Отчет по лабораторной работе № 1

Задание 1

Раздел "Management":

Раздел "Management" предоставляет инструменты для администрирования и управления сервером MySQL. Основные функциональные группы включают:

Общая информация о сервере:

- Отображение основных параметров сервера: имя хоста, порт подключения, версию MySQL.

- Статус соединения с сервером и время его работы.

Конфигурация каталогов:

- Просмотр путей к основным каталогам сервера: директория данных, временные файлы, логи.

- Информация о расположении конфигурационных файлов.

Управление SSL-соединениями:

- Настройка и мониторинг параметров безопасного соединения.

- Конфигурация сертификатов и ключей для шифрования.

Мониторинг ресурсов системы:

- Отображение использования системных ресурсов: CPU, память, диск.

- Графическое представление загрузки ресурсов во времени.

Управление журналами:

- Просмотр и анализ лог-файлов сервера.

- Фильтрация и поиск событий в журналах ошибок и событий.

Настройка параметров сервера:

- Динамическое изменение глобальных и сессионных переменных.

- Мониторинг значений системных переменных и их состояния.

Управление резервными копиями:

- Создание и восстановление бэкапов баз данных.

- Планирование автоматических резервных копий.

Сервисное обслуживание:

- Инструменты для оптимизации таблиц и проверки их целостности.

- Репарация поврежденных таблиц при необходимости.

Раздел “Instance”: Раздел «Instance» предоставляет комплексные инструменты для управления экземпляром базы данных MySQL. В данном разделе можно выделить следующие ключевые функциональные группы:

Обзор состояния сервера:

- Отображение текущего статуса работы сервера MySQL, включая время его запуска и продолжительность работы.

- Информация о количестве активных подключений и их состоянии.

Настройки конфигурации:

- Возможность просмотра и изменения параметров конфигурационного файла сервера.

- Редактирование настроек производительности, таких как размер буферов, лимиты соединений и другие параметры.

Управление службами:

- Инструменты для запуска, остановки и перезапуска сервера MySQL.

- Просмотр журнала событий сервера для анализа ошибок и важных уведомлений.

Пользователи и привилегии:

- Администрирование пользователей базы данных: создание новых пользователей, изменение паролей, назначение прав доступа.

- Настройка ограничений для пользователей по IP-адресам или конкретным базам данных.

Бэкапы и восстановление:

- Создание резервных копий баз данных с использованием различных методов.

- Восстановление баз данных из ранее созданных резервных копий.

Схемы данных:

- Управление существующими схемами (базами данных): создание, удаление, модификация.

- Обзор содержимого схем, включая таблицы, представления, триггеры и хранимые процедуры.

Раздел “Performance”: Раздел «Performance» предназначен для мониторинга и оптимизации производительности базы данных MySQL. Он позволяет детально анализировать использование ресурсов и выявлять потенциальные проблемы.

Мониторинг ресурсов:

- Отображение информации о загрузке процессора, использования оперативной памяти и дисковых операций.

- Графики производительности для визуализации трендов и пиков нагрузки.

Статистика запросов:

- Анализ выполненных SQL-запросов, включая их частоту, длительность и объем обработанных данных.

- Выявление медленных запросов и запросов с высокой нагрузкой на систему.

Оптимизация индексов:

- Инструменты для анализа использования индексов таблиц.

- Рекомендации по добавлению или удалению индексов для повышения производительности запросов.

Кэширование:

- Мониторинг эффективности использования кэша запросов и буферов данных.

- Настраиваемые параметры для управления размером и поведением кэша.

Журналы производительности:

- Просмотр и анализ журналов производительности, где фиксируются различные события, связанные с работой сервера.

- Использование этих данных для долгосрочного планирования и оптимизации.

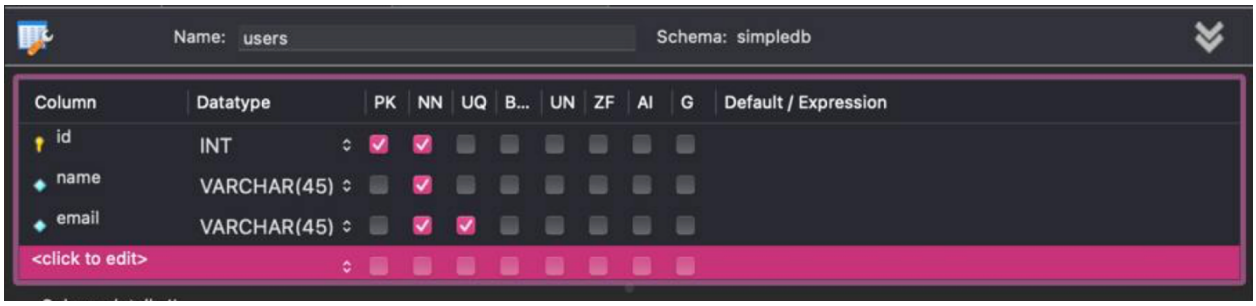
Аудит и отчеты:

- Создание отчетов о производительности за определенные периоды времени.

- Экспорт данных для дальнейшего анализа или представления руководству.

- Эти инструменты помогают администраторам баз данных своевременно реагировать на проблемы производительности и поддерживать оптимальную работу системы.

Задание 2



Задание 3

Созданная таблица:

```
CREATE TABLE `new_table` (
`id` int NOT NULL AUTO_INCREMENT,
`name` varchar(45) NOT NULL,
`email` varchar(45) NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `email_UNIQUE` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
```

Задание 4

После создания таблицы мы получаем такой SQL-запрос:

```
INSERT INTO `simple`.`new_table` (`name`, `email`) VALUES ('Paul',
'paul@superpochta.com');
INSERT INTO `simple`.`new_table` (`name`, `email`) VALUES ('Marina',
'marina.ivanova@hotmail.com');
INSERT INTO `simple`.`new_table` (`name`, `email`) VALUES ('Ekaterina',
'ekaterina.petrova@outlook.com');
```

Мы получаем такой SQL-запрос после изменения ячейки «name»:

```
UPDATE `simpledb`.`new_table` SET `name` = 'Bob' WHERE (`id` = '1');
```

Задание 5.2

Объяснение из документации: При формировании текущей даты/времени создавать строку не требу ется – следующие встроенные функции организуют доступ к систем ным часам и возвратят текущую дату и/или время в виде строки: CURRENT_TIMESTAMP( );

Столбец, отслеживающий время последнего изменения пользовате лем определенной строки таблицы, использовал бы тип **timestamp** (временная метка). Этот тип содержит ту же информацию, что и тип datetime (год, месяц, день, час, минуту, секунду), но при добавлении или изменении строки таблицы сервер MySQL автоматически за полнит столбец **timestamp** текущими значениями даты/времени.

После построения таблицы получили:

```
ALTER TABLE `simpledb`.`new_table`
ADD COLUMN `gender` ENUM('M', 'F') NULL AFTER `email`,
ADD COLUMN `bday` DATE NULL AFTER `gender`,
ADD COLUMN `postal` VARCHAR(10) NULL AFTER `bday`,
ADD COLUMN `rating` FLOAT NULL AFTER `postal`,
CHANGE COLUMN `id` `id` INT GENERATED ALWAYS AS () ;
```

Задание 5.3

Посмотрев таблицу, NULL могут иметь все значения, кроме id, так как оно явно указано NOT NULL. Поле created имеет значение по умолчанию CURRENT_TIMESTAMP(), но если оно не было явно объявлено как NOT NULL, то технически оно тоже может быть NULL.

Я не мог создать таблицу в графическом интерфейсе, поэтому я создал таблицу через терминал:

```
mysql> ALTER TABLE new_table
    -> ADD (gender ENUM('M', 'F'),
    -> bday DATE,
    -> postal VARCHAR(10),
    -> rating FLOAT,
    -> created TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
Query OK, 0 rows affected (0,01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Задание 6

Мы добавили дополнительные данные о людях в нашей БД

```
+----+-----------+-----------------------------+--------+------------+--------+--------+---------------------+
| id | name      | email                       | gender | bday       | postal | rating | created             |
+----+-----------+-----------------------------+--------+------------+--------+--------+---------------------+
|  1 | Paul      | paul@superpochta.ru         | M      | 1990-05-10 | 98765  |    4.5 | 2025-02-07 09:51:12 |
|  2 | Marina    | marina.ivanova@hotmail.ru   | F      | 1991-06-17 | 12345  |      5 | 2025-02-07 09:51:12 |
|  3 | Ekaterina | ekaterina.petrova@outlook.com | F      | 2001-10-21 | 13579  |      7 | 2025-02-07 09:51:12 |
+----+-----------+-----------------------------+--------+------------+--------+--------+---------------------+
```

Задание 7

Представленный файл содержит таблицу new_table, в которой есть данные о трех людях.

SELECT – извлекает данные из таблицы

INSERT INTO – добавляет новую запись в таблицу

UPDATE – обновляет существующие записи в таблице

DELETE – удаляет записи из таблицы

CREATE TABLE – создает новую таблицу
ALTER TABLE – изменяет структуру существующей таблицы

DROP TABLE – удаляет всю таблицу вместе со всеми ее данными

Задание 8

При удалении id из таблицы users удаляется вся строка с резюме в таблице resumes

```
mysql> SELECT * FROM resumes;
Empty set (0,00 sec)
```

Задание 9

Можно создавать бесконечное количество резюме для каждого пользователя, просто буде генерироваться новый resumeid и каждое из них удалится, если удалить пользователя в таблице user.

```
+----------+--------+--------------------+--------------------------+---------------------+
| resumeid | userid | title              | skills                   | created             |
+----------+--------+--------------------+--------------------------+---------------------+
|     1237 |      2 | Software Developer | Python, SQL, JavaScript | 2025-02-07 11:41:49 |
|     1238 |      2 | Software Developer | Python, SQL              | 2025-02-07 11:42:30 |
+----------+--------+--------------------+--------------------------+---------------------+
```

Невозможно добавить в таблицу resumes пользователя с id, которого не существует.

```
mysql> INSERT INTO resumes (userid, title, skills)
    -> VALUES (10, 'Software Developer', 'Python, SQL');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`simpledb`.`resumes`, CONSTRAINT `simpleuser` FOREIGN KEY
(`userid`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE)
```

Задание 10

После удаления id пользователя, все резюме, которые относились к этому пользователю – были удалены.

Если изменить id пользователя, то он и изменится в таблицу resumes.

```
+----+------------+--------------------------------+--------+------------+--------+--------+---------------------+
| id | name       | email                          | gender | bday       | postal | rating | created             |
+----+------------+--------------------------------+--------+------------+--------+--------+---------------------+
|  1 | John Doe   | john.doe@example.com           | M      | 1990-05-10 | 12345  |    4.5 | 2025-02-07 11:51:59 |
|  3 | Ekaterina  | ekaterina.petrova@outlook.com  | F      | 2001-10-21 | 13579  |      7 | 2025-02-07 09:51:12 |
|  4 | Jane Smith | jane.smith@example.com         | F      | 1985-08-15 | 67890  |    4.8 | 2025-02-07 11:53:23 |
+----+------------+--------------------------------+--------+------------+--------+--------+---------------------+
3 rows in set (0,01 sec)

mysql> SELECT * FROM resumes;
Empty set (0,00 sec)

mysql> INSERT INTO resumes (userid, title, skills)
    -> VALUES (1, 'Software Developer', 'Python, SQL, JavaScript');
Query OK, 1 row affected (0,01 sec)

mysql> SELECT * FROM resumes;
+----------+--------+--------------------+-------------------------+---------------------+
| resumeid | userid | title              | skills                  | created             |
+----------+--------+--------------------+-------------------------+---------------------+
|     1240 |      1 | Software Developer | Python, SQL, JavaScript | 2025-02-07 12:06:13 |
+----------+--------+--------------------+-------------------------+---------------------+
```

```
+-----+------------+--------------------------------+--------+------------+--------+--------+---------------------
| id  | name       | email                          | gender | bday       | postal | rating | created
+-----+------------+--------------------------------+--------+------------+--------+--------+---------------------
|   3 | Ekaterina  | ekaterina.petrova@outlook.com  | F      | 2001-10-21 | 13579  |      7 | 2025-02-07 09:51:12
|   4 | Jane Smith | jane.smith@example.com         | F      | 1985-08-15 | 67890  |    4.8 | 2025-02-07 11:53:23
| 101 | John Doe   | john.doe@example.com           | M      | 1990-05-10 | 12345  |    4.5 | 2025-02-07 11:51:59
+-----+------------+--------------------------------+--------+------------+--------+--------+---------------------
3 rows in set (0,00 sec)

mysql> SELECT * FROM resumes;
+----------+--------+--------------------+-------------------------+---------------------+
| resumeid | userid | title              | skills                  | created             |
+----------+--------+--------------------+-------------------------+---------------------+
|     1240 |    101 | Software Developer | Python, SQL, JavaScript | 2025-02-07 12:06:13 |
+----------+--------+--------------------+-------------------------+---------------------+
```