Например, пользователь в системе заполняет карточку:

- ФИО
- Дата рождения
- ИНН
- Телефон — отдельно код страны, города и номер
- Адрес — тоже разбит на несколько полей

В базе данных у нас есть несколько таблиц:

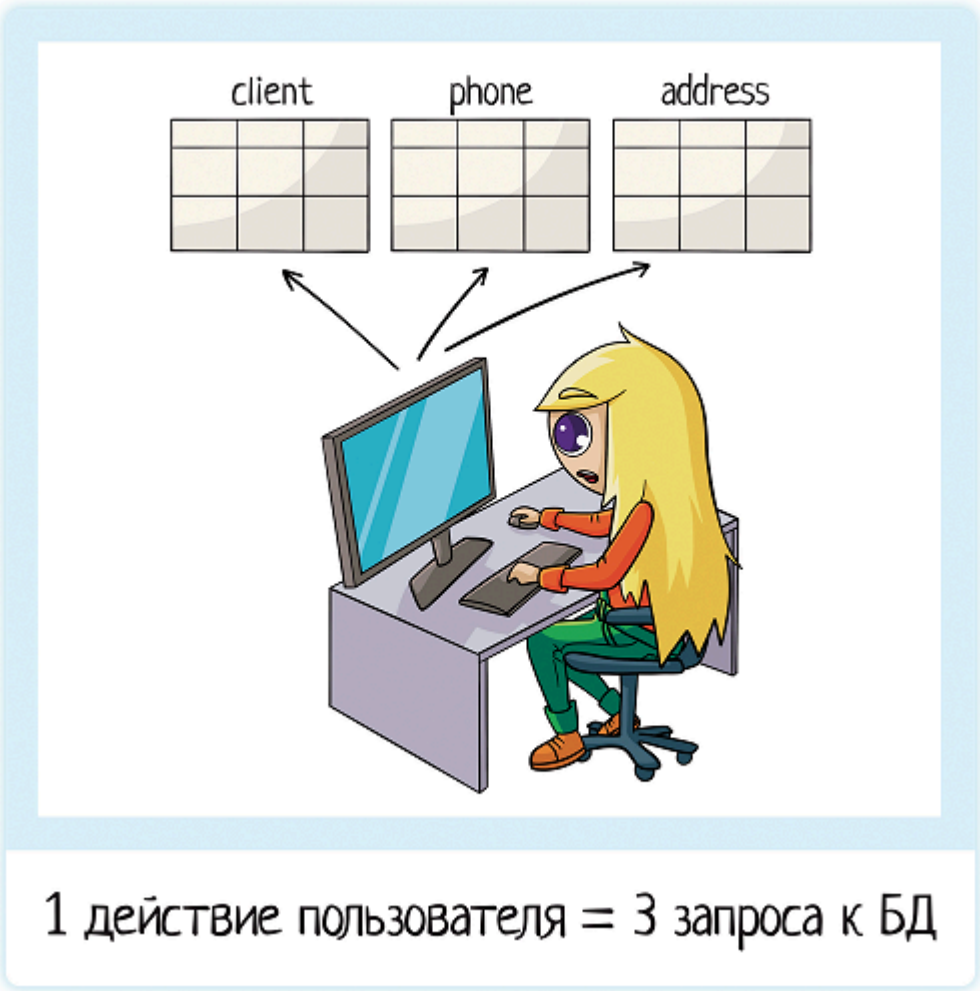
- client
- phone
- address

Так что когда пользователь заполнил форму и нажал «сохранить», система отправляет в базу данных 3 запроса:

```
insert into client… -- вставить в таблицу клиентов такие-то данные

insert into phone…

insert into address…
```



Можно отправить 3 разных запроса, но лучше сделать одну транзакцию, внутри которой будут эти 3 запроса.

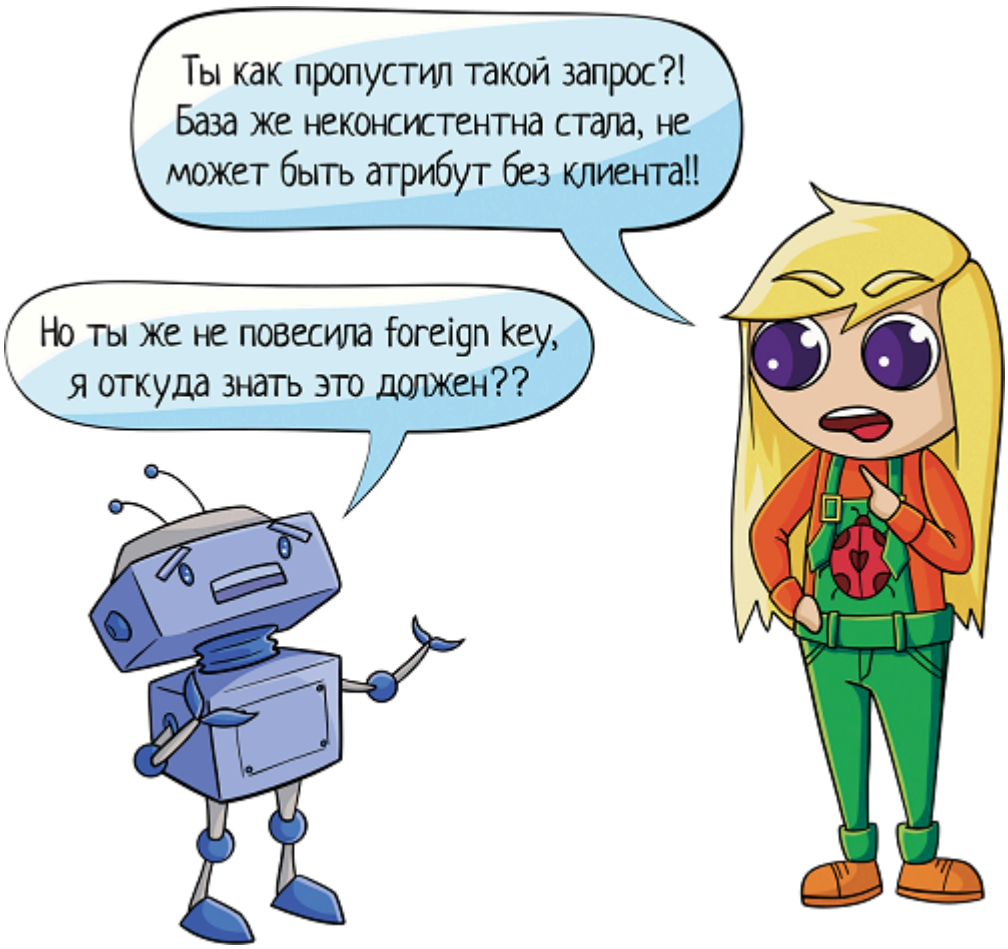
Атомарность гарантирует, что не получится такого, что адрес с телефоном сохранились, а сам клиент — нет. Это сделало бы базу неконсистентной, ведь у нас бы появились атрибуты, «висящие в воздухе», никому не принадлежащие. Что, в свою очередь, приведет к ошибкам в системе.

За консистентностью должен следить разработчик. Ведь это вопрос скорее бизнес-логики, чем технологий. Те же атрибуты, «висящие в воздухе» — это разработчик знает,

что:

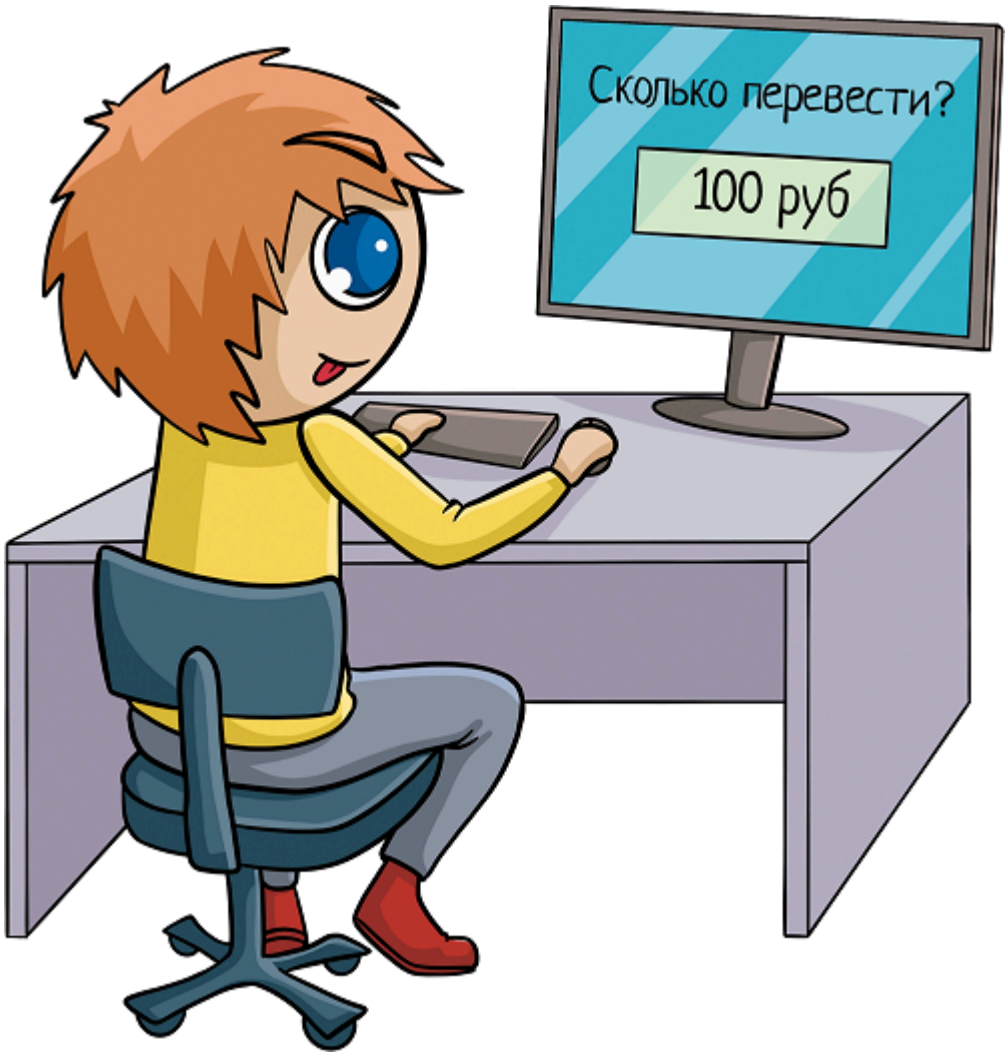
- если есть телефон в таблице phone
- он должен ссылаться на таблицу client

База об этом не знает ничего, если ей не рассказать. И она легко пропустит запрос «добавь в базу телефон без ссылки на клиента», если сам по себе запрос корректный, а разработчик не повесил на таблицу *foreign key*.

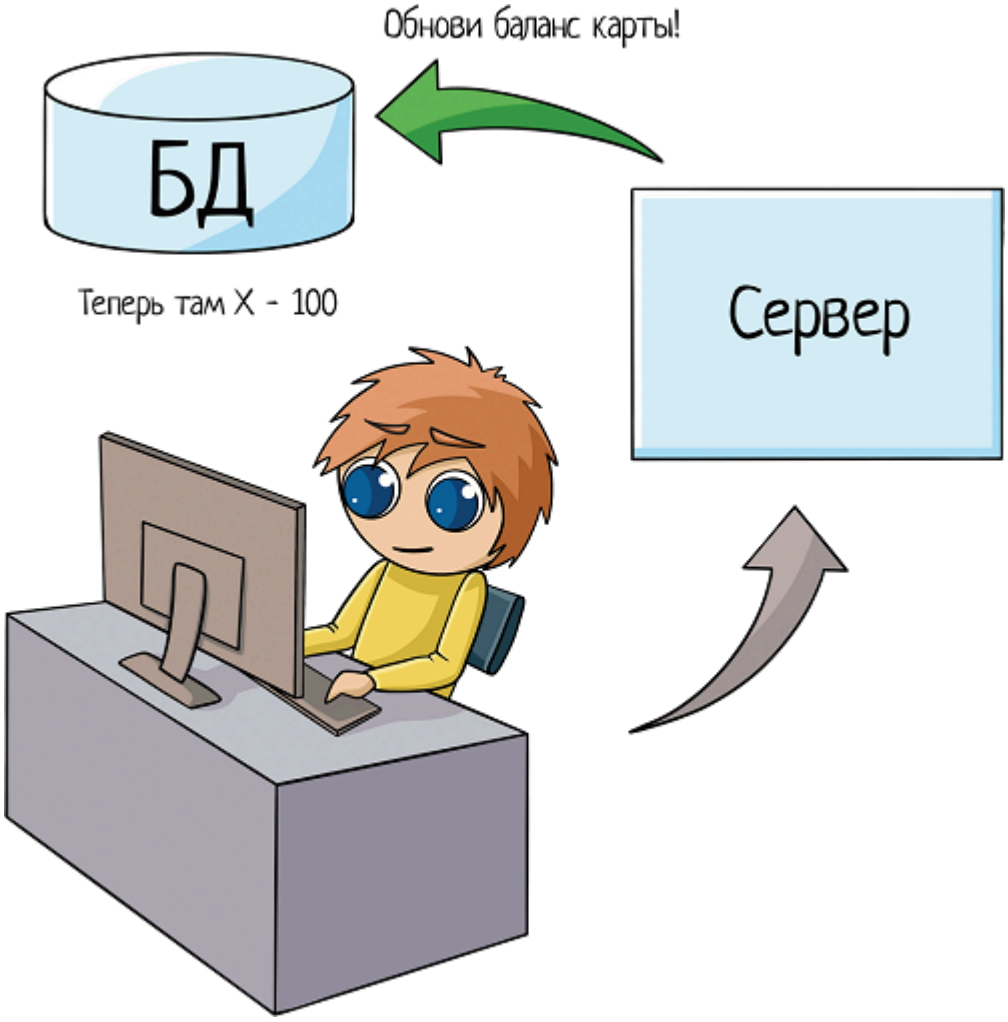


Можно повесить на таблицу *constraint*. Например, «баланс строго положительный». Тогда сценарий с ошибкой будет выглядеть так:

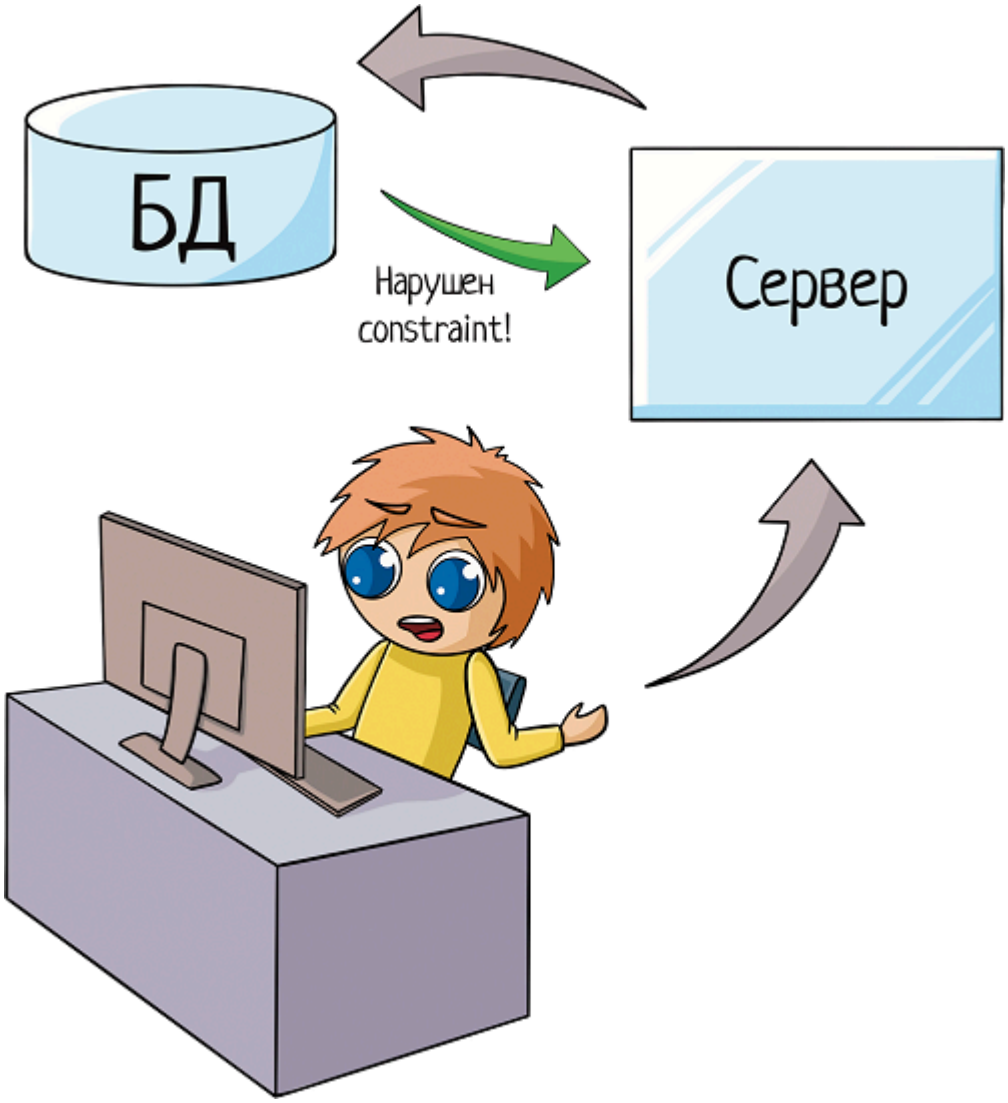
1. Пользователь пытается перевести другу 100р, хотя у него самого 10



2. Система отправляет в базу запрос — «обнови баланс карты, теперь там X – 100».



3. База пытается выполнить запрос, но ой! Нарушен constraint, в итоге операции баланс стал отрицательным, эту ошибку она и возвращает.



4. Система обрабатывает ошибку и выводит ее пользователю в читаемом виде.

К сожалению, нет единого механизма рассказать базе о том, какое состояние считается согласованным. Разработчик может использовать *foreign* ключи, какие-то констрейнты — это БД проверит. Но что с одного счета списалось, а на другой пришло — это БД уже не проверит. Это бизнес-логика.

Разработчик пишет код, пошагово переводящий БД в нужное согласованное состояние и, если где-то посередине возникает ошибка или нежданчик, откатывает всю транзакцию. То есть можно после каждого шага делать запрос, проверяя какое-то поле:

— Эй, баланс, ты ведь положительный остался?

— Ку-ку, тебе деньги пришли?

Если вдруг проверка не прошла, то кидаем ошибку и делаем откат.

Isolation — Изолированность