

## **Лабораторная работа 1.**

**Задание 1.** Используя справочные материалы по MySQL Workbench, опишите назначение пунктов меню Management (“Управление”), “Instance” (“Экземпляр БД”) и “Performance” (“Производительность”).

### **Раздел “Management”:**

Раздел "Server Status" содержит информацию о состоянии сервера и подключении к нему. Он предоставляет общую информацию о сервере, включая название хоста, номер порта, версию БД, настройки сервера, каталоги сервера, сводку по используемым ресурсам и настройки SSL.

Раздел "Client Connections" предоставляет информацию о текущих подключениях к серверу базы данных MySQL. Здесь можно увидеть список активных подключений, IP-адреса клиентов, имена пользователей, номера потоков запросов и другую информацию о текущей активности. Также можно управлять подключениями, завершая их при необходимости.

Раздел "User and Privileges" позволяет управлять пользователями и их правами доступа к базе данных MySQL. Здесь можно создавать новых пользователей, удалять существующих, изменять их пароли и другие атрибуты учетных записей. Также можно определять привилегии доступа для каждого пользователя и управлять ролями пользователей.

Раздел "Status and System Variables" предоставляет информацию о текущем статусе сервера базы данных MySQL и позволяет просматривать и изменять системные переменные, которые влияют на работу сервера. Здесь можно просмотреть общую информацию о состоянии сервера, системные переменные, мониторить ресурсы и настраивать параметры сервера.

Раздел "Data Export" позволяет экспортировать данные из базы данных MySQL в различные форматы для создания резервных копий, обмена данными или анализа информации. Здесь можно выбрать данные для экспорта, выбрать формат экспорта, настроить параметры экспорта и создать расписание экспорта.

Раздел "Data Import/Restore" предназначен для импорта данных в базу данных MySQL из внешних источников и восстановления данных из резервных копий или дампов баз данных. Здесь можно выбрать источник данных, настроить параметры импорта, обработать конфликты и восстановить данные.

### **Раздел “Instance”:**

Раздел "Startup / Shutdown" предоставляет возможность управлять процессом запуска и остановки экземпляра базы данных MySQL. Здесь можно инициировать запуск сервера, его остановку или перезапуск.

Раздел "Server Logs" позволяет просматривать и анализировать логи действий на сервере. Здесь можно увидеть различные типы логов, информацию о событиях, ошибках, успешных операциях и других действиях на сервере. Также можно анализировать производительность базы данных и искать нужную информацию в логах.

Раздел "Options File" предназначен для работы с конфигурационным файлом сервера базы данных MySQL. Здесь можно просмотреть текущие настройки сервера, редактировать параметры, сохранять изменения и откатывать настройки.

### **Раздел "Performance":**

Раздел "Dashboard" предоставляет обзорную информацию о производительности сервера. Здесь можно увидеть сводные показатели о состоянии сервера, графики и диаграммы, мониторить ресурсы и анализировать производительность запросов.

Раздел "Performance Reports" позволяет генерировать отчеты о производительности сервера. Здесь можно получить различные типы отчетов, анализировать метрики производительности и получать рекомендации по оптимизации.

Раздел "Performance Schema Setup" предоставляет возможность настройки и оптимизации схемы производительности MySQL. Здесь можно активировать Performance Schema, настроить параметры, отслеживать метрики и оптимизировать работу сервера.

**Задание 2.** Создать и настроить новую базу данных `simpledb`.

**Review the SQL Script to be Applied on the Database**

Online DDL

Algorithm: 

Default

 Lock Type: 

Default

```
1 CREATE SCHEMA `simpledb` DEFAULT CHARACTER SET utf8 ;
2
```

SCHEMAS

Filter objects

simpledb


Tables

Views

Stored Procedures

Functions

sys



Connection Name

mysql-test1

Host: 347d52e041ae

Socket: /var/run/mysqld/mysqld.sock

Port: 3306

Version: 8.0.36 (MySQL Community Server - GPL)

Compiled For: Linux (x86\_64)

Configuration File: unknown

Running Since: Mon Apr 15 19:48:06 2024 (0:03)

Refresh

Available Server Features

Performance Schema:	<input checked="" type="radio"/> On	Windows Authentication:	<input type="radio"/> Off
Thread Pool:	<input type="radio"/> n/a	Password Validation:	<input type="radio"/> n/a
Memcached Plugin:	<input type="radio"/> n/a	Audit Log:	<input type="radio"/> n/a
Semisync Replication Plugin:	<input type="radio"/> n/a	Firewall:	<input type="radio"/> n/a
SSL Availability:	<input checked="" type="radio"/> On	Firewall Trace:	<input type="radio"/> n/a

Administration

Schemas

### Задание 3

Скопируйте запрос, соответствующий созданию этой таблицы и вставьте его в отчет по выполнению этой лабораторной работы.

```
CREATE TABLE `simplified`.`users` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `name` VARCHAR(45) NOT NULL,  
  `email` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE);
```

#### Задание 4

Добавьте несколько примеров-записей в созданную таблицу.

```
INSERT INTO `simplifiedb`.`users` (`name`, `email`) VALUES ('Paul',  
'smith@example.com');  
INSERT INTO `simplifiedb`.`users` (`name`, `email`) VALUES ('Marina',  
'doe@example.com');  
INSERT INTO `simplifiedb`.`users` (`name`, `email`) VALUES ('Ekaterina',  
'williams@example.com');
```

После этого обновите одно или несколько полей (например, name и/или email), нажмите Apply и сохраните полученный SQL-запрос в отчете.

Какой SQL-запрос при этом выполнится?

```
UPDATE `simplifiedb`.`users` SET `name` = 'Mark' WHERE (`id` = '3');
```

	id	name	email
	1	Paul	smith@example.com
	2	Marina	doe@example.com
▶	3	Mark	williams@example.com
★	NULL	NULL	NULL

### Задание 5.

Дополните таблицу users так, чтобы получилась таблица со следующими полями и параметрами: 1. id int pk, not null 2. name varchar(50) 3. email varchar(45) 4. gender ENUM('M', 'F') 5. bday Date 6. postal\_code varchar(10) 7. rating float 8. created TIMESTAMP CURRENT\_TIMESTAMP()

```
ALTER TABLE `simplifiedb`.`users`
```

```
ADD COLUMN `gender` ENUM('M', 'F') NULL AFTER `email`,
```

```
ADD COLUMN `bday` DATE NULL AFTER `gender`,
```

```
ADD COLUMN `postal_code` VARCHAR(10) NULL AFTER `bday`,
```

```
ADD COLUMN `rating` FLOAT NULL AFTER `postal_code`,
```

```
ADD COLUMN `created` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP()  
AFTER `rating`,
```

```
CHANGE COLUMN `name` `name` VARCHAR(50) NOT NULL ;
```

Тип данных TIMESTAMP: В MySQL поле типа TIMESTAMP предназначено для хранения информации о времени (дате и времени). Чаще всего оно используется для отслеживания времени создания или обновления записей в таблице.

Значение по умолчанию CURRENT\_TIMESTAMP(): Использование функции CURRENT\_TIMESTAMP() в качестве значения по умолчанию для поля TIMESTAMP означает, что если при добавлении новой записи в таблицу не указано значение для поля created, система автоматически установит текущее время и дату в это поле. Все поля, кроме первых трёх, могут содержать значение NULL.

Все поля, кроме первых трех, могут иметь значение NULL.

**Задание 6.**

Дополните таблицу, добавив данные двумя способами:

- с помощью внесения данных вручную (как это было сделано ранее);
- с помощью выполнения SQL-запросов ниже;

```
INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`, `rating`)
VALUES ('Ekaterina', 'ekaterina.petrova@outlook.com', '145789', 'f', '2000-02-11', '1.123');
INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`, `rating`)
VALUES ('Paul', 'paul@superpochta.ru', '123789', 'm', '1998-08-12', '1');
```

**Вручную:**

```
UPDATE `simplifiedb`.`users` SET `gender` = 'M', `postal_code` = '123456', `rating` = '1'
WHERE (`id` = '1');
```

```
UPDATE `simplifiedb`.`users` SET `gender` = 'F', `postal_code` = '654321', `rating` = '2' WHERE
(`id` = '2');
```

```
UPDATE `simplifiedb`.`users` SET `gender` = 'M', `postal_code` = '234156', `rating` = '3'
WHERE (`id` = '3');
```

	id	name	email	gender	bday	postal_code	rating	created
	1	Paul	StickQuicksand@gmail.com	M	NULL	123456	1	2024-02-21 14:37:46
	2	Marina	ScriptMultiBlue@gmail.com	F	NULL	654321	2	2024-02-21 14:37:46
▶	3	Jonh	PandaSquidSeal@gmail.com	M	NULL	234156	3	2024-02-21 14:37:46
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**С помощью запросов:**

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	email	gender	bday	postal_code	rating	created
▶	1	Paul	StickQuicksand@gmail.com	M	NULL	123456	1	2024-02-21 14:37:46
	2	Marina	ScriptMultiBlue@gmail.com	F	NULL	654321	2	2024-02-21 14:37:46
	3	Jonh	PandaSquidSeal@gmail.com	M	NULL	234156	3	2024-02-21 14:37:46
	4	Ekaterina	ekaterina.petrova@outlook.com	F	2000-02-11	145789	1.123	2024-02-21 14:55:36
	5	Paul	paul@superpochta.ru	M	1998-08-12	123789	1	2024-02-21 14:55:37
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 1 | SQL File 2\* x users | Limit to 1000 rows

```
1 • INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`,
2   `rating`) VALUES ('Ekaterina', 'ekaterina.petrova@outlook.com', '145789', 'f',
3   '2000-02-11', '1.123');
4 • INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`,
5   `rating`) VALUES ('Paul', 'paul@superpochta.ru', '123789', 'm', '1998-08-12', '1');
```

### Задание 7.

С помощью кнопки “Export recordset to external file” и получите файл с SQL-запросами.

/\*

-- Query: SELECT \* FROM simplifiedb.users

LIMIT 0, 1000

-- Date: 2024-02-21 17:59

\*/

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (1,'Paul','StickQuicksand@gmail.com','M',NULL,'123456',1,'2024-02-21 14:37:46');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (2,'Marina','ScriptMultiBlue@gmail.com','F',NULL,'654321',2,'2024-02-21 14:37:46');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (3,'Jonh','PandaSquidSeal@gmail.com','M',NULL,'234156',3,'2024-02-21 14:37:46');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (4,'Ekaterina','ekaterina.petrova@outlook.com','F','2000-02-11','145789',1.123,'2024-  
02-21 14:55:36');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (5,'Paul','paul@superpochta.ru','M','1998-08-12','123789',1,'2024-02-21 14:55:37');
```



### Задание 8.

Создайте еще одну таблицу с названием resume со следующей структурой: - resumeid, INT, PK, NN, AI - userid, INT, NN - title, VARCHAR(100), NN - skills, TEXT - created, TIMESTAMP, Default / Expression: CURRENT\_TIMESTAMP().

Определите так внешний ключ , который будет определять связь между текущей таблицей resume и уже созданной таблицей user. Введите в таблицу слева в столбец Foreign Key (внешний ключ): userid и определите таблицу, где они будут находиться: simplifiedb.users В таблицу Foreign key details 'userid' щелкните мышкой рядом с полем userid так, чтобы оно было выделено и определите столбец-источник для значений - id.

Последний шаг: определить действия при операциях On Update и On Delete: для обоих выберите - Cascade.

```
CREATE TABLE `simplifiedb`.`resume` (  
  `resumeid` INT NOT NULL AUTO_INCREMENT,  
  `userid` INT NOT NULL,  
  `title` VARCHAR(100) NOT NULL,  
  `skills` MEDIUMTEXT NULL,  
  `created` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP(),  
  PRIMARY KEY (`resumeid`),  
  INDEX `userid_idx` (`userid` ASC) VISIBLE,  
  CONSTRAINT `userid`  
    FOREIGN KEY (`userid`)  
    REFERENCES `simplifiedb`.`users` (`id`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE);
```

**Как будет вести себя СУБД при удалении связанных записей из таблиц users и resume.**

При удалении связанных записей из таблиц users и resume, поведение СУБД будет следующим:

При удалении записи из таблицы users:

Если у пользователя нет записей в таблице resume, то его удаление пройдет без проблем.

Если же у пользователя есть связанные записи в таблице resume, то при удалении пользователя из таблицы users, все связанные записи из таблицы resume будут также удалены автоматически благодаря внешнему ключу с опцией ON DELETE CASCADE.

При удалении записи из таблицы resume:

Удаление записи из таблицы resume не повлияет на данные в таблице users, так как удаление из дочерней таблицы (resume) не требует каскадного удаления из родительской таблицы (users).

### Задание 9.

Наполните вторую таблицу данными так, чтобы в ней была информация хотя бы о нескольких резюме, связанных с уже существующими пользователями из таблицы users.

\*

```
-- Query: SELECT * FROM simplifiedb.resume  
LIMIT 0, 1000
```

```
-- Date: 2024-02-21 18:27
```

\*/

```
INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (1,1,'Web  
Developer','JavaScript, React, Node.js','2024-02-21 15:25:57');
```


```
INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (2,2,'Data  
Analyst','SQL, Python, Data Visualization','2024-02-21 15:25:57');
```

**Попробуйте добавить в таблицу resume строчку с userid несуществующего пользователя (такого пользователя id которого нет в таблице users).**

При выполнении подобной операции будет выдано сообщение об ошибке, указывающее на нарушение целостности данных из-за отсутствия соответствующего user\_id в таблице users. Это одно из важных требований для поддержания целостности данных в базе данных при использовании внешних ключей для связей между таблицами.

### Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution.  
Press Show Logs to see the execution logs.

 Execute SQL Statements

Error: There was an error while applying the SQL script to the database.

#### Message Log

```
Executing:
INSERT INTO `simplifiedb`.`resume` (`userid`, `title`, `skills`) VALUES (6, 'Graphic Designer', 'Adobe Photoshop, UI/UX Design');

Operation failed: There was an error while applying the SQL script to the database.
ERROR 1452: 1452: Cannot add or update a child row: a foreign key constraint fails
(`simplifiedb`.`resume`, CONSTRAINT `userid` FOREIGN KEY (`userid`) REFERENCES `users` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE)
SQL Statement:
INSERT INTO `simplifiedb`.`resume` (`userid`, `title`, `skills`) VALUES (6, 'Graphic Designer', 'Adobe Photoshop, UI/UX Design')
```

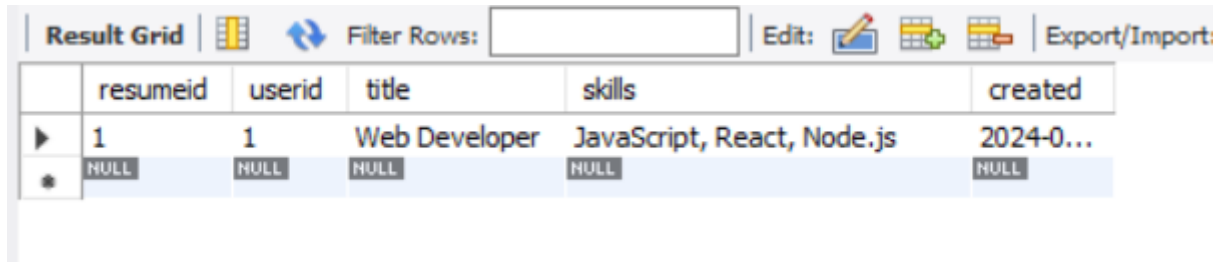
### Задание 10.

Удалите одного или двух таких пользователей, что для них существуют записи в таблице resume.

```
DELETE FROM `simplifiedb`.`users` WHERE (`id` = '2');
```

**Что произойдет со связанными сущностями в таблице resume?**

Соответствующее резюме будет удалено.

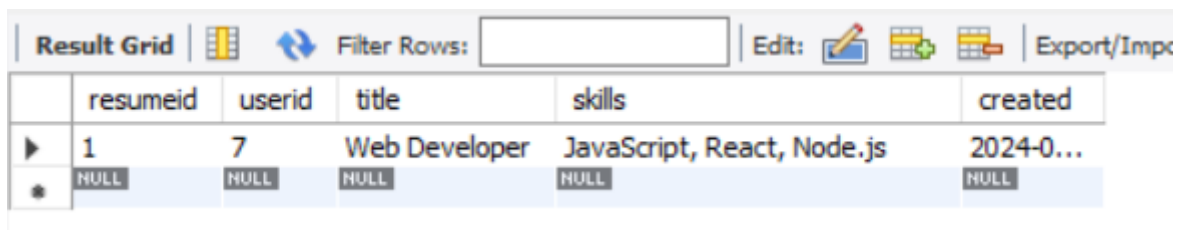


	resumeid	userid	title	skills	created
▶	1	1	Web Developer	JavaScript, React, Node.js	2024-0...
✱	NULL	NULL	NULL	NULL	NULL

**Что произойдет, если в таблице users будет изменен id какого-то существующего пользователя.**

```
UPDATE `simplifiedb`.`users` SET `id` = '7' WHERE (`id` = '1');
```

Произойдет автоматическое обновление.



	resumeid	userid	title	skills	created
▶	1	7	Web Developer	JavaScript, React, Node.js	2024-0...
✱	NULL	NULL	NULL	NULL	NULL