

Создание таблиц в СУБД MySQL

Проектирование таблиц

Рассмотрим создание таблиц на примере 2-х сущностей: **СТУДЕНТЫ** и **ГРУППЫ** студентов.

Сначала создадим реляционную модель данных (РМД, концептуальная модель), описывающих вышеуказанные сущности.

Фамилия (**Surn**)

Имя (**Name**)

Отчество (**Patr**)

Дата рождения (**Birth**)

Пол (**Gender**)

Группа (**Grp**)

Год набора (**sYear**)

Зеленым цветом выделены ключевые поля РМД (составной первичный ключ)

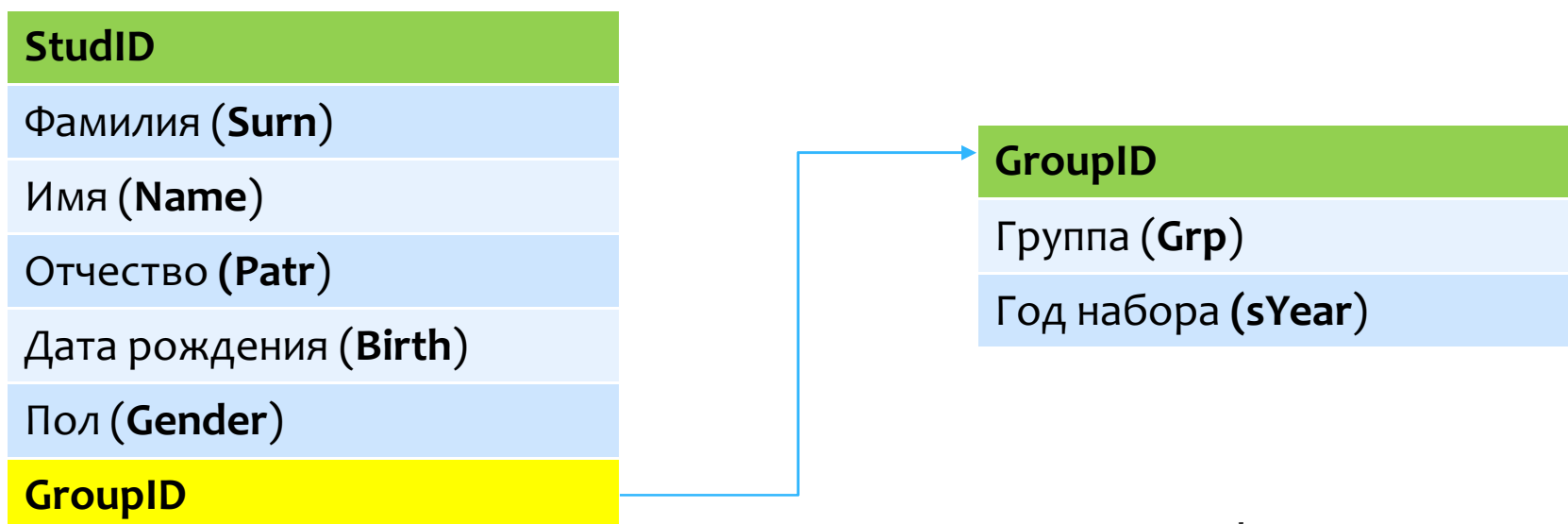
Поля, выделенные **желтым** и **синим** цветом являются зависимыми от части первичного ключа

Добавим для каждой части первичного ключа отдельное ключевое поле и нормализуем таблицу (приведем ее ко 2-ой НФ)

Проектирование таблиц

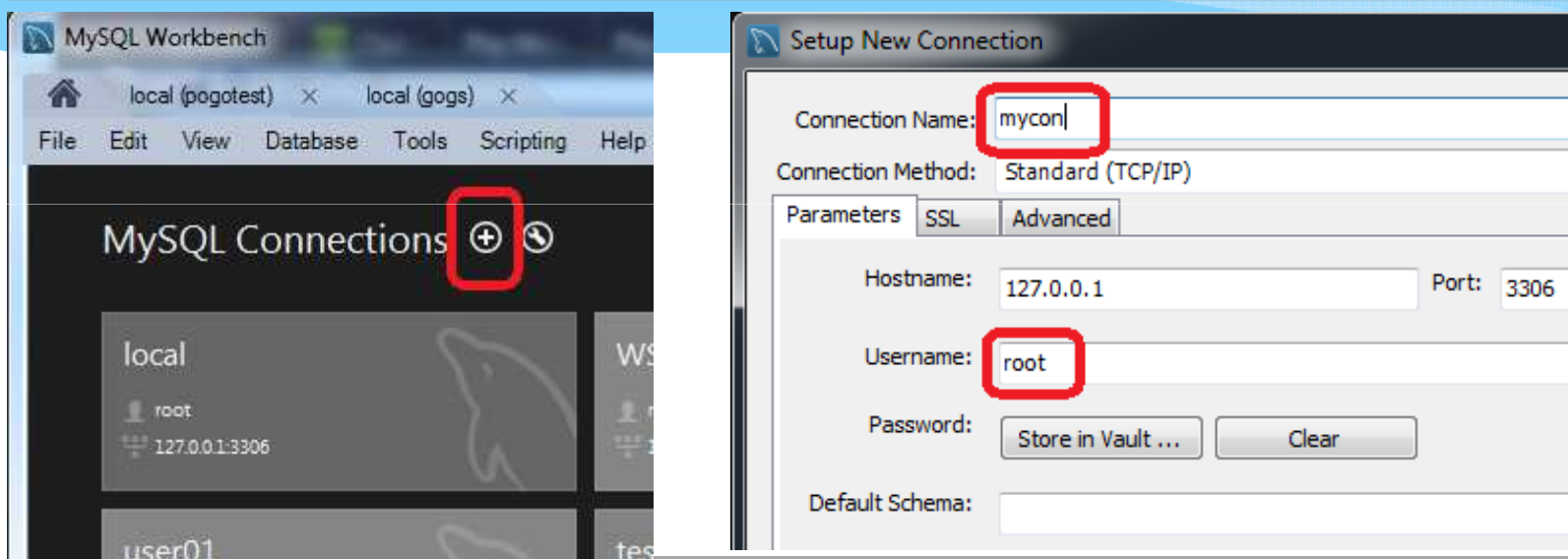
Данные отношения находясь во 2-ой НФ также будут находиться и во 3-ой НФ.

В двух полученных таблицах **зеленым** цветом отмечены первичные ключи, **желтым** – внешние.



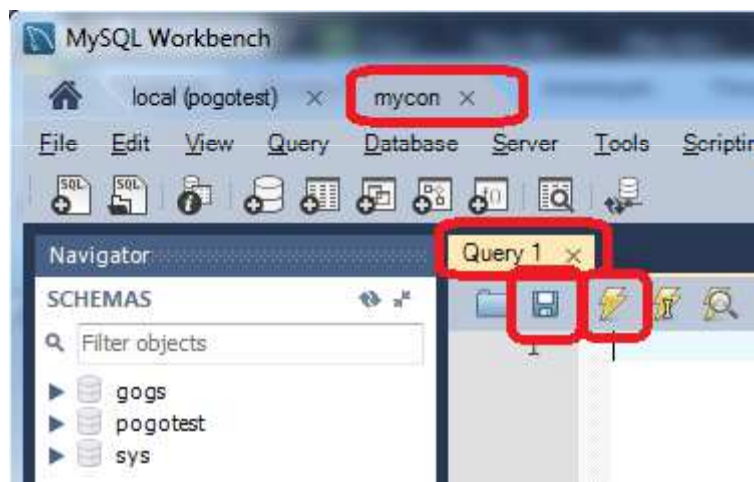
Создание таблиц

Для создания таблиц с помощью SQL запросов, запустим **MySQL Workbench** и создадим новое соединение с сервером. В появившемся окне необходимо указать имя соединения.



Для проверки соединения нажать кнопку **Test Connection**. Если службы сервера работают и пароль верен – соудинение будет успешным. В этом случае пароль можно сохранить в кэш кнопкой **Store in Vault**

Создание таблиц



Перед запуском запроса, желательно его сохранить во избежание потери информации.

После создания соединения, необходимо подключиться к серверу с его использованием. Выбираем соединение и вводим пароль. Откроется окно, содержащее соединение и возможность создать и выполнить SQL запрос на создание БД.

На вкладке сверху будет указано имя соединения.

Создание БД

SQL - скрипт начинается с проверки наличия уже созданной версии БД и ее удаления. Затем создается новая БД.

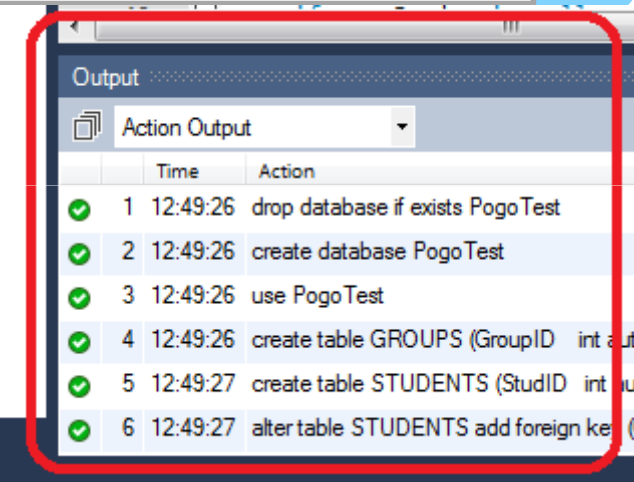
```
drop database if exists MyDB;  
create database if not exists MyDB;
```

После того, как база данных создана, перед добавлением таблиц, необходимо произвести подключение к ней.

```
use MyDB;
```

Для запуска запроса на выполнение можно использовать сочетание клавиш **Shift+Ctrl+Enter**.

Скрипт должен выполняться без ошибок, в чем можно удостовериться в окне **Output**.



	Time	Action
✓ 1	12:49:26	drop database if exists PogoTest
✓ 2	12:49:26	create database PogoTest
✓ 3	12:49:26	use PogoTest
✓ 4	12:49:26	create table GROUPS (GroupID int aut
✓ 5	12:49:27	create table STUDENTS (StudID int au
✓ 6	12:49:27	alter table STUDENTS add foreign key (

Создание таблиц

В случае возникновения ошибок - проверить написание скрипта и наличие дублирующихся имен БД. Приступать к созданию таблиц только после успешного создания БД

Создание таблиц начнем с таблицы, не имеющей внешних ключей – таблицы **GROUPS**. Запросы на создание таблиц будут выглядеть следующим образом

```
create table GROUPS
(GroupID int auto_increment primary key,
GroupName varchar(10) ,
sYear numeric) ;
```

Здесь **GroupID** – первый ключ таблицы **GROUPS**, соответственно имеет ограничение требующее уникальности этого поля. Помимо этого, первичный ключ не может быть пустым и должен быть уникальным – флаг **auto_increment** создает автозаполняющееся инкрементное поле.

Создание таблиц

В таблице **STUDENTS** первичный ключ таблицы – **StudID** – имеет все те же ограничения, что и первичный ключ предыдущей таблицы.

```
create table STUDENTS
(StudID int auto_increment primary key,
 GroupID int,
 Surn varchar(30) ,
 Name varchar(20) ,
 Patr varchar(30) ,
 Gender varchar (1) ,
 Birth date) ;
-- Добавление внешнего ключа
alter table STUDENTS add foreign key (GroupID)
references GROUPS (GroupID) on update cascade on delete
```

Внешний ключ **GroupID** связывает таблицу **STUDENTS** с таблицей **GROUPS** по ее первичному ключу **GroupID**. В случае изменения первичного ключа в родительской таблице предусмотрено каскадное изменение дочерней (**on update cascade**), а при удалении – сброс значения внешнего ключа в **null** (**on delete set null**).

Создание таблиц

Для удаления таблиц используется команда **drop table**

```
drop table STUDENTS ;
```

После выполнения скрипта необходимо закрыть транзакцию с сохранением ее работы. Для этого в скрипты с запросами на создание таблиц необходимо добавлять **commit**. В данном случае готовый запрос на создание таблиц будет выглядеть примерно так (описание полей опущено)

```
create table GROUPS  
(...) ;
```

```
create table STUDENTS  
(...) ;  
commit;
```

Создание триггеров

Для контроля за заполнением полей корректными значениями, используются **триггеры**. Для контроля года создания группы, создадим триггер для проверки этого поля на пустое значение (**null**). В этом случае будем считать текущий год годом создания группы. Запрос на создание такого триггера **bi_group** для таблицы **GROUPS** выглядит следующим образом.

```
delimiter ^
create trigger bi_group before insert on GROUPS
for each row
begin
    if new.gYear is null then
        set new.gYear =Year(Now()) ;
    end if;
end; ^
delimiter ;
commit;
```

delimiter служит для определения начала и окончания команды создания триггера.

Создание триггеров

Создадим аналогичный триггер для автозаполнения пола студентов, если он не указан или указан некорректно. Будем считать, что пол может обозначаться одним латинским (M/F) или кириллическим (М/Ж) символом. Запрос на их создание такого триггера `bi_students` для таблицы `STUDENTS` выглядит следующим образом.

```
delimiter ^
create trigger bi_students before insert on STUDENTS
for each row
begin
    if new.Gender is null or
        Locate(Upper(new.Gender), 'MFMЖ')=0 then
        set new.Gender = 'M';
    end if;
end; ^
delimiter ;
commit;
```

Этот триггер устанавливает по мужской пол, если таковой не указан.

Заполнение БД

Заполнение таблицы осуществляется SQL запросами `insert`.

Добавление двух записей в таблицу **GROUPS**. Первый запрос тестирует триггер заполнения года.

```
insert into GROUPS (GroupName) values ('ПР10-10');  
insert into GROUPS (GroupName,sYear) values ('ПР11-10',2016);
```

Добавление студентов в таблицу **STUDENTS**. Значения поля **GroupID** выбирается исходя из данных таблицы **GROUPS**.

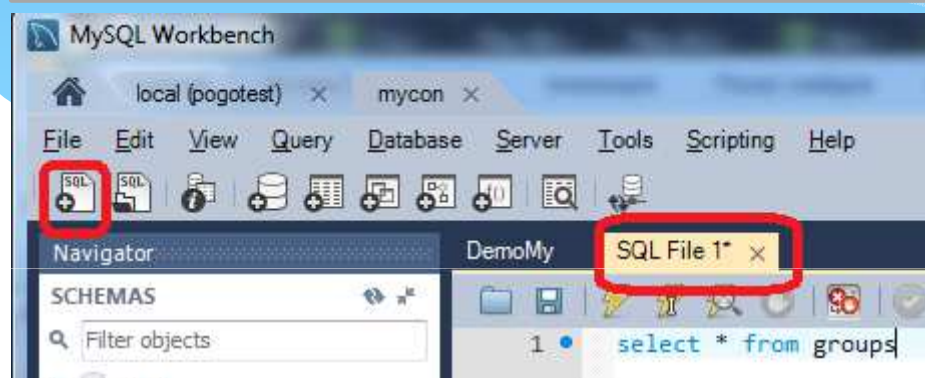
```
insert into STUDENTS  
(GroupID,Surn,Name,Patr,Birthday) values  
(1,'Рюрик','Иоанн','Васильевич','2000.10.01');
```

Добавьте больше студентов в таблицу **STUDENTS**. Сформируйте запросы таким образом, чтобы протестировать работоспособность генератора `bi_students`.

Просмотр БД

Контроль добавления можно осуществить выполнив запросы **select**. Для этого откроем новое окно запросов и впишем туда

```
select * from GROUPS
```



В окне редактора появится результат запроса в виде таблицы. Если заполнение было проведено корректно, отобразятся данные из предыдущих запросов.

Внешним видом результат можно управлять с помощью фильтров и заголовков столбцов.

A screenshot of the 'Result Grid' in MySQL Workbench. It displays the results of the 'select * from groups' query as a table with three columns: GroupID, GroupName, and gYear. The first three rows contain data, and the fourth row is a summary row with 'NULL' values. The first row is highlighted in blue.

	GroupID	GroupName	gYear
1	1	ПР2000	2018
2	2	ПР2000	2018
3	3	ПР2016	2016
*	NULL	NULL	NULL

Просмотр БД

Создадим более сложный запрос, объединяющий обе таблицы.

```
select GroupName, Surn, Name, Patr, Gender from STUDENTS S  
left join GROUPS G on G.GroupID=S.GroupID  
order by GroupName, Surn, Name
```

Добавьте к запросу еще одно поле: `Year(Now())-Year(Birthday) as Age`

Проанализируйте полученные результаты.