Минобрнауки России

федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный технологический институт
(технический университет)»

Кафедра систем автоматизированного проектирования и управления

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к практической работе № 7 по курсу

БАЗЫ ДАННЫХ

ст. преподаватель

Иванов А.А.

2023

Практическая работа № 7

Часть 1. Работа с базами данных MySQL.

MySQL представляет собой СУБД, обеспечивающую создание информационных систем с архитектурой «клиент-сервер», в которой он играет роль сервера базы данных. MySQL удовлетворяет требованиям, предъявляемым к системам распределенной обработки информации. Эта СУБД поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных на недорогих аппаратных платформах, отличается простатой управления и использования, а также обеспечивает тесную интеграцию баз данных в Web.

Цель работы:

— изучить основные принципы работы с базами данных MySQL (создание базы данных, импорт данных, резервное копирование, восстановление);
— изучить работу с запросами в базе данных MySQL;
— изучить обращение к базам данных MySQL из клиентских программ (MS Access).

Порядок выполнения работы

1.Конвертация базы данных MS Access в MySQL.

Для создания объектов базы данных в MySQL необходимо провести конвертацию модели базы данных разработанной для MS Access в модель базы данных MySQL. Для этого необходимо открыть модель базы данных в Toad Data Modeler, выбрать в меню «Model» пункт «Database Conversion» и в выпадающем списке «Convert to» указать СУБД, в которую будет конвертирована модель базы данных. Затем необходимо сгенерировать скрипт для создания объектов базы данных.

2.Создание базы данных.

Необходимо отметить, что скрипт не создает саму базу данных. Поэтому для создания самой базы данных в начало скрипта необходимо добавить следующую последовательность команд:

```
CREATE DATABASE имя_базы_данных;
USE имя_базы_данных;
```

Где USE – команда для выбора текущей базы данных,
CREATE DATABASE – команда создания новой базы данных;

Затем откройте MySQL Workbench, в случае необходимости создайте новое соединение с сервером mysql, используя учетные данные root@root, откройте созданный скрипт через меню File -> Open SQL Script… и выполните его.

3.Подключение базы данных к MS Access.

Создайте копию файла базы данных MS Access и откройте ее. Откройте вкладку внешние данные и выберите в ней кнопку «Базы данных ODBC», а в следующем открывшемся окне укажите «Создать связанную таблицу для связи с источником данных». В следующем окне на вкладке «Источники данных компьютера» нажмите кнопку «Создать…». Скорее всего Вам будет предложено создать пользовательский источник данных. Нажмите кнопку «Далее > » и выберите драйвер, для которого задается источник. Еще раз нажмите «Далее > », а затем «Готово».

В открывшемся окне MySQL Connector укажите учетные данные сервера, к которому будете подключаться, а также выберите созданную Вами базу данных. Нажмите «ОК», в открывшемся окне выделите только пользовательские таблицы, нажмите еще раз кнопку «ОК».

4.Перенос данных из MS Access в MySQL.

Выполните перенос данных из нескольких таблиц MS Access в соответствующие таблицы MySQL с помощью запросов на добавление (INSERT … SELECT …).

5.Выполнение запросов в MySQL

Для выполнения запросов создайте новый запрос в MySQL Workbench. Убедитесь, что запрос будет обращаться к нужной базе данных. Введите (скопируйте в окно из MS Access) текст запроса. Выполните запрос.

6.Создание резервной копии базы данных.

Для создания резервной копии базы данных выберите в Навигаторе пункт «Data Export» укажите соответствующую базу данных (Schema) и таблицы и выполните экспорт данных. После создания резервной копии измените/удалите какие-либо данные в базе данных и перейдите к восстановлению базы данных.

7.Восстановление базы данных из резервной копии.

Выполните восстановление данных из резервной копии с помощью команды «Data Import/Restore». Покажите, что данные восстановились в полном объеме.

Содержание 1 части отчета:

- листинг SQL-скриптов – создания базы данных из Toad Data Modeler и резервной копии базы данных;
- скриншот вкладки «Поля» одной из таблиц базы данных MySQL;
- скриншоты результата выполнения запроса;
- скриншот вкладки таблицы главного окна базы данных MS Access;
- выводы по лабораторной работе о соответствии типов данных, а также различии технологий хранения и использования данных в MS Access и MySQL.

Часть 2. Администрирование сервера MySQL.

Цели работы:

- приобретение навыков администрирования сервера MySQL;
- приобретение навыков администрирования баз данных сервера MySQL;
- освоение графических утилит MySQL.

Теоретическая часть

1. Управление учетными записями пользователей сервера MySQL.

1.1. Синтаксис команд GRANT и REVOKE

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    TO  user_name [IDENTIFIED BY [PASSWORD] 'password']
      [,user_name [IDENTIFIED BY [PASSWORD] 'password'] ...]
    [WITH GRANT OPTION]

REVOKE priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    FROM user_name [, user_name ...]
```

Команды GRANT и REVOKE позволяют системным администраторам создавать пользователей MySQL, а также предоставлять права пользователям или лишать их прав на четырех уровнях привилегий:

Глобальный уровень
Глобальные привилегии применяются ко всем базам данных на указанном сервере. Эти привилегии хранятся в таблице **mysql.user.**

Уровень базы данных
Привилегии базы данных применяются ко всем таблицам указанной базы данных. Эти привилегии хранятся в таблицах **mysql.db** и **mysql.host.**

Уровень таблицы
Привилегии таблицы применяются ко всем столбцам указанной таблицы. Эти привилегии хранятся в таблице **mysql.tables_priv.**

Уровень столбца
Привилегии столбца применяются к отдельным столбцам указанной таблицы. Эти привилегии хранятся в таблице **mysql.columns_priv.**

Если привилегии предоставляются *пользователю, которого не существует*, то этот пользователь *создается*.

В таблице приведен список возможных значений параметра `priv_type`:

ALL [PRIVILEGES]	Задает все простые привилегии, кроме WITH GRANT OPTION
ALTER	Разрешает использование ALTER TABLE
CREATE	Разрешает использование CREATE TABLE
CREATE TEMPORARY TABLES	Разрешает использование CREATE TEMPORARY TABLE

DELETE	Разрешает использование DELETE
DROP	Разрешает использование DROP TABLE
EXECUTE	Разрешает пользователю запускать хранимые процедуры (для MySQL 5.0)
FILE	Разрешает использование SELECT ... INTO OUTFILE и LOAD DATA INFILE
INDEX	Разрешает использование CREATE INDEX and DROP INDEX
INSERT	Разрешает использование INSERT
LOCK TABLES	Разрешает использование LOCK TABLES на таблицах, для которых есть привилегия SELECT
PROCESS	Разрешает использование SHOW FULL PROCESSLIST
REFERENCES	Зарезервировано для использования в будущем
RELOAD	Разрешает использование FLUSH
REPLICATION CLIENT	Предоставляет пользователю право запрашивать местонахождение головного и подчиненных серверов
REPLICATION SLAVE	Необходимо для подчиненных серверов при репликации (для чтения информации из бинарных журналов головного сервера)
SELECT	Разрешает использование SELECT
SHOW DATABASES	SHOW DATABASES выводит все базы данных
SHUTDOWN	Разрешает использование mysqladmin shutdown
SUPER	Позволяет установить одно соединение (один раз), даже если достигнуто значение max_connections, и запускать команды CHANGE MASTER, KILL thread,mysqladmin debug, PURGE MASTER LOGS и SET GLOBAL
UPDATE	Разрешает использование UPDATE
USAGE	Синоним для "без привилегий"
GRANT OPTION	Синоним для WITH GRANT OPTION

Значение USAGE можно задавать, если необходимо создать пользователя без привилегий.

Для ***таблицы*** можно указать только следующие значения **priv_type**:
SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX и ALTER.

Для ***столбца*** можно указать только следующие значения **priv_type** (при использовании оператора **column_list**): **SELECT, INSERT и UPDATE.**

Глобальные привилегии можно задать, воспользовавшись синтаксисом ON *.*,
а ***привилегии базы данных*** - при помощи синтаксиса **ON `db_name.*`.**

Если указать **ON *** *при открытой текущей базе данных*, то привилегии будут заданы для этой базы данных. (**Предупреждение**: если указать **ON *** при **отсутствии** открытой текущей базы данных, это повлияет на глобальные привилегии!)

Примечание. В операторе GRANT шаблонные символы **`'%'`** не допускаются в определении имени баз данных.

С тем, чтобы можно было определять права пользователям с конкретных компьютеров, в MySQL обеспечивается возможность указывать имя пользователя (`user_name`) в форме user@host. Если необходимо указать строку user, в которой содержатся специальные символы (такие как `'-'`) или строку `host`, в которой содержатся специальные или групповые символы (такие как `'%'`), можно заключить имя удаленного компьютера или пользователя в кавычки (например, **`'test-user'@'test-hostname'`**).

В имени удаленного компьютера также можно указывать групповые символы. Например, `user@'%.loc.gov'` относится к user всех удаленных компьютеров домена `loc.gov`, а `user@'144.155.166.%'` относится к user всех удаленных компьютеров подсети 144.155.166.

Простая форма **user** является синонимом для **user@"%" .**

В MySQL *не поддерживаются групповые символы в именах пользователей*. Анонимные пользователи определяются вставкой записей `User=""` в таблицу `mysql.user` или созданием пользователя с пустым именем при помощи команды GRANT.

Примечание: если анонимным пользователям разрешается подсоединяться к серверу MySQL, необходимо также предоставить привилегии всем локальным пользователям как `user@localhost`, поскольку в противном случае при попытке пользователя зайти в MySQL с локального компьютера в таблице `mysql.user` будет использоваться вход для анонимного пользователя!

Чтобы проверить, происходит ли подобное на вашем компьютере, выполните следующий запрос:

```
mysql> SELECT Host,User FROM mysql.user WHERE User='';
```

Привилегии для таблицы или столбца формируются при помощи логического оператора OR из привилегий каждого из четырех уровней. Например, если в таблице **mysql.user** указано, что у пользователя есть глобальная привилегия SELECT, эта привилегия не отменяется на уровне базы данных, таблицы или столбца.

Привилегии для столбца могут быть вычислены следующим образом:

глобальные привилегии
OR (привилегии базы данных AND привилегии удаленного компьютера)
OR привилегии таблицы
OR привилегии столбца

В большинстве случаев права пользователя определяются только на одном уровне привилегий, поэтому обычно эта процедура не настолько сложна, как описано выше.

Если привилегии предоставляются сочетанию пользователь/удаленный компьютер, которое отсутствует в таблице **mysql.user**, то в последнюю добавляется запись, которая остается в таблице до тех пор, пока не будет удалена при помощи команды DELETE. Иначе говоря, *команда **GRANT** может создавать записи **user** в таблице, но команда REVOKE не может их удалить. Это необходимо делать при помощи команды **DELETE**.*

Если в MySQL создан новый пользователь или предоставлены глобальные привилегии, пароль пользователя будет назначаться оператором **IDENTIFIED BY**, если он указан. Если у пользователя уже есть пароль, то этот пароль будет заменен новым.

Предупреждение: если при создании нового пользователя не указать оператор IDENTIFIED BY, будет создан пользователь без пароля. Это ненадежно с точки зрения безопасности.

Пароли также можно задавать при помощи команды **SET PASSWORD.**

Если у пользователя нет никаких привилегий для таблицы, то таблица не отображается, когда пользователь запрашивает список таблиц (например, при помощи оператора SHOW TABLES).

Оператор **WITH GRANT OPTION** предоставляет пользователю возможность наделять других пользователей любыми привилегиями, которые он сам имеет на указанном уровне привилегий. При предоставлении привилегии GRANT необходимо проявлять осмотрительность, так как *два пользователя с разными привилегиями могут объединить свои привилегии*!

Нельзя предоставить другому пользователю привилегию, которой нет у вас самого. Привилегия GRANT позволяет предоставлять только те привилегии, которыми вы обладаете.

Учтите, что если пользователю назначена привилегия **GRANT** на определенном уровне привилегий, то все привилегии, которыми этот пользователь уже обладает (или которые будут ему назначены в будущем!) на этом уровне, также могут назначаться этим пользователем. Предположим, пользователю назначена привилегия **INSERT** в базе данных. Если потом в базе данных назначить привилегию **SELECT** и указать **WITH GRANT OPTION**, пользователь сможет назначать не только привилегию **SELECT**, но также и **INSERT**. Если затем в базе данных предоставить пользователю привилегию **UPDATE**, пользователь сможет после этого назначать **INSERT**, **SELECT** и **UPDATE.**

Не следует назначать привилегии **ALTER** *обычным пользователям*. Это дает пользователю возможность разрушить систему привилегий путем переименования таблиц!

Изменения в таблицах назначения привилегий, которые осуществляются при помощи команд GRANT и REVOKE, *обрабатываются сервером немедленно*. Если изменять таблицы назначения привилегий вручную (используя команды INSERT, UPDATE и т.д.), необходимо запустить оператор **FLUSH PRIVILEGES** или **mysqladmin flush-privileges**, чтобы указать серверу на необходимость перезагрузки таблиц назначения привилегий.

1.2. Задание изначальных привилегий MySQL

После установки MySQL инициализирует таблицы предоставления привилегий со следующим набором привилегий:

- В качестве суперпользователя создается MySQL root, который может делать все, что угодно. Соединения должны устанавливаться с локального компьютера. **Примечание**: Изначально пароль root пуст, поэтому кто угодно может подсоединиться в качестве root *без пароля* и получить все привилегии.

- Создается анонимный пользователь, который может выполнять любые операции над базами данных с именами test или начинающимися с test_. Соединения должны устанавливаться с локального компьютера. Это означает, что любой локальный пользователь может подключиться без пароля и будет воспринят сервером как анонимный пользователь.

- Остальные привилегии запрещены. Например, обычный пользователь не может использовать команды **mysqladmin shutdown** или **mysqladmin processlist**.

Поскольку сразу после установки программа совершенно не защищена, первым делом необходимо задать пароль для пользователя MySQL **root**.

Это можно сделать следующим образом

Задание пароля администратора (обратите внимание, что пароль указывается при помощи функции PASSWORD()):

1)

```
shell> mysql -u root mysql
mysql> SET PASSWORD FOR root@localhost=PASSWORD('new_password');
```

2) Опытные пользователи могут работать непосредственно с таблицами назначения привилегий:

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('new_password')
    ->     WHERE user='root';
mysql> FLUSH PRIVILEGES;- это обязательно
```

3) Еще один способ задать пароль - воспользоваться командой mysqladmin:

```
shell> mysqladmin -u root password new_password
```

*Изменять пароли других пользователей могут **только** пользователи с правом записи/обновления базы данных mysql. Все **обычные** пользователи (не анонимные) могут модифицировать только свой собственный пароль при помощи указанных выше команд или команды **SET PASSWORD=PASSWORD('new_password')**.*

После того, как был задан пароль **root**, этот пароль необходимо будет вводить, подсоединяясь к серверу как **root**.

В большинстве случаев для задания пользователей и их паролей следует пользоваться командой GRANT.

1.3. Добавление новых пользователей в MySQL

Пользователей можно добавлять двумя различными способами - при помощи команды GRANT или напрямую в таблицы назначения привилегий MySQL. Предпочтительнее использовать команду GRANT - этот способ проще и дает меньше ошибок.

Существует также большое количество программ (таких как MySQL Administrator или phpmyadmin), которые служат для создания и администрирования пользователей.

```
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
    ->     IDENTIFIED BY 'some_pass' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@"%"
        -> IDENTIFIED BY 'some_pass' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO admin@localhost;
mysql> GRANT USAGE ON *.* TO dummy@localhost;
```

Эти команды GRANT создают трех новых пользователей:

monty

Полноценный суперпользователь - он может подсоединяться к серверу откуда угодно, но должен использовать для этого пароль some_pass. Обратите внимание на то, что мы должны применить операторы GRANT как для monty@localhost, так и для monty@"%". Если не добавить запись с localhost, запись анонимного пользователя для localhost, которая создается при помощи mysql_install_db, будет иметь преимущество при подсоединении с локального компьютера, так как в ней указано более определенное значение для поля Host, и она расположена раньше в таблице user.

admin

Пользователь, который может подсоединяться с localhost без пароля; ему назначены административные привилегии RELOAD и PROCESS. Эти привилегии позволяют пользователю запускать команды mysqladmin reload, mysqladmin refresh и mysqladmin flush-*, а также mysqladmin processlist. Ему не назначено никаких привилегий, относящихся к базам данных (их можно назначить позже, дополнительно применив оператор GRANT).

dummy

Пользователь, который может подсоединяться к серверу без пароля, но только с локального компьютера. Все глобальные привилегии установлены в значение 'N'-тип привилегии USAGE, который позволяет создавать пользователей без привилегий. Предполагается, что относящиеся к базам данных привилегии будут назначены позже.

Можно напрямую добавить точно такую же информацию о пользователе при помощи оператора INSERT, а затем дать серверу команду перезагрузить таблицы назначения привилегий:

```
mysql> INSERT INTO user VALUES('localhost','monty',PASSWORD('some_pass'),
    ->         'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user VALUES('%','monty',PASSWORD('some_pass'),
    ->         'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user SET Host='localhost',User='admin',
    ->          Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
    ->          VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

Для пользователя admin используется более удобочитаемый расширенный синтаксис команды INSERT.

Обратите внимание: чтобы создать суперпользователя, необходимо создать запись таблицы **user** с полями привилегий, установленными в значение 'Y'. Нет необходимости задавать значения в записях таблиц db или host.

Столбцы привилегий в таблице user в последнем операторе INSERT (для пользователя dummy) не были заданы явно, поэтому данным столбцам был присвоено принятое по умолчанию значение 'N'. Точно так же действует команда GRANT USAGE.

Пример добавления пользователя

Добавляется пользователь custom, который может подсоединяться с компьютеров localhost, server.domain и whitehouse.gov. Он хочет получать доступ к базе данных bankaccount только с компьютера localhost, к базе данных expenses - только с whitehouse.gov, и к базе данных customer - со всех трех компьютеров, а также использовать пароль stupid при подсоединении со всех трех компьютеров.

Чтобы задать эти привилегии пользователя при помощи оператора GRANT, выполните следующие команды:

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->         ON bankaccount.*
    ->         TO custom@localhost
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON expenses.*
    ->     TO custom@whitehouse.gov
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON customer.*
    ->     TO custom@'%'
    ->     IDENTIFIED BY 'stupid';
```

Задание привилегий пользователя путем непосредственного внесения изменений в таблицы назначения привилегий (обратите внимание на команду FLUSH PRIVILEGES в конце примера):

```
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('localhost','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('server.domain','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('whitehouse.gov','custom',PASSWORD('stupid'));
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    -> Create_priv,Drop_priv)
    -> VALUES
    -> ('localhost','bankaccount','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    -> Create_priv,Drop_priv)
    -> VALUES
    -> ('whitehouse.gov','expenses','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    -> Create_priv,Drop_priv)
```

```
    -> VALUES('%','customer','custom','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
```

Первые три оператора INSERT добавляют в таблицу user записи, которые позволят пользователю custom подключаться с различных компьютеров с указанным паролем, но не дают ему никаких привилегий (все привилегии установлены в принятое по умолчанию значение 'N'). Следующие три оператора INSERT добавляют записи в таблицу db, в которой назначаются привилегии для пользователя custom по отношению к базам данных bankaccount, expenses и customer, но только если доступ осуществляется с определенных компьютеров. Как обычно, после внесения изменений непосредственно в таблицы назначения привилегий серверу необходимо дать команду на перезагрузку этих таблиц (при помощи FLUSH PRIVILEGES), чтобы внесенные изменения вступили в силу.

Если необходимо предоставить определенному пользователю доступ с любого компьютера к определенному домену, можно воспользоваться оператором GRANT следующим образом:

```
mysql> GRANT ...
    ->         ON *.*
    ->   TO myusername@"%.mydomainname.com"
    ->   IDENTIFIED BY 'mypassword';
```

Чтобы сделать то же самое путем непосредственного внесения изменений в таблицы назначения привилегий, выполните следующие действия:

```
mysql> INSERT INTO user VALUES ('%.mydomainname.com', 'myusername',
    ->           PASSWORD('mypassword'),...);
mysql> FLUSH PRIVILEGES;
```

Примечание. Команда SHOW PRIVILEGES показывает список системных привилегий, которые поддерживаются сервером MySQL.

```
mysql> show privileges;
+------------+--------------------------+--------------------------------
| Privilege  | Context                  | Comment
+------------+--------------------------+--------------------------------
| Select     | Tables                   | To retrieve rows from
| Insert     | Tables                   | To insert data into
| Update     | Tables                   | To update existing
| Delete     | Tables                   | To delete existing
| Index      | Tables                   | To create or drop
| Alter      | Tables                   | To alter the
| Create     | Databases,Tables,Indexes | To create new databases and
| Drop       | Databases,Tables         | To drop databases and tables
| Grant      | Databases,Tables         | To give to other users those
                                          privileges you possess
    | References | Databases,Tables         | To have references on
tables
| Reload     | Server Admin             | To reload or refresh tables,
                                          logs and privileges
| Shutdown   | Server Admin             | To shutdown the
| Process    | Server Admin             | To view the plain text of
                                          currently executing queries
| File       | File access on server    | To read and write files on the
                                          server
+------------+--------------------------+--------------------------------
14 rows in set (0.00 sec)
```

Этот же список в переводе на русский язык:

```
mysql> show privileges;
+------------+---------------------+-----------------------------------
| Привилегия | Контекст            | Описание
+------------+---------------------+-----------------------------------
| Select     | Таблица             | Для выборки строк из таблицы
| Insert     | Таблица             | Для вставки данных в таблицы
| Update     | Таблица             | Для обновления существующих записей
| Delete     | Таблица             | Для удаления существующих записей
| Index      | Таблица             | Для создания или удаления индексов
| Alter      | Таблица             | Для изменения таблиц
| Create     | База,Таблица,Индекс | Для создания новых БД и таблиц
| Drop       | База,Таблица        | Для удаления баз данных и таблиц
| Grant      | База,Таблица        | Для раздачи другим пользователям
                                   |  имеющихся у вас прав
| References | База,Таблица        | Для обладания ссылок на таблицы
| Reload     | Администрирование   | Для обновления таблиц, журналов и
                                   | привилегий
| Shutdown   | Администрирование   | Для остановки сервера
| Process    | Администрирование   | Для просмотра текста выполняющихся
                                   | запросов
| File       | Доступ к файлам     | Для чтения и записи файлов на
                                     сервере
+------------+---------------------+-----------------------------------
14 rows in set (0.00 sec)
```

2. Администрирование базы данных

Администраторы СУБД сами создают пользователей и дают им привилегии. С другой стороны – *администраторы баз данных, которые создают таблицы, сами имеют права на управление этими таблицами*.

2.1. Команда GRANT

Предположим, что пользователь Diane имеет таблицу Заказчиков и хочет разрешить пользователю Adrian выполнить запрос к ней. Diane должна в этом случае ввести следующую команду:

```
GRANT SELECT ON Salespeople TO Diane;
```

Теперь Adrian может выполнить запросы к таблице Заказчиков. Но не может предоставить право SELECT другому пользователю: таблица еще принадлежит Diane (далее мы покажем, как Diane может дать право Adrian предоставлять SELECT другим пользователям).

Группы привилегий и пользователей.

Приемлемыми являются списки привилегий или пользователей, разделяемых запятыми.

Stephen может предоставить и SELECT, и INSERT в таблице Заказов для Adrian и Diane:

```
GRANT SELECT, INSERT ON Заказы TO Adrian, Diane;
```

Ограничение привилегий на определенные столбцы

Все привилегии объекта используют один и тот же синтаксис, кроме команды UPDATE в которой не обязательно указывать имена столбцов.

Привилегию UPDATE можно предоставлять наподобие других привилегий:

```
GRANT UPDATE ON Продавцы TO Diane;
```

Эта команда позволит Diane изменять значения в любом или во всех столбцах таблицы Продавцов. Однако, если Adrian хочет ограничить Diane в изменении, например, комиссионных, он может ввести:

```
GRANT UPDATE (city, comm) ON Продавцы TO Diane;
```

Здесь в круглых скобках после имени таблицы указаны конкретные столбцы, к которому привилегия UPDATE должна быть применена. Имена нескольких столбцов таблицы могут указываться в любом порядке, отделяемые запятыми.

2.2. Использование аргумента ALL

SQL поддерживает аргумент для команды GRANT, который имеют специальное значение: ALL PRIVILEGES (ВСЕ ПРИВИЛЕГИИ), или просто ALL.

ALL используется вместо имён привилегий в команде GRANT, чтобы передать все привилегии в таблице.

Например, Diane может выдать Stephen весь набор привилегий в таблице Заказчиков с помощью такой команды:

```
GRANT ALL ON Продавцы TO Diane;
```

(при этом привилегия UPDATE применяется ко всем столбцам.)

Конечно, вы можете предоставить любые или все привилегии каждому, но это, очевидно, нежелательно. Все привилегии, за исключением SELECT, позволяют пользователю изменять содержание таблицы. Разрешение всем пользователям изменять содержание ваших таблиц создаст проблемы.

Даже если вы имеете небольшую компанию и в ней работают все ваши текущие пользователи, способные выполнять команды модификации в данной таблице, было бы лучше предоставить привилегии каждому пользователю индивидуально, чем одни и те же привилегии для всех.

2.3. Предоставление привилегий с помощью предлжения «WITH GRANT OPTION»

Иногда создателю таблицы необходимо, чтобы другие пользователи могли получить привилегии в его таблице. Обычно это делается в системах, где один человек или более создают несколько (или все) базовые таблицы в базе данных, а затем передают ответственность за них тем, кто будет фактически с ними работать. SQL позволяет делать это с помощью предложения WITH GRANT OPTION.

Если бы Diane хотела, чтобы Adrian имел право предоставлять привилегию SELECT в таблице Заказчиков другим пользователям, она дала бы ему привилегию SELECT с использованием предложения WITH GRANT OPTION:

```
GRANT SELECT ON Заказчики TO Adrian WITH GRANT OPTION;
```

После этого Adrian получил право передавать привилегию SELECT третьим лицам; он может выдать команду

```
GRANT SELECT ON Diane.Заказчики TO Stephen;
```

или даже

```
GRANT SELECT ON Diane.Заказчики TO Stephen WITH GRANT OPTION;
```

2.4. Отмена привилегий

Команда GRANT позволяет вам давать привилегии пользователям, но не позволяет отобрать их обратно.

Удаление привилегии выполняется командой REVOKE. Синтаксис команды REVOKE похож на GRANT, но имеет обратный смысл.

Чтобы удалить привилегию INSERT для Adrian в таблице Заказов, вы можете ввести

```
REVOKE INSERT ON Заказы FROM Adrian;
```

Использование *списков привилегий и пользователей* здесь также допустимы, как и в случае с GRANT, так что вы можете ввести следующую команду:

```
REVOKE INSERT, DELETE ON Заказчики FROM Adrian, Stephen;
```

Привилегии отменяются пользователем, который их предоставил, и отмена будет каскадироваться, то есть она будет автоматически распространяться на всех пользователей, получивших от него эту привилегию.

Порядок выполнения лабораторной работы.

Лабораторная работа состоит из трех частей.

1. Администрирование MySQL - сервера.
   1.1. Изучить теоретический материал.
   1.2. Изучить работу с графической утилитой MySQL Workbench.
   1.3. Используя MySQL Workbench, создать базу данных.
   1.4. Используя MySQL Workbench, создать двух пользователей (условно пользователь А и пользователь Б). Под этими пользователями вы будете выполнять вторую и третью части задания. **Дайте пользователям уникальные имена.** Сложные пароли можно не изобретать, берите самые простые. Одного из пользователей (А) сделать администратором вашей базы данных, т.е. дать ему необходимые привилегии. Второму привилегий не давать.

1.5. Используя MySQL Workbench проверьте содержимое таблиц ***user***, ***db*** и ***host*** в базе данных ***mysql***. Убедитесь, что пользователи созданы правильно. ***Необходимые фрагменты таблиц привести в отчете.***

2. Создание таблиц.
   2.1. Подключиться к MySQL серверу в качестве пользователя ***A*** (т.е. как администратор базы данных).
   2.2. Создать в базе данных таблицы с учетом ограничения ссылочной целостности данных.
   2.3. Выполнить команду SHOW INDEX FROM tbl_name для каждой таблицы. ***Результаты с пояснениями привести в отчете.***
   2.4. Заполнить таблицы данными любым способом, например используя редактор SQL. ***Полученные таблицы с данными привести в отчете.***
   2.5. Убедиться с помощью запросов DELETE, UPDATE, INSERT, что выбранные вами стратегии поддержания ссылочной целостности данных действительно обеспечиваются в СУБД MySQL. ***Результаты с пояснениями привести в отчете.***
3. Администрирование баз данных сервера MySQL.
   3.1. Подключиться к MySQL серверу в качестве пользователя ***A*** (т.е. как администратор базы данных).
   Примечание. Войдя в программу MySQL Workbench под именем пользователя ***A*** (и соответственно подключившись к MySQL серверу не как администратор MySQL) вы не сможете передать ваши привилегии на вашу базу данных другим пользователям. Поэтому дальнейшие действия необходимо производить путем SQL - запросов. Из командной строки или с помощью MySQL Workbench (также войдя в него под именем пользователя ***A***).
   3.2. Используя теоретический материал, выполнить следующее. *В отчете привести название темы, словесную формулировку запросов, запросы, результат проверки (с помощью соответствующих запросов), необходимые фрагменты таблиц* ***user***, ***db*** *и* ***host*** *в базе данных mysql*:
   - С помощью команды GRANT предоставить пользователю ***Б*** отдельные привилегии на свои таблицы. Подключиться к серверу как пользователь ***Б*** и убедиться, что привилегии действуют.
   - Аналогично изучить действие команды GRANT с использованием аргумента ALL.
   - Аналогично изучить предоставление привилегий с помощью предложения «WITH GRANT OPTION».
   - Аналогично изучить отмену привилегий с помощью команды REVOKE.

Содержание 2 части отчета:

1. Цель работы.
2. Фрагменты таблиц ***user***, ***db*** и ***host*** *(см. Задание. Часть 1.5).*
3. Команды SHOW INDEX FROM tbl_name для каждой таблицы. Результаты с пояснениями *(см. Задание. Часть 2.3).*
4. Таблицы вашей базы данных *(см. Задание. Часть 2.3).*

5. Использование команд GRANT и REVOUKE при администрировании базы данных *(см. Задание. Часть 3.2)*.
6. Общие выводы по вашим результатам.