Существует три директории в каталоге $PGDATA, которые имеют вид «pg\_\*log».

## pg\_log

$PGDATA/pg\_log является по умолчанию местом, где хранятся журналы деятельности. Они включают в себя сообщения об ошибках, записи о запросах, и сообщения во время старта\выключения СУБД. Вы можете свободно удалять, переименовывать, сжимать и перемещать файлы из pg\_log без опаски, при условии что пользователь postgres имеет право на запись в каталог.

## pg\_xlog

$PGDATA/pg\_xlog — это место, где PostgreSQL хранит журнал транзакций. Этот набор бинарных файлов, с названиями вида '00000001000000000000008E', которые содержат образы данных последних транзакций. В отличие от pg\_log, вы не можете свободно удалять, перемещать или сжимать файлы в этом каталоге. Удаление файлов из pg\_xlog может привести к невосстановимому повреждению базы данных. Всё время в процессе работы PostgreSQL ведёт журнал предзаписи (WAL), который расположен в подкаталоге pg\_xlog/ каталога с данными кластера баз данных. В этот журнал записываются все изменения, вносимые в файлы данных. Прежде всего, журнал существует для безопасного восстановления после краха сервера: если происходит крах, целостность СУБД может быть восстановлена в результате «воспроизведения» записей, зафиксированных после последней контрольной точки.

## pg\_clog

$PGDATA/pg\_clog содержит журналы метаданных транзакций. Этот журнал говорит серверу, какие транзакции завершены, а какие нет. Если вы когда-нибудь удалите файлы из pg\_clog, вы можете смело удалить и весь каталог базы данных. Не существует способа восстановить базу данных без этих журналов.

* Используя много команд в одной транзакции, можно улучшить производительность, так как система обрабатывает сразу все команды в транзакции, а не каждую команду в отдельности.
* Каждый пользователь может определить транзакцию. Не требуется никаких разрешений ни для одного выражения транзакции.
* Внутри пользовательских транзакций не могут создаваться временные таблицы.
* Имена транзакций должны удовлетворять правилам именования объектов.
* Некоторые системные процедуры не могут быть использованы внутри транзакций, так как они создают временные таблицы.

В PostgreSQL транзакция определяется набором SQL-команд, окружённым командами BEGIN и COMMIT. Например, напишем транзакцию по добавлению ткани к заказу. Для этого нам нужно уменьшить остаток на складе и указать позицию в таблице заказов. Код транзакции будет выглядеть следующим образом (рисунок ыыы).

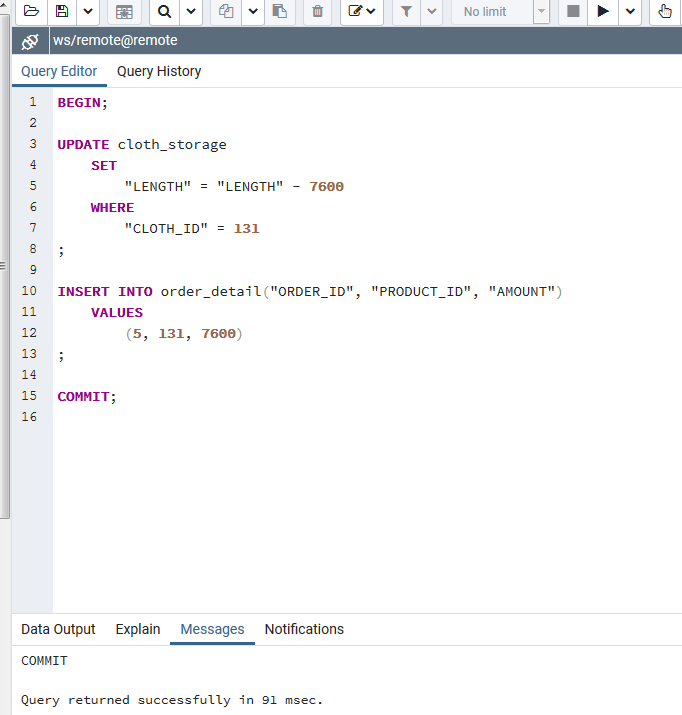


Рисунок ыыы – Код транзакции

Если транзакция должна быть прервана до своего завершения – или по причине сбоев, или из-за действий пользователя, – то все её команды должны быть отменены. Транзакции могут быть прерваны с помощью команды ROLLBACK. Эта команда должна быть выдана до команды COMMIT. Можно «откатить» всю транзакцию или её часть. Естественно, никто не может «откатить» транзакцию после её завершения. После вызова ROLLBACK транзакция завершается.

Точка сохранения представляет собой маркер, размещаемый пользователем внутри транзакции, чтобы отметить место, к которому можно «откатить» транзакцию (не считая, естественно, начала – BEGIN). Точки сохранения определяются внутри транзакции с помощью команды SAVEPOINT <имя>. Например, мы изменим транзакцию из примера выше, как показано на рисунке иаивыв.

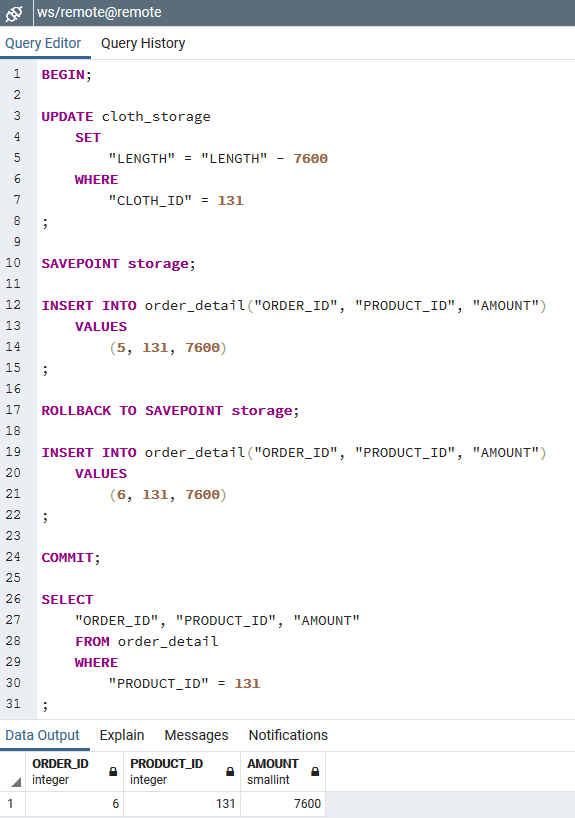


Рисунок иаивыв – Транзакция с точками сохранения

После изменения остатка на складе мы создали точку сохранения. Затем добавили ткань к заказу №5, но поняли, что это было ошибкой и, на самом деле, нужный заказ это №6. Мы откатываемся к состоянию точки сохранения и делаем новый запрос на добавление уже с правильным заказом. Вывод после коммита показывает, что всё произошло так, как было описано.

В программной реализации такую логику можно применить следующим образом:

* Имеется бизнес-процесс оформления заказа в текстильном магазине;
* Менеджер должен составить заказ, а затем оформить его;
* Когда менеджер инициирует процедуру заказа мы начинаем транзакцию;
* Добавление каждого товара сопровождается созданием точки сохранения;
* Возврат происходит при отмене последнего действия или же при редактировании позиций;
* Если произошел отказ, делаем rollback;
* Когда заказ собран, происходит commit;

Такая логика позволит при добавлении позиций отлавливать ошибки, такие как – наличие на складе, наименования, цены, а правильно выбранный режим изоляции облегчит резервирование товаров во время оформления заказа.

PostgreSQL не поддерживает вложенных транзакций, для каждого клиента необходимо вызывать их пследовательно. Транзакции поддерживают только нативный SQL, использовать PL/pgSQL не удастся (объявление переменных, условные конструкции и т.д.). Это связано с тем, что хранимые процедуры и функции по умолчанию выполняются как транзакции (либо успешное выполнение всех блоков, либо изменения не применяются), они же как раз и поддерживают данный синтаксис. Правила обработки исключительных ситуаций можно прочитать в документации PostgreSQL.