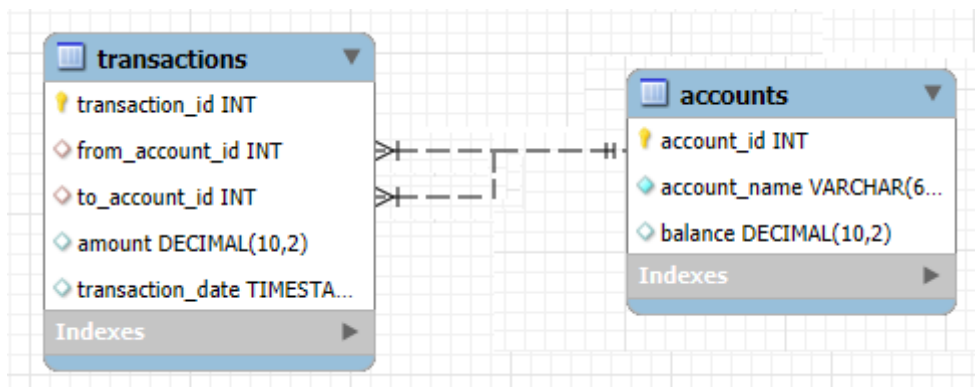


Пример использования транзакций в MySQL.

Создадим базу данных и две связанные таблицы для симуляции банковских операций:



Поле **balance** должно иметь ограничение **CHECK (balance >= 0)**

1. ROLLBACK

Вставим данные в таблицу, а затем сделаем откат (ROLLBACK), например:

```
3 • INSERT INTO accounts (account_name, balance) VALUE ('Ivanov', 555.50);
4
5 • START TRANSACTION;
6 • INSERT INTO accounts (account_name, balance) VALUE ('Petrov', 1000);
7 • SELECT * FROM accounts WHERE account_name = 'Petrov';
8 • ROLLBACK;
9 • SELECT * FROM accounts WHERE account_name = 'Petrov';
10
11
```

account_id	account_name	balance
3	Petrov	1000.00
NULL	NULL	NULL

Проверим, что изменения не применились к таблице.

2. COMMIT.

Вставим данные в таблицу и сделаем COMMIT, например:

```
7 • START TRANSACTION;
8 • INSERT INTO accounts (account_name, balance) VALUE ('Zuev', 2000);
9 • COMMIT;
10 • SELECT * FROM accounts WHERE account_name = 'Zuev';
11
```

account_id	account_name	balance
4	Zuev	2000.00
NULL	NULL	NULL

Видим, что данные сохранены в таблице.

Обратите внимание, что в случае SQL ошибки, транзакция сама по себе не откатится. Обычно ошибки обрабатываются в приложении.

3. Использование транзакций.

Для симуляции денежного перевода от одного пользователя другому будем использовать Python. На балансе пользователя с id=1 1000\$, переведем 500\$ пользователю с id=2:

```
52     try:
53         send_money( money: 500, sender: 1, recipient: 2)
54         check_money()
55     finally:
56         connection.close()

Перевод завершен успешно!
{'account_id': 1, 'account_name': 'Ivanov', 'balance': Decimal('500.00')}
{'account_id': 2, 'account_name': 'Zuev', 'balance': Decimal('1500.00')}
{'account_id': 3, 'account_name': 'Petrov', 'balance': Decimal('0.00')}
```

Повторим перевод, но только теперь 600\$, а на счету только 500\$:

```
try:
    send_money( money: 600, sender: 1, recipient: 2)
    check_money()
finally:
    connection.close()
```

```
Недостаточно средств у отправителя!
{'account_id': 1, 'account_name': 'Ivanov', 'balance': Decimal('500.00')}
{'account_id': 2, 'account_name': 'Zuev', 'balance': Decimal('1500.00')}
{'account_id': 3, 'account_name': 'Petrov', 'balance': Decimal('0.00')}
```

Перевод отменен, балансы не изменились

Проверим записи в таблице transaction:

```

1 • USE transaction;
2 • SELECT * FROM transactions;

```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:					
Export/Import:					
	transaction_id	from_account_id	to_account_id	amount	transaction_date
▶	1	1	2	500.00	2024-09-16 08:35:03
*	NULL	NULL	NULL	NULL	NULL

Триггеры.

Добавим ограничение на максимальную сумму перевода (1000\$):

```

1 • USE transaction;
2   DELIMITER //
3 • CREATE TRIGGER prevent_large_transfer
4   BEFORE UPDATE ON accounts
5   FOR EACH ROW
6   BEGIN
7       -- Проверка, что разница в балансе между новым и старым значением не больше 1000
8       IF OLD.balance - NEW.balance > 1000 THEN
9           -- Если разница больше 1000, вызываем ошибку
10          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Перевод суммы больше 1000$ запрещен!';
11      END IF;
12  END //
13  DELIMITER ;

```

BEFORE UPDATE ON accounts: Триггер срабатывает **до обновления** данных в таблице accounts, чтобы проверить сумму перевода.

FOR EACH ROW: Триггер применяется к каждой строке, которая будет обновлена.

OLD.balance - NEW.balance: Мы сравниваем старый баланс (до обновления) и новый баланс (после обновления). Если разница больше \$1000 (т.е. сумма перевода больше \$1000), триггер блокирует операцию.

SIGNAL SQLSTATE '45000': Генерируем ошибку с сообщением "Перевод суммы больше 1000\$ запрещен!", если условие выполняется.

DELIMITER //: Временно меняем разделитель команд с ; на //, чтобы внутри триггера можно было использовать стандартный разделитель ; для SQL-запросов.

BEGIN ... END: Блок кода триггера, где выполняются проверки.

SIGNAL: вызывается ошибка, если сумма перевода больше 1000.

45000 — это общее значение для произвольных ошибок, определённых пользователем.

Проверим, что триггер создан:

```
16 • SELECT TRIGGER_NAME, EVENT_MANIPULATION, EVENT_OBJECT_TABLE, ACTION_TIMING
17 FROM INFORMATION_SCHEMA.TRIGGERS
18 WHERE TRIGGER_SCHEMA = 'transaction';
19
20
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
TRIGGER_NAME	EVENT_MANIPULATION	EVENT_OBJECT_TABLE	ACTION_TIMING
prevent_large_transfer	UPDATE	accounts	BEFORE

Этот запрос вернёт список всех триггеров, созданных в базе данных **transaction**, с указанием их имени, события, для которого они настроены, таблицы и времени срабатывания.

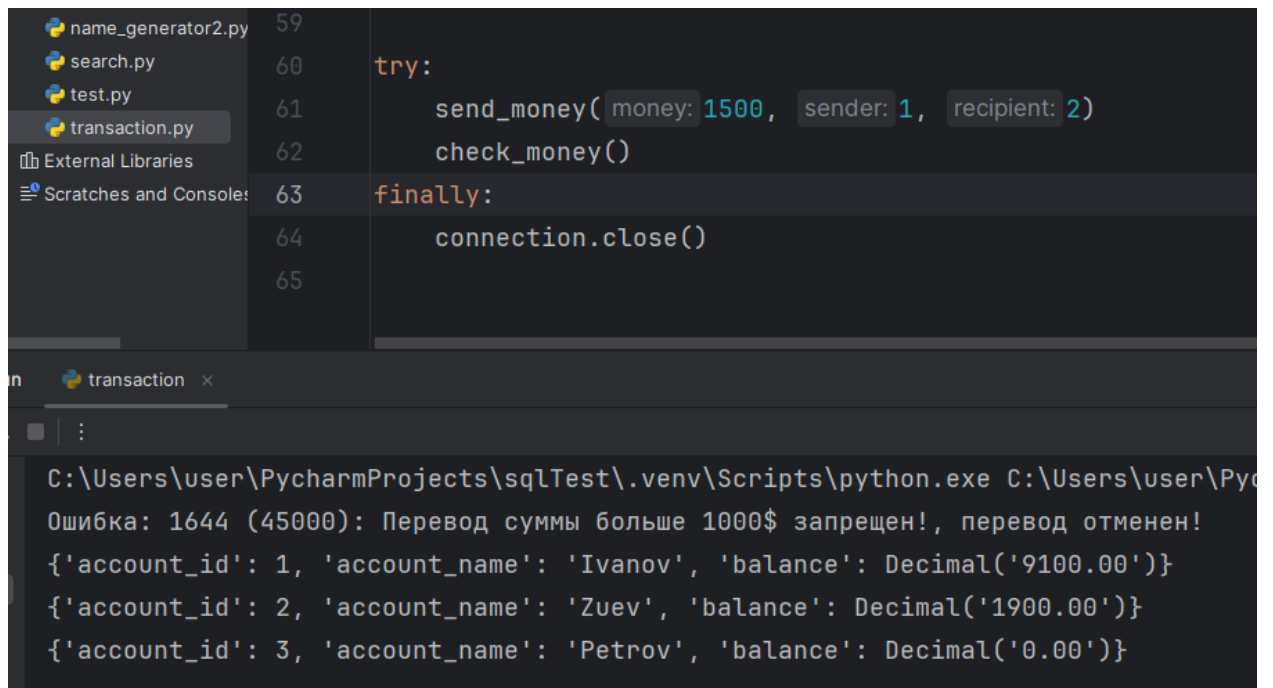
Проверим, что триггер работает. Перевод 900\$ - всё ОК:

```
search.py 60 try:
test.py 61     send_money( money: 900, sender: 1, recipient: 2)
transaction.py 62     check_money()
> External Libraries 63 finally:
Scratches and Console: 64     connection.close()
65
```

Run transaction x

```
C:\Users\user\PycharmProjects\sqlTest\.venv\Scripts\python.exe C:\Users\user\
Перевод завершен успешно!
{'account_id': 1, 'account_name': 'Ivanov', 'balance': Decimal('9100.00')}
{'account_id': 2, 'account_name': 'Zuev', 'balance': Decimal('1900.00')}
{'account_id': 3, 'account_name': 'Petrov', 'balance': Decimal('0.00')}
```

Перевод 1500\$ - ошибка:



The screenshot shows a PyCharm IDE with a file explorer on the left and a console at the bottom. The file explorer lists several Python files: `name_generator2.py`, `search.py`, `test.py`, and `transaction.py`. The `transaction.py` file is currently open, showing a `try-finally` block. The code attempts to send money from account 1 to account 2, but it fails due to an insufficient balance. The console output shows the error message in Russian and the current balances of the three accounts.

```
59
60
61
62
63
64
65
```

```
try:
    send_money(money: 1500, sender: 1, recipient: 2)
    check_money()
finally:
    connection.close()
```

transaction x

C:\Users\user\PycharmProjects\sqlTest\.venv\Scripts\python.exe C:\Users\user\Pyd
Ошибка: 1644 (45000): Перевод суммы больше 1000\$ запрещен!, перевод отменен!
{'account_id': 1, 'account_name': 'Ivanov', 'balance': Decimal('9100.00')}
{'account_id': 2, 'account_name': 'Zuev', 'balance': Decimal('1900.00')}
{'account_id': 3, 'account_name': 'Petrov', 'balance': Decimal('0.00')}