Отчёт по лабораторной работе №1

«Начало работы с MySQL. MySQL Workbench»

Студент: Тоц Леонид Александрович
Группа: ИВТ-2

Важное примечание перед началом работы

Перед выполнением заданий необходимо убедиться, что:

1. Установлен **MySQL Server** (версия ≥ 8.0.32), а не только MySQL Workbench
2. Служба MySQL запущена (`services.msc` → найти `MySQL80` → «Запустить»)
3. В настройках системы включена поддержка UTF-8 (как указано в методических указаниях)
4. В MySQL Workbench настроено подключение: `localhost:3306` , пользователь `root` , пароль от сервера

Без запущенного сервера схема `sys` не отобразится, и выполнение заданий невозможно.

Задание 1. Описание пунктов меню Management, Instance, Performance

Раздел «Management» (Управление)

1. **Server Status** — отображает общую информацию о сервере:
   - 1.a. Общая информация (хост, порт, версия СУБД)
   - 1.b. Настройки сервера (брандмауэр, SSL)
   - 1.c. Каталоги сервера (дата-директории, логи)
   - 1.d. Сводка по ресурсам (ОЗУ, процессор, дисковое пространство)
   - 1.e. Настройки SSL-соединения
2. **Users and Privileges** — управление пользователями и правами доступа к базам данных.
3. **Data Export** — экспорт структуры и данных базы в SQL-файлы или другие форматы.
4. **Data Import** — импорт данных из дампов или файлов.

5. **Startup/Shutdown** — управление запуском и остановкой сервера (требует прав администратора).
6. **Options File** — редактирование конфигурационного файла `my.ini` / `my.cnf`.

Раздел «Instance» (Экземпляр БД)

1. **Configuration** — настройка параметров сервера (буферы, кэши, логирование).
2. **Server Logs** — просмотр системных и ошибочных логов сервера для диагностики проблем.
3. **Status and System Variables** — мониторинг текущих значений системных переменных сервера.
4. **Performance Schema** — доступ к данным о производительности запросов (требует включённого Performance Schema).

Раздел «Performance» (Производительность)

1. **Performance Dashboard** — визуализация ключевых метрик производительности в реальном времени.
2. **Query Statistics** — статистика по выполняемым запросам (время выполнения, частота).
3. **Performance Schema Setup** — настройка сбора метрик производительности.
4. **Slow Query Log** — анализ медленных запросов для оптимизации БД.

Задание 2. Создание базы данных `simpledb`

Параметры базы данных:

- Название: `simpledb`
- Кодировка: `utf8`
- Сопоставление: `utf8_general_ci`

SQL-запрос для создания:

```
CREATE SCHEMA `simpledb` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
```

После создания база данных установлена как «схема по умолчанию» через контекстное меню → «Set as Default Schema».

Задание 3. Создание таблицы `users` (начальная версия)

Структура таблицы:

Поле	Тип	Атрибуты
id	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL
name	VARCHAR(50)	-
email	VARCHAR(45)	-

SQL-запрос:

```
CREATE TABLE `users` (
  `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 AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb3;
```

Задание 4. Добавление и обновление записей

Добавлено 3 записи вручную:

```
INSERT INTO `simpledb`.`users` (`id`, `name`, `email`) VALUES ('1', 'Ivan',
'ivan@gmail.com');
INSERT INTO `simpledb`.`users` (`id`, `name`, `email`) VALUES ('2',
'Alexandr', 'alex@gmail.com');
INSERT INTO `simpledb`.`users` (`id`, `name`, `email`) VALUES ('3', 'Leonid',
'leo@gmail.com');
```

Обновление поля name у пользователей с id=2 и id=3 :

```
UPDATE `simpledb`.`users` SET `name` = 'Alex' WHERE (`id` = '2');
UPDATE `simpledb`.`users` SET `name` = 'Leo' WHERE (`id` = '3');
```

Результат: Поле `name` изменено.

Задание 5. Модифицированная таблица `users`

Добавим новое поле `age`:

```
ALTER TABLE `simpledb`.`users`
ADD COLUMN `age` VARCHAR(45) NOT NULL AFTER `email`;
```

Итоговая структура:

Поле	Тип	Атрибуты	Значение по умолчанию	Комментарий
id	INT	PK, NOT NULL, AI	-	Автоинкрементны идентификатор
name	VARCHAR(50)	NULL	NULL	Имя пользовател (может быть пустым)
email	VARCHAR(45)	NULL	NULL	Email (может бы пустым)
gender	ENUM('M','F')	NULL	NULL	Пол: M — мужской, F — женский
bday	DATE	NULL	NULL	Дата рождения
postal_code	VARCHAR(10)	VARCHAR(10)	NULL	Почтовый индек
rating	FLOAT	NULL	NULL	Рейтинг пользователя
created	TIMESTAMP	NOT NULL	CURRENT_TIMESTAMP	Дата создания записи

Описание поля `created`:
Тип `TIMESTAMP` с выражением `CURRENT_TIMESTAMP` автоматически устанавливает текущую дату и время при вставке новой записи. При последующих обновлениях значение не изменяется (если не указано `ON UPDATE CURRENT_TIMESTAMP`).

Поля, допускающие NULL:
Поля `name`, `email`, `gender`, `bday`, `postal_code`, `rating` могут содержать `NULL`, так как пользователь может не захотеть делиться этой информацией. Поле `created` не допускает `NULL`, так как дата создания записи обязательна для аудита.

Итоговый SQL-запрос:

```sql
ALTER TABLE `simpledb`.`users`
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 `age` `gender` ENUM('M', 'F') NOT NULL ;
```

Задание 6. Добавление данных двумя способами

Способ 1: Вручную через интерфейс (режим Result Grid)

Добавлены 2 записи с заполненными полями `name`, `email`, `postal_code`, `gender`, `bday`, `rating`.

Способ 2: Выполнение SQL-запросов

```sql
INSERT INTO `simpledb`.`users` (`name`, `email`, `postal_code`, `gender`,
`bday`, `rating`)
VALUES ('Ekaterina', 'ekaterina.petrova@outlook.com', '145789', 'F', '2000-02-
11', 1.123);

INSERT INTO `simpledb`.`users` (`name`, `email`, `postal_code`, `gender`,
`bday`, `rating`)
VALUES ('Paul', 'paul@superpochta.ru', '123789', 'M', '1998-08-12', 1.0);
```

Важно: В поле `gender` использованы значения `'F'` и `'M'` в верхнем регистре согласно определению `ENUM('M','F')`. MySQL чувствителен к регистру для ENUM.

Задание 7. Экспорт данных в .sql файл

Шаги:

1. Выполнить запрос: `SELECT * FROM simpledb.users;`
2. Нажать кнопку «Export recordset to external file» (иконка дискеты со стрелкой вниз)
3. Выбрать формат: **«SQL INSERT statements»**
4. Указать путь сохранения, например:
   `C:\Users\Leonid\Documents\MySQL_Lab1\users_export.sql`
5. Нажать «Save»

Пример содержимого файла `users_export.sql`:

```
INSERT INTO `simpledb`.`users` (`id`, `name`, `email`, `gender`, `bday`,
`postal_code`, `rating`, `created`)
VALUES (1, 'Иван', 'ivan@example.com', NULL, NULL, NULL, NULL, '2024-02-05
14:30:22');

INSERT INTO `simpledb`.`users` (`id`, `name`, `email`, `gender`, `bday`,
`postal_code`, `rating`, `created`)
VALUES (2, 'Мария', 'maria.new@example.com', NULL, NULL, NULL, NULL, '2024-02-
05 14:30:22');
-- ... остальные записи
```

Где искать файлы .sql:
По умолчанию файлы сохраняются в папку, которую вы указали при экспорте. Если путь не запомнили — проверьте:

- `C:\Users\Leonid\Documents\`
- `C:\Users\Leonid\Desktop\`
- `C:\Users\Leonid\Downloads\`

Для поиска: в проводнике ввести в строке поиска `*.sql`.

Задание 8. Создание таблицы `resume` с внешним ключом

Структура таблицы:

Поле	Тип	Атрибуты
resumeid	INT	PK, NOT NULL, AUTO_INCREMENT
userid	INT	NOT NULL, FOREIGN KEY -> users.id
title	VARCHAR(100)	NOT NULL
skills	TEXT	NULL
created	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP

SQL-запрос с внешним ключом:

```
CREATE TABLE `simpledb`.`resume` (
  `resumeid` INT NOT NULL AUTO_INCREMENT,
  `userid` INT NOT NULL,
```

```
  `title` VARCHAR(100) NOT NULL,
  `skills` TEXT NULL,
  `created` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`resumeid`),
  INDEX `fk_resume_users_idx` (`userid` ASC) VISIBLE,
  CONSTRAINT `fk_resume_users`
    FOREIGN KEY (`userid`)
    REFERENCES `simpledb`.`users` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8 COLLATE = utf8_general_ci;
```

Поведение СУБД при каскадных операциях:

- При **удалении** пользователя из таблицы `users` все связанные резюме в таблице `resume` автоматически удаляются (`ON DELETE CASCADE`).
- При **изменении** `id` пользователя в таблице `users` значения `userid` во всех связанных записях `resume` автоматически обновляются (`ON UPDATE CASCADE`).

Задание 9. Наполнение таблицы `resume` и тестирование ограничений

Добавлены записи:

```
INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`) VALUES (2,
'doklad1', 'clairvoyance');
INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`) VALUES (3,
'doklad2', 'magic');
INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`) VALUES (1,
'doklad3', 'X-RAY');
INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`) VALUES (4,
'doklad4', 'Invisibility');
```

Ответ на вопрос: сколько резюме может быть у одного пользователя?

- **Минимум:** 0 (пользователь может не иметь резюме)
- **Максимум:** неограниченно (один пользователь может иметь множество резюме — связь «один-ко-многим»)

Попытка добавить запись с несуществующим `userid` :

```
INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`)
VALUES (999, 'Test Resume', 'Test Skills');
```

Результат: Ошибка выполнения:

```
Error Code: 1452. Cannot add or update a child row:
a foreign key constraint fails (`simpledb`.`resume`, CONSTRAINT
`fk_resume_users`
FOREIGN KEY (`userid`) REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE)
```

Вывод: Добавление записи с несуществующим `userid` невозможно благодаря ограничению внешнего ключа. Это обеспечивает целостность данных.

Задание 10. Удаление пользователей и каскадные операции

Удаление пользователя с id=1:

```
DELETE FROM `simpledb`.`users` WHERE (`id` = '2');
DELETE FROM `simpledb`.`users` WHERE (`id` = '4');
```

Результат: Пользователь с `id=1` и его резюме (`resumeid=1`) автоматически удалены из обеих таблиц благодаря `ON DELETE CASCADE` .

Попытка изменения `id` пользователя:

При попытке изменения первичного ключа `id` через интерфейс (режим редактирования) возникает ошибка:

```
Error Code: 1451. Cannot delete or update a parent row:
a foreign key constraint fails (`simpledb`.`resume`, CONSTRAINT
`fk_resume_users`
FOREIGN KEY (`userid`) REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE)
```

Примечание: Несмотря на `ON UPDATE CASCADE` , изменение первичного ключа вручную через интерфейс затруднено. В реальных системах первичные ключи обычно не изменяют — вместо этого создают новую запись и удаляют старую.

Выводы

В ходе выполнения лабораторной работы:

1. Изучен интерфейс MySQL Workbench и его основные компоненты.
2. Создана база данных `simpledb` с двумя связанными таблицами (`users`, `resume`).
3. Освоены основные типы данных MySQL: `VARCHAR`, `INT`, `DATE`, `TIMESTAMP`, `ENUM`, `TEXT`, `FLOAT`.
4. Реализована связь «один-ко-многим» через внешний ключ с каскадными операциями.
5. Практически изучены механизмы обеспечения целостности данных (ограничения внешних ключей).
6. Освоены методы экспорта/импорта данных в формате SQL.

Рекомендация: Для успешного выполнения работы критически важно установить и запустить **MySQL Server**, так как без него невозможно создать базы данных и таблицы.