Учебно-методическое пособие по теме:

«Разделение прав доступа и проведения транзакций в соответствии с уровнем пользователей»

Цель: разделение прав доступа и проведения транзакций в соответствии с уровнем пользователей.

Содержание:

1. Права доступа, типы прав доступа.....1
2. Создание нового пользователя…….2
3. GRANT – определить права доступа.....5
4. REVOKE – отозвать права доступа.....9
5. Транзакции. Понятие транзакции.....10

ПРАВА ДОСТУПА

Когда в базе данных создаётся объект, ему назначается владелец. Владельцем обычно становится роль, с которой был выполнен оператор создания. Для большинства типов объектов в исходном состоянии только владелец (или суперпользователь) может делать с объектом всё, что угодно. Чтобы разрешить использовать его другим ролям, нужно дать им права.

Существует несколько типов прав: SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE и USAGE. Набор прав, применимых к определённому объекту, зависит от типа объекта (таблица, функция и т. д.)

Неотъемлемое право изменять или удалять объект имеет только владелец объекта.

Объекту можно назначить нового владельца с помощью команды ALTER для соответствующего типа объекта, например ALTER TABLE(ALTER TABLE — изменить определение таблицы. ALTER TABLE меняет определение существующей таблицы. Несколько её разновидностей описаны ниже. Заметьте, что для разных разновидностей могут требоваться разные уровни блокировок. Если явно не отмечено другое, требуется блокировка ACCESS EXCLUSIVE. При перечислении нескольких подкоманд будет запрашиваться самая сильная блокировка из

требуемых ими.). Суперпользователь может делать это без ограничений, а обычный пользователь, только если он является одновременно текущим владельцем объекта (или членом роли владельца) и членом новой роли.

Для назначения прав применяется команда GRANT. Например, если в базе данных есть роль joe и таблица accounts, право на изменение таблицы можно дать этой роли так:

```
GRANT UPDATE ON accounts TO joe;
```

Если вместо конкретного права написать ALL, роль получит все права, применимые для объекта этого типа.

Для назначения права всем ролям в системе можно использовать специальное имя «роли»: PUBLIC.

Чтобы лишить пользователей прав, используется команда REVOKE:

```
REVOKE ALL ON accounts FROM PUBLIC;
```

Особые права владельца объекта (то есть права на выполнение DROP, GRANT, REVOKE и т. д.) всегда неявно закреплены за владельцем и их нельзя назначить или отобрать. Но владелец объекта может лишить себя обычных прав, например, разрешить всем, включая себя, только чтение таблицы.

Обычно распоряжаться правами может только владелец объекта (или суперпользователь). Однако возможно дать право доступа к объекту «с правом передачи», что позволит получившему такое право назначать его другим. Если такое право передачи впоследствии будет отозвано, то все, кто получил данное право доступа (непосредственно или по цепочке передачи), потеряют его.

Для того, чтобы создать нового пользователя используется команда: createuser.

```
Синтаксис: createuser [параметр-подключения...][параметр...][имя_пользователя]
```

Createuser — это обёртка для SQL-команды CREATE ROLE. Создание пользователей с её помощью по сути не отличается от выполнения того же действия при обращении к серверу другими способами.

Createuser принимает следующие аргументы:

имя_пользователя

Задаёт имя создаваемого пользователя PostgreSQL. Это имя должно отличаться от имён всех существующих ролей в данной инсталляции PostgreSQL.

-c номер
--connection-limit=номер

Устанавливает максимальное допустимое количество соединений для создаваемого пользователя. По умолчанию ограничение в количестве соединений отсутствует.

d
--createdb

Разрешает новому пользователю создавать базы данных.

-D
--no-createdb

Запрещает новому пользователю создавать базы данных. Это поведение по умолчанию.

-e
--echo

Вывести команды к серверу, генерируемые при выполнении createuser.

-E
--encrypted

Шифровать пароль пользователя, хранимый в базе. Если флаг не указан, то для пароля используется поведение по умолчанию.

-g role
--role=role

Указывает роль, к которой будет добавлена текущая роль в качестве члена группы. Допускается множественное использование флага -g.

-i
--inherit

Создаваемая роль автоматически унаследует права ролей, в которые она включается. Это поведение по умолчанию.

-I

--no-inherit

Роль не будет наследовать права ролей, в которые она включается.

-l

--login

Новый пользователь сможет подключаться к серверу (то есть его имя может быть идентификатором начального пользователя сеанса). Это свойство по умолчанию.

-L

--no-login

Новый пользователь не сможет подключаться к серверу.

-r

--createrole

Разрешает новому пользователю создавать другие роли, что означает наделение привилегией CREATEROLE.

-s

--superuser

Создаваемая роль будет иметь права суперпользователя.

-S

--no-superuser

Новый пользователь не будет суперпользователем. Это поведение по умолчанию.

--replication

Создаваемый пользователь будет наделён правом REPLICATION.

--no-replication

Создаваемый пользователь не будет иметь привилегии REPLICATION.

-?

--help

Вывести помощь по команде createuser.

createuser также принимает из командной строки параметры подключения:

-h сервер

--host=сервер

Указывает имя компьютера, на котором работает сервер. Если значение начинается с косой черты, оно определяет каталог Unix-сокета.

-p порт

--port=порт

Указывает TCP-порт или расширение файла локального Unix-сокета, через который сервер принимает подключения.

-w

--no-password

Не выдавать запрос на ввод пароля. Если сервер требует аутентификацию по паролю и пароль не доступен с помощью других средств, таких как файл .pgpass, попытка соединения не удастся.

-W

--password

Принудительно запрашивать пароль перед подключением к базе данных.

Чтобы создать роль joe на сервере, используемом по умолчанию:

```
$ createuser username
```

Чтобы создать роль username с правами суперпользователя и предустановленным паролем:

```
$ createuser -P -s -e username

Введите пароль для новой роли: xyzzy

Подтвердите ввод пароля: xyzzy

CREATE ROLE joe PASSWORD 'md5b5f5ba1a423792b526f799ae4eb3d59e' SUPERUSER CREA
TEDB CREATEROLE INHERIT LOGIN;
```

В приведённом примере введённый пароль отображается лишь для отражения сути, на деле же он не выводится на экран. Как видно в выводе журнала команд, пароль зашифрован. Если же указан флаг --unencrypted, то он отобразиться в этом журнале неизменным, а также, возможно, и в других журналах сервера. По этой причине в этой ситуации стоит использовать флаг -e с особой осторожностью.

GRANT

GRANT — определить права доступа

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER
}
    [, ...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] имя_таблицы [, ...]
         | ALL TABLES IN SCHEMA имя_схемы [, ...] }
    TO указание_роли [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | INSERT | UPDATE | REFERENCES } ( имя_столбца [, ...] )
    [, ...] | ALL [ PRIVILEGES ] ( имя_столбца [, ...] ) }
    ON [ TABLE ] имя_таблицы [, ...]
    TO указание_роли [, ...] [ WITH GRANT OPTION ]

GRANT { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { SEQUENCE имя_последовательности [, ...]
         | ALL SEQUENCES IN SCHEMA имя_схемы [, ...] }
    TO указание_роли [, ...] [ WITH GRANT OPTION
```

Команда GRANT имеет две основные разновидности: первая назначает права для доступа к объектам баз данных (таблицам, столбцам, представлениям, сторонним таблицам, последовательностям, базам данных, обёрткам сторонних данных, сторонним серверам, функциям, процедурным языкам, схемам или табличным пространствам), а вторая назначает одни роли членами других. Эти разновидности во многом похожи, но имеют достаточно отличий, чтобы рассматривать их отдельно.

GRANT ДЛЯ ОБЪЕКТОВ БАЗ ДАННЫХ

Эта разновидность команды GRANT даёт одной или нескольким ролям определённые права для доступа к объекту базы данных. Эти права добавляются к списку имеющихся, если роль уже наделена какими-то правами.

Также можно дать роли некоторое право для всех объектов одного типа в одной или нескольких схемах. Эта функциональность в настоящее время поддерживается только для таблиц, последовательностей и функций (но заметьте, что указание ALL TABLES распространяется также на представления и сторонние таблицы).

Ключевое слово PUBLIC означает, что права даются всем ролям, включая те, что могут быть созданы позже. PUBLIC можно воспринимать как неявно определённую группу, в которую входят все роли. Любая конкретная роль получит в сумме все права, данные непосредственно ей и ролям, членом которых она является, а также права, данные роли PUBLIC.

Если указано WITH GRANT OPTION, получатель права, в свою очередь, может давать его другим. Без этого указания распоряжаться своим правом он не сможет. Группе PUBLIC право передачи права дать нельзя.

Нет необходимости явно давать права для доступа к объекту его владельцу (обычно это пользователь, создавший объект), так как по умолчанию он имеет все права. (Однако владелец может лишить себя прав в целях безопасности.)

Право удалять объект или изменять его определение произвольным образом не считается назначаемым; оно неотъемлемо связано с владельцем, так что отозвать это право или дать его кому-то другому нельзя. (Однако похожий эффект можно получить, управляя членством в роли, владеющей объектом; см. ниже.) Владелец также неявно получает право распоряжения всеми правами для своего объекта.

PostgreSQL по умолчанию назначает группе PUBLIC определённые права для некоторых типов объектов. Для таблиц, столбцов, последовательностей, обёрток сторонних данных, сторонних серверов, больших объектов, схем или табличных пространств PUBLIC по умолчанию никаких прав не имеет.

Все возможные права перечислены ниже:

SELECT

Позволяет выполнять SELECT для любого столбца или перечисленных столбцов в заданной таблице, представлении или последовательности. Также позволяет выполнять COPY TO. Помимо того, это право требуется для обращения к существующим значениям столбцов в UPDATE или DELETE.

INSERT

Позволяет вставлять строки в заданную таблицу с помощью INSERT. Если право ограничивается несколькими столбцами, только их значение можно будет задать в команде INSERT (другие столбцы получат значения по умолчанию). Также позволяет выполнять COPY FROM.

UPDATE

Позволяет изменять (с помощью UPDATE) данные во всех, либо только перечисленных, столбцах в заданной таблице. (На практике для любой нетривиальной команды UPDATE потребуется и право SELECT, так как она

должна обратиться к столбцам таблицы, чтобы определить, какие строки подлежат изменению, и/или вычислить новые значения столбцов.)

DELETE

Позволяет удалять строки из заданной таблицы с помощью DELETE.

TRUNCATE

Позволяет опустошить заданную таблицу с помощью TRUNCATE (Команда TRUNCATE быстро удаляет все строки из набора таблиц. Она действует так же, как безусловная команда DELETE для каждой таблицы, но гораздо быстрее, так как она фактически не сканирует таблицы. Более того, она немедленно высвобождает дисковое пространство, так что выполнять операцию VACUUM после неё не требуется. Наиболее полезна она для больших таблиц.).

REFERENCES

Позволяет создавать ограничение внешнего ключа, ссылающееся на определённую таблицу либо на определённые столбцы таблицы.

TRIGGER

Позволяет создавать триггеры в заданной таблице.

CREATE

Для баз данных это право позволяет создавать схемы и публикации в заданной базе.

CONNECT

Позволяет пользователю подключаться к указанной базе данных. Это право проверяется при установлении соединения (в дополнение к условиям, определённым в конфигурации pg_hba.conf).

TEMP

Позволяет создавать временные таблицы в заданной базе данных.

EXECUTE

Позволяет выполнять заданную функцию и применять любые определённые поверх неё операторы. Это единственный тип прав, применимый к функциям.

ALL PRIVILEGES

Даёт целевой роли все права сразу. Ключевое слово PRIVILEGES является необязательным в PostgreSQL, хотя в строгом SQL оно требуется.

GRANT для ролей

Эта разновидность команды GRANT включает роль в члены одной или нескольких других ролей. Членство в ролях играет важную роль, так как права, данные роли, распространяются и на всех её членов.

С указанием WITH ADMIN OPTION новоиспечённый член роли сможет, в свою очередь, включать в члены этой роли, а также исключать из неё другие роли. Без этого указания обычные пользователи не могут это делать. Считается, что роль не имеет права WITH ADMIN OPTION для самой себя, но ей позволено управлять своими членами из сеанса, в котором пользователь сеанса соответствует данной роли. Суперпользователи баз данных могут включать или исключать любые роли из любых ролей. Роли с правом CREATEROLE могут управлять членством в любых ролях, кроме ролей суперпользователей.

В отличие от прав, членство в ролях нельзя назначить группе PUBLIC. Заметьте также, что эта форма команды не принимает избыточное слово GROUP.

REVOKE:

REVOKE — отозвать права доступа

Команда REVOKE лишает одну или несколько ролей прав, назначенных ранее. Ключевое слово PUBLIC обозначает неявно определённую группу всех ролей.

Любая конкретная роль получает в сумме права, данные непосредственно ей, права, данные любой роли, в которую она включена, а также права, данные группе PUBLIC. Поэтому, например, лишение PUBLIC права SELECT не обязательно будет означать, что все роли лишатся права SELECT для данного объекта: оно сохранится у тех ролей, которым оно дано непосредственно или косвенно, через другую роль. Подобным образом, лишение права SELECT какого-либо пользователя может не повлиять на его возможность пользоваться правом SELECT, если это право дано группе PUBLIC или другой роли, в которую он включён.

Если указано GRANT OPTION FOR, отзывается только право передачи права, но не само право. Без этого указания отзывается и право, и право распоряжаться им.

Если пользователь обладает правом с правом передачи, и он дал его другим пользователям, последнее право считается зависимым. Когда первый пользователь лишается самого права или права передачи и существуют зависимые права, эти

зависимые права также отзываются, если дополнительно указано CASCADE; в противном случае операция завершается ошибкой. Это рекурсивное лишение прав затрагивает только права, полученные через цепочку пользователей, которую можно проследить до пользователя, являющегося субъектом команды REVOKE. Таким образом, пользователи могут в итоге сохранить это право, если оно было также получено через других пользователей.

Когда отзывается право доступа к таблице, с ним вместе автоматически отзываются соответствующие права для каждого столбца таблицы (если такие права заданы). С другой стороны, если роли были даны права для таблицы, лишение роли таких же прав на уровне отдельных столбцов ни на что не влияет.

Замечания по совместимости, приведённые для команды GRANT, справедливы и для REVOKE. Стандарт требует обязательного указания ключевого слова RESTRICT или CASCADE, но PostgreSQL подразумевает RESTRICT по умолчанию.

Транзакции

Транзакции — это фундаментальное понятие во всех СУБД. Суть транзакции в том, что она объединяет последовательность действий в одну операцию "всё или ничего". Промежуточные состояния внутри последовательности не видны другим транзакциям, и если что-то помешает успешно завершить транзакцию, ни один из результатов этих действий не сохранится в базе данных.

Например, рассмотрим базу данных банка, в которой содержится информация о счетах клиентов, а также общие суммы по отделениям банка. Предположим, что мы хотим перевести 100 долларов со счёта Алисы на счёт Боба. Простоты ради, соответствующие SQL-команды можно записать так:

```
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';
UPDATE branches SET balance = balance - 100.00
    WHERE name = (SELECT branch_name FROM accounts WHERE name = 'Alice');
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';
UPDATE branches SET balance = balance + 100.00
    WHERE name = (SELECT branch_name FROM accounts WHERE name = 'Bob');
```

Точное содержание команд здесь не важно, важно лишь то, что для выполнения этой довольно простой операции потребовалось несколько отдельных действий. При этом с точки зрения банка необходимо, чтобы все эти действия

выполнились вместе, либо не выполнились совсем. Если Боб получит 100 долларов, но они не будут списаны со счёта Алисы, объяснить это сбоем системы определённо не удастся. И наоборот, Алиса вряд ли будет довольна, если она переведёт деньги, а до Боба они не дойдут. Нам нужна гарантия, что если что-то помешает выполнить операцию до конца, ни одно из действий не оставит следа в базе данных. И мы получаем эту гарантию, объединяя действия в одну транзакцию. Говорят, что транзакция атомарна: с точки зрения других транзакций она либо выполняется и фиксируется полностью, либо не фиксируется совсем.

Нам также нужна гарантия, что после завершения и подтверждения транзакции системой баз данных, её результаты в самом деле сохраняются и не будут потеряны, даже если вскоре произойдёт авария. Например, если мы списали сумму и выдали её Бобу, мы должны исключить возможность того, что сумма на его счёте восстановится, как только он выйдет за двери банка. Транзакционная база данных гарантирует, что все изменения записываются в постоянное хранилище (например, на диск) до того, как транзакция будет считаться завершённой.

Другая важная характеристика транзакционных баз данных тесно связана с атомарностью изменений: когда одновременно выполняется множество транзакций, каждая из них не видит незавершённые изменения, произведённые другими. Например, если одна транзакция подсчитывает баланс по отделениям, будет неправильно, если она посчитает расход в отделении Алисы, но не учтёт приход в отделении Боба, или наоборот. Поэтому свойство транзакций "всё или ничего" должно определять не только, как изменения сохраняются в базе данных, но и как они видны в процессе работы. Изменения, производимые открытой транзакцией, невидимы для других транзакций, пока она не будет завершена, а затем они становятся видны все сразу.

В PostgreSQL транзакция определяется набором SQL-команд, окружённым командами BEGIN и COMMIT. Таким образом, наша банковская транзакция должна была бы выглядеть так:

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';
COMMIT;
```

Если в процессе выполнения транзакции мы решим, что не хотим фиксировать её изменения (например, потому что оказалось, что баланс Алисы стал отрицательным), мы можем выполнить команду ROLLBACK вместо COMMIT, и все наши изменения будут отменены.

PostgreSQL на самом деле отрабатывает каждый SQL-оператор как транзакцию. Если вы не вставите команду BEGIN, то каждый отдельный оператор будет неявно окружён командами BEGIN и COMMIT (в случае успешного завершения). Группу операторов, окружённых командами BEGIN и COMMIT иногда называют блоком транзакции.

Операторами в транзакции можно также управлять на более детальном уровне, используя точки сохранения. Точки сохранения позволяют выборочно отменять некоторые части транзакции и фиксировать все остальные. Определив точку сохранения с помощью SAVEPOINT, при необходимости вы можете вернуться к ней с помощью команды ROLLBACK TO. Все изменения в базе данных, произошедшие после точки сохранения и до момента отката, отменяются, но изменения, произведённые ранее, сохраняются.

Всё это происходит в блоке транзакции, так что в других сеансах работы с базой данных этого не видно. Совершённые действия становятся видны для других сеансов все сразу, только когда вы фиксируете транзакцию, а отменённые действия не видны вообще никогда.

Вернувшись к банковской базе данных, предположим, что мы списываем 100 долларов со счёта Алисы, добавляем их на счёт Боба, и вдруг оказывается, что деньги нужно было перевести Уолли. В данном случае мы можем применить точки сохранения:

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';
SAVEPOINT my_savepoint;
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';
-- ошибочное действие... забыть его и использовать счёт Уолли
ROLLBACK TO my_savepoint;
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Wally';
COMMIT;
```

ROLLBACK TO — это единственный способ вернуть контроль над блоком транзакций, оказавшимся в прерванном состоянии из-за ошибки системы, не считая возможности полностью отменить её и начать снова. [1]

Вывод: В данном методическом пособии рассмотрены понятия прав доступа и транзакции. Приведены примеры с описанием переменных и частями кода.

Контрольные вопросы:

1 В чем отличие обычного пользователя от суперпользователя?

2 Для чего нужны транзакции и точки сохранения в них?

3 Что такое права доступа и как их назначить и(или) отозвать?

Задания для выполнения:

1 Создать новую учетную запись с одним параметром-подключения и тремя параметрами.

2 Определить права доступа для нового пользователя.

3 Отозвать права доступа для нового пользователя.

4 Произвести транзакцию как при пополнении счёта мобильного телефона(т.е. снятие определённой суммы и перевод на другой счёт).

Источники:

1. Postgres Pro Standart: Документация . https://postgrespro.ru/docs/postgrespro