**Лабораторная № 1**

**«Безопасность строк в PostgreSQL 9.5»**

**Цель работы:**

Ознакомиться с механизмом безопасности строк таблиц в PostgreSQL 9.5.

**Теоретическая информация:**

Row-level security или безопасность на уровне строк — механизм разграничения доступа к информации к БД, позволяющий ограничить доступ пользователей к отдельным строкам в таблицах.

Работает это следующим образом: описываются правила для конкретной таблицы, согласно которым ограничивается доступ к конкретным строкам при выполнении определённых команд, с помощью выражения **CREATE POLICY**. Каждое правило содержит некое логическое выражение, которое должно быть истинным, чтобы строка была видна в запросе. Затем правила активируются с помощью выражения **ALTER TABLE… ENABLE ROW LEVEL SECURITY**. Затем при попытке доступа, например при запросе **SELECT**, проверяется, имеет ли пользователь право на доступ к конкретной строке и если нет, то они ему не показываются. Суперпользователь по умолчанию может видеть все строки, так как у него по умолчанию выставлен флаг BYPASSRLS, который означает, что для данной роли проверки осуществляться не будут.  
  
Синтаксис выражения **CREATE POLICY**:

**CREATE** **POLICY** **name** **ON** table\_name

[ **FOR** { ALL | **SELECT** | **INSERT** | **UPDATE** | **DELETE** } ]

[ **TO** { role\_name | **PUBLIC** | **CURRENT\_USER** | **SESSION\_USER** } [, ...] ]

[ **USING** ( using\_expression ) ]

[ **WITH** **CHECK** ( check\_expression ) ]

Правила создаются для конкретных таблиц, поэтому в БД может быть несколько правил с одним и тем же именем для различных таблиц.  
После выражения **FOR** указывается, для каких именно запросов применяется правило, по умолчанию — **ALL**, то есть для всех запросов.  
  
После **TO** — для каких ролей, по умолчанию — **PUBLIC**, то есть для всех ролей.  
  
Далее, в выражении **USING** указывается булевское выражение, которое должно быть true, чтобы конкретная строка была видна пользователю в запросах, которые используют уже имеющиеся данные (**SELECT**, **UPDATE**, **DELETE**). Если булевское выражение вернуло null или false, то строка видна не будет.  
  
В выражении **WITH CHECK** указывается булевское выражение, которое должно быть true, чтобы запрос, добавляющий или изменяющий данные (**INSERT** или **UPDATE**), прошел успешно. В случае, если булевское выражение вернет null или false, то будет ошибка. Выражение **WITH CHECK** выполняется после триггеров **BEFORE** (если они присутствуют) и до любых других проверок. Поэтому, если триггер **BEFORE** модифицирует строку таким образом, что условие не вернет true, будет ошибка. Для успешного выполнения **UPDATE** необходимо, чтобы оба условия вернули true, в том числе, если в запросе **INSERT… ON CONFILCT DO UPDATE** произойдет конфликт и запрос попытается модифицировать данные. Если выражение **WITH CHECK** опущено, вместо него будет подставляться условие из выражения **USING**.  
В условиях нельзя использовать аггрегирующие или оконные функции.  
  
Обычно, требуется управлять доступом, исходя из того, какой пользователь БД запрашивает данные, поэтому нам пригодятся функции, возвращающие информацию о системе ([System Information Functions](http://www.postgresql.org/docs/9.5/static/functions-info.html)).

**Задание:**

1. Установить PostrgeSQL 9.5

2. Создать несколько (не меньше 3-х) ролей пользователей.

3. Создать таблицу с произвольным количеством строк и столбцов и произвольными данными. Одним из столбцов таблицы будет столбец owner, который будет содержать имена созданных ранее ролей.

Пример таблицы:

Допустим имеется три роли пользователей: ivanov, petrov, sidorov. Тогда таблица будет выглядеть следующим образом:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Произвольный столбец 1 | Произвольный столбец 2 | … | **owner** |
| Произвольная строка 1 |  |  |  | ivanov |
| Произвольная строка 2 |  |  |  | pertov |
| … |  |  |  |  |
| Произвольная строка n-1 |  |  |  | ivanov |
| Произвольная строка n |  |  |  | sidorov |

4. Создать политику безопасности, позволяющую определенной роли пользователя просматривать, изменять, удалять только те строки таблицы, где он указан как owner.

5. По результатам работы предоставить отчет, который будет содержать код политики безопасности и скриншоты, демонстрирующие работу безопасности строк.

6.\* Придумать свою политику безопасности строк таблицы (любой).

**Отчет необходимо оформить по шаблону с сайта «ЛЭТИ» (титульник, цель работы, ход выполнения работы, вывод).**

**Критерии оценивания лабораторной работы: 5 пунктов – оценка 4,**

**6 пунктов – оценка 5.**

**Отчеты присылать на почту** [ZOC.leti@yandex.ru](https://vk.com/write?email=ZOC.leti@yandex.ru) **. В теме письма должно содержаться имя, фамилия, группа, номер лабораторной работы и слово «PostgreSQL».**