Store a total separately from the log,
require they be updated together

```
class UserBalance(models.model):
    user = models.OneToOneField(User)
    total = models.DecimalField(max_digits=20, decimal_places=2)
```

A single-row lock must now be obtained on the **total**
before adding new BalanceTransfer rows for that user.

Full example using locking

```
def send_money(src, dst, amt):
    with transaction.atomic():
        # Lock balance rows, preventing other threads from making changes
        src_bal = UserBalance.objects.select_for_update().filter(id=src)
        dst_bal = UserBalance.objects.select_for_update().filter(id=dst)

        if src_bal[0].total < amt:
            raise Exception('Not enough balance to complete transaction')

        # Update the totals and add a BalanceTransfer log row together
        BalanceTransfer.objects.create(src=src, dst=dst, amt=amt)
        src_bal.update(total=F('total') - amt)
        dst_bal.update(total=F('total') + amt)
```

Side benefit: no need to scan entire BalanceTransfer table anymore to get a user's balance