САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 4
Управление привилегиями

Агафонов Антон Александрович
к.т.н., доцент кафедры ГИиИБ

Самара

План лекции

- MySQL
  - Привилегии
  - Роли
- PostgreSQL
  - Привилегии
  - Роли
- MongoDB
  - Роли

Привилегии

Привилегия – разрешение на использование определенной услуги управления данными для доступа к объекту данных, предоставляемое идентифицированному пользователю.

Роль – это именованная совокупность привилегий, которые могут быть предоставлены пользователям или другим ролям.

Ролевая модель управления доступом — развитие политики избирательного управления доступом, при этом права доступа (привилегии) субъектов системы на объекты группируются с учётом специфики их применения, образуя роли.

- Один субъект может иметь несколько ролей.
- Одну роль могут иметь несколько субъектов.
- Одна роль может иметь несколько разрешений.
- Одно разрешение может принадлежать нескольким ролям.

Управление привилегиями в SQL

Предоставление привилегий:

```
GRANT {привилегия на объект [,…] | имя роли [,…]}
ON имя объекта
TO {получатель привилегии [,…]}
[WITH GRANT OPTION | WITH ADMIN OPTION]
```

Отзыв привилегий:

```
REVOKE [GRANT OPTION FOR] {привилегия на объект [,…] | имя роли [,…]}
ON имя объекта
FROM {получатель привилегии [,…]}
[RESTRICT | CASCADE]
```

Управление привилегиями в SQL

Привилегия	Описание	Применимо к объектам
ALL PRIVILEGIES	Назначить все привилегии	Ко всем объектам
SELECT | INSERT | UPDATE | DELETE	Право на просмотр, вставку, редактирование и удаление данных в таблице (столбце)	Таблицы, столбцы, представления (только SELECT)
REFERENCES	Право управления ограничением внешнего ключа (FOREIGIN KEY), право использовать столбцы в любом ограничении	Таблицы и столбцы
USAGE	Дает право использовать данный объект для определения другого объекта	Домены, пользовательские типы данных, наборы символов, порядки сравнения и сортировки, трансляции
UNDER	Право на создание подтипов или объектных таблиц	Структурные типы
TRIGGER	Право на создание триггера	Таблицы
EXECUTE	Запуск на выполнение	Хранимые процедуры и функции

Привилегии MySQL различаются в зависимости от контекста, в котором они применяются, и уровня их работы:

- Административные привилегии позволяют пользователям управлять работой сервера MySQL. Эти привилегии являются глобальными, потому что они не связаны с конкретной БД.
- Привилегии базы данных применяются к БД и ко всем объектам в ней. Эти привилегии могут быть предоставлены для определенных баз данных или глобально для всех.
- Привилегии для объектов базы данных, таких как таблицы, индексы, представления и хранимые процедуры, могут предоставляться для конкретных объектов в базе данных, для всех объектов данного типа в базе данных (например, для всех таблиц в базе данных) или глобально для всех объектов данного типа во всех базах данных.
- Привилегии столбцов обращаются к одиночным столбцам в указанной таблице.

Для просмотра привилегий пользователя можно выполнить команду

```
SHOW GRANTS FOR {CURRENT_USER | имя_пользователя};
```

MySQL. Глобальные привилегии

Глобальные привилегии применяются ко всем базам данных на данном сервере.

Административные привилегии, такие как `CREATE USER`, `FILE`, `PROCESS`, `RELOAD`, `REPLICATION CLIENT`, `REPLICATION SLAVE`, `SHOW DATABASES`, `SHUTDOWN` и `SUPER`, могут предоставляться только глобально. Другие привилегии могут быть предоставлены глобально или на более определенных уровнях.

Чтобы назначить глобальные привилегии, используется синтаксис `ON *.*`:

Создадим пользователя:

```
CREATE USER 'user'@'localhost' IDENTIFIED BY 'P@ssw0rd';
```

Назначим права:

```
GRANT ALL ON *.* TO 'user'@'localhost';

GRANT SELECT, INSERT ON *.* TO 'user'@'localhost';
```

MySQL. Привилегии на уровне БД

Привилегии базы данных применяются ко всем объектам в данной базе данных.

Для назначения привилегии на уровне базы данных используется синтаксис `ON db_name.*`:

```
GRANT ALL ON lectures.* TO 'user'@'localhost';

GRANT SELECT, INSERT ON lectures.* TO 'user'@'localhost';
```

Привилегии `CREATE`, `DROP`, `EVENT`, `LOCK TABLES` и `REFERENCES` можно указать на уровне базы данных.

Привилегии для таблицы или хранимой процедуры/функции также могут быть указаны на уровне базы данных. В этом случае они применяются ко всем таблицам или хранимым процедурам/функциям в базе данных.

MySQL. Привилегии на уровне таблицы

Привилегии для таблицы применяются ко всем столбцам в данной таблице.

Для назначения привилегии на уровне таблицы используется синтаксис `ON db_name.tbl_name`:

```
GRANT ALL ON lectures.movies TO 'user'@'localhost';

GRANT SELECT, INSERT ON lectures.movies TO 'user'@'localhost';
```

Допустимые значения привилегий на уровне таблицы: `ALTER`, `CREATE VIEW`, `CREATE`, `DELETE`, `DROP`, `GRANT OPTION`, `INDEX`, `INSERT`, `REFERENCES`, `SELECT`, `SHOW VIEW`, `TRIGGER` и `UPDATE`.

Привилегии уровня таблицы применяются к базовым таблицам и представлениям. Они не применяются к таблицам, созданным с помощью `CREATE TEMPORARY TABLE`, даже если имена таблиц совпадают.

MySQL. Привилегии на уровне столбцов

Для назначения привилегий уровня столбцов для конкретной таблицы указываются конкретные столбцы:

```
GRANT SELECT (data) ON lectures.movies TO 'user'@'localhost';
```

Допустимыми привилегиями на уровне столбца являются `INSERT`, `REFERENCES`, `SELECT` и `UPDATE`.

MySQL. Роли

Роли в MySQL не ограничиваются определением именованного набора привилегий, это по сути те же учетные записи, которые могут быть присвоены другим учетным записям. В связи с этим именование ролей, как и пользователей, подразумевает указание имени и хоста.

Однако есть два основных отличия: во-первых, имя роли не может быть пустым, т.е. не бывает анонимных ролей, и во-вторых, если имя хоста для роли не указано, то оно интерпретируется как `'%'`. По этой причине имена ролей часто задаются, используя только часть имени пользователя без указания имени хоста.

В MySQL существует такое понятие, как **обязательные роли** (mandatory roles). Сервер интерпретирует обязательную роль как роль, предоставляемую всем пользователям, т.е. без необходимости её явного предоставления какой-либо учетной записи. Указание обязательных ролей выполняется с помощью их перечисления в значении системной переменной `mandatory_roles` в конфигурационном файле сервера.

MySQL. Управление ролями

Создание роли:

```
CREATE ROLE [IF NOT EXISTS] role;
```

Удаление роли:

```
DROP ROLE [IF EXISTS] role;
```

Просмотр активных ролей в текущем сеансе:

```
SELECT CURRENT_ROLE();
```

Установка активной роли в текущем сеансе:

```
SET ROLE {DEFAULT | NONE | ALL | ALL EXCEPT role [, role] | role [, role] }
```

Установка активной роли по умолчанию:

```
SET DEFAULT ROLE {NONE | ALL | role [, role] } TO user [, user]
```

MySQL. Управление ролями. Примеры

Создание ролей

```
CREATE ROLE 'app_developer', 'app_read', 'app_write';
```

Назначение привилегий ролям

```
GRANT ALL ON lectures.* TO 'app_developer';
GRANT SELECT ON lectures.* TO 'app_read';
GRANT INSERT, UPDATE, DELETE ON lectures.* TO 'app_write';
```

Предоставление ролей пользователям

```
GRANT 'app_developer' TO 'agafonov'@'localhost’;

REVOKE ALL ON *.* FROM 'user'@'localhost';
GRANT 'app_read', 'app_write' TO 'user'@'localhost';
```

MySQL. Управление ролями. Примеры

Просмотр назначенных привилегий

```
SHOW GRANTS FOR CURRENT_USER;
```

	Grants for user@localhost
	GRANT USAGE ON *.* TO `user`@`localhost`
	GRANT `app_read`@`%`,`app_write`@`%` TO `user`@`localhost`

Просмотр активных ролей

```
SELECT CURRENT_ROLE();
```

	CURRENT_ROLE()
	NONE

MySQL. Управление ролями. Примеры

Выбор активных ролей

```
SET ROLE ALL EXCEPT 'app_write';
SELECT CURRENT_ROLE();
```

	CURRENT_ROLE()
	`app_read`@`%`

Установка ролей по умолчанию и активация их в сеансе пользователя

```
SET DEFAULT ROLE ALL TO 'user'@'localhost';
SET ROLE DEFAULT; -- или переподключиться
SELECT CURRENT_ROLE();
```

	CURRENT_ROLE()
	`app_read`@`%`,`app_write`@`%`

MySQL. Фиксированные роли

Роль	Описание
DBA	Разрешено выполнять любые действия на сервере.
MaintenanceAdmin	Предоставляет права на администрирование сервера.
ProcessAdmin	Предоставляет права на доступ, мониторинг и завершение клиентских процессов.
UserAdmin	Предоставляет права на создание пользователей и сброс паролей.
SecurityAdmin	Предоставляет права на управления учетными записями, а также предоставление и отзыв серверных привилегий.
MonitorAdmin	Предоставляет права на для мониторинга сервера.
DBManager	Предоставляет полные права на все базы данных.
DBDesigner	Предоставляет права на создание и модификацию схемы любой базы данных.
ReplicationAdmin	Предоставляет права на настройку и управление репликацией данных.
BackupAdmin	Предоставляет минимальные права, необходимые для выполнения резервного копирования любой базы данных.

PostgreSQL. Привилегии

В отличие от MySQL, в PostgreSQL существует понятие «**владельца**» объекта.

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 выполнять с объектом любые операции. Чтобы разрешить использовать его другим ролям, нужно назначить им привилегии.

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 имя_таблицы OWNER TO новый_владелец;
```

PostgreSQL. Управление ролями

Создание роли:

```
CREATE ROLE имя [ [ WITH ] параметр [ ... ] ]
```

Изменение роли:

```
ALTER ROLE имя [ WITH ] параметр [ ... ];
```

Удаление роли:

```
DROP ROLE [IF EXISTS] имя;
```

Просмотр текущей роли и сессионной роли:

```
SELECT CURRENT_ROLE, SESSION_USER;
```

Установка активной роли в текущем сеансе:

```
SET ROLE role | NONE
RESET ROLE
```

PostgreSQL. Управление ролями

Роль также можно рассматривать как группу пользователей, членство в этой группе выдаётся ролям индивидуальных пользователей, права назначаются для всей группы.

Для настройки групповой роли сначала нужно создать саму роль:

```
CREATE ROLE групповая_роль;
```

После того, как групповая роль создана, в неё можно добавлять или удалять членов группы:

```
GRANT групповая_роль TO роль1, …;
REVOKE групповая_роль FROM роль1, …;
```

Роли, имеющие атрибут INHERIT, автоматически используют права всех ролей, членами которых они являются, в том числе и унаследованные этими ролями права.

PostgreSQL. Назначение привилегий

Привилегии доступа к объектам базы данных для ролей также определяются с помощью операторов GRANT и REVOKE. Синтаксис этих операторов позволяет указывать как отдельные, так и все возможные привилегии, как на отдельные объекты, так и на группы объектов и т.д.

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO app_role;

GRANT SELECT(first_name, last_name) ON actor TO app_role;

GRANT INSERT, UPDATE, DELETE ON actor TO app_role;
```

Доступ роли к объектам базы данных определяется выданными привилегиями, но есть два исключения:

- Суперпользователи - для них проверки разграничения доступа не выполняются, они имеют доступ ко всем объектам.
- Владельцы объектов - они сразу получают полный набор привилегий для этого объекта, но эти привилегии могут быть отозваны.

PostgreSQL. Управление ролями. Примеры

Создание ролей

```
CREATE ROLE app_user WITH LOGIN PASSWORD '1';
CREATE ROLE app_developer;
CREATE ROLE app_read;
CREATE ROLE app_write;
CREATE ROLE app_read_write NOINHERIT;
```

Назначение привилегий ролям

```
GRANT SELECT ON film TO app_developer;
GRANT SELECT(first_name, last_name) ON actor TO app_read;
GRANT INSERT, UPDATE, DELETE ON actor TO app_write;
```

PostgreSQL. Управление ролями. Примеры

Предоставление ролей ролям

```
GRANT app_developer TO app_user;
GRANT app_read, app_write TO app_read_write;
GRANT app_read_write TO app_user;
```

Проверка доступных привилегий (подключение от имени app_user)

```
SELECT CURRENT_ROLE, SESSION_USER;
```

	current_role	session_user
	app_user	app_user

```
SELECT title, release_year FROM film LIMIT 2;
```

	title	release_year
	ACADEMY DINOSAUR	2006
	ACE GOLDFINGER	2006

Проверка доступных ролей (подключение от имени app_user)

```
SELECT first_name, last_name FROM actor LIMIT 2;
```

ERROR: ОШИБКА: нет доступа к таблице actor

```
SET ROLE app_read_write;
SELECT first_name, last_name FROM actor LIMIT 2;
```

ERROR: ОШИБКА: нет доступа к таблице actor

```
SET ROLE app_read;
SELECT CURRENT_ROLE, SESSION_USER;
```

	current_role	session_user
	app_read	app_user

```
SELECT first_name, last_name FROM actor LIMIT 2;
```

	first_name	last_name
	PENELOPE	GUINESS
	NICK	WAHLBERG

PostgreSQL. Фиксированные роли

Роль	Разрешаемый доступ
pg_read_all_data	Читать все данные (таблицы, представления, последовательности), как будто роль имеет права SELECT на эти объекты и права USAGE на все схемы, но при этом явным образом такие права ей не назначены.
pg_write_all_data	Записывать все данные (таблицы, представления, последовательности), как будто роль имеет права INSERT, UPDATE и DELETE на эти объекты и права USAGE на все схемы, но при этом явным образом такие права ей не назначены.
pg_read_all_settings	Читать все конфигурационные переменные, даже те, что обычно видны только суперпользователям.
pg_read_all_stats	Читать все представления pg_stat_* и использовать различные расширения, связанные со статистикой.
pg_database_owner	Никакого. Неявным образом включает в себя владельца текущей базы данных.

```
GRANT pg_read_all_data TO app_user;
```

PostgreSQL. Политики защиты строк

В дополнение к стандартной системе прав SQL, управляемой командой GRANT, на уровне таблиц можно определить политики защиты строк (RLS, Row-Level Security), ограничивающие для пользователей наборы строк, которые могут быть возвращены или модифицированы обычными запросами.

Команда включения защиты строк для таблицы:

```
ALTER TABLE таблица ENABLE ROW LEVEL SECURITY;
```

Если политика для таблицы не определена, применяется политика запрета по умолчанию, так что никакие строки в этой таблице нельзя увидеть или модифицировать. При создании политики можно также задать, для каких ролей она будет работать (по умолчанию — для всех) и для каких операторов (по умолчанию — также для всех).

Суперпользователи и роли с атрибутом BYPASSRLS всегда обращаются к таблице, минуя систему защиты строк.

PostgreSQL. Политики защиты строк

Оператор CREATE POLICY позволяет определить политику защиты строк для указанной таблицы:

```
CREATE POLICY имя ON имя таблицы
[ FOR { ALL | SELECT | INSERT | UPDATE | DELETE } ]
[ TO { имя роли | PUBLIC| CURRENT_USER } [,…] ]
[ USING ( выражение USING ) ]
[ WITH CHECK ( выражение WITH CHECK) ]
```

Для строк определяются предикаты (выражения логического типа), существующие строки таблицы проверяются по выражению, указанному в USING, тогда как строки, которые могут быть созданы командами INSERT или UPDATE проверяются по выражению, указанному в WITH CHECK.

PostgreSQL. Удаление ролей

Так как роли могут владеть объектами баз данных и иметь права доступа к объектам других ролей, удаление роли не сводится к немедленному выполнению **DROP ROLE**. Сначала должны быть удалены или переданы другим владельцам все объекты, принадлежащие роли; также должны быть отозваны все права, данные роли.

Владение объектами можно передавать в индивидуальном порядке:

```
ALTER TABLE название таблицы OWNER TO новая роль
```

Для переназначения какой-либо другой роли владения сразу всеми объектами, принадлежащих удаляемой роли, можно применить команду **REASSIGN OWNED**.

Объекты, принадлежащие удаляемой роли, можно удалить с помощью команды **DROP OWNED**. Эта команда также удаляет все права, которые даны целевой роли для объектов, не принадлежащих ей.

При попытке выполнить **DROP ROLE** для роли, у которой сохраняются зависимые объекты, будут выданы сообщения, говорящие, какие объекты нужно передать другому владельцу или удалить.

Роль предоставляет привилегии для выполнения указанных действий над ресурсом. Каждая привилегия либо явно указывается в роли, либо наследуется от другой роли, либо и то, и другое.

Привилегия состоит из указанного ресурса и действий, разрешенных для ресурса.

Ресурс — это база данных, коллекция, набор коллекций или кластер. Если ресурсом является кластер, выполняемые действия влияют на состояние системы, а не на конкретную базу данных или коллекцию.

Роль может включать в свое определение одну или несколько существующих ролей, и в этом случае роль наследует все привилегии включенных ролей.

Для просмотра привилегий для роли можно выполнить команду:

```
db.runCommand(
    {
      rolesInfo : { role: <name>, db: <db> },
      showPrivileges: true,
      showBuiltinRoles: true
    }
)
```

MongoDB. Встроенные роли пользователя БД

Роль	Описание
`read`	Предоставляет возможность чтения данных для всех несистемных коллекций и коллекции `system.js`. Роль предоставляет доступ для чтения, разрешая выполнять следующие действия: `find, dbStats, dbHash, listCollections, listIndexes` и др.
`readWrite`	Предоставляет все привилегии роли `read`, а также возможность изменять данные во всех несистемных коллекциях и коллекции `system.js`. Роль позволяет выполнять следующие действия над этими коллекциями: `createCollection, dropCollection, createIndex, dropIndex, insert, update, remove` и др.

MongoDB. Встроенные роли администратора БД

Роль	Описание
dbAdmin	Предоставляет возможность выполнять административные задачи: задачи, связанные с управлением схемой, индексированием и сбором статистики. Эта роль не предоставляет привилегий для управления пользователями и ролями.
dbOwner	Владелец базы данных может выполнять любые административные действия с базой данных. Эта роль сочетает в себе привилегии, предоставляемые ролями readWrite, dbAdmin и userAdmin.
userAdmin	Предоставляет возможность создавать и изменять роли и пользователей в текущей базе данных. Поскольку роль userAdmin позволяет пользователям предоставлять любые привилегии любому пользователю, включая себя, эта роль также косвенно предоставляет доступ суперпользователя либо к базе данных, либо, если она ограничена базой данных администратора, к кластеру.

MongoDB. Встроенные роли администрирования кластера

База данных `admin` включает роли для администрирования всей системы, а не только одной базы данных.

Роль	Описание
`clusterAdmin`	Обеспечивает максимальный доступ к управлению кластером. Эта роль объединяет в себе привилегии, предоставляемые ролями `clusterManager`, `clusterMonitor` и `hostManager`.
`clusterManager`	Обеспечивает действия по управлению и мониторингу в кластере.
`clusterMonitor`	Предоставляет доступ только для чтения к инструментам мониторинга.
`hostManager`	Предоставляет возможность мониторинга и управления серверами.
`backup`	Предоставляет минимальные привилегии, необходимые для резервного копирования данных.
`restore`	Предоставляет необходимые привилегии для восстановления данных из резервных копий.

MongoDB. Определяемые пользователем роли

Оператор	Описание
`db.createRole()`	Создает роль и указывает ее привилегии.
`db.dropRole()`	Удаляет пользовательскую роль.
`db.dropAllRoles()`	Удаляет все пользовательские роли, связанные с базой данных.
`db.getRole()`	Возвращает информацию об указанной роли.
`db.getRoles()`	Возвращает информацию обо всех пользовательских ролях в базе данных.
`db.grantPrivilegesToRole()`	Назначает привилегии пользовательской роли.
`db.revokePrivilegesFromRole()`	Удаляет указанные привилегии из пользовательской роли.
`db.grantRolesToRole()`	Указывает роли, от которых пользовательская роль наследует привилегии.
`db.revokeRolesFromRole()`	Удаляет унаследованные роли из роли.
`db.updateRole()`	Обновляет пользовательскую роль.

MongoDB . Назначение ролей. Пример

Создание пользователя

```
use db_lectures
db.createUser( {
    user: "lecturesReader",
    pwd: passwordPrompt(), // или указать пароль
    roles: []
} )

db.auth("lecturesReader", passwordPrompt())
```

Проверка чтения из БД db_lectures

```
db.inventory.find( ).count()
```

MongoServerError: not authorized on db_lectures to execute command

MongoDB . Назначение ролей. Пример

Назначение роли

```
use admin
db.auth("myUserAdmin", passwordPrompt())

use db_lectures
db.grantRolesToUser(
    "lecturesReader",
    [ { role: "read", db: "db_lectures" } ]
)

db.auth("lecturesReader", passwordPrompt())
```

Проверка чтения из БД db_lectures

```
db.inventory.find( ).count()

11
```

Управление доступом и привилегиями. Итоги

- Модели управления доступом: дискреционная, мандатная, ролевая.
- Идентификация / аутентификация / авторизация.
- Аутентификация в СУБД: имя пользователя + пароль, внешние системы аутентификации (LDAP, Kerberos).
- Привилегии / роли.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
к.т.н., доцент кафедры ГИиИБ