Решение тестового задания

1. Создать новую базу данных PGSQL с именем test\_db:  
  
postgres=# CREATE DATABASE test\_db OWNER test\_user;

Подключиться к данной БД:  
  
\c test\_db test\_user localhost 5432

2. В базе данных создать таблицу с именем «People», содержащую 3 столбца с разными типами данных (порядковый Index, Name, DataOfBirth) с разными типами данных, например: int, varchar[255], date.

Создание таблицы:

CREATE TABLE People (

Index serial PRIMARY KEY,

Name text,

DataOfBirth date

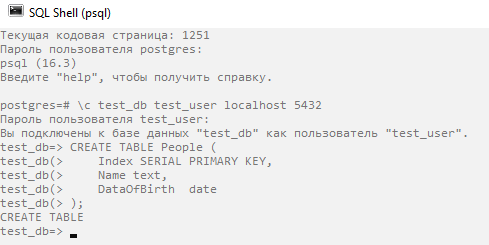
);

**Обоснование выбора конкретных типов данных.**

Для данной таблицы для столбца Index, который подразумевает порядковый номер, выбран тип данных serial потому, что определение данного типа в PostgreSQL равнозначно типу integer и автоматически добавляет числа, аналогично свойству AUTO\_INCREMENT. Особенностью данного типа является то, что при удалении данных и добавлении новых счётчик не уменьшится, будет только увеличиваться даже в случае неудачной попытки вставить строки. Это надо учитывать при добавлении новых строк и явно указывать столбец Index и его правильное порядковое значение, если были произведены изменения. Добавлено ограничение PRIMARY KEY для того, чтобы избежать повторяющиеся и неопределённые значения.

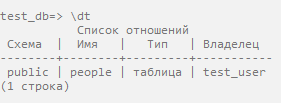
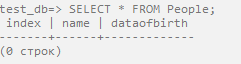
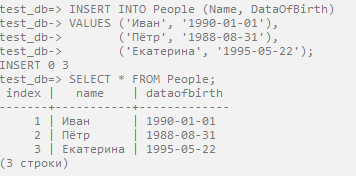
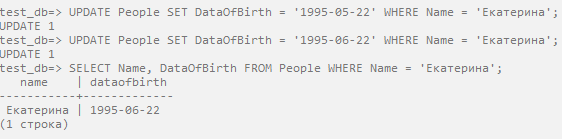
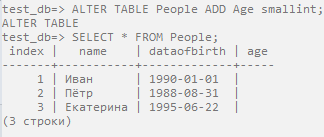
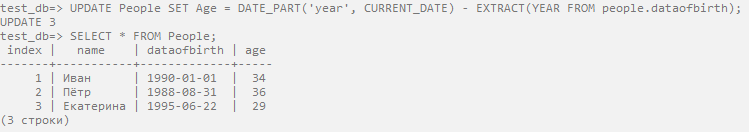
Для столбца Name выбран тип данных text для того, чтобы записывать имя символьной строкой переменной длины.

Для столбца DataOfBirth выбран тип данных date для указания года, месяца и дня рождения людей.

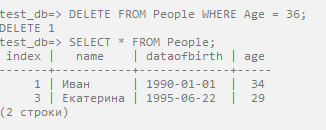


**Позитивные и негативные сценарии наполнения таблицы данными и присутствия данных в таблице с использованием SQL-команд (Select, Insert, Update, Delete и т.д., дополнив их соответствующим WHERE или другими параметрами).**

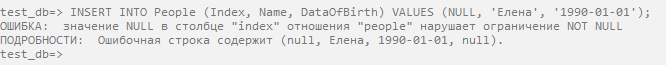
Позитивные сценарии.

1. Проверка существования тестируемой таблицы.  
     
   Отправить запрос psql, который выводит список таблиц: \dt.  
   Ожидаемый результат: таблица "People" присутствует в списке таблиц.  
     
   
2. Проверка присутствия столбцов «Index», «Name», «DataOfBirth».  
     
   Отправить запрос: SELECT \* FROM People;  
   Ожидаемый результат: таблица содержит только столбцы «Index», «Name», «DataOfBirth».  
     
   
3. Проверка вставки данных всех столбцов.  
     
   Отправить запрос с данными:   
   INSERT INTO People (Name, DataOfBirth)   
   VALUES (‘Иван’, ’1990-01-01’),  
    (‘Пётр’, ’1988-08-31’),  
    (‘Екатерина’, ’1995-05-22’);  
   Ожидаемый результат: ответ INSERT 0 3.  
     
   Отправить запрос для проверки данных: SELECT \* FROM People;  
   Ожидаемый результат: в таблицу добавлены данные.  
     
   
4. Проверка изменения одного значения столбца.  
     
   Отправить запрос на изменение «DataOfBirth»:   
   UPDATE People SET DataOfBirth = ’1995-06-22’   
   WHERE Name = ‘Екатерина’;  
   Ожидаемый результат: ответ UPDATE 1.  
     
   Отправить запрос для проверки изменённых данных:   
   SELECT Name, DataOfBirth FROM People;  
   Ожидаемый результат: данные изменены.  
     
   
5. Проверка добавления столбцов.  
     
   Отправить запрос на создание столбца «Age»:   
   ALTER TABLE People ADD Age smallint;  
   Ожидаемый результат: ответ ALTER TABLE.  
     
   Отправить запрос для проверки данных: SELECT \* FROM People;  
   Ожидаемый результат: столбца «Age» добавлен.  
     
   
6. Проверка изменения всех данных столбца.  
     
   Отправить запрос на вычисление и добавление возраста людей в столбец «Age»:  
   UPDATE People SET Age = DATE\_PART('year', CURRENT\_DATE) - EXTRACT(YEAR FROM people.dataofbirth);  
   Ожидаемый результат: ответ UPDATE 3.  
     
   Отправить запрос: SELECT \* FROM People;  
   Ожидаемый результат: столбец «Age» заполнен корректными данными.  
   
7. Проверка удаления данных.

Отправить запрос на удаление данных:   
DELETE FROM People WHERE Age = 36;  
Ожидаемый результат: ответ DELETE 1.  
  
Отправить запрос для проверки данных: SELECT \* FROM People;  
Ожидаемый результат: данные удалены, номера Index не по порядку.



Негативные сценарии.

1. Проверка добавления не поддерживаемых символов.  
     
   Отправить запрос с некорректным символом:   
   INSERT INTO People (Name, DataOfBirth) VALUES (‘Елена’, ’1990-01-01’);  
   Ожидаемый результат: ОШИБКА: неверная последовательность байт для кодировки "UTF8": 0xe2 0x22 0x29.  
     
   
2. Проверка добавления некорректного формата данных data.  
     
   Отправить запрос на добавление с некорректным форматом даты:   
   INSERT INTO People (Name, DataOfBirth) VALUES (‘Елена’, ’19-90-01-01’);  
   Ожидаемый результат: ОШИБКА: неверный синтаксис для типа date: "19-90-01-01"  
   СТРОКА 1: ...le (Index, Name, DataOfBirth) VALUES (2, 'Елена', '19-90-01-...  
     
   
3. Проверка добавления некорректного количества значений.  
     
   Отправить запрос с большим количеством значений, чем столбцов:   
   INSERT INTO People (Name, DataOfBirth) VALUES (2, 'Елена', '1990-01-01');  
   Ожидаемый результат: ОШИБКА: INSERT содержит больше выражений, чем целевых столбцов  
   СТРОКА 1: ...TO People (Name, DataOfBirth) VALUES (2, 'Елена', '1990-01-0...  
   
4. Проверка обновления некорректного формата данных data.  
     
   Отправить запрос на обновление с некорректным форматом даты:   
   UPDATE People SET DataOfBirth = '19-95-07-22' WHERE Name = 'Екатерина';  
   Ожидаемый результат: ОШИБКА: неверный синтаксис для типа date: "19-95-07-22"  
   СТРОКА 1: UPDATE People SET DataOfBirth = '19-95-07-22' WHERE Name = '...  
     
   
5. Проверка целостности путём добавления порядкового номера со значением NULL:  
     
   Отправить запрос на добавление значению Index NULL:   
   INSERT INTO People (Index, Name, DataOfBirth) VALUES (NULL, 'Елена', '1990-01-01');  
   Ожидаемый результат: ОШИБКА: значение NULL в столбце "index" отношения "people" нарушает ограничение NOT NULL   
     
   
6. Проверка целостности путём добавления порядкового номера с дублирующим значением Index.  
     
   Отправить запрос на добавление дублирующего значения Index:   
   INSERT INTO People (Index, Name, DataOfBirth) VALUES (1, 'Елена', '1990-01-01');  
   Ожидаемый результат: ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "people\_pkey"  
   ПОДРОБНОСТИ: Ключ "(index)=(1)" уже существует.  
     
   

3. В базе данных создать таблицу с именем "People", содержащую следующие столбцы "Index", "FirstName", "FamilyName", "DateOfBirth", "PlaceOfBirth", "Occupation", "Hobby")

Так как одинаковые названия таблиц запрещены, удаляем таблицу «People» из п.2:

DROP TABLE People;

Создание таблицы:

CREATE TABLE People (

Index serial PRIMARY KEY,

FirstName text,

FamilyName text,

DataOfBirth date,

PlaceOfBirth text,

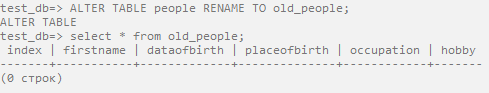
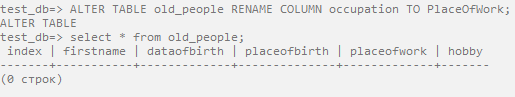
Occupation text,

Hobby text

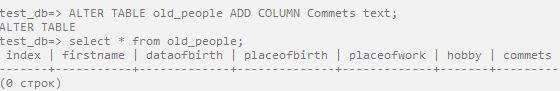
);

**Позитивные и негативные сценарии изменения таблицы (переименовать таблицу/столбец, добавить/удалить столбец/таблицу, изменить тип данных и т.д.)**

Позитивные сценарии.

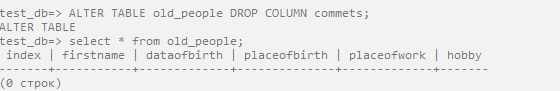
1. Проверка существования тестируемой таблицы.  
     
   Отправить запрос psql, который выводит список таблиц: \dt.  
   Ожидаемый результат: таблица "People" присутствует в списке таблиц.
2. Проверка присутствия созданных столбцов.  
     
   Отправить запрос: SELECT \* FROM People;  
   Ожидаемый результат: таблица содержит все указанные при создании столбы.
3. Проверка переименования таблицы.  
     
   Отправить запрос для переименования таблицы:   
   ALTER TABLE people RENAME TO old\_people;  
   Ожидаемый результат: таблица переименована.  
   
4. Проверка переименования столбца.  
     
   Отправить запрос для переименования столбца:   
   ALTER TABLE old\_people RENAME COLUMN occupation TO PlaceOfWork;  
   Ожидаемый результат: столбец переименован.  
   
5. Добавить столбец.

Отправить запрос для добавления столбца:  
ALTER TABLE old\_people ADD COLUMN Commets text;  
Ожидаемый результат: столбец добавлен.



1. Удалить столбец.

Отправить запрос для удаления столбца:  
ALTER TABLE old\_people DROP COLUMN commets;  
Ожидаемый результат: столбец добавлен.



1. Изменить тип данных.

Отправить запрос для изменения типа столбца Index на bigint:  
ALTER TABLE old\_people ALTER COLUMN index TYPE bigint;  
Ожидаемый результат: тип данных изменен.  


Негативные сценарии провести аналогично первой таблице People.

4. Предложение дополнительных позитивные и негативные сценарии DML операций.

Позитивные:

1. Добавление/удаления ограничения.
2. Добавить/удалить значения по умолчанию.
3. Добавить отдельные проверки с порядковыми номерами (добавлением/удалением).

Негативные:

1. Отправлять количество знаков, большее, чем размер предусмотренных для них.
2. Проверить граничные значения для даты.
3. Проверить недоступность таблицы под другим пользователем.

5. Предложение нагрузочного тестирования.

Цель:

- проверить отсутствие негативного влияния пользовательской нагрузки на производительность БД;

- найти места в продукте более подверженные воздействию нагрузки;

- выяснить максимальное количество пользователей.

Подготовка: настроить окружение для мониторинга (например, zabbix), выделить основные параметры, которые нужно мониторить (процессор, память сервера, где работает БД), выбрать инструмент для тестирования и вид отчета.

Варианты действий для проверок:

- за определённое время отправить множество одинаковых/самых частых запросов:

- передача и приём больших файлов.

- запуск других приложений на сервере для дополнительной нагрузки.

Ожидания: продукт работает стабильно (среднее время запросов и ответов не превышено, и пропускная способность не меняется, частота ошибок сервера минимальная).