ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | Н.В. Путилова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| Триггеры. Обеспечение активной целостности данных базы данных |
| по курсу: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4236 |  |  |  | А.В. Панченко |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Цель работы:** Получение умений и навыков по проектированию и созданию триггеров баз данных, включая их использование для поддержания активной ссылочной целостности.

**Описание задания**

Реализовать для своей базы данных триггеры для всех событий (insert,delete, update) до и после.(6 триггеров) Часть из которых будет обеспечивать ссылочную целостность, остальные могут иметь другое назначение из других предложенных , но не менее 2 различных . - Вычисление/поддержание в актуальном состоянии вычислимых (производных) атрибутов - логирование (запись) изменений; - проверка корректности проводимых действий.). Вычислимые поля можно добавить при необходимости.

**Вариант 18:**

18. служба доставки: адрес доставки, контактное лицо, стоимость посылки, диапазон желаемого времени доставки. время доставки фактическое, вес посылки, отметка о доставке, фирма отправитель

а. все посылки, отправляемые в район, в названии которой есть начало «моск», но это не единственные буквы в нем

б. Улица, на которой нет адресов доставки

в. контактное лицо, получавшее посылки на улице Строителей и на проспекте Ленина

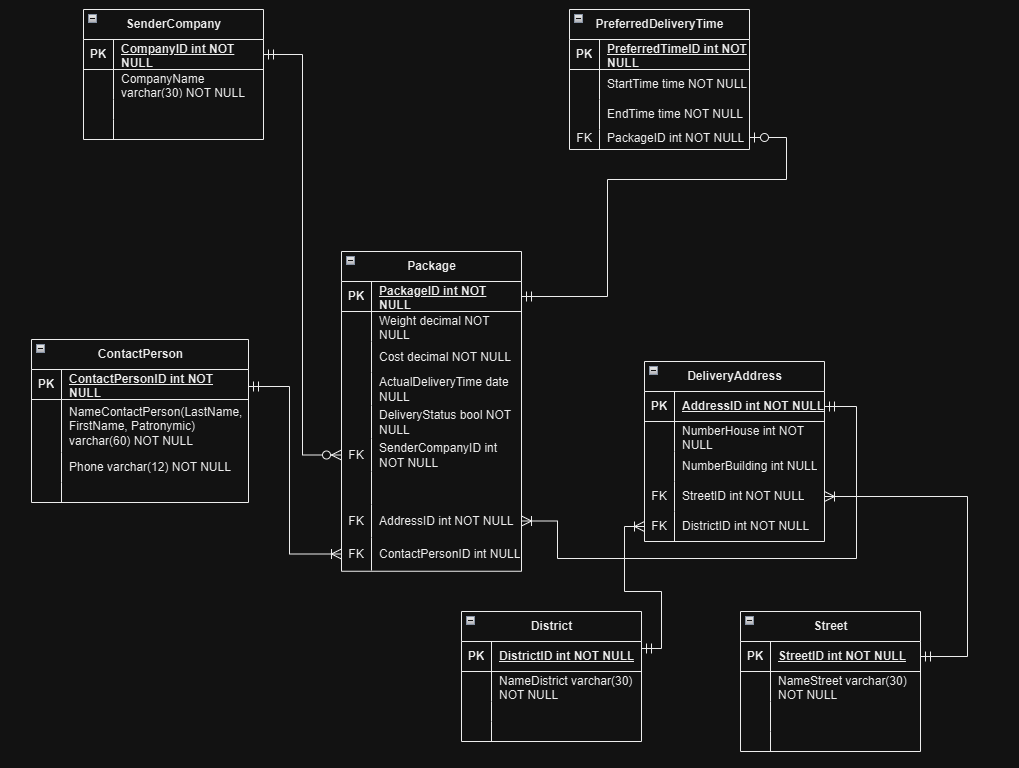
г. фирма, отправившая больше среднего посылок

д. посылка с весом меньше среднего

е. контактное лицо, получавшее посылки от всех фирм на букву Б

ж. контактное лицо, никогда не получавшее посылок в прошлом году, но получавшее посылки в феврале текущего

**Физическая модель БД:**



**Назначение разработанных триггеров текстом:**

1. **Триггер check\_cost\_before\_insert**

* **Тип**: BEFORE INSERT
* **Назначение**: Этот триггер проверяет значение поля Cost перед вставкой новой записи в таблицу Package. Если стоимость пакета отрицательная, триггер вызывает ошибку и предотвращает вставку. Это обеспечивает целостность данных и предотвращает добавление некорректных значений в базу данных.

2. **Триггер check\_cost\_before\_update**

* **Тип**: BEFORE UPDATE
* **Назначение**: Аналогично триггеру для вставки, этот триггер проверяет значение поля Cost перед обновлением существующей записи. Если новая стоимость пакета отрицательная, триггер вызывает ошибку и не позволяет обновить запись. Это также помогает поддерживать целостность данных.

3. **Триггер update\_status\_before\_insert**

* **Тип**: BEFORE INSERT
* **Назначение**: Этот триггер автоматически обновляет поле DeliveryStatus перед вставкой новой записи в таблицу Package. Если поле ActualDeliveryTime не равно NULL, статус доставки устанавливается на TRUE (доставлено); если равно NULL, статус устанавливается на FALSE (в пути). Это позволяет автоматически поддерживать актуальный статус доставки без необходимости ручного ввода.

4. **Триггер update\_status\_before\_update**

* **Тип**: BEFORE UPDATE
* **Назначение**: Этот триггер выполняет ту же функцию, что и триггер для вставки, но применяется при обновлении существующих записей. Он обновляет поле DeliveryStatus на основе значения поля ActualDeliveryTime, что позволяет поддерживать актуальность статуса доставки даже после изменения других атрибутов пакета.

5. **Триггер log\_after\_insert**

* **Тип**: AFTER INSERT
* **Назначение**: Этот триггер записывает информацию о каждой вставке новой записи в таблицу package\_log. Он фиксирует идентификатор пакета, действие (INSERT) и время изменения. Это позволяет отслеживать все изменения в базе данных и предоставляет возможность аудита действий пользователей.

6. **Триггер log\_after\_update**

* **Тип**: AFTER UPDATE
* **Назначение**: Аналогично триггеру для вставки, этот триггер фиксирует информацию о каждом обновлении существующей записи в таблице package\_log. Он записывает идентификатор пакета, действие (UPDATE) и время изменения. Это также способствует аудиту изменений и позволяет отслеживать историю изменений.

7. **Триггер log\_after\_delete**

* **Тип**: AFTER DELETE
* **Назначение**: Этот триггер фиксирует информацию о каждом удалении записи из таблицы Package. Он записывает идентификатор удаленного пакета, действие (DELETE) и время изменения в таблицу логов. Это позволяет сохранить историю удаленных записей и обеспечивает возможность восстановления информации при необходимости.

**8. Триггер before\_package\_delete**

* **Тип**: BEFORE DELETE
* **Назначение**: Этот триггер срабатывает перед удалением записи из таблицы Package. Он гарантирует, что перед удалением записи из Package будут удалены все связанные записи из таблицы PreferredDeliveryTime, которые ссылаются на удаляемую запись. Это обеспечивает целостность данных и предотвращает наличие "висячих" ссылок в таблице PreferredDeliveryTime, если пакет удаляется.

**Скрипт для создания триггеров:**

-- Функция для проверки стоимости перед вставкой

CREATE OR REPLACE FUNCTION check\_cost\_before\_insert()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.Cost < 0 THEN

RAISE EXCEPTION 'Cost cannot be negative';

END IF;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

-- Триггер для проверки стоимости перед вставкой

CREATE TRIGGER check\_cost\_before\_insert

BEFORE INSERT ON Package

FOR EACH ROW

EXECUTE FUNCTION check\_cost\_before\_insert();

-- Функция для проверки стоимости перед обновлением

CREATE OR REPLACE FUNCTION check\_cost\_before\_update()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.Cost < 0 THEN

RAISE EXCEPTION 'Cost cannot be negative';

END IF;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

-- Триггер для проверки стоимости перед обновлением

CREATE TRIGGER check\_cost\_before\_update

BEFORE UPDATE ON Package

FOR EACH ROW

EXECUTE FUNCTION check\_cost\_before\_update();

-- Функция для обновления статуса доставки

CREATE OR REPLACE FUNCTION update\_delivery\_status()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.ActualDeliveryTime IS NOT NULL THEN

NEW.DeliveryStatus := TRUE; -- Пакет доставлен

ELSE

NEW.DeliveryStatus := FALSE; -- Пакет в пути

END IF;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

OWNER TO andrejpancenko;

-- Триггер перед вставкой для обновления статуса доставки

CREATE TRIGGER update\_status\_before\_insert

BEFORE INSERT ON Package

FOR EACH ROW

EXECUTE FUNCTION update\_delivery\_status();

-- Триггер перед обновлением для обновления статуса доставки

CREATE TRIGGER update\_status\_before\_update

BEFORE UPDATE ON Package

FOR EACH ROW

EXECUTE FUNCTION update\_delivery\_status();

-- Функция для логирования изменений

CREATE OR REPLACE FUNCTION log\_package\_changes()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

INSERT INTO package\_log (PackageID, Action, ChangeTime)

VALUES (NEW.PackageID, 'INSERT', CURRENT\_TIMESTAMP);

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO package\_log (PackageID, Action, ChangeTime)

VALUES (NEW.PackageID, 'UPDATE', CURRENT\_TIMESTAMP);

ELSIF TG\_OP = 'DELETE' THEN

INSERT INTO package\_log (PackageID, Action, ChangeTime)

VALUES (OLD.PackageID, 'DELETE', CURRENT\_TIMESTAMP);

END IF;

RETURN OLD; -- Для DELETE возвращаем OLD

END;

$$

LANGUAGE plpgsql;

-- Создает таблицу package\_log

CREATE TABLE IF NOT EXISTS public.package\_log

(

logid integer NOT NULL DEFAULT nextval('package\_log\_logid\_seq'::regclass),

packageid integer NOT NULL,

action character varying(10) COLLATE pg\_catalog."default" NOT NULL,

changetime timestamp without time zone DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT package\_log\_pkey PRIMARY KEY (logid)

)

TABLESPACE pg\_default;

-- Устанавливает владельца таблицы на andrejpancenko

ALTER TABLE IF EXISTS public.package\_log

-- Триггер для логирования после вставки

CREATE TRIGGER log\_after\_insert

AFTER INSERT ON Package

FOR EACH ROW

EXECUTE FUNCTION log\_package\_changes();

-- Триггер для логирования после обновления

CREATE TRIGGER log\_after\_update

AFTER UPDATE ON Package

FOR EACH ROW

EXECUTE FUNCTION log\_package\_changes();

-- Триггер для логирования после удаления

CREATE TRIGGER log\_after\_delete

AFTER DELETE ON Package

FOR EACH ROW

EXECUTE FUNCTION log\_package\_changes();

-- Функция для удаления связанных записей в PreferredDeliveryTime перед удалением из Package

CREATE OR REPLACE FUNCTION delete\_preferred\_time\_before\_package\_delete()

RETURNS TRIGGER AS $$

BEGIN

-- Удаляет все записи из PreferredDeliveryTime, связанные с удаляемым пакетом

DELETE FROM PreferredDeliveryTime WHERE PackageID = OLD.PackageID;

RETURN OLD;

END;

$$

LANGUAGE plpgsql;

-- Триггер, вызывающий удаление из PreferredDeliveryTime перед удалением из Package

CREATE TRIGGER before\_package\_delete

BEFORE DELETE ON Package

FOR EACH ROW

EXECUTE FUNCTION delete\_preferred\_time\_before\_package\_delete();

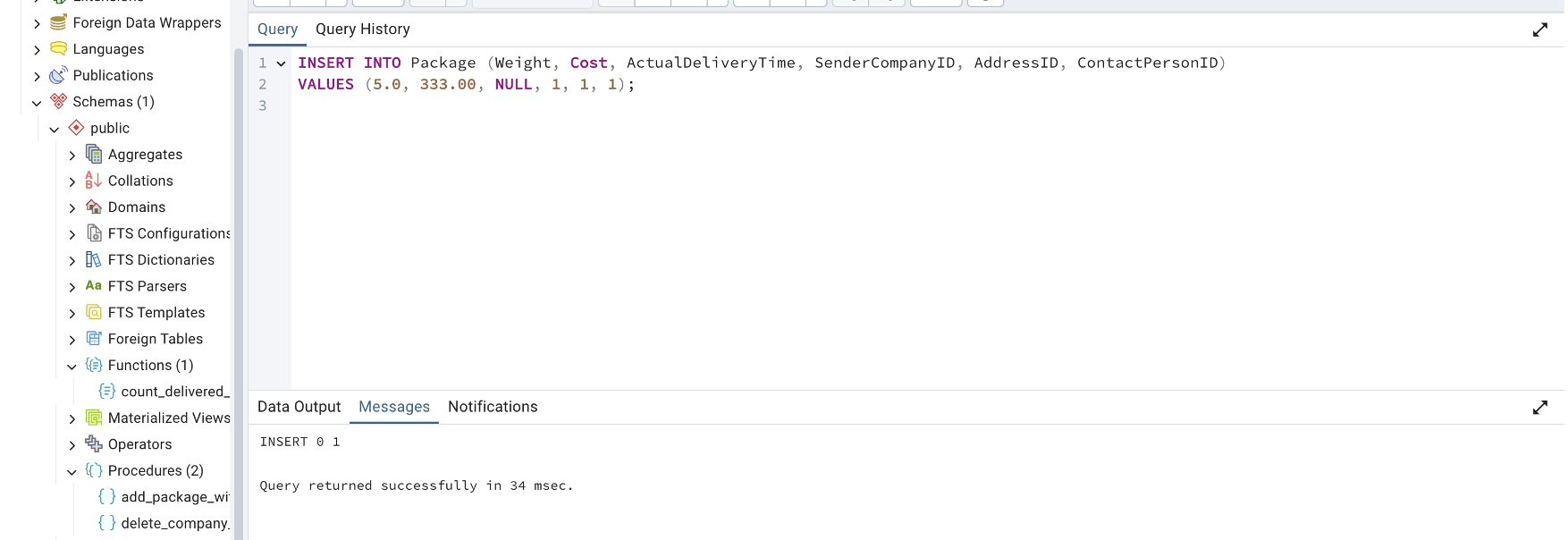
**SQL операторы и скриншоты наборов данных, иллюстрирующие работу триггеров**

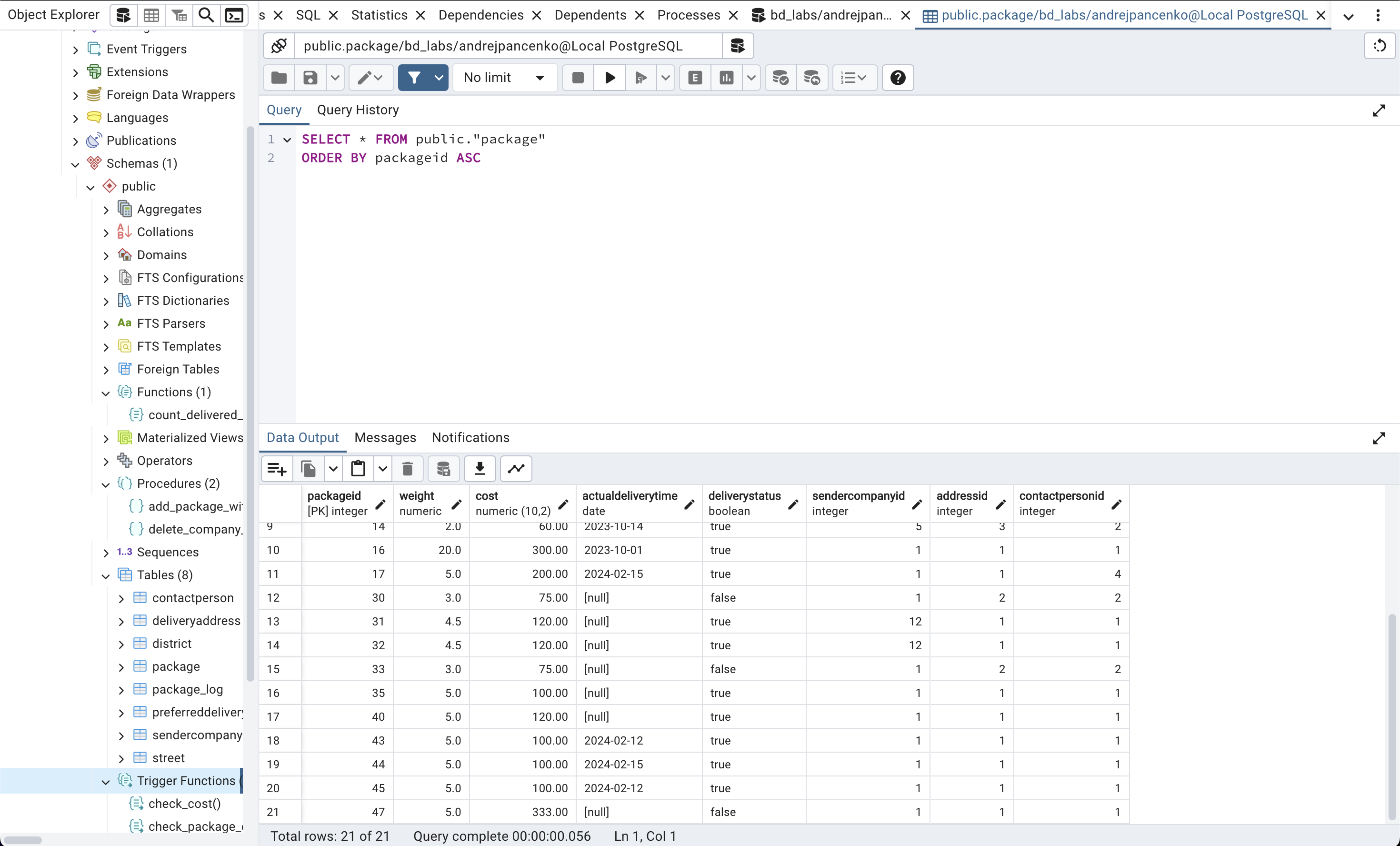
1. Триггер: Проверка стоимости перед вставкой (check\_cost\_before\_insert)

Вставка с корректной стоимостью (должен пройти успешно):

INSERT INTO Package (Weight, Cost, ActualDeliveryTime, SenderCompanyID, AddressID, ContactPersonID)

VALUES (5.0, 100.00, NULL, 1, 1, 1);





Вставка с некорректной стоимостью (должен вызвать ошибку):

INSERT INTO Package (Weight, Cost, ActualDeliveryTime, SenderCompanyID, AddressID, ContactPersonID)

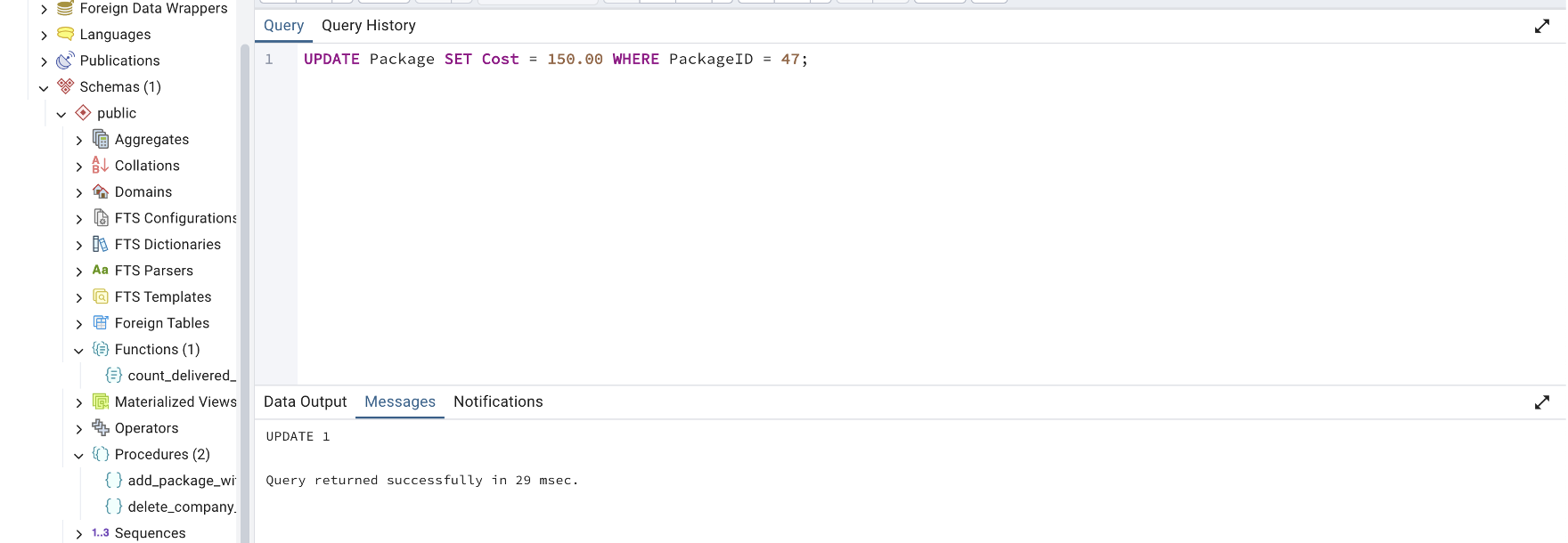
VALUES (5.0, -50.00, NULL, 1, 1, 1); -- Это вызовет ошибку из-за триггера check\_cost\_before\_insert.



2. Триггер: Проверка стоимости перед обновлением (check\_cost\_before\_update)

Обновление с корректной стоимостью (должен пройти успешно):

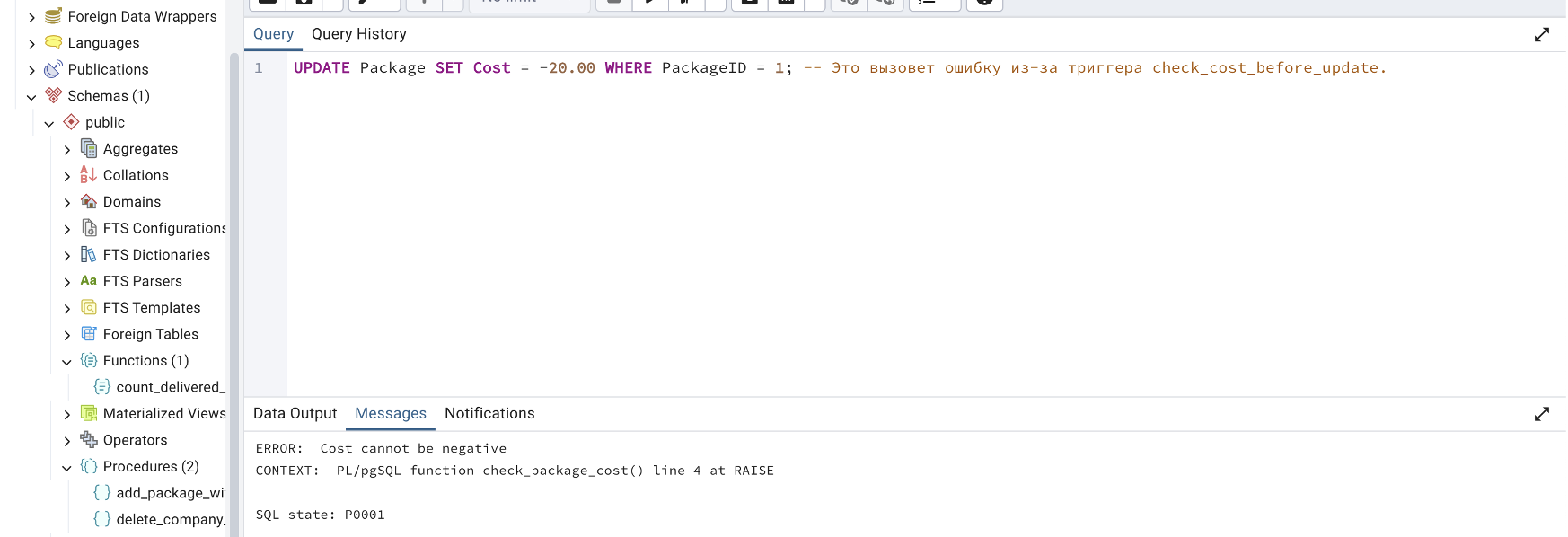
UPDATE Package SET Cost = 150.00 WHERE PackageID = 47;





Обновление с некорректной стоимостью (должен вызвать ошибку):

UPDATE Package SET Cost = -20.00 WHERE PackageID = 1; -- Это вызовет ошибку из-за триггера check\_cost\_before\_update.

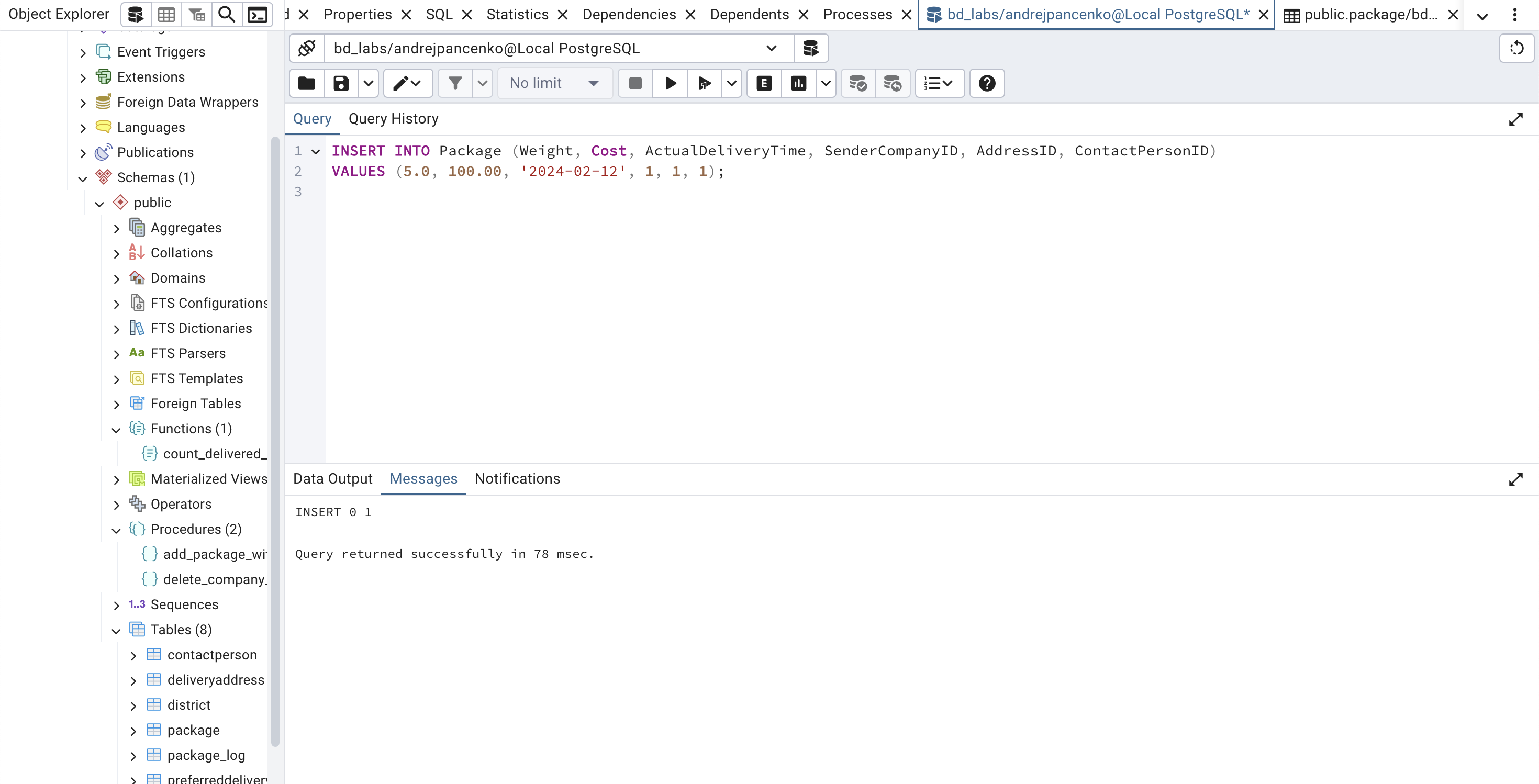


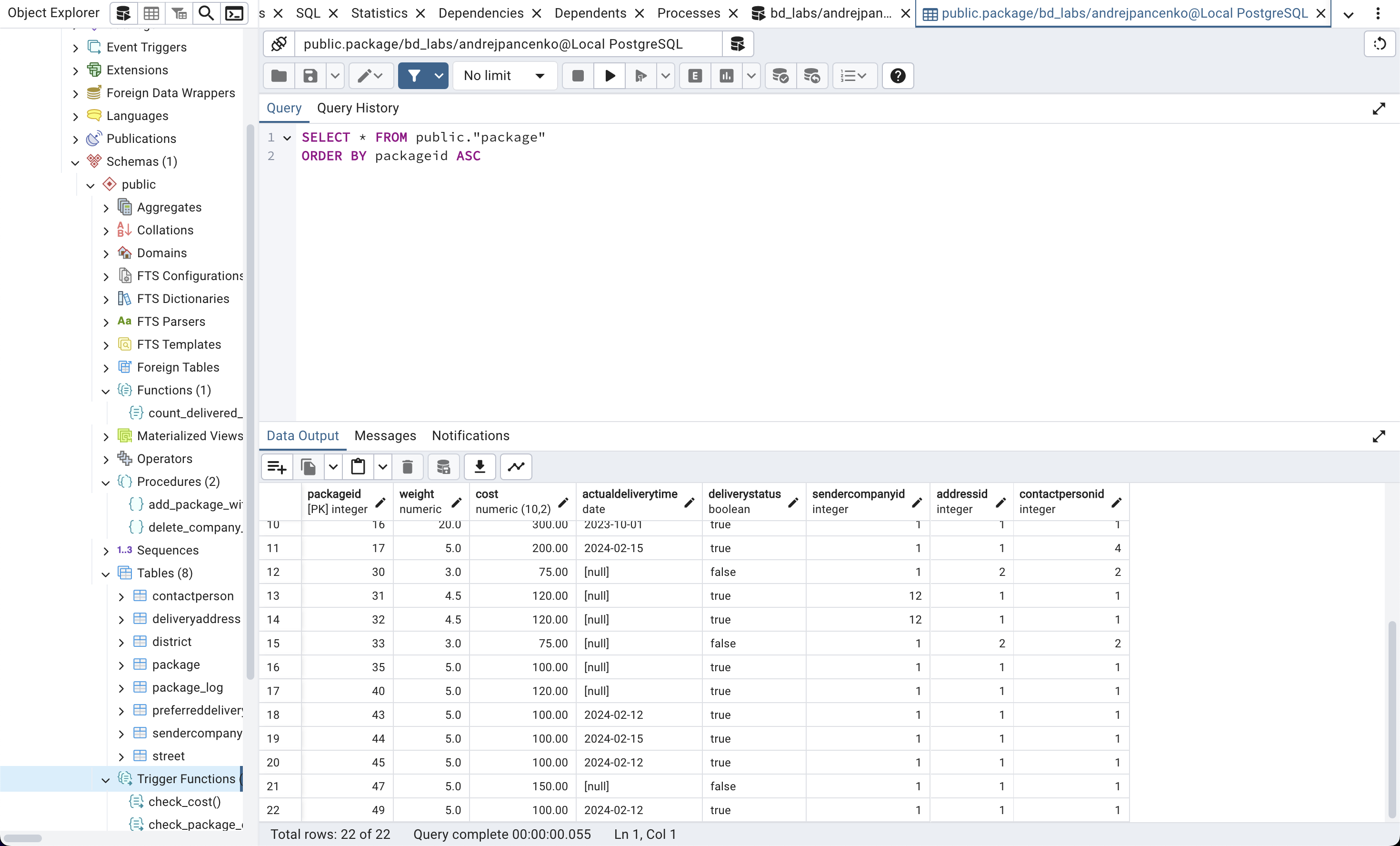
3. Триггер: Обновление статуса доставки перед вставкой (update\_status\_before\_insert)

Вставка с корректной датой доставки (статус должен быть TRUE):

INSERT INTO Package (Weight, Cost, ActualDeliveryTime, SenderCompanyID, AddressID, ContactPersonID)

VALUES (5.0, 100.00, '2024-02-12', 1, 1, 1);

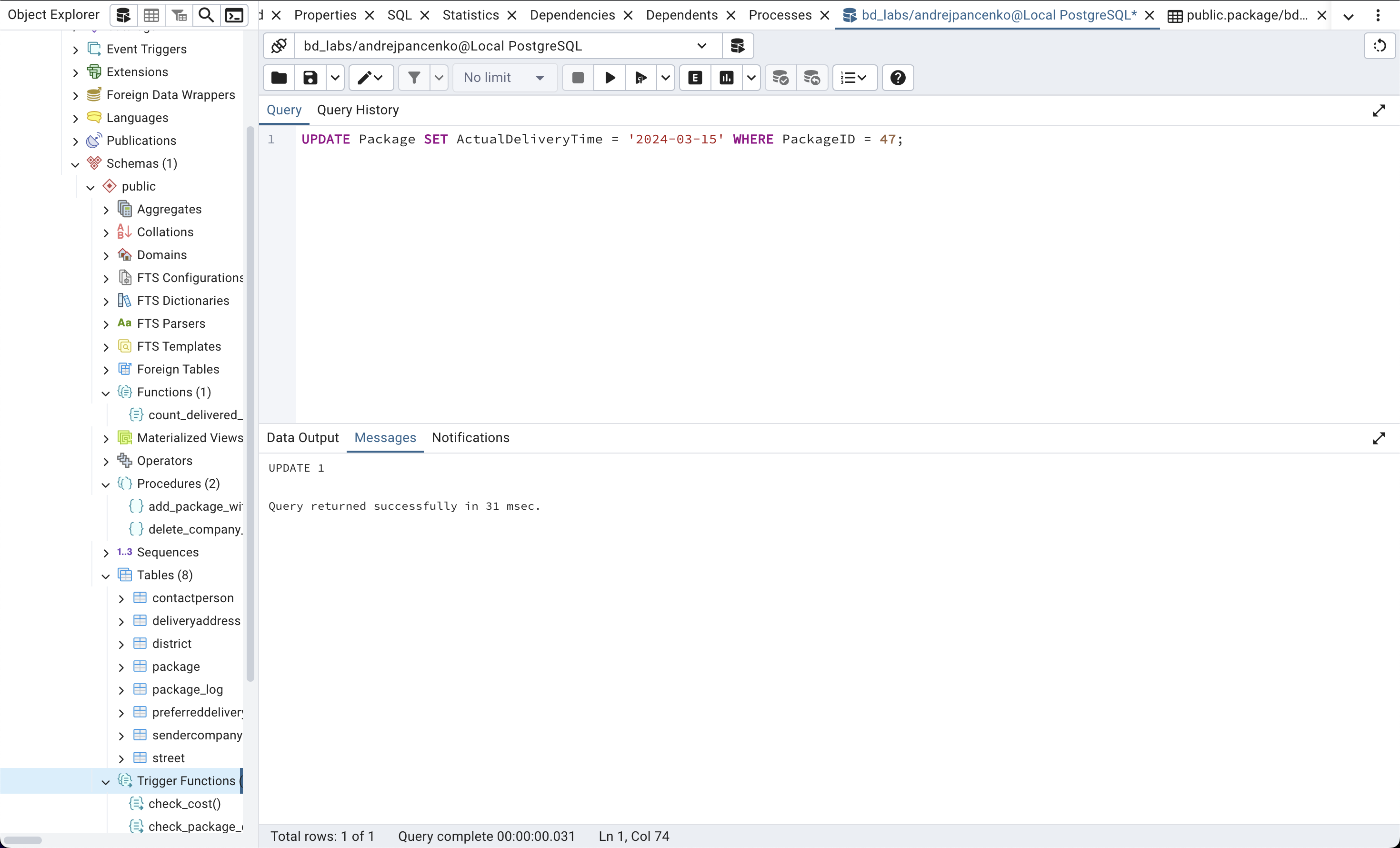


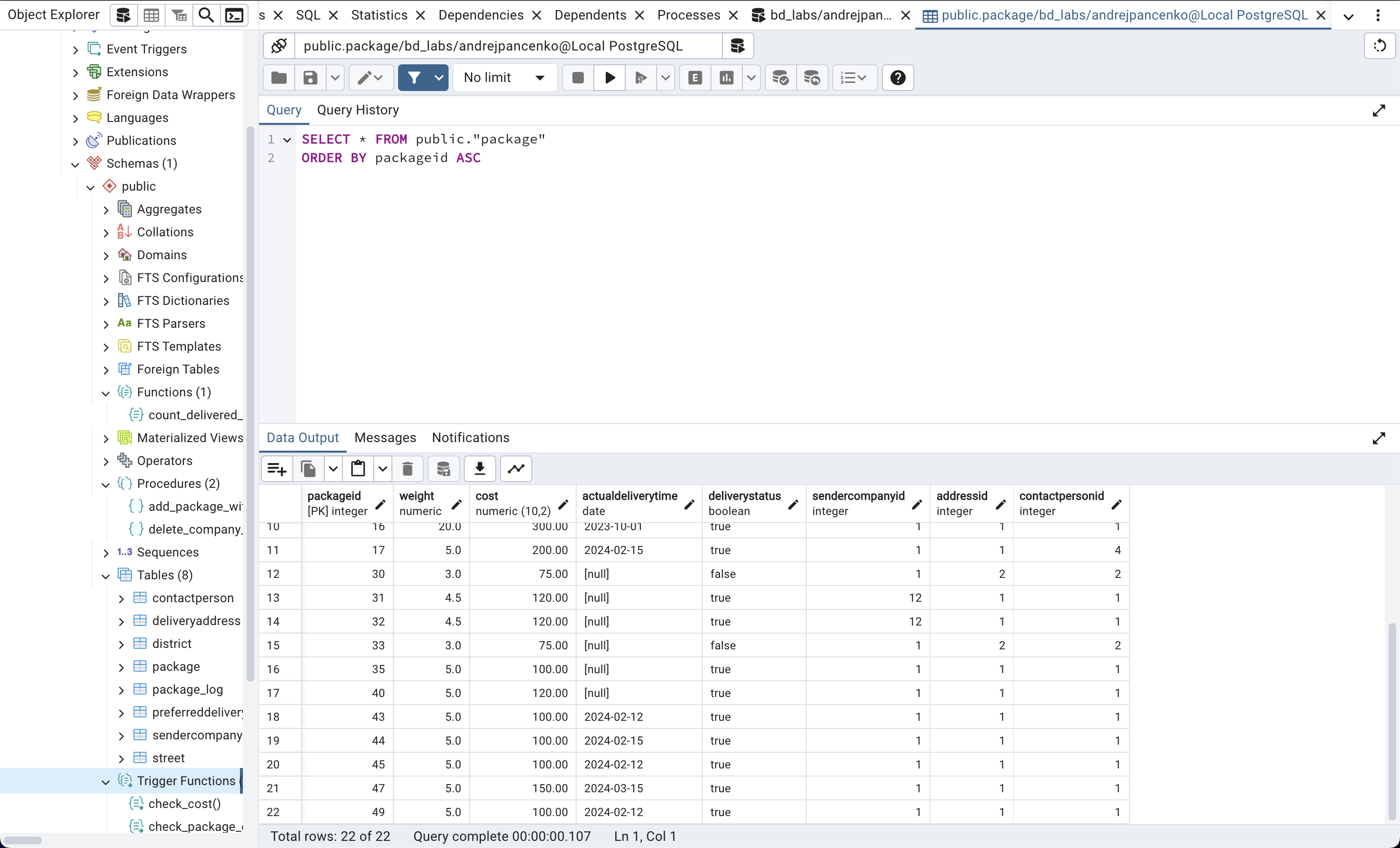


4. Триггер: Обновление статуса доставки перед обновлением (update\_status\_before\_update)

Обновление с новой датой доставки (статус должен измениться на TRUE):

UPDATE Package SET ActualDeliveryTime = '2024-03-15' WHERE PackageID = 1;



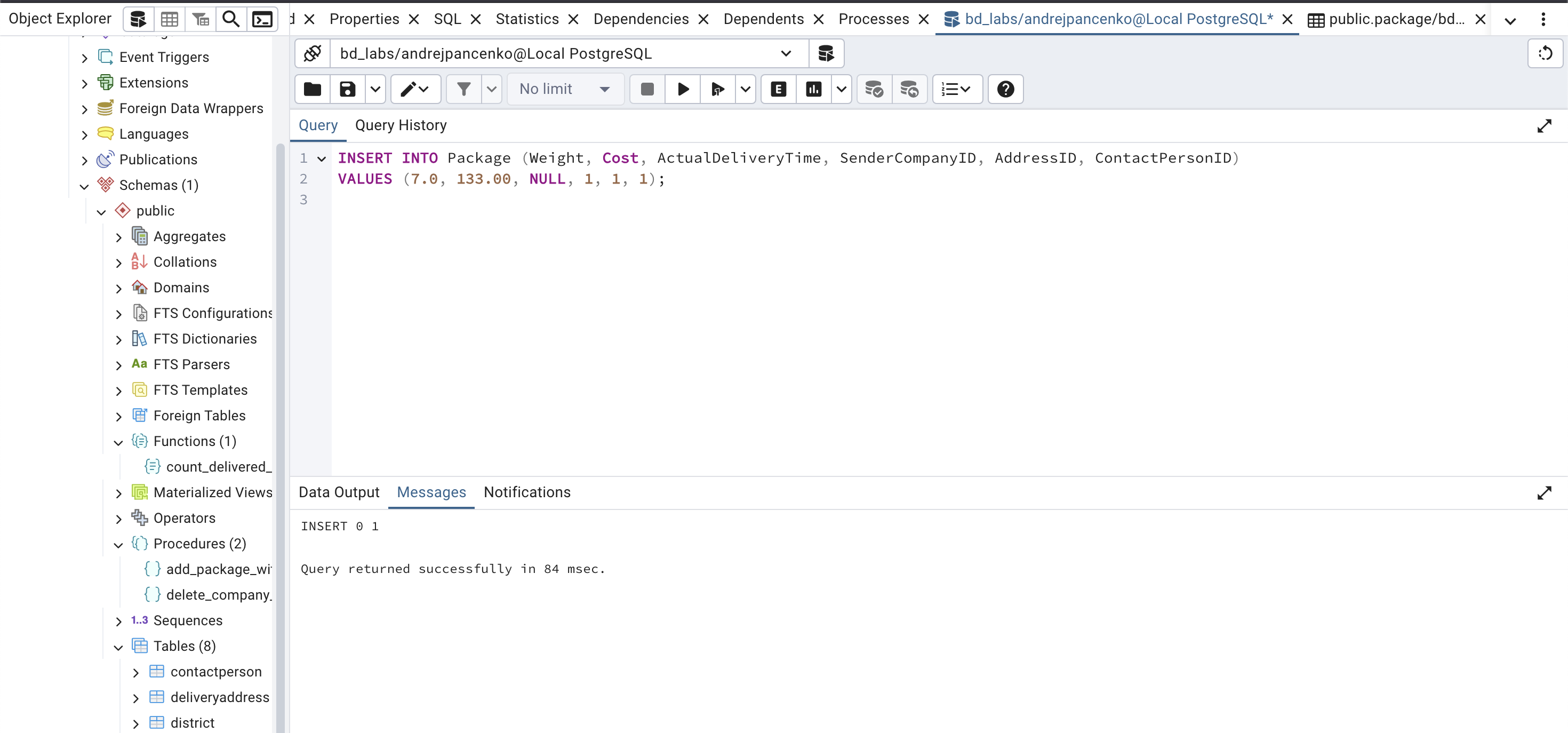


5. Триггер: Логирование изменений после вставки (log\_after\_insert)

Вставка с корректной стоимостью (должен записаться лог):

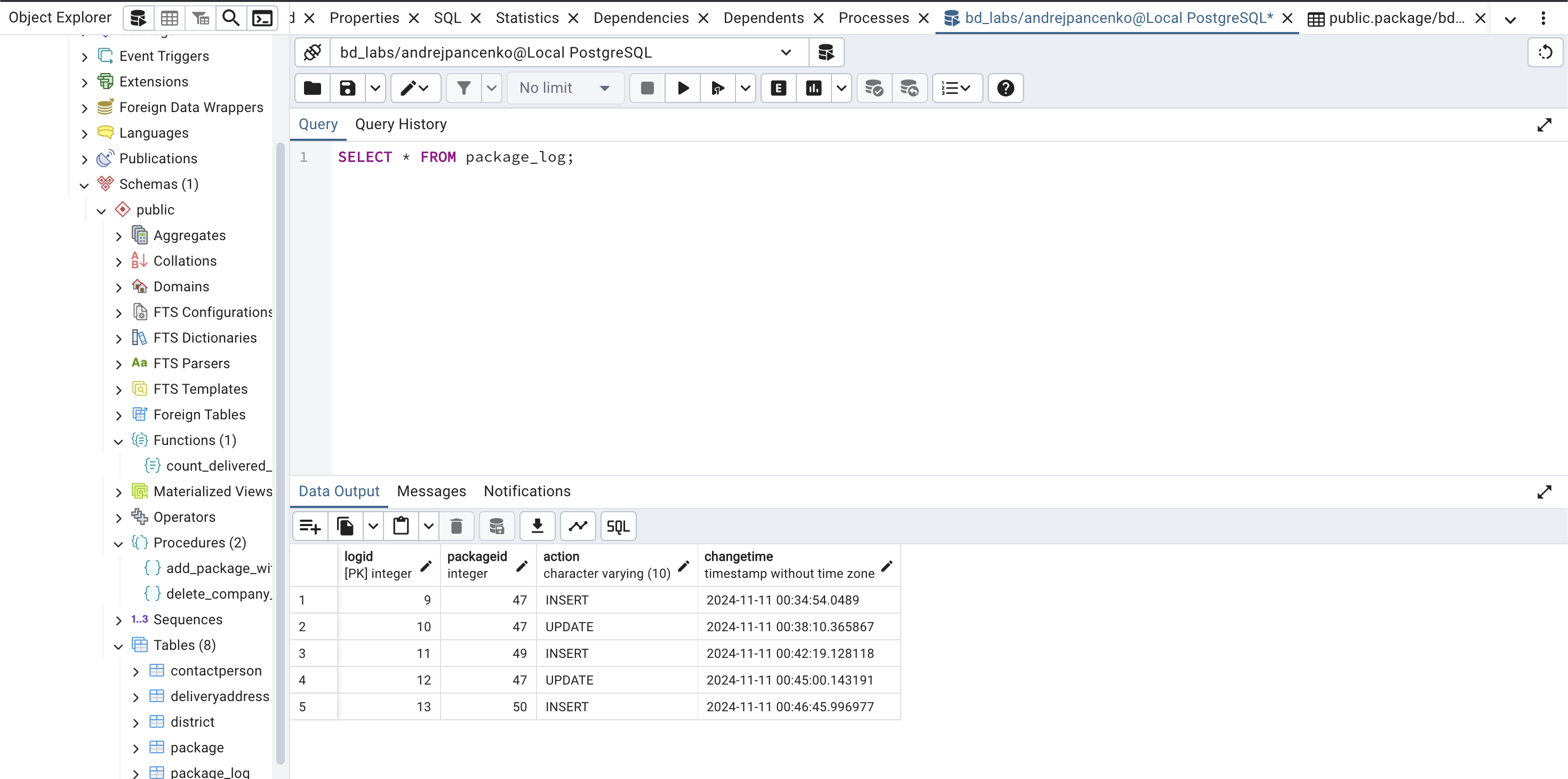
INSERT INTO Package (Weight, Cost, ActualDeliveryTime, SenderCompanyID, AddressID, ContactPersonID)

VALUES (5.0, 100.00, NULL, 1, 1, 1);



Проверка логов после вставки:

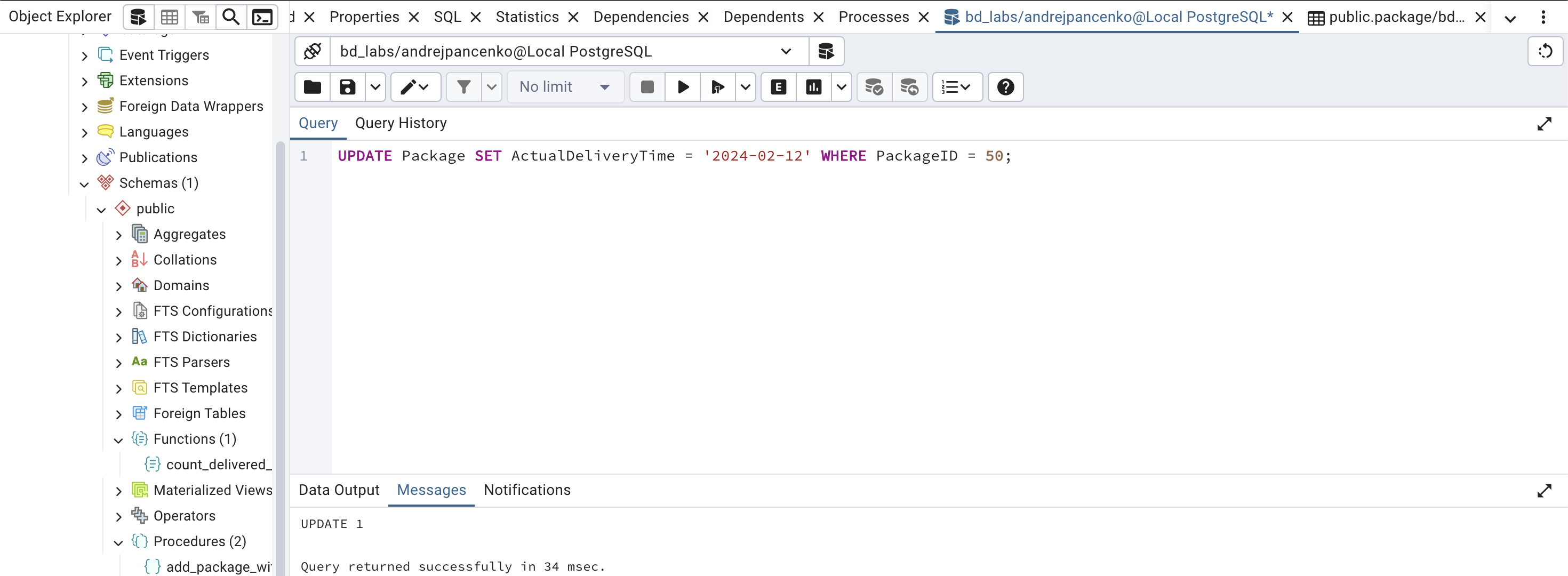
SELECT \* FROM package\_log;



6. Триггер: Логирование изменений после обновления (log\_after\_update)

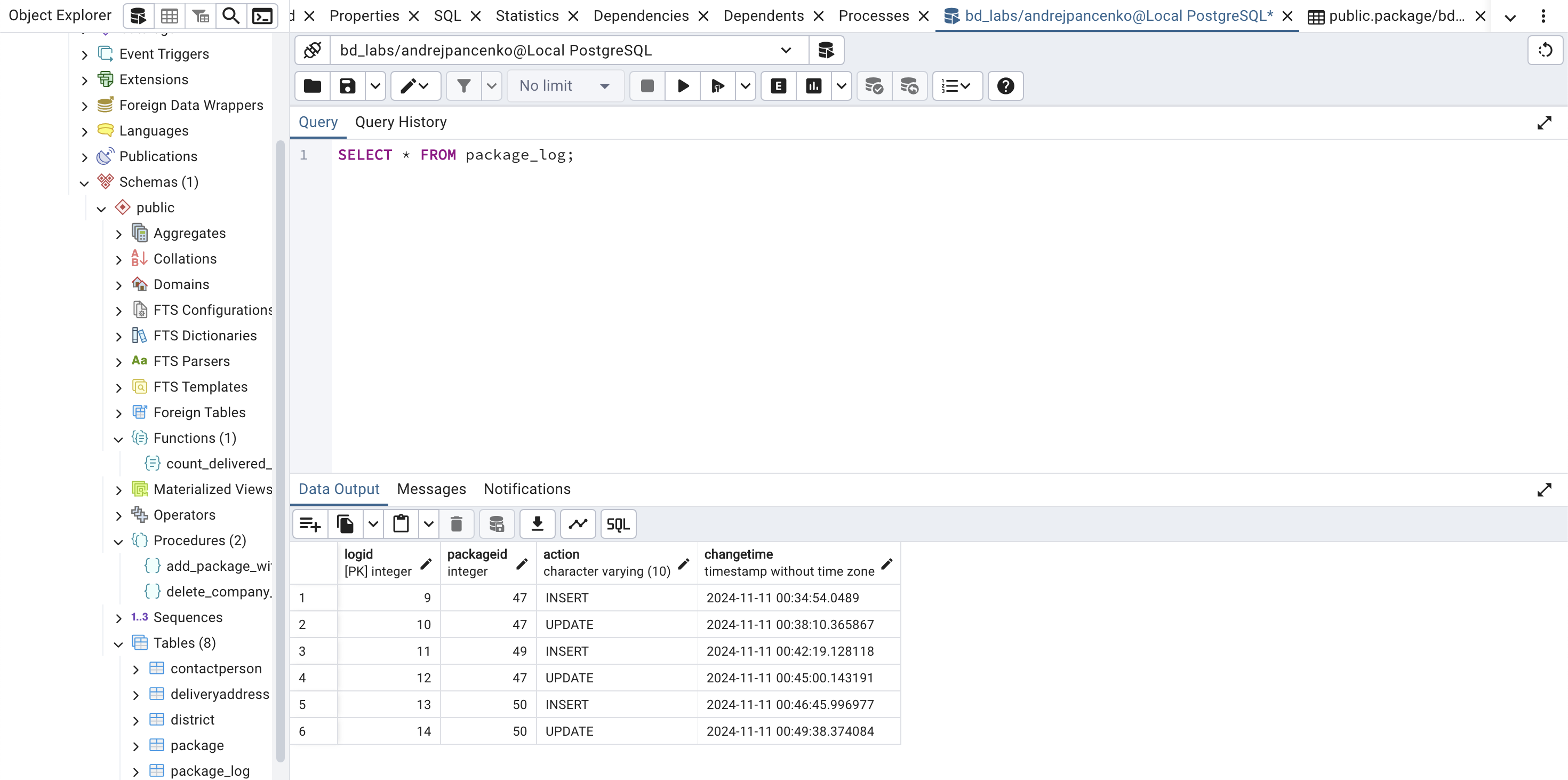
Обновление существующей записи (должен записаться лог):

UPDATE Package SET ActualDeliveryTime = '2024-02-12' WHERE PackageID = 1;



Проверка логов после обновления:

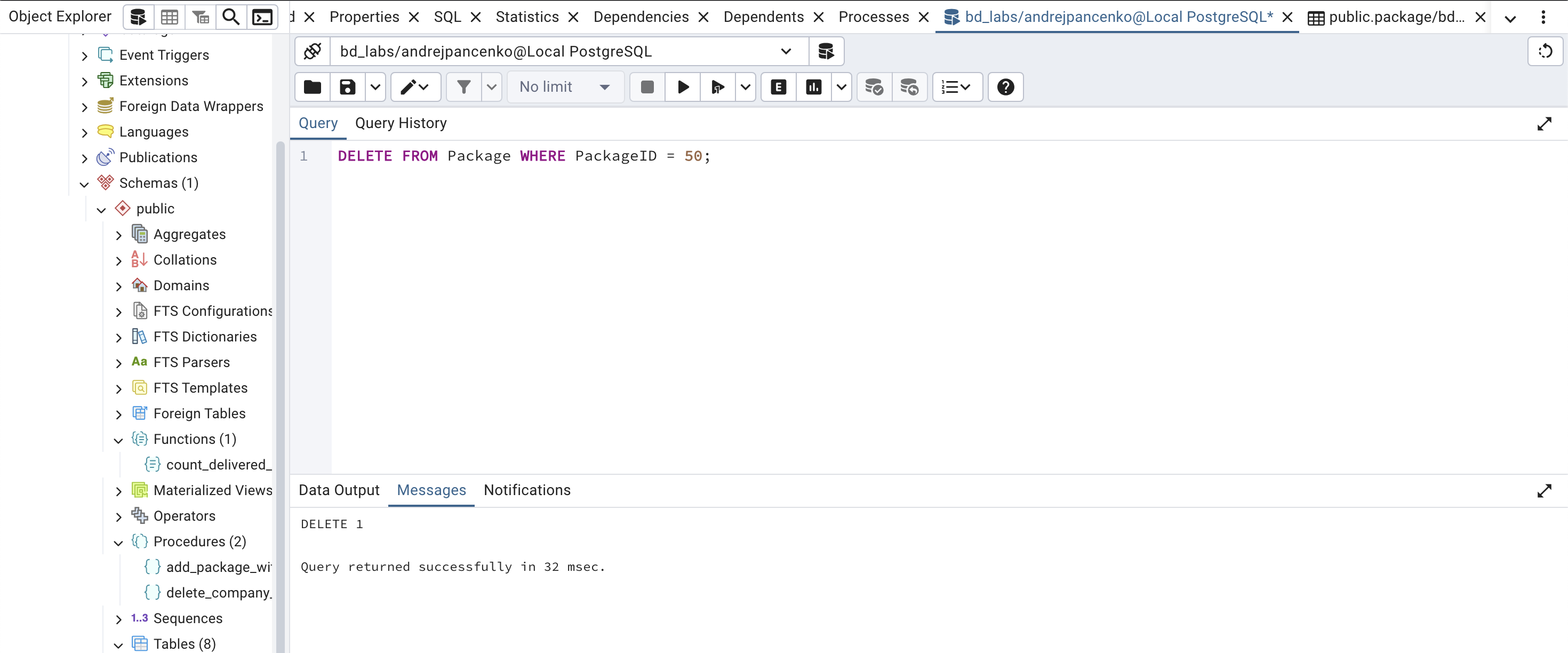
SELECT \* FROM package\_log;



7. Триггер: Логирование изменений после удаления (log\_after\_delete)

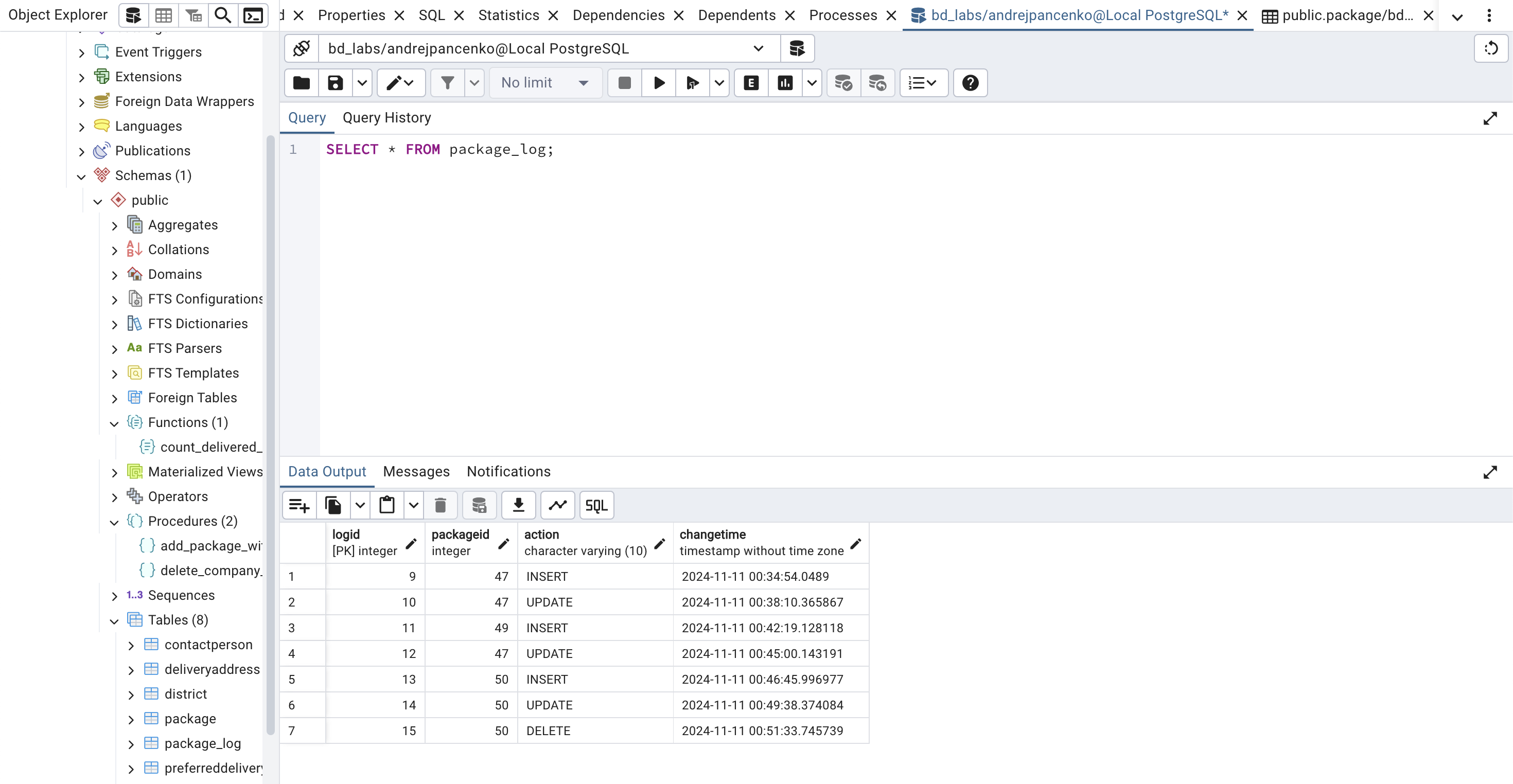
Удаление записи (должен записаться лог):

DELETE FROM Package WHERE PackageID = 50;



Проверка логов после удаления:

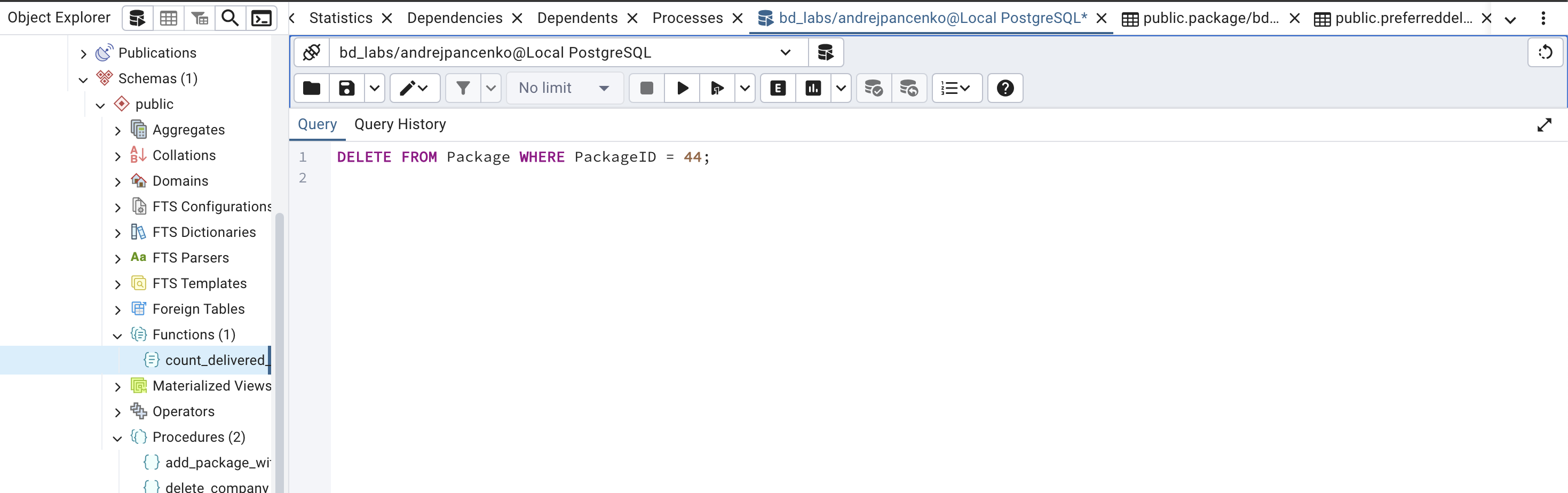
SELECT \* FROM package\_log;

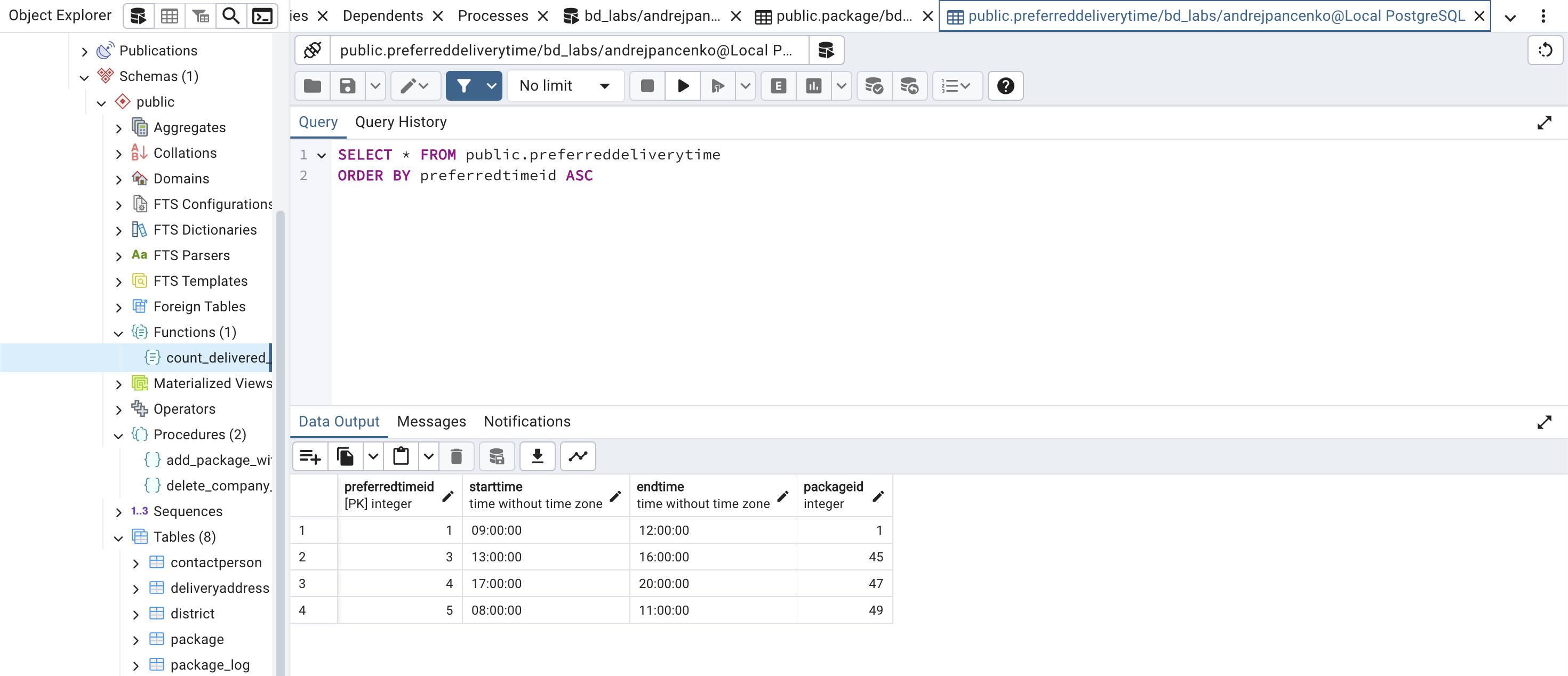


8. Триггер: Удаление связанных записей из PreferredDeliveryTime перед удалением из Package

Удаление записи из таблицы Package (должны быть удалены связанные записи из таблицы PreferredDeliveryTime):

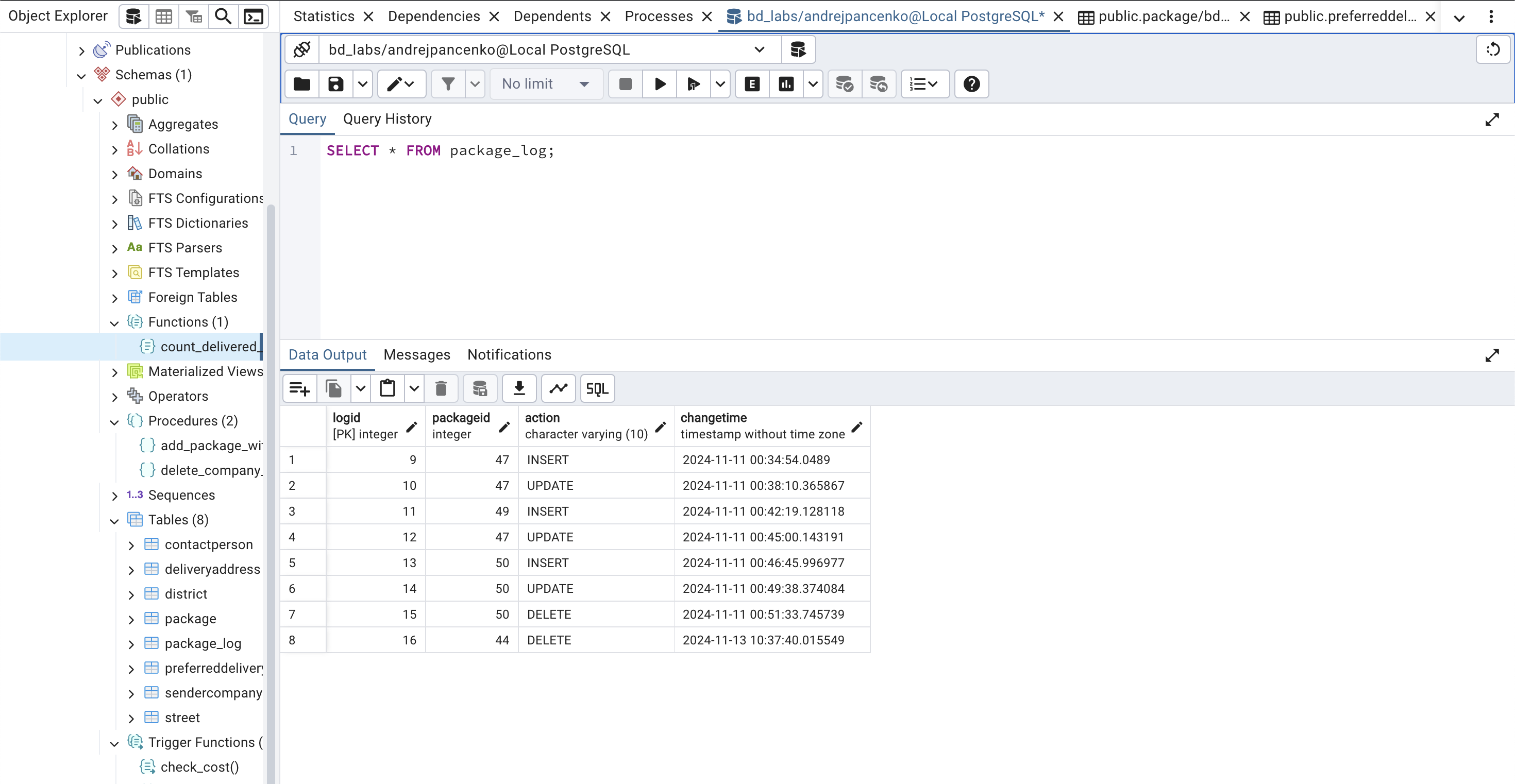
DELETE FROM Package WHERE PackageID = 44;





Проверка логов после удаления:

SELECT \* FROM package\_log;



**Выводы об использовании триггеров в разработанной базе данных:**

Использование триггеров в разработанной базе данных существенно улучшает управление данными и повышает их целостность. Триггеры, такие как check\_cost\_before\_insert и check\_cost\_before\_update, обеспечивают проверку корректности вводимых данных, предотвращая вставку или обновление записей с некорректными значениями, например, с отрицательной стоимостью. Это помогает поддерживать целостность базы данных и избегать ошибок. Триггеры update\_status\_before\_insert и update\_status\_before\_update автоматически устанавливают статус доставки пакета в зависимости от наличия фактической даты доставки, что упрощает управление данными и гарантирует их актуальность. Логирование изменений с помощью триггеров log\_after\_insert, log\_after\_update и log\_after\_delete фиксирует все действия, происходящие в таблице Package, позволяя вести аудит и отслеживать историю изменений. Это полезно для анализа и отладки. Кроме того, триггеры позволяют реализовать сложные бизнес-правила непосредственно на уровне базы данных, снижая необходимость в дополнительной логике на уровне приложения. Они также могут откатывать транзакции в случае обнаружения ошибок или нарушений целостности данных, что обеспечивает дополнительный уровень защиты. В целом, использование триггеров снижает нагрузку на клиентские приложения и делает систему более устойчивой к ошибкам, что в конечном итоге улучшает качество работы с данными и повышает эффективность разработки.