Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему:

БАЗА ДАННЫХ СИСТЕМЫ ДЛЯ ОРГАНИЗАЦИИ ФЕСТИВАЛЕЙ

БГУИР КП 1-40 01 01 003 ПЗ

Студент: гр. 651001 Арабей Д. И.

Руководитель: Фадеева Е.Е.

Минск 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день широко распространены веб-сервисы для организации и участия в различных мероприятиях. Данные ресурсы позволяют охватить более широкую аудиторию, повысить скорость отклика. Зачастую лица, заинтересованные в посещении такого рода мероприятий хотели бы иметь возможность рассмотреть различные варианты, узнать более детальную информацию.

В данном курсовом проекте производится разработка веб-приложения для организации и принятия участия в таком виде мероприятий.

Программный комплекс должен обеспечивать высокий уровень защиты данных пользователей, так как в базе будут храниться контактные данные клиентов, платежные данные, информация о произведенных транзакциях, реквизиты. Кроме того, система должна обладать достаточной гибкостью, так как в рамках постоянно изменяющегося рынка функциональность будет меняться. Данный сервис должен иметь возможность интегрироваться с системами других служб, а также быть легко модифицируемым, так как предметная область предполагает постоянное добавление нового функционала, а так же усложенение уже существующего за счет, например, усложения модели оплаты.

В качестве курсовой работы было решено разработать базу данных для программного комплекса, автоматизирующего регистрацию и оплату желаемого к посещению мероприятия.

В качестве системы управления базами данных была выбрана обьектно-реляционная MySQL.

В данной пояснительной записке отображены следующие этапы. В первой части - анализ предметной области, информационных потребностей пользователей и постановка задачи. Во второй - выбор системы управления базами данных и иных технических средств разработки. Построение инфологической модели базы данных. Далее будет построение непосредственно самой схемы данных и модели БД. И в конечном итоге будет представлена разработка бизнес-логики базы данных.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ существующих аналогов

1.1.1 GetYourGuide

GetYourGuide (сайт на русском — getyourguide.ru) — крупнейший международный сервис, где можно найти развлечения по всему миру. Это не только групповые и индивидуальные экскурсии, но и, например, различные мастер-классы (кулинария, йога, рисование и т.д.), входные билеты в музеи, чтобы избежать очередей, туристические автобусы (city sightseeing) и множество других.

Минус у GetYourGuide только один. Большинство экскурсий проводятся на английском языке.

1.1.2 Musement

Musement (сайт — musement.com) — лидирующий онлайн сервис для поиска и бронирования 25 000+ развлечений в Лондоне, Барселоне, Риме, Флоренции и 1000+ других направлениях по всему миру. Сайт доступен на 8 языках, в том числе и на русском. Перевод на русский язык сделан автоматически.

Минус у сервиса такой же как и у предыдущего конкурента — большинство мероприятий на английском.

Преимущества Musement: можно найти некоторые активности, которых нет больше ни у кого, включая временные и эксклюзивные. А также множество входных билетов без очереди, практически во все достопримечательности в Европе.

1.1.3 Klook

Klook (сайт — klook.com) — огромнейшая площадка на которой представлены десятки тысяч развлечений по всем странам Азии.

В этой части мира Klook несомненный фаворит и предлагает эксклюзивные билеты и мероприятия, которые больше нельзя забронировать нигде. Кроме того, у Klook есть скидки, по которым билеты даже дешевле, чем напрямую у организаторов, так например, билеты в Gardens by Bay (Сады у залива) в Сингапуре можно купить дешевле на 35%. А в Гонконге на канатную дорогу отдельная ВИП очередь для тех, кто купил билет на Klook.

1.2 Анализ предполагаемых пользователей

К предполагаемым пользователям относятся:

- Незарегистрированный пользователь;
- Зарегистрированный пользователь;
- Администратор;
- Выступающие на мероприятиях.

С учётом вышеописанных особенностей предметной области можно выделить

следующие функциональные требования:

- Регистрация пользователя;
- Авторизация пользователя;
- Работа пользователя с каталогом (просмотр);
- Оформление заявки пользователем;
- Подтверждение заявки администратором;
- Работа администратора с каталогом мероприятий (добавление, редактирование, удаление);
- Добавление, редактирование, удаление выступающих на мероприятии.
- Просмотр администатором информации о мероприятии, такой как: количество проданных билетов, полученная прибыль, сумма расходов, необходимое для мероприятия оборудование;
- Возможность пользователя оплатить билет, заказать себе его;
- Возможность пользователей увидеть историю своих платежей;
- Возможность администраторов увидеть спонсоров мероприятия, это могут быть как как компании, так и индивиды;
- Возможность администраторов увидеть организаторов мероприятия, это могут быть как как компании, так и индивиды;
- Возожность администратора редактировать детальную информацию о мероприятии;
- Наличие служебной информации в случаи ошибки в приложении
- Наличие информации о сотрудниках комапнии-организатора, о их профессиональных способностях;
- Каталог выступающих на мероприятии;
- Возможность просмореть информацию о месте проведения мероприятия(адресс, вместимость);
- Возможность получить более детальную информацию о каждом из выступающих;
- Каталог персонала, обслуживающего представленное мероприятие, а также место их работы;
- Возможность осуществлять поиск для нахождения любого вида информации, представленного в приложении;
- Возможность оценить мероприятие;
- Возможность оценить организаторов;
- Возможность написания сообщения в службу поддержки;
- Наличие информации о количестве свободных мест;
- Возможность на некоторое время забронировать место с последующей оплатой;

1.3 Постановка задачи

Предметная область представляет собой базу данных сайта для организации мероприятий. В данной БД хранится информация о пользователях, мероприятиях, выступающих. База данных является связующим звеном между поставщиком услуг – администратором сайта и потребителем – клиентом сайта. База данных должна справляться с достаточно большим количеством подключений и обработкой данных в режиме реального времени, обеспечивать высокий уровень защиты данных и обладать достаточной гибкостью.

2 РАЗРАБОТКА МОДЕЛИ БАЗЫ ДАННЫХ

2.1 Выбор СУБД и иных средств разработки

В качестве системы управления базой данных используется обьектно-реляционная MySQL.

Преимущества MySQL:

1) Быстродействие. Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко.

2)Безопасность. Довольно высокий уровень безопасности обеспечивается благодаря базе данных mysql, создающейся при установке пакета и содержащей пять таблиц. При помощи этих таблиц можно описать, какой пользователь из какого домена с какой таблицей может работать и какие команды он может применять. Пароли, хранящиеся в базе данных, можно зашифровать при помощи встроенной в MySQL функции password().

3)Лицензия. Раньше лицензирование MySQL было немного запутанным; сейчас эта программа для некоммерческих целей распространяется бесплатно.

4)Открытость кода. Благодаря этому вы сможете сами добавлять в пакет нужные функции, расширяя его функциональность так, как вам требуется. Кстати, за отдельную плату для вас это могут сделать и сами авторы MySQL. Чтобы заказать расширение MySQL у создателей пакета, просто зайдите на сайт http://www.mysql.com и заполните соответствующую форму.

5)Надежность. Этот пакет довольно стабилен и его трудно вывести из строя.

6)Ресурсы. Это может зависеть от разных факторов, но в любом случае суперкомпьютер вам не потребуется.

7)Сообщество. Как следствие открытости кода, бесплатности программы, стабильной и надежной ее работы образовалось сообщество людей, которые не просто лояльны к MySQL, но и всячески участвуют как в развитии самого пакета, так и в обучении менее опытных людей работе с ним. Существует огромное количество листов рассылки и конференций, где можно получить бесплатную помощь в любое время суток.

8)Переносимость. В настоящее время существуют версии программы для большинства распространенных компьютерных платформ. Это говорит о том, что вам не навязывают определенную операционную систему. Вы сами можете выбрать, с чем работать, например с Linux или Windows, но даже в случае замены ОС вы не потеряете свои данные и вам даже не понадобятся дополнительные инструменты для их переноса.

Недостатки MySQL:

1) Недостаточная надежность. В вопросах надежности некоторых процессов по работе с данными (например, связь, транзакции, аудит) MySQL уступает некоторым другим СУБД.

2) Низкая скорость разработки. Как и многим другим программным

продуктам с открытым кодом, MySQL не достает некоторого технического совершенства, что порой сказывается на эффективности процессов разработки.

Так как серверная часть системы разрабатывается на Java и учитывая достаточно высокое количество запросов, требующих постоянных расчетов в реальном времени расстояния/времени/цены, многие вычисления производятся на стороне базы данных. Таким образом MySql выбран в качестве базы данных благодаря быстроте, достаточной надёжности, переносимости.

А также плюсом несомненно является то, что данное средство с открытым кодом.

2.2 Разработка инфологической модели предметной области

В базе данных будут использоваться следующие объекты:
1) Пользователь — человек, который может использовать функционал сайта.
2) Роль пользователя – (администратор, зарегистрированный / незарегистрированный пользователь).
3)Регистрационная информация
4)Контактная информация – информация, идентифицирующая пользователей.
5)Выступающий (performer) – человек, выступающий на мероприятии.
6)Участник (participant) – человек, участвующий в мероприятии.
7)Заявка — запрос на участие в мероприятии.
8)Тип мероприятия (type).
9)Место проведения(place) – место, где проводится мероприятие.
10) Мероприятие — кино, научный, исторический фестиваль;
11)Платёж (payment) –оплата участником билета;
12)Метод платежа(paymentMethod) - наличные, перевод, ЕРИП;
13) Роль у пользователя(user has role) – сапоставление роли и пользователя;
14)Информация о мероприятии (event Details) ;
15) Расходы мероприятия (outlay);
16) Тип расходов (outlayType);
17) Прибыль мероприятия (profit);
18) Тип прибыли (profitType);
19) Требования для мероприятия(requirement) –какое оборудование необходимо, его цена и количество;
20)Оборудование (equipment);
21) Тип оборудования (equipmentType);
22)Адресс места проведения (address);
23)Здание места проведения (building);
24) Зал места проведения (room);
25) Город места проведения (city);
26) Страна места проведения (country);
27)Выступающий и мероприятия;
28)Высупающий и навыки (performer has skills);
29) Навыки выступаюих(performerSkills);

30) Лог ошибок (Error Log) ;
31) Транспорт у мероприятия (Event has transport);
32) Транспорт;
33) Тип транспорта;
34) Организаторы у мероприятия (event has organizer);
35) Спонсоры у мероприятия (event has sponsors);
36) Комапния;
37) Информация о компании;
38) Тип компании;
39) Сотрудники компании(company has employee);
40) Сотрудники (employee);
41) Информация о сотрудниках (employee details);
42) Навыки сотрудников(employee has skills);
43)Навыки;
44)Выполняемая сотрудниками работа;
45) Работа сотрудников (employee Work);
46) Тип работы, выполняемой сотрудником(workType);
47)Спонсоры;
48) Информация о спонсорах.

Сущности и их атрибуты:

Пользователь — уникальный идентификатор, идентификатор контактной информации и регистрационной;

Роль пользователя – уникальный идентификатор, название, описание;

Регистрационная информация – уникальный идентификатор, логин, пароль;

Контактная информация – уникальный идентификатор, имя, фамилия, возраст, почта, телефон;

Выступающий – уникальный идентификатор, контактная информация, род деятельности, награды;

Заявка — мероприятие, номер платежа, пользователь;

Тип мероприятия – уникальный идентификатор, название, описание;

Место проведения– уникальный идентификатор, адресс, максимально количество посктителей;

Мероприятие — тип, место, детали, название, расходы, прибыль, уникальный идентификатор;

Платёж – уникальный идентификатор, дата, результат, количество денег, метод платежа;

Метод платежа– уникальный идентификатор, название, описание;

Роль у пользователя– уникальный идентификатор, название, описание;

Информация о мероприятии – уникальный идентификатор, дата, описание, количество участников;

Расходы мероприятия – уникальный идентификатор, количество, тип, название;

Тип расходов – уникальный идентификатор, название;

Прибыль мероприятия – уникальный идентификатор, количество, тип, название;
Тип прибыли - уникальный идентификатор, название;
Требования для мероприятия– количество, цена;
Оборудование – уникальный идентификатор, название, доступность;
Тип оборудования – уникальный идентификатор, название;
Адресс места проведения – уникальный идентификатор, строение, город;
Здание места проведения – улица, название,номер;
Зал места проведения - – уникальный идентификатор, номер, вместимость;
Город места проведения - уникальный идентификатор, назание, идентификатор страны;
Страна места проведения - уникальный идентификатор, название;
Выступающий и мероприятия – идентификатор выступающего и мероприятия;
Высупающий и навыки – идентификаторы выстуающего и навыков;
Навыки выступаюих - - уникальный идентификатор, название;
Лог ошибок - - уникальный идентификатор, состояние, сообщение, время;
Транспорт у мероприятия – идентификатор транспорта и мероприятия;
Транспорт – уникальный идентификатор, количество, идентификатор типа;
Тип транспорта- уникальный идентификатор, название;
Организаторы у мероприятия – идентификатор организатора и меропрриятия
Спонсоры у мероприятия – уникальный идентификатор, контактная информация;
Компания – уникальный идентификатор, название;
Информация о компании – идентификатор компании, описание, идентификатор типа;
Тип компании – уникальный идентификатор, название;
Сотрудники компании- идентификаторы компании и сотрудников;
Сотрудники – уникальный идентификатор, профессия, идентификатор на подробную информацию;
Информация о сотрудниках – национальность, виза, водительские права, резюме;
Навыки сотрудников – идентификаторы навыков и сотрудников;
Навыки – уникальный идентификатор, назание, описание;
Выполняемая сотрудниками работа- идентификатор вида деятельности и сотрудника;
Работа сотрудников – уникальный идентификатор, потраченное время, зарплата, рабочее время, идентификатор типа работы;
Тип работы, выполняемой сотрудником – уникальный идентификатор, название;
Спонсоры – профессия, контактная информация, уникальный идентификатор;
Информация о спонсорах – уникальный идентификатор, контактная информация, количество выданных денег.

2.3 Особенности нормализации

При сохранении данных платежа (например, при оплате заказа или проверке платежеспособности при привязке банковской карты к аккаунту) в таблицу payment в поле помещается полный ответ сервиса, использующегося для проведения оплаты. Он представляет собой JSON с данными платежа, содержащего вложенные обьекты и состоящего из примерно сотни полей. Такой способ хранения не совсем соответствует первой нормальной форме. Но так как эти данные требуются лишь в исключительных случаях (сбоях системы), было решено не тратить ресурсы на то, чтобы сделать их атомарными.

3 РАЗРАБОТКА БИЗНЕС-ЛОГИКИ БАЗЫ ДАННЫХ

3.1 Разработка триггеров базы данных

Триггер update_count_event_trigger срабатывает при заказе нового билета пользователем, таким образом количество доступных билетов становится меньше на то количество, которое заказали.

Триггеры profit_balance_trigger и outlay_balance_trigger обновляют данные о gприбыли и расходах проведённого мероприятия, в зависимости от спонсоров и организаторов.

Триггер check_validation_trigger проверяет валидность регистрации и авторизации при вводе логина и пароля пользователя.

3.2 Разработка процедур базы данных

Процедура update_payment обновляет данные платежа при оплате заказа и заносит данные в таблицу транзакций.

Процедура profit_add_balance пополняет баланс прибыли, сохраняя данные в соответствующие таблицы.

Процедура set_organizer устанавливает организатора, начальную стоимость и статус заказа, а также обновляет необходимую для выполнения работу.

Процедура outlay_balance вычисляет расход мероприятия. Деньги потраченные на оборудование, персонал, транспорт, а также другие необходимые реквизиты для выступающих.

Процедура update_event обновляет количество оставшихся билетов.

Процедура show_user показывает ограниченое количество пользователей в целях быстродействия.

Процедура show_event показывает ограниченое количество мероприятий в целях быстродействия.

Процедура show_places показывает ограниченое количество мест проведения мероприятия в целях быстродействия.

3.3 Индексы

Для быстрого поиска и получения данных из таблиц было решено использовать следующие индексы:

1) Role:
- ID;
- name;

2) User:
- ID;
- RegInfoID;
- ContactInfoID;

3) Bells :
- TermID, LessonNumberID, WeekDayID (unique);

4) eventDetails:
- ID;
- date;
- countOfMembers;

5) event:
- ID;
- name(unique);

6) outlay:
- ID;
- amount;

7) profit:
- ID;
- amount;

8) ContactInfo:
- ID (unique);
- firstname;
- surnname;

9) place:
- ID;
- AddressID;
- maxAmountOfParticipant

10) address:
- ID;
- CityId

11) Employees:
- ID;
- employee_Details_ID, ContactInfo (unique);

12) Performer:
- ID;
- profession;
- ContactInfoID;

13) Organizer:
- ID;
- profession;

14) Sponsor:
- ID,
- profession;

15) Company:
- ID;
- name;

16) CompanyDetails:
- ID;
- ComponyType_ID;

17) Skills:
- ID;

18) employeeWork:
- ID
- salary, workingHours;

19) typeOfWork:
- ID;
- Name;

20) typeOfEvent:

- ID;
- Name(unique);

21) Building:
- ID;
- Street;
-Number;

22) Country:
- ID;
- Name;

23) performerSkills:
- ID;
- name;

24) room:
- Id;
- Number;

25) equipment:
- ID;
- Name;

26) equipmentType:
- ID;
- Name;

ЗАКЛЮЧЕНИЕ

В результате работы над курсовым проектом была разработана база приложения для организации мероприятий, удовлетворяющая разработанным требованиям.

Работа была разделена на этапы, такие как изучение предметной области, разработка требований по безопасности, производительности и гибкости архитектуры базы данных, постановка функциональных требований, разработка алгоритма обработки данных, выбор сиситемы управления базами данных и вспомогательных технических средств, построение инфологической модели предметной области, схемы базы данных на ее основе, разработка процедур, функций и тригерров, заполнение базы данными и проверка ее работы.

База данных успешно справляется с высокой нагрузкой, обеспечивает требуемый уровень защиты данных пользователей, имеет возможность интегрироваться с системами других служб, а также является легко модифицируемой, так как предметная область предполагает постоянное добавление нового функционала и расширение существующего.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Куликов, С. С. К90 Работа с MySQL, MS SQL Server и Oracle в примерах : практ. пособие. / С. С. Куликов. — Минск: БОФФ,. 2016. — 556 с.

[2] MySQL Documentation [Электронный ресурс] / – Режим доступа: https://dev.mysql.com/doc// – Дата доступа: 25.11.2019

[3] PostGIS Documentation – Режим доступа: https://postgis.net/documentation/ – Дата доступа: 28.11.2019

[4] MySQL Workbench - MySQL Tools [Электронный ресурс] / – Режим доступа: https://www.workbenchMySQL.org/ – Дата доступа: 28.11.2019

[5] dbdesc Official Site [Электронный ресурс] / – Режим доступа: https://dbmstools.com/tools/dbdesc#database-documentation-tools– Дата доступа: 28.11.2019

[6] ГОСТ 19.701–90 (ИСО 5807–85) [Текст]. – Единая система программной документации: Сб. ГОСТов. - М.: Стандартинформ, 2005 с.

ПРИЛОЖЕНИЕ А
Схема базы данных на языке SQL

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_
DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema fest
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema fest
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `fest` DEFAULT CHARACTER SET utf8 ;
-- -----------------------------------------------------
-- Schema fest
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema fest
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `fest` DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci ;
USE `fest` ;

-- -----------------------------------------------------
-- Table `fest`.`RegInfo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`RegInfo` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `login` VARCHAR(100) NOT NULL,
  `password` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `login_UNIQUE` (`login` ASC) INVISIBLE,
  INDEX `regInfo` (`login` ASC, `password` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`Contact_info`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`Contact_info` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(100) NOT NULL,
  `surname` VARCHAR(100) NOT NULL,
```

```
  `age` INT NULL,
  `mail` VARCHAR(45) NULL,
  `phone` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `firstname_surname` (`firstname` ASC, `surname` ASC) VISIBLE,
  INDEX `contactInfo` (`firstname` ASC, `surname` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`user` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `RegInfo_ID` INT NOT NULL,
  `Contact_info_ID` INT NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_user_RegInfo1_idx` (`RegInfo_ID` ASC) VISIBLE,
  INDEX `fk_user_Contact_info1_idx` (`Contact_info_ID` ASC) VISIBLE,
  CONSTRAINT `fk_user_RegInfo1`
    FOREIGN KEY (`RegInfo_ID`)
    REFERENCES `fest`.`RegInfo` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_user_Contact_info1`
    FOREIGN KEY (`Contact_info_ID`)
    REFERENCES `fest`.`Contact_info` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`eventType`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`eventType` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `typeEvent` (`name` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`country`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`country` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
```

```
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`city`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`city` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `country_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_city_country1_idx` (`country_id` ASC) VISIBLE,
  CONSTRAINT `fk_city_country1`
    FOREIGN KEY (`country_id`)
    REFERENCES `fest`.`country` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`building`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`building` (
  `id` INT NOT NULL,
  `street` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NULL,
  `number` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`adress`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`adress` (
  `id` INT NOT NULL,
  `city_id` INT NOT NULL,
  `building_id` INT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_adress_city1_idx` (`city_id` ASC) VISIBLE,
  INDEX `fk_adress_building1_idx` (`building_id` ASC) VISIBLE,
  INDEX `addres` (`building_id` ASC, `city_id` ASC) VISIBLE,
  CONSTRAINT `fk_adress_city1`
    FOREIGN KEY (`city_id`)
    REFERENCES `fest`.`city` (`id`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_adress_building1`
    FOREIGN KEY (`building_id`)
    REFERENCES `fest`.`building` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`place`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`place` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `maxAmountOfParticipants` INT NOT NULL,
  `adress_id` INT NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_place_adress1_idx` (`adress_id` ASC) VISIBLE,
  CONSTRAINT `fk_place_adress1`
    FOREIGN KEY (`adress_id`)
    REFERENCES `fest`.`adress` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`eventDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`eventDetails` (
  `id` INT NOT NULL,
  `date` DATE NOT NULL,
  `description` VARCHAR(145) NULL,
  `countOfMembers` INT NOT NULL,
  `availableCountOfMembers` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `eventDate` (`date` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`outlayType`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`outlayType` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `fest`.`outlay`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`outlay` (
  `id` INT NOT NULL,
  `amount` VARCHAR(45) NULL,
  `outlayType_id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_outlay_outlayType1_idx` (`outlayType_id` ASC) VISIBLE,
  INDEX `outlayName` (`name` ASC) VISIBLE,
  CONSTRAINT `fk_outlay_outlayType1`
    FOREIGN KEY (`outlayType_id`)
    REFERENCES `fest`.`outlayType` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`profitType`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`profitType` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`profit`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`profit` (
  `id` INT NOT NULL,
  `amountMoney` VARCHAR(45) NULL,
  `profitType_id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_profit_profitType1_idx` (`profitType_id` ASC) VISIBLE,
  INDEX `profitName` (`name` ASC) VISIBLE,
  CONSTRAINT `fk_profit_profitType1`
    FOREIGN KEY (`profitType_id`)
    REFERENCES `fest`.`profitType` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `fest`.`Event`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`Event` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `type_id` INT NOT NULL,
  `place_ID` INT NOT NULL,
  `eventDetails_id` INT NOT NULL,
  `name` VARCHAR(145) NOT NULL,
  `outlay_id` INT NOT NULL,
  `profit_id` INT NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_Event_type1_idx` (`type_id` ASC) VISIBLE,
  INDEX `fk_Event_place1_idx` (`place_ID` ASC) VISIBLE,
  INDEX `fk_Event_eventDetails1_idx` (`eventDetails_id` ASC) VISIBLE,
  INDEX `fk_Event_outlay1_idx` (`outlay_id` ASC) VISIBLE,
  INDEX `fk_Event_profit1_idx` (`profit_id` ASC) VISIBLE,
  CONSTRAINT `fk_Event_type1`
    FOREIGN KEY (`type_id`)
    REFERENCES `fest`.`eventType` (`id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Event_place1`
    FOREIGN KEY (`place_ID`)
    REFERENCES `fest`.`place` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Event_eventDetails1`
    FOREIGN KEY (`eventDetails_id`)
    REFERENCES `fest`.`eventDetails` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_outlay1`
    FOREIGN KEY (`outlay_id`)
    REFERENCES `fest`.`outlay` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_profit1`
    FOREIGN KEY (`profit_id`)
    REFERENCES `fest`.`profit` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`Performer`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `fest`.`Performer` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `profession` VARCHAR(255) NULL,
  `Contact_info_ID` INT NOT NULL,
  `awards` VARCHAR(255) NULL,
  PRIMARY KEY (`ID`),
  INDEX `fk_Performer_Contact_info1_idx` (`Contact_info_ID` ASC) VISIBLE,
  INDEX `performerInd` (`profession` ASC, `Contact_info_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Performer_Contact_info1`
    FOREIGN KEY (`Contact_info_ID`)
    REFERENCES `fest`.`Contact_info` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`Performer_has_Event`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`Performer_has_Event` (
  `Performer_ID` INT NOT NULL,
  `Event_ID` INT NOT NULL,
  INDEX `fk_Performer_has_Event_Event1_idx` (`Event_ID` ASC) VISIBLE,
  INDEX `fk_Performer_has_Event_Performer1_idx` (`Performer_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Performer_has_Event_Performer1`
    FOREIGN KEY (`Performer_ID`)
    REFERENCES `fest`.`Performer` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Performer_has_Event_Event1`
    FOREIGN KEY (`Event_ID`)
    REFERENCES `fest`.`Event` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`role`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`role` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(145) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`role_has_user`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`role_has_user` (
  `role_id` INT NOT NULL,
  `user_ID` INT NOT NULL,
  INDEX `fk_role_has_user_user1_idx` (`user_ID` ASC) VISIBLE,
  INDEX `fk_role_has_user_role1_idx` (`role_id` ASC) VISIBLE,
  CONSTRAINT `fk_role_has_user_role1`
    FOREIGN KEY (`role_id`)
    REFERENCES `fest`.`role` (`id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_role_has_user_user1`
    FOREIGN KEY (`user_ID`)
    REFERENCES `fest`.`user` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`paymentMethod`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`paymentMethod` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  `commission` INT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`payment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`payment` (
  `id` INT NOT NULL,
  `date` DATETIME NOT NULL,
  `result` VARCHAR(145) NOT NULL,
  `amount` INT NOT NULL,
  `paymentMethod_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_payment_paymentMethod1_idx` (`paymentMethod_id` ASC) VISIBLE,
  INDEX `paymentInd` (`date` ASC, `result` ASC, `amount` ASC) VISIBLE,
  CONSTRAINT `fk_payment_paymentMethod1`
    FOREIGN KEY (`paymentMethod_id`)
    REFERENCES `fest`.`paymentMethod` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `fest`.`request`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`request` (
  `user_ID` INT NOT NULL,
  `Event_ID` INT NOT NULL,
  `payment_id` INT NOT NULL,
  INDEX `fk_user_has_Event_Event1_idx` (`Event_ID` ASC) VISIBLE,
  INDEX `fk_user_has_Event_user1_idx` (`user_ID` ASC) VISIBLE,
  INDEX `fk_request_payment1_idx` (`payment_id` ASC) VISIBLE,
  CONSTRAINT `fk_user_has_Event_user1`
    FOREIGN KEY (`user_ID`)
    REFERENCES `fest`.`user` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_user_has_Event_Event1`
    FOREIGN KEY (`Event_ID`)
    REFERENCES `fest`.`Event` (`ID`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_request_payment1`
    FOREIGN KEY (`payment_id`)
    REFERENCES `fest`.`payment` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`performerSkills`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`performerSkills` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(45) NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`Performer_has_performerSkills`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`Performer_has_performerSkills` (
  `Performer_ID` INT NOT NULL,
  `performerSkills_id` INT NOT NULL,
  INDEX `fk_Performer_has_performerSkills_performerSkills1_idx` (`performerSkills_id`
ASC) VISIBLE,
  INDEX `fk_Performer_has_performerSkills_Performer1_idx` (`Performer_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Performer_has_performerSkills_Performer1`
```

```
    FOREIGN KEY (`Performer_ID`)
    REFERENCES `fest`.`Performer` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Performer_has_performerSkills_performerSkills1`
    FOREIGN KEY (`performerSkills_id`)
    REFERENCES `fest`.`performerSkills` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`transportType`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`transportType` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`transport`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`transport` (
  `id` INT NOT NULL,
  `count` VARCHAR(45) NULL,
  `transportType_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_transport_transportType1_idx` (`transportType_id` ASC) VISIBLE,
  CONSTRAINT `fk_transport_transportType1`
    FOREIGN KEY (`transportType_id`)
    REFERENCES `fest`.`transportType` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`Event_has_transport`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`Event_has_transport` (
  `Event_ID` INT NULL,
  `transport_id` INT NULL,
  INDEX `fk_Event_has_transport_transport1_idx` (`transport_id` ASC) VISIBLE,
  INDEX `fk_Event_has_transport_Event1_idx` (`Event_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Event_has_transport_Event1`
    FOREIGN KEY (`Event_ID`)
```

```
    REFERENCES `fest`.`Event` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_has_transport_transport1`
    FOREIGN KEY (`transport_id`)
    REFERENCES `fest`.`transport` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`company`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`company` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `nameCompany` (`name` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`organizer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`organizer` (
  `id` INT NOT NULL,
  `profession` VARCHAR(145) NULL,
  `Contact_info_ID` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_organizer_Contact_info1_idx` (`Contact_info_ID` ASC) VISIBLE,
  CONSTRAINT `fk_organizer_Contact_info1`
    FOREIGN KEY (`Contact_info_ID`)
    REFERENCES `fest`.`Contact_info` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`Event_has_organization`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`Event_has_organization` (
  `Event_ID` INT NULL,
  `company_id` INT NULL,
  `organizer_id` INT NULL,
  INDEX `fk_Event_has_company_company1_idx` (`company_id` ASC) VISIBLE,
  INDEX `fk_Event_has_company_Event1_idx` (`Event_ID` ASC) VISIBLE,
  INDEX `fk_Event_has_organization_organizer1_idx` (`organizer_id` ASC) VISIBLE,
```

```
  CONSTRAINT `fk_Event_has_company_Event1`
    FOREIGN KEY (`Event_ID`)
    REFERENCES `fest`.`Event` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_has_company_company1`
    FOREIGN KEY (`company_id`)
    REFERENCES `fest`.`company` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_has_organization_organizer1`
    FOREIGN KEY (`organizer_id`)
    REFERENCES `fest`.`organizer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`sponsorDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`sponsorDetails` (
  `id` INT NOT NULL,
  `moneyAmount` INT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`sponsor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`sponsor` (
  `id` INT NOT NULL,
  `profession` VARCHAR(45) NULL,
  `Contact_info_ID` INT NOT NULL,
  `sponsorDetails_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_sponsor_Contact_info1_idx` (`Contact_info_ID` ASC) VISIBLE,
  INDEX `fk_sponsor_sponsorDetails1_idx` (`sponsorDetails_id` ASC) VISIBLE,
  CONSTRAINT `fk_sponsor_Contact_info1`
    FOREIGN KEY (`Contact_info_ID`)
    REFERENCES `fest`.`Contact_info` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_sponsor_sponsorDetails1`
    FOREIGN KEY (`sponsorDetails_id`)
    REFERENCES `fest`.`sponsorDetails` (`id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`Event_has_sponsor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`Event_has_sponsor` (
  `Event_ID` INT NOT NULL,
  `company_id` INT NULL,
  `sponsor_id` INT NULL,
  `sponsorDetails_id` INT NULL,
  INDEX `fk_Event_has_company_company2_idx` (`company_id` ASC) VISIBLE,
  INDEX `fk_Event_has_company_Event2_idx` (`Event_ID` ASC) VISIBLE,
  INDEX `fk_Event_has_sponsor_sponsor1_idx` (`sponsor_id` ASC) VISIBLE,
  INDEX `fk_Event_has_sponsor_sponsorDetails1_idx` (`sponsorDetails_id` ASC) VISIBLE,
  CONSTRAINT `fk_Event_has_company_Event2`
    FOREIGN KEY (`Event_ID`)
    REFERENCES `fest`.`Event` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_has_company_company2`
    FOREIGN KEY (`company_id`)
    REFERENCES `fest`.`company` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_has_sponsor_sponsor1`
    FOREIGN KEY (`sponsor_id`)
    REFERENCES `fest`.`sponsor` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Event_has_sponsor_sponsorDetails1`
    FOREIGN KEY (`sponsorDetails_id`)
    REFERENCES `fest`.`sponsorDetails` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

USE `fest` ;

-- -----------------------------------------------------
-- Table `fest`.`name`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`name` (
  `idname` INT(11) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idname`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- -----------------------------------------------------
-- Table `fest`.`ErrorLog`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`ErrorLog` (
  `LogId` INT NOT NULL,
  `ErrorState` VARCHAR(45) NOT NULL,
  `ErrorMessage` VARCHAR(45) NULL,
  `UserId` INT NOT NULL,
  `CreatedOn` DATE NULL,
  PRIMARY KEY (`LogId`),
  UNIQUE INDEX `LogId_UNIQUE` (`LogId` ASC) VISIBLE,
  INDEX `log` (`ErrorState` ASC, `ErrorMessage` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`requirement`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`requirement` (
  `id` INT NOT NULL,
  `quantity` INT NULL,
  `cost_planned` DOUBLE NULL,
  `cost_actual` DOUBLE NULL,
  `Event_ID` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_requirement_Event_idx` (`Event_ID` ASC) VISIBLE,
  CONSTRAINT `fk_requirement_Event`
    FOREIGN KEY (`Event_ID`)
    REFERENCES `fest`.`Event` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`equipment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`equipment` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `available` VARCHAR(45) NOT NULL,
  `requirement_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_equipment_requirement1_idx` (`requirement_id` ASC) VISIBLE,
  INDEX `equipment` (`name` ASC) VISIBLE,
  CONSTRAINT `fk_equipment_requirement1`
    FOREIGN KEY (`requirement_id`)
```

```
    REFERENCES `fest`.`requirement` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`equipmentType`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`equipmentType` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `equipment_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_equipmentType_equipment1_idx` (`equipment_id` ASC) VISIBLE,
  CONSTRAINT `fk_equipmentType_equipment1`
    FOREIGN KEY (`equipment_id`)
    REFERENCES `fest`.`equipment` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`employeeDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`employeeDetails` (
  `id` INT NOT NULL,
  `nationality` VARCHAR(45) NOT NULL,
  `visa` VARCHAR(45) NOT NULL,
  `drivingLicense` VARCHAR(45) NULL,
  `resume` VARCHAR(45) NOT NULL,
  `Contact_info_ID` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_employeeDetails_Contact_info1_idx` (`Contact_info_ID` ASC) VISIBLE,
  CONSTRAINT `fk_employeeDetails_Contact_info1`
    FOREIGN KEY (`Contact_info_ID`)
    REFERENCES `fest`.`Contact_info` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`employee` (
  `ID` INT NOT NULL,
```

```
  `profession` VARCHAR(45) NOT NULL,
  `employeeDetails_id` INT NOT NULL,
  `Contact_info_ID` INT NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,
  INDEX `fk_employee_employeeDetails1_idx` (`employeeDetails_id` ASC) VISIBLE,
  INDEX `fk_employee_Contact_info1_idx` (`Contact_info_ID` ASC) VISIBLE,
  CONSTRAINT `fk_employee_employeeDetails1`
    FOREIGN KEY (`employeeDetails_id`)
    REFERENCES `fest`.`employeeDetails` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_employee_Contact_info1`
    FOREIGN KEY (`Contact_info_ID`)
    REFERENCES `fest`.`Contact_info` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`typeOfWork`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`typeOfWork` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`employeesWork`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`employeesWork` (
  `id` INT NOT NULL,
  `spendHours` INT NULL,
  `salary` INT NOT NULL,
  `workingHours` INT NOT NULL,
  `typeOfWork_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_employeesWork_typeOfWork1_idx` (`typeOfWork_id` ASC) VISIBLE,
  INDEX `work` (`salary` ASC, `typeOfWork_id` ASC) VISIBLE,
  CONSTRAINT `fk_employeesWork_typeOfWork1`
    FOREIGN KEY (`typeOfWork_id`)
    REFERENCES `fest`.`typeOfWork` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `fest`.`employee_has_employeesWork`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`employee_has_employeesWork` (
  `employee_ID` INT NOT NULL,
  `employeesWork_id` INT NOT NULL,
  INDEX `fk_employee_has_employeesWork_employeesWork1_idx` (`employeesWork_id` ASC)
VISIBLE,
  INDEX `fk_employee_has_employeesWork_employee1_idx` (`employee_ID` ASC) VISIBLE,
  CONSTRAINT `fk_employee_has_employeesWork_employee1`
    FOREIGN KEY (`employee_ID`)
    REFERENCES `fest`.`employee` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_employee_has_employeesWork_employeesWork1`
    FOREIGN KEY (`employeesWork_id`)
    REFERENCES `fest`.`employeesWork` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`skills`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`skills` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(255) NULL,
  `level` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `skillNames` (`name` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`skills_has_employeeDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`skills_has_employeeDetails` (
  `skills_id` INT NOT NULL,
  `employeeDetails_id` INT NOT NULL,
  INDEX `fk_skills_has_employeeDetails_employeeDetails1_idx` (`employeeDetails_id` ASC)
VISIBLE,
  INDEX `fk_skills_has_employeeDetails_skills1_idx` (`skills_id` ASC) VISIBLE,
  CONSTRAINT `fk_skills_has_employeeDetails_skills1`
    FOREIGN KEY (`skills_id`)
    REFERENCES `fest`.`skills` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
```

```
  CONSTRAINT `fk_skills_has_employeeDetails_employeeDetails1`
    FOREIGN KEY (`employeeDetails_id`)
    REFERENCES `fest`.`employeeDetails` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`room`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`room` (
  `id` INT NOT NULL,
  `number` INT NOT NULL,
  `capacity` INT NULL,
  `building_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_room_building1_idx` (`building_id` ASC) VISIBLE,
  CONSTRAINT `fk_room_building1`
    FOREIGN KEY (`building_id`)
    REFERENCES `fest`.`building` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`company_has_employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`company_has_employee` (
  `companyOrganizer_id` INT NULL,
  `employee_ID` INT NULL,
  INDEX `fk_companyOrganizer_has_employee_employee1_idx` (`employee_ID` ASC) VISIBLE,
  INDEX `fk_companyOrganizer_has_employee_companyOrganizer1_idx` (`companyOrganizer_id`
ASC) VISIBLE,
  CONSTRAINT `fk_companyOrganizer_has_employee_companyOrganizer1`
    FOREIGN KEY (`companyOrganizer_id`)
    REFERENCES `fest`.`company` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_companyOrganizer_has_employee_employee1`
    FOREIGN KEY (`employee_ID`)
    REFERENCES `fest`.`employee` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
```

```
-- Table `fest`.`companyType`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`companyType` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  PRIMARY KEY (`id`),
  INDEX `companyType` (`name` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `fest`.`companyDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `fest`.`companyDetails` (
  `id` INT NOT NULL,
  `description` VARCHAR(45) NULL,
  `companyType_id` INT NOT NULL,
  `company_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `fk_companyDetails_companyType1_idx` (`companyType_id` ASC) VISIBLE,
  INDEX `fk_companyDetails_company1_idx` (`company_id` ASC) VISIBLE,
  CONSTRAINT `fk_companyDetails_companyType1`
    FOREIGN KEY (`companyType_id`)
    REFERENCES `fest`.`companyType` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_companyDetails_company1`
    FOREIGN KEY (`company_id`)
    REFERENCES `fest`.`company` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;


INSERT INTO `fest`.`role` (`id`, `name`, `description`) VALUES ('1', 'admin',
'Администратор');
INSERT INTO `fest`.`role` (`id`, `name`, `description`) VALUES ('2', 'user',
'Зарегистрированный');
INSERT INTO `fest`.`role` (`id`, `name`, `description`) VALUES ('3', 'nonReg',
'Незарегистрированный');
INSERT INTO `fest`.`user` (`ID`, `RegInfo_ID`, `Contact_info_ID`) VALUES ('1', '1',
'1');
```

```
INSERT INTO `fest`.`user` (`ID`, `RegInfo_ID`, `Contact_info_ID`) VALUES ('2', '2',
'2');
INSERT INTO `fest`.`user` (`ID`, `RegInfo_ID`, `Contact_info_ID`) VALUES ('3', '3',
'3');

INSERT INTO `fest`.`reginfo` (`ID`, `login`, `password`) VALUES ('1', 'petr45', 'gf45');
INSERT INTO `fest`.`reginfo` (`ID`, `login`, `password`) VALUES ('2', 'kol78',
'fvfgd45');
INSERT INTO `fest`.`reginfo` (`ID`, `login`, `password`) VALUES ('3', 'r78', 'dx98');

ALTER TABLE role_has_user
MODIFY user_ID int NOT NULL;

alter table role_has_user
  add unique index roleUser (user_id );

INSERT INTO `fest`.`country` (`id`, `name`) VALUES ('1', 'France');
INSERT INTO `fest`.`country` (`id`, `name`) VALUES ('2', 'Italy');

INSERT INTO `fest`.`city` (`id`, `name`, `country_id`) VALUES ('1', 'Milan', '2');
INSERT INTO `fest`.`city` (`id`, `name`, `country_id`) VALUES ('2', 'Paris', '1');

INSERT INTO `fest`.`building` (`id`, `street`, `name`, `number`) VALUES ('1',
'Sovtskay', 'ТЦ МОЛ', '98');
INSERT INTO `fest`.`building` (`id`, `street`, `name`, `number`) VALUES ('2', 'Петруся
Бровки', 'ТЦ GREEN', '1');
INSERT INTO `fest`.`room` (`id`, `number`, `capacity`, `building_id`) VALUES ('1', '12',
'200', '2');
INSERT INTO `fest`.`room` (`id`, `number`, `capacity`, `building_id`) VALUES ('2', '25',
'1000', '1');


DELIMITER //

CREATE PROCEDURE ShowPeople(rowsCount SMALLINT)
BEGIN
  select r.login, r.password, c.age, c.firstname, c.mail, c.phone, c.surname from user
AS u
    inner join reginfo AS r On  r.id = u.RegInfo_ID
  inner join contact_info AS c On  c.id = u.Contact_Info_ID
    limit rowsCount;
END //

DELIMITER ;

INSERT INTO `fest`.`eventtype` (`id`, `name`) VALUES ('1', 'Музыкальный');
INSERT INTO `fest`.`eventtype` (`id`, `name`) VALUES ('2', 'Исторический');

INSERT INTO `fest`.`outlaytype` (`id`, `name`) VALUES ('1', 'оборудование');
INSERT INTO `fest`.`outlaytype` (`id`, `name`) VALUES ('2', 'зарплата работников');
```

```
INSERT INTO `fest`.`outlay` (`id`, `amount`, `outlayType_id`, `name`) VALUES ('1',
'200', '1', 'оборудование');
INSERT INTO `fest`.`outlay` (`id`, `amount`, `outlayType_id`, `name`) VALUES ('2',
'1000', '2', 'зарплата');
INSERT INTO `fest`.`profittype` (`id`, `name`) VALUES ('1', 'билеты');
INSERT INTO `fest`.`profittype` (`id`, `name`) VALUES ('2', 'сувениры');
INSERT INTO `fest`.`profit` (`id`, `amountMoney`, `profitType_id`, `name`) VALUES ('1',
'2000', '1', 'билеты');
INSERT INTO `fest`.`profit` (`id`, `amountMoney`, `profitType_id`, `name`) VALUES ('2',
'100', '2', 'сувениры');
INSERT INTO `fest`.`transporttype` (`id`, `name`) VALUES ('1', 'машина');
INSERT INTO `fest`.`transporttype` (`id`, `name`) VALUES ('2', 'автобус');
INSERT INTO `fest`.`transport` (`id`, `count`, `transportType_id`) VALUES ('1', '20',
'1');
INSERT INTO `fest`.`transport` (`id`, `count`, `transportType_id`) VALUES ('2', '3',
'2');
INSERT INTO `fest`.`adress` (`id`, `city_id`, `building_id`) VALUES ('1', '1', '1');
INSERT INTO `fest`.`adress` (`id`, `city_id`, `building_id`) VALUES ('2', '2', '2');
INSERT INTO `fest`.`place` (`ID`, `maxAmountOfParticipants`, `adress_id`) VALUES ('1',
'20000', '1');
INSERT INTO `fest`.`place` (`ID`, `maxAmountOfParticipants`, `adress_id`) VALUES ('2',
'500', '2');
INSERT INTO `fest`.`event` (`ID`, `type_id`, `place_ID`, `eventDetails_id`, `name`,
`outlay_id`, `profit_id`) VALUES ('1', '1', '1', '1', 'музыкал)', '1', '1');
INSERT INTO `fest`.`event` (`ID`, `type_id`, `place_ID`, `eventDetails_id`, `name`,
`outlay_id`, `profit_id`) VALUES ('2', '2', '2', '2', 'Калиновский', '2', '2');

DELIMITER //

CREATE PROCEDURE ShowEvent(rowsCount SMALLINT)
BEGIN
  select u.* from event AS u
    inner join eventtype AS t On  t.id = u.type_id
  inner join place AS p On  p.id = u.place_ID
    inner join eventDetails AS d ON d.ID = u.eventDetails_id
    inner join outlay AS o ON o.ID = u.outlay_id
    inner join profit AS pr ON pr.ID = u.profit_id
    limit rowsCount;
END //

DELIMITER ;


DELIMITER //

CREATE PROCEDURE ShowPlaces(rowsCount SMALLINT)
BEGIN
  select u.* from place AS u
    inner join adress AS t On  t.id = u.adress_id
  inner join city AS p On  p.id = t.city_id
```

```
    inner join building AS d ON d.ID = t.building_id
    inner join country AS o ON o.ID = p.country_id

    limit rowsCount;
END //

DELIMITER ;
```

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1–40 01 01 003 ПЗ	Пояснительная записка	41 с.
	Графические документы	
БГУИР 651001 003 ПД	Модель БД	Формат А1
БГУИР 651001 003 ПД	Схема данных	Формат А1

					БГУИР КП 1-40 01 01 003 Д1		
Изм.	Л.	№ докум.	Подп.	Дата	База данных системы для организации фестивалей	Лист	Листов
Разраб.		Арабей Д.И.				Т 41	41
Пров.		Фадеева Е.Е.			Ведомость курсового проекта	Кафедра ПОИТ	
						гр. 651001	