**Лабораторная работа №2**

**Транзакции и ограничения в среде PostgreSQL**

*Цель работы:* Приобрести навыки создания пользователей баз данных, работы с транзакциями в среде PostgreSQL. Изучить основные команды по работе с базой данных. Получение навыков работы с ограничениями.

*Задание. Общая часть:* Ознакомиться с теоретическими сведениями о возможностях создания пользователей баз данных, использования транзакций в PostgreSQL. Создать нового пользователя и зайти под его именем. Ознакомиться с теоретическими сведениями о возможностях создания ограничений в PostgreSQL. Наложить ограничения согласно своему варианту. Проверить работоспособность ограничений путем добавления в таблицы данных, удовлетворяющих и не удовлетворяющих условиям ограничений. Создать транзакционный блок, в котором производится добавление в таблицы произвольных полей, создать несколько производных таблиц, просмотреть структуру измененных таблиц. Не завершая транзакции параллельно запустить еще одно окно терминала, подключиться к базе и попробовать добавить и удалить записи в таблицы. Сделать откат транзакций, просмотреть структуру таблиц. Просмотреть и проанализировать полученную в результате выполнения операций информацию.

*Вариант 1*: Заполнить базу данных игроков хоккейной команды (при необходимости). Запретить ввод роста хоккеиста более 210 см и менее 150 см. Поле «хват клюшки» должно содержать только значения правый и левый. «Игровая позиция» может быть только вратарь, нападающий, защитник, полузащитник. Вместительность стадиона не может быть отрицательной. Продемонстрировать результаты работы.

*Вариант 2*: Заполнить базу данных для учета работы продуктового магазина (при необходимости). Ограничить поле «цена» и «стоимость» так, чтобы они не содержали значений ниже 10 р. и выше 15000р. Проданное количество товара не может быть отрицательным. Дата покупки должна соответствовать действительности (не должна быть в будущем). Ограничить заполнение поля «тип товара» следующими наименованиями: бакалея, молочные продукты, кондитерские изделия. Продемонстрировать результаты работы.

*Вариант 3*: Заполнить базу данных для службы такси (при необходимости). Ограничить ввод даты рождения водителя таким образом, чтобы нельзя было ввести данные о лицах, не достигших возраста 21 год. Расстояние не может быть отрицательным. Ограничить поле «водительский стаж» так, чтобы это поле не содержало значений ниже 3 лет. На поле «стоимость заказа» наложить ограничение таким образом, чтобы это поле не содержало значений ниже 100 р. Продемонстрировать результаты работы.

*Вариант 4*: Заполнить базу данных для обработки данных по работе книжной лавки (при необходимости). Ограничить заполнение поля «жанр» следующими наименованиями: детектив, фантастика, комедия, мелодрама, боевик. На поле «стоимость» наложить ограничение таким образом, чтобы нельзя было ввести цену ниже 50 р. Год выпуска книги не должен быть в будущем (должен соответствовать действительности). Для поля «тираж» ограничить ввод значений менее 100 штук. Продемонстрировать результаты работы.

*Вариант 5*: Заполнить базу данных для осуществления учета работы автосалона (при необходимости). Наложить ограничение на поле «привод»: оно должно содержать только «полный», «передний», «задний». Поле «руль» может содержать только значения правый и левый. Дата выпуска должна соответствовать действительности (не должна быть больше текущей даты). Ограничить заполнение поля «тип кузова» следующими наименованиями: седан, универсал, микроавтобус. Продемонстрировать результаты работы.

*Вариант 6*: Заполнить базу данных учета поступления товара на овощную базу (при необходимости). Для поля «вес» ограничить ввод значений менее 10 кг. Дата поставки должна соответствовать действительности (не должна быть в будущем). Стоимость не должна быть отрицательной. Ограничить заполнение поля «сорт» следующими наименованиями: второй, первый, высший. Продемонстрировать результаты работы.

*Вариант 7*: Заполнить базу данных для учета работы типографии (при необходимости). Поле «количество» не должно содержать значений менее 100 шт. Поле «способ расчета» может содержать только наименования: наличный и безналичный. Стоимость заказа не должна быть отрицательной. Ограничить заполнение поля «тип продукции» следующими наименованиями: этикетка, коробка, блокнот. Продемонстрировать результаты работы.

*Вариант 8*: Заполнить базу данных жилищной управляющей компании (при необходимости). Ограничить размер ремонта, она не должна превышать 5000 руб. и быть менее 500 руб. Длительность ремонта не должна быть отрицательной. Поле «способ расчета» может содержать только наименования: наличный и безналичный. Информация об оплате ремонта может содержать только наименования: оплачено полностью, не оплачено, оплачено частично. Продемонстрировать результаты работы.

*Вариант 9*: Заполнить базу данных ремонтной мастерской (при необходимости). Поле «способ расчета» может содержать только наименования: наличный и безналичный. Возраст мастеров не должен быть менее 18 лет. Стоимость не должна быть отрицательной. Ограничить заполнение поля «вид ремонтируемого предмета» следующими наименованиями: холодильник, пылесос, кофеварка, фен, мультиварка, микроволновка. Продемонстрировать результаты работы.

*Вариант 10*: Заполнить базу данных туристического агентства (при необходимости). Ограничить заполнение поля «вид тура» следующими

наименованиями: автобусный, железнодорожный, авиа. Возраст руководителей туров не должен быть менее 25 лет. Стоимость тура не должна быть отрицательной. Количество участников тура не должно превышать 25 человек и быть менее 10 человек. Продемонстрировать результаты работы.

*Содержание отчета*

Отчет должен содержать титульный лист, цель работы, задание, коды команд на каждом этапе выполнения работы, результаты (скриншоты) выполнения команд (проверка работоспособности ограничений в соответствии с вариантом), результаты выполнения команд при работе с транзакциями (скриншоты), выводы и анализ результатов работы.

## Использование транзакций в PostgreSQL

Транзакции являются одним из фундаментальных концептов всех СУБД. Сущность транзакции состоит в связывании нескольких шагов в одну операцию по принципу «все или ничего». Внутренние промежуточные состояния между шагами не видны для других конкурирующих транзакций и если во время выполнения транзакции случится ошибка, которая помешает транзакции завершится, то в базе данных никаких изменений сделано не будет.

Транзакция является атомарным действием с точки зрения других транзакций и либо она завершится полностью успешно, либо никакие действия, составляющие транзакцию, выполнены не будут.

Мы также хотим гарантировать, что одна полностью завершившаяся и подтверждённая СУБД транзакция является действительно сохранённой и не может быть потеряна, даже если после её выполнения произойдет крах системы. Традиционные СУБД гарантируют, что все обновления, осуществляемые в одной транзакции, протоколируются в надежное

хранилище (т.е. на диск) перед тем как СУБД сообщит о завершении транзакции.

Другое важное свойство транзакционных СУБД состоит в строгой изоляции транзакций: когда несколько транзакций запускаются конкурентно, каждая из них не видит тех неполных изменений, которые производят другие транзакции. Таким образом, транзакции должны выполнять принцип «все или ничего» не только в плане нерушимости тех изменений, которые они производят в базе данных, но и также в плане того, что они видят в момент работы. Обновления, которые вносит открытая транзакция, являются невидимыми для других транзакций пока данная транзакция не завершиться, после чего все внесенные ей изменения станут видимыми.

В PostgreSQL транзакция - это список команд SQL, которые находятся внутри блока, начинающегося командой *BEGIN* и заканчивающегося командой *COMMIT*.

Если во время выполнения транзакции мы решаем, что не хотим завершать ее, то мы вместо команды *COMMIT* вводим команду *ROLLBACK* и все наши изменения от начала транзакции, будут отменены.

PostgreSQL фактически считает каждый оператор SQL запущенным в транзакции. Если вы не указываете команду *BEGIN*, то каждый отдельный оператор имеет неявную команду *BEGIN* перед оператором и (при успешной отработке оператора) команду *COMMIT* после оператора. Группа операторов, заключаемая в блок между *BEGIN* и *COMMIT*, иногда называется транзакционным блоком.

Можно управлять операторами в транзакции и на более детализированном уровне с помощью "точек сохранения" (savepoints). Точки сохранения позволяют выборочно отбрасывать части транзакции, в то же время выполняя остаток транзакции. После того как вы зададите точку сохранения с помощью оператора *SAVEPOINT имя_точки*, вы можете, если понадобится, откатить транзакцию до этой точки сохранения с помощью оператора *ROLLBACK TO*. Все изменения базы данных внутри транзакции

между точкой сохранения и местом откуда вызван откат теряются, но изменения, которые были сделаны до точки сохранения остаются.

После отката к точке сохранения, она продолжает оставаться заданной и, таким образом, вы можете делать к ней откат несколько раз. И наоборот, если вы уверены, что вам не нужен снова откат к определённой точке сохранения, она может быть убрана, чтобы система могла освободить некоторые ресурсы. Запомните, что откат к некоторой точке сохранения или ее удаление, автоматически удаляет все точки сохранения, которые были заданы после нее.

Всё это происходит внутри транзакционного блока, так что ничего из этого не будет видно в других сессиях. Когда и если вы завершаете транзакционный блок, выполняемые действия становятся видимыми в других сессиях как единичная операция, в то время как действия по откату транзакции никогда не становятся видимыми для других сессий.

## Создание ограничений

Ограничение (constraint) представляет собой особый атрибут таблицы, который устанавливает критерии допустимости для содержимого ее полей. Соблюдение этих правил помогает предотвратить заполнение базы ошибочными или неподходящими данными.

Ограничения задаются в секции *CONSTRAINT* при создании таблицы командой *CREATE TABLE*. Они делятся на два типа - ограничения полей и ограничения таблиц.

Ограничения полей всегда относятся только к одному полю, тогда как ограничения таблиц могут устанавливаться как для одного, так и для нескольких полей.

В команде *CREATE TABLE* ограничения полей задаются сразу же после определения поля, тогда как ограничение таблицы устанавливается в специальном блоке, отделенном запятой от всех определений полей. Поля, на

которые распространяется ограничение таблицы, задаются самим определением, а не его расположением в команде.

Ниже описаны различные правила, устанавливаемые при помощи ограничений.

При выполнении команды *\h CREATE TABLE* клиент psql выводит несколько подробных синтаксических диаграмм для ограничений, которые могут устанавливаться для таблиц. Список команд интерактивного терминала приведен в приложении Г. Синтаксис ограничения поля выглядит так:

*[CONSTRAINT ограничение]*
*{NOT NULL | UNIQUE | PRIMARY KEY | DEFAULT значение |*
*CHECK (условие) | REFERENCES таблица [(поле)]*
*[MATCH FULL | MATCH PARTIAL] [ON DELETE операция]*
*[ON UPDATE операция] [DEFERRABLE | NOT DEFERRABLE]*
*[INITIALLY DEFERRED | INITIALLY IMMEDIATE]}*

Определение следует в команде *CREATE TABLE* сразу же за типом ограничиваемого поля и предшествует запятой, отделяющей его от следующего поля. Ограничения могут устанавливаться для любого количества полей, а ключевое слово *CONSTRAINT* и идентификатор *ограничение* не обязательны.

Существует шесть типов ограничений полей, задаваемых при помощи специальных ключевых слов. Некоторые из них косвенно устанавливаются при создании ограничений другого типа. Типы ограничений полей перечислены ниже.

*NOT NULL*. Поле не может содержать псевдозначение NULL. Ограничение *NOT NULL* эквивалентно ограничению *CHECK (поле NOT NULL)*.

*UNIQUE*. Поле не может содержать повторяющиеся значения. Следует учитывать, что ограничение *UNIQUE* допускает многократное вхождение

псевдозначений NULL, поскольку формально NULL не совпадает ни с каким другим значением.

*PRIMARY KEY*. Автоматически устанавливает ограничения *UNIQUE* и *NOT NULL*, а для заданного поля создается индекс. В таблице может устанавливаться только одно ограничение первичного ключа.

*DEFAULT значение*. Пропущенные значения поля заменяются заданной величиной. Значение по умолчанию должно относиться к типу данных, соответствующему типу поля

*CHECK (условие)*. Команда *INSERT* или *UPDATE* для записи завершается успешно лишь при выполнении заданного условия (выражения, возвращающего логический результат). При установке ограничения поля в секции *CHECK* может использоваться только поле, для которого устанавливается ограничение.

*REFERENCES*. Это ограничение состоит из нескольких секций. *REFERENCES таблица [(поле)]*. Входные значения ограничиваемого поля сравниваются со значениями другого поля в заданной таблице. Если совпадения отсутствуют, команда *INSERT* или *UPDATE* завершается неудачей. Если параметр *поле* не указан, проверка выполняется по первичному ключу.

*MATCH FULL | MATCH PARTIAL*. Секция *MATCH* указывает, разрешается ли смешивание значений NULL и обычных значений при вставке в таблицу, у которой внешний ключ ссылается на несколько полей. Таким образом, на практике секция *MATCH* приносит пользу лишь в ограничениях таблиц, хотя формально она может использоваться и при ограничении полей. Конструкция *MATCH FULL* запрещает вставку данных, у которых часть полей внешнего ключа содержит псевдозначение NULL (кроме случая, когда NULL содержится во всех полях). Если секция *MATCH* отсутствует, считается, что поля с псевдозначениями NULL удовлетворяют ограничению. Также будет уместно напомнить, что ограничения полей

относятся лишь к одному полю, поэтому секция *MATCH* используется лишь в ограничениях таблиц.

*ON DELETE операция*. При выполнении команды *DELETE* для заданной таблицы с ограничиваемым полем выполняется одна из следующих операций: *NO ACTION* (если удаление приводит к нарушению целостности ссылок, происходит ошибка; используется по умолчанию, если операция не указана), *RESTRICT* (аналогично *NO ACTION*), *CASCADE* (удаление всех записей, содержащих ссылки на удаляемую запись), *SET NULL* (поля, содержащие ссылки на удаляемую запись, заменяются псевдозначениями NULL), *SET DEFAULT* (полям, содержащим ссылки на удаляемую запись, присваивается значение по умолчанию).

*ON UPDATE операция*. При выполнении команды *UPDATE* для заданной таблицы выполняется одна из операций, описанных выше. По умолчанию используется значение *NO ACTION*. Если выбрана операция *CASCADE*, все записи, содержащие ссылки на обновляемую запись, обновляются новым значением (вместо удаления, как в случае с *ON DELETE CASCADE*).

*DEFERRABLE | NOT DEFERRABLE*. Значение *DEFERRABLE* позволяет отложить выполнение ограничения до конца транзакции (вместо немедленного выполнения после завершения команды). Значение *NOT DEFERRABLE* означает, что ограничение всегда проверяется сразу же после завершения очередной команды. В этом случае пользователь не может отложить проверку ограничения до конца транзакции. По умолчанию выбирается именно этот вариант.

*INITIALLY DEFERRED | INITIALLY IMMEDIATE*. Секция *INITIALLY* задается только для ограничений, определенных с ключевым словом *DEFERRED*. Значение *INITIALLY DEFERRED* откладывает проверку ограничения до конца транзакции, а при установке значения *INITIALLY IMMEDIATE* проверка производится после каждой команды. При отсутствии

секции *INITIALLY* по умолчанию используется значение *INITIALLY IMMEDIATE*.

В ограничениях таблиц, в отличие от ограничений полей, могут участвовать сразу несколько полей таблицы. Синтаксис ограничения таблицы:

*[CONSTRAINT ограничение] {UNIQUE (поле [...]) |*
*PRIMARY KEY (поле [...]) | CHECK (условие) |*
*FOREIGN KEY (поле [...]) REFERENCES таблица [(поле [...])]*
*[MATCH FULL | MATCH PARTIAL] [ON DELETE операция]*
*[ON UPDATE операция] [DEFERRABLE | NOT DEFERRABLE]*
*[INITIALLY DEFERRED | INITIALLY IMMEDIATE]*

Секция *CONSTRAINT* ограничение определяет необязательное имя. Ограничениям рекомендуется присваивать содержательные имена вместо автоматически сгенерированных имен, не несущих никакой полезной информации. В будущем имя также может пригодиться и для удаления ограничения (например, в секции *DROP CONSTRAINT* команды *ALTER TABLE*). Другие секции относятся к четырем разновидностям ограничений таблиц.

*PRIMARY KEY (поле [...])*. Ограничение таблицы *PRIMARY KEY* имеет много общего с аналогичным ограничением поля. В ограничении таблицы *PRIMARY KEY* могут перечисляться несколько полей, разделенных запятыми. Для перечисленных полей автоматически строится индекс. Как и в случае с ограничением поля, комбинация значений всех полей должна быть уникальной и не может содержать NULL.

*UNIQUE (поле [...])*. Ограничение означает, что комбинация значений полей, перечисленных за ключевым словом *UNIQUE*, принимает только уникальные значения. Допускается многократное вхождение псевдозначения NULL, поскольку оно формально не совпадает ни с одним значением.

*CHECK (условие)*. Команда *INSERT* или *UPDATE* для записи завершается успешно лишь при выполнении заданного условия (выражения,

возвращающего логический результат). Используется по аналогии с ограничениями полей, но в секции *CHECK* может содержать ссылки на несколько полей.

*FOREIGN KEY (поле [...]) REFERENCES таблица [(поле [...])]*. В качестве прототипа для секции *REFERENCES* можно перечислить несколько полей. Синтаксис части, следующей за секцией *FOREIGN KEY*, идентичен синтаксису ограничения *REFERENCES* для полей.

Команда *ALTER TABLE* позволяет включать ограничения в существующую таблицу. Установка ограничений в команде *ALTER TABLE* имеет следующий синтаксис:

*ALTER TABLE таблица ADD | DROP [CONSTRAINT ограничение]*

*{CHECK (условие) | FOREIGN KEY (поле [...])*

*REFERENCES таблица [(поле [....])] [MATCH FULL | MATCH PARTIAL]*

*[ON DELETE операция] [ON UPDATE операция]*

*[DEFERRABLE | NOT DEFERRABLE]*

*[INITIALLY DEFERRED | INITIALLY IMMEDIATE]}*

*Контрольные вопросы*

1. Какими правами можно наделять пользователей?
2. Какими свойствами обладают транзакции?
3. Для чего нужен откат транзакций?
4. Как выполнить откат транзакции?
5. Как создать точку останова внутри группы транзакций?
6. Как сохранить транзакцию?
7. Можно ли осуществить откат части транзакции? Если да, то как?
8. Что такое ограничение? Как можно задать ограничение?
9. Какие типы ограничений полей существуют?
10. Каким образом можно добавить ограничения в существующую таблицу?
11. Каким образом можно увидеть все ограничения, наложенные на поля таблицы?
12. Как удалить ограничение?

**Список источников**


1. Малыхина М.П. Базы данных: основы, проектирование, использование: учебное пособие для вузов по направлению подготовки "Информатика и вычислительная техника" / М. П. Малыхина. - СПб., 2006. - 517 с. : ил.
2. Хомоненко А.Д. Базы данных: учебник для вузов по техническим и экономическим специальностям / [Хомоненко А. Д., Цыганков В. М., Мальцев М. Г.] ; под ред. А. Д. Хомоненко. - М., 2006. - 736 с. : ил., табл.
3. Разработка приложений на C# с использованием СУБД PostgreSQL : учебное пособие / [И. А. Васюткина и др.] ; Новосиб. гос. техн. ун-т. - Новосибирск, 2015. - 141, [1] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000220068
4. Дж Уорсли Дж., Дрейк Дж. PostgreSQL. Для профессионалов - СПб.: Питер, 2003. -496 с.
5. PostgreSQL (русский) [Электронный ресурс]: Документация по PostgreSQL. – Режим доступа: http://postgresql.ru.net . - загл. с экрана.
6. Википедия свободная энциклопедия (русский) [Электронный ресурс]: PostgreSQL. – Режим доступа: http://ru.wikipedia.org. - загл. с экрана.
7. Компьютерная документация (русский) [Электронный ресурс]: Функции для работы с датой и временем. – Режим доступа: http://www.hardline.ru/ . - загл. с экрана.
8. Linux и Windows: помощь админам и пользователям (русский) [Электронный ресурс]: 15 команд для управления PostgreSQL. – Режим доступа: http://www.guruadmin.ru . - загл. с экрана.