
    User = get_user_model()
    user = get_object_or_404(User.objects.annotate(
        followers_count=Count('followers'),
        followings_count=Count('followings'),
    ), pk=user_pk)
    followings = user.followings.prefetch_related(
        Prefetch('score_set', queryset=Score.objects.order_by('-value'), to_attr='score_set_value_order'))
    context = {'user_info': user, 'followings': followings}
    return render(request, 'accounts/detail.html', context)
```

Python ▾

```
{% for follow in followings %} <p>{{ follow.username }}의 최고 평점</p>
<p>{{ follow.score_set_value_order.0.movie.title }}</p> <p>{{
follow.score_set_value_order.0.value }}</p> <hr> {% endfor %}
```

HTML ▾



짤! 중복된 내용이 없단다. 이게 얼마나 큰 이슈냐고?

아래의 코드로 shell_plus에서 100명을 추가하고 평점 남기게끔하고 내가 팔로우를 하도록 하자.

```
me = User.objects.get(pk=2) for i in range(100): u =
User.objects.create_user(f'user{i}', f'user{i}@gmail.com',
'govlgozld!') Score.objects.create( content='hi', value=3,
movie_id=1, user=u ) u.followers.add(me)
```

Python ▾

- 원래 코드대로 돌렸다면 이렇게 날아갔을 거다.
- 모든 팔로워들이 `score_set` 에 미친듯이 쿼리를 날리고 있다.

SQL queries from 1 connection			
default 32.20 ms (311 queries including 308 similar and 306 duplicates)			
Query	Timeline	Time (ms)	액션
<code>SELECT ... FROM "accounts_user" WHERE "accounts_user"."id" = 2</code> [2 similar queries.] Duplicated 2 times.		0.61	Sel Expl
<code>SELECT ... FROM "django_session" WHERE ("django_session"."expire_date" > "2019-04-20 13:30:49.012557" AND "django_session"."session_key" = "q8k5a1j3agnbed4ccolvvt8mmpbdsu3r")</code>		0.19	Sel Expl
<code>SELECT ... FROM "accounts_user" WHERE "accounts_user"."id" = 2</code> [2 similar queries.] Duplicated 2 times.		0.13	Sel Expl
<code>SELECT ... FROM "movies_score" WHERE "movies_score"."user_id" = 2</code>		0.20	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 10</code> [102 similar queries.]		0.17	Sel Expl
<code>SELECT ... FROM "accounts_user" INNER JOIN "accounts_user_followers" ON ("accounts_user"."id" = "accounts_user_followers"."from_user_id" WHERE "accounts_user_followers"."to_user_id" = 2</code>		0.21	Sel Expl
<code>SELECT ... FROM "movies_score" WHERE "movies_score"."user_id" = 1 ORDER BY "movies_score"."id" ASC LIMIT 1</code> [204 similar queries.] Duplicated 2 times.		0.21	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 2</code> [102 similar queries.]		0.11	Sel Expl
<code>SELECT ... FROM "movies_score" WHERE "movies_score"."user_id" = 1 ORDER BY "movies_score"."id" ASC LIMIT 1</code>		0.10	Sel

- 개선한 상황에서는 다음과 같다.

SQL queries from 1 connection			
default 19.05 ms (108 queries including 102 similar and 100 duplicates)			
Query	Timeline	Time (ms)	액션
<code>SELECT ... FROM "accounts_user" LEFT OUTER JOIN "accounts_user_followers" ON ("accounts_user"."id" = "accounts_user_followers"."from_user_id" LEFT OUTER JOIN "accounts_user_followers" T4 ON ("accounts_user"."id" = T4."to_user_id" WHERE "accounts_user"."id" = 2 GROUP BY "accounts_user"."id", "accounts_user"."password", "accounts_user"."last_login", "accounts_user"."is_superuser", "accounts_user"."username", "accounts_user"."first_name", "accounts_user"."last_name", "accounts_user"."email", "accounts_user"."is_staff", "accounts_user"."is_active", "accounts_user"."date_joined"</code>		6.94	Sel Expl
<code>SELECT ... FROM "django_session" WHERE ("django_session"."expire_date" > "2019-04-20 13:39:28.678911" AND "django_session"."session_key" = "q8k5a1j3agnbed4ccolvvt8mmpbdsu3r")</code>		0.21	Sel Expl
<code>SELECT ... FROM "accounts_user" WHERE "accounts_user"."id" = 2</code>		0.26	Sel Expl
<code>SELECT ... FROM "movies_score" WHERE "movies_score"."user_id" = 2</code>		0.17	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 10</code> [102 similar queries.]		0.22	Sel Expl
<code>SELECT ... FROM "accounts_user" INNER JOIN "accounts_user_followers" ON ("accounts_user"."id" = "accounts_user_followers"."from_user_id" WHERE "accounts_user_followers"."to_user_id" = 2</code>		0.22	Sel Expl
<code>SELECT ... FROM "movies_score" WHERE "movies_score"."user_id" IN (1, 4, 5, 16, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104) ORDER BY "movies_score"."value" DESC</code>		0.79	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 3</code> [102 similar queries.]		0.13	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 1</code> [102 similar queries.] Duplicated 100 times.		0.09	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 1</code> [102 similar queries.] Duplicated 100 times.		0.09	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 1</code> [102 similar queries.] Duplicated 100 times.		0.08	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 1</code> [102 similar queries.] Duplicated 100 times.		0.09	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 1</code> [102 similar queries.] Duplicated 100 times.		0.08	Sel Expl
<code>SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 1</code> [102 similar queries.]		0.09	Sel

- 반토막 났다! 근데 또 중복이라고? 해결 가능하다.

2-3. select_related

- ▼ 현재의 코드에서 중복되는 내용은 아래의 코드이다.

```
<p>{{ follow.score_set_value_order.0.movie.title }}</p>
```

Python ▾

- 각각 스코어 셋의 첫번째가 해당하는 영화를 찾는 쿼리를 미친듯이 날리기 때문.
- 이때는 `select_related('movie')` 를 통해서 평점 정보 가져올 때 영화까지 함께 들고 오도록 하자!

```
followings = user.followings.prefetch_related( Prefetch(
    'score_set',
    queryset=Score.objects.select_related('movie').order_by('-value'),
    to_attr='score_set_value_order' ))
```

Python ▾

SQL queries from 1 connection

default
8.47 ms (7 queries)

Query	Timeline	Time (ms)	액션
SELECT ... FROM "accounts_user" LEFT OUTER JOIN "accounts_user_followers" ON ("accounts_user"."id" = "accounts_user_followers"."from_user_id") LEFT OUTER JOIN "accounts_user_followers" T4 ON ("accounts_user"."id" = T4."to_user_id") WHERE "accounts_user"."id" = 2 GROUP BY "accounts_user"."id", "accounts_user"."password", "accounts_user"."last_login", "accounts_user"."is_superuser", "accounts_user"."username", "accounts_user"."first_name", "accounts_user"."last_name", "accounts_user"."email", "accounts_user"."is_staff", "accounts_user"."is_active", "accounts_user"."date_joined"		6.72	Sel Expl
SELECT ... FROM "django_session" WHERE ("django_session"."expire_date" > "2019-04-20 13:36:46.970560" AND "django_session"."session_key" = "q8k5a13agrbefccokvwt8mxpbsdu3")		0.19	Sel Expl
SELECT ... FROM "accounts_user" WHERE "accounts_user"."id" = 2		0.19	Sel Expl
SELECT ... FROM "movies_score" WHERE "movies_score"."user_id" = 2		0.16	Sel Expl
SELECT ... FROM "movies_movie" WHERE "movies_movie"."id" = 10		0.14	Sel Expl
SELECT ... FROM "accounts_user" INNER JOIN "accounts_user_followers" ON ("accounts_user"."id" = "accounts_user_followers"."from_user_id") WHERE "accounts_user_followers"."to_user_id" = 2		0.23	Sel Expl
SELECT ... FROM "movies_score" INNER JOIN "movies_movie" ON ("movies_score"."movie_id" = "movies_movie"."id") WHERE "movies_score"."user_id" IN (1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104) ORDER BY "movies_score"."value" DESC		0.85	Sel Expl

- 째@_@