****

**Занятие 11. Динамический SQL. Представления. Пакеты**

**Практическое задание**

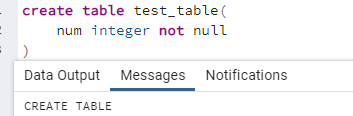
Практическое задание сдается в виде отчета, в котором для каждого задания, подразумевающего практическую работу с экземпляром Postgres, приводятся снимки экрана, на которых видны фрагменты соответствующих экранных форм, комментированные исходные коды запросов, а также результатов их выполнения на экземпляре. Также сдаётся текстовый файл \*.sql, в который включены комментированные исходные коды всех запросов по всему заданию. В случае, если задание требует использования возможности, поддержка которой в Postgres (или используемой версии Postgres) отсутствует, то необходимо отразить это в отчете и использовать альтернативный путь решения задачи.

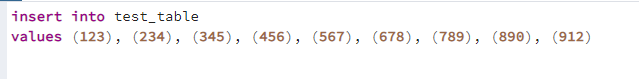
В качестве «полигона» используйте таблицы (если необходимо), созданные в рамках подготовки ДЗ с одного из предыдущих занятий.

1. Напишите процедуру, которая в зависимости от переданных ей параметров выполняет одно из действий: удаляет таблицу/удаляет все строки из таблицы/создаёт копию таблицы. Все необходимые данные передаются процедуре в качестве параметров. Процедура реализована на уровне пакета.

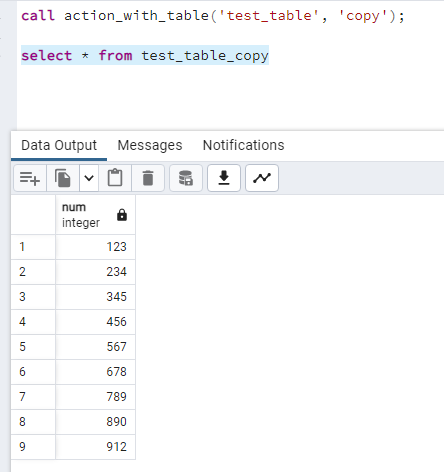


Объявляется переменная "status" с типом "boolean" (логический), которая устанавливается в результат запроса. Этот запрос проверяет, существует ли таблица с именем "test\_table" в схеме "public". Результат запроса будет булевым значением (true или false), и оно будет присвоено переменной "status".

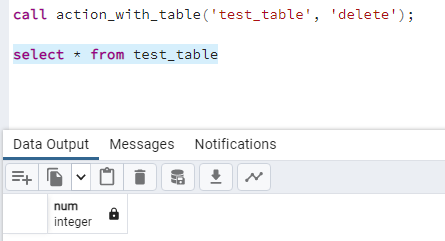




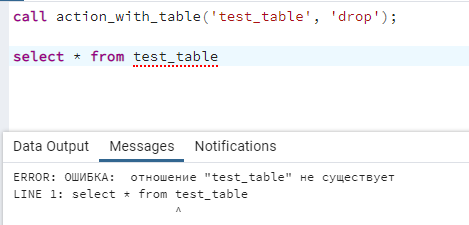
Значения test\_table  копируются в таблицу test\_table\_copy



Все значения test\_table удаляются из таблицы



Удаляем таблицу и делаем проверку(таблицы не существует)



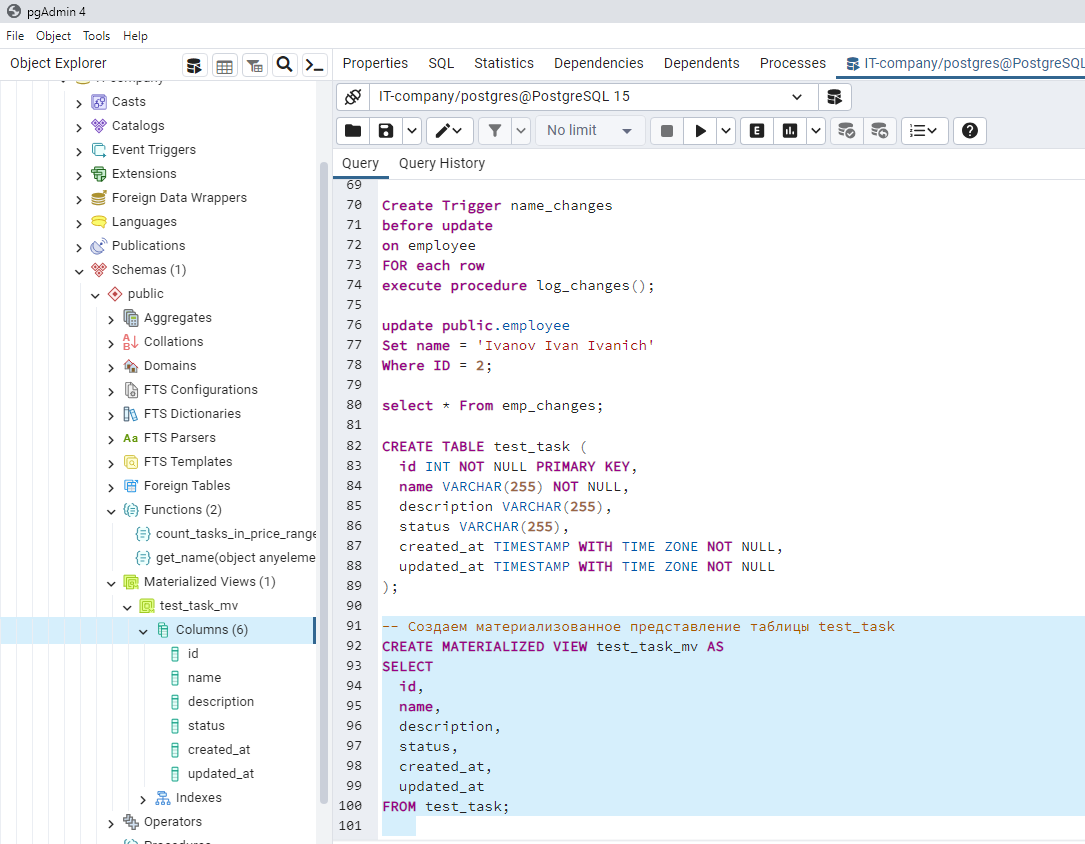
1. \* Создайте материализованное представление. При этом его «материализованность» должна быть обоснована доводами. Попробуйте в деле различные варианты настройки его актуализации.

Материализованное представление упрощает сложные данные путем сохранения результатов запросов - Вам не придется создавать новый запрос всякий раз, когда вам потребуется доступ к соответствующей информации.

Основное, что выделяет материализованное представление, это копия данных запроса, который не выполняется в реальном времени. Она занимает немного больше места, но зато и данные извлекаются очень быстро. Вы можете настроить материализованные представления на получение обновленных данных по расписанию с тем, чтобы обновленная информация не осталась незамеченной.

К сожалению, использование материализованных представлений подходит не во всех ситуациях. Во-первых, не всякая база данных поддерживает материализованные представления.

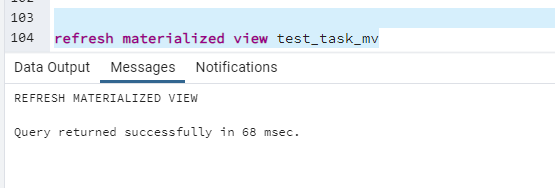
Есть и другие проблемы. Материализованные представления могут использоваться только на чтение. Это означает, что вы не можете обновлять таблицы из материализованного представления так, как вы это делаете с помощью обычного представления. Кроме того, хотя материализованные представления довольно безопасны, все же имеется некоторый риск, поскольку не все функции безопасности присутствуют. Например, вы не можете создать ключи безопасности или ограничения на материализованном представлении.



Когда вы добавляете предложение «WITH NO DATA» к материализованному представлению, оно создает пустое представление. Это материализованное представление не запрашивается. Как вы можете видеть на следующем изображении, он не получает записи во вновь созданном представлении

Из-за опции with no data  вы не можете запрашивать данные из представления. Если вы попытаетесь это сделать, вы получите следующее сообщение об ошибке

PostgreSQL полезен, чтобы дать вам подсказку, чтобы запросить загрузку данных в представление. Давайте сделаем это, выполнив следующую инструкцию:



Оператор REFRESH MATERIALIZED VIEW используется для импорта данных в материализованное представление. Заполните материализованное представление, выполнив следующий запрос REFRESH MATERIALIZED VIEW в оболочке. Как видите, этот запрос сработал эффективно.

Опять же, выберите записи материализованного представления order\_mview с помощью оператора SELECT в оболочке. На этот раз запрос SELECT работает правильно, потому что оператор REFRESH загрузил содержимое в материализованное представление.

актуальность каждый раз будет поддерживаться добавлением новых заказов

1. Немного отдохните. Это сложная и интересная тема. Вы проделали большую работу ☺ Но мы, как обычно, затронули только лишь верхушку. Курс по ООП СУБД завершается…

Если будет время, то

**\* Рекомендации что почитать дополнительно (темы, не вошедшие в курс, но про которые лучше что-то знать хотя-бы в общих чертах)**

1. Про резервное копирование и восстановление. Попробуйте поделать различные типы бекапов и восстанавливаться из них.
2. Немного более подробно про архитектуру и устройство СУБД. Про способы подключения к экземпляру. Файлы параметров, Listeners, протоколы, способы аутентификации… Это конечно более «админские» вещи, а не «программистские», но, тем не менее, лучше их немного понимать.
3. Про ретроспективные запросы. Попробуйте настроить такую возможность у себя на экземпляре и попишите запросы с конструкцией AS OF.
4. Про технологию автоматической индексации версии 19с. И вообще про индексирование поглубже.
5. Про оптимизацию запросов. Про CBO. Про хинты… Углубитесь насколько возможно.
6. Про систему трассировки, логирования, отслеживания событий. Как инструмент поиск проблемных и узким мест.
7. Про распределённые транзакции
8. Про репликацию данных.
9. Про интеграцию Java и СУБД.
10. И т.д. и т.п.