**Федеральное агентство связи**
Федеральное государственное бюджетное образовательное учреждение высшего образования
**«Поволжский государственный университет телекоммуникаций и информатики»**

Кафедра «Прикладная информатика»

**«УТВЕРЖДАЮ»**

Зав. кафедрой ПИ
профессор Маслов О.Н.

« 31 » августа 2019 г.

# «ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ»

## Методические указания для выполнения курсовой работы

для специальности 090301

**Составитель: Горожанина Е.И.**

Самара

2019

# Введение

Большой популярностью среди программистов пользуются базы данных MySQL, а для управления ими – PHPMyAdmin. Переходя от маленьких проектов к большим, от cms к фреймворкам, многие используют MySQL.

Однако для проектирования сложной базы данных с большим количеством таблиц и связей, возможностей PHPMyAdmin катастрофически не хватает. Тогда возможно использование MySQL Workbench.

MySQL Workbench — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL.

Программа позволяет быстро создавать схемы данных проекта, проектировать сущности и связи между ними, безболезненно внедрять изменения в схему и так же быстро и безболезненно синхронизировать её с удалённым сервером. А графический редактор ER-диаграмм позволяет увидеть общую картину модели данных. После первой же пробы этот инструмент становится незаменимым помощником в боевом арсенале веб-программиста.

### Цели и задачи работы

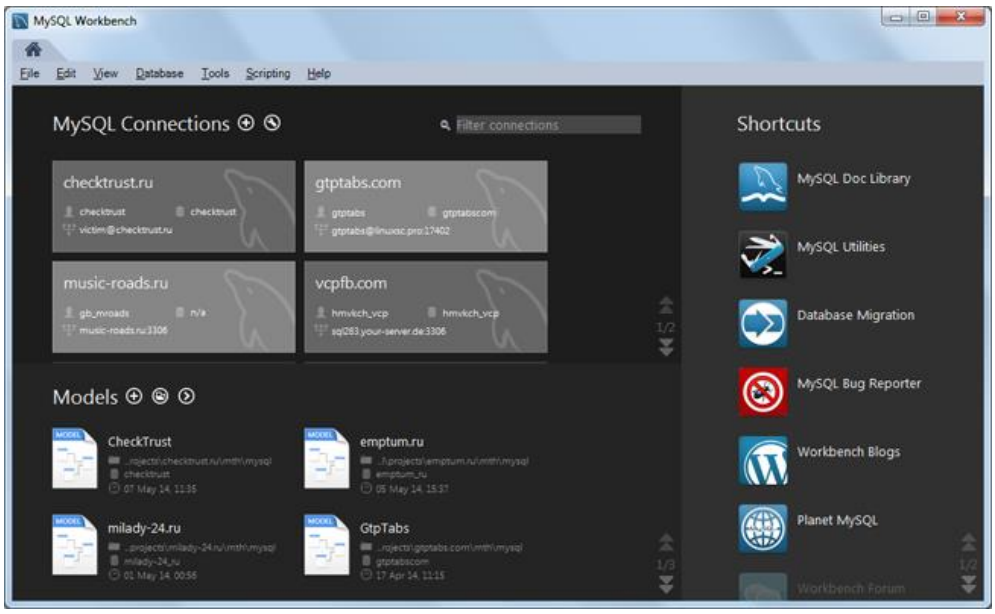
Цель курсовой работы – применение на практике знаний, полученных в процессе изучения курса "Проектирование баз данных", и приобретение практических навыков при проектировании баз данных в MySQL.

### Задание

Реализовать базу данных проекта в СУБД MySQL согласно варианту. Оформить согласно РД ПГУТИ 2.18.7 – 2017.

### Начало работы

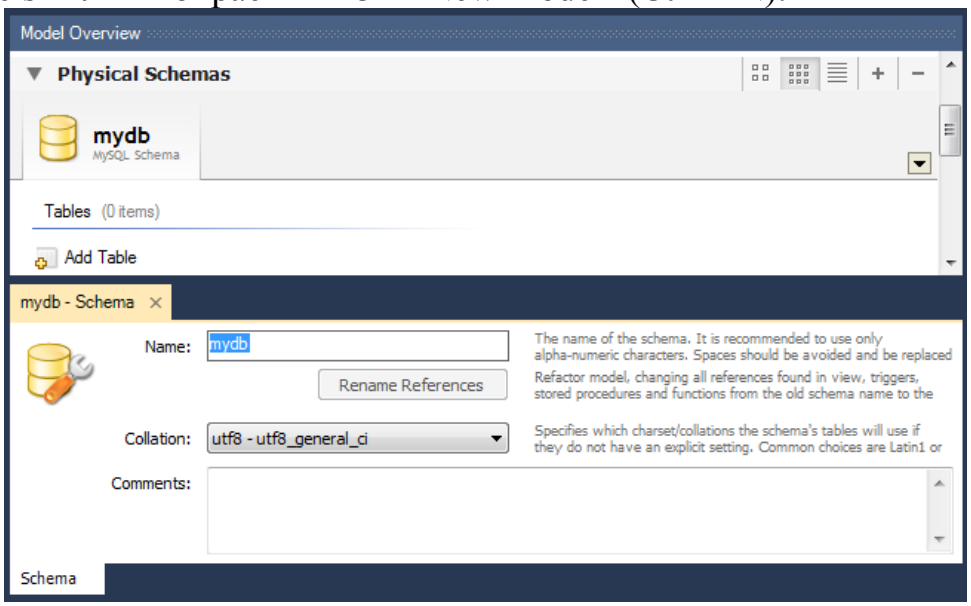
Стартовый экран программы отражает основные направления её функциональности - проектирование моделей баз данных и их администрирование:




В верхней части экрана находится список подключений к MySQL серверам ваших проектов, а список последних открытых моделей данных - в нижней части экрана. Работа обычно начинается с создания схемы данных или загрузки существующей структуры в MySQL Workbench.

**Создание и редактирование модели данных**

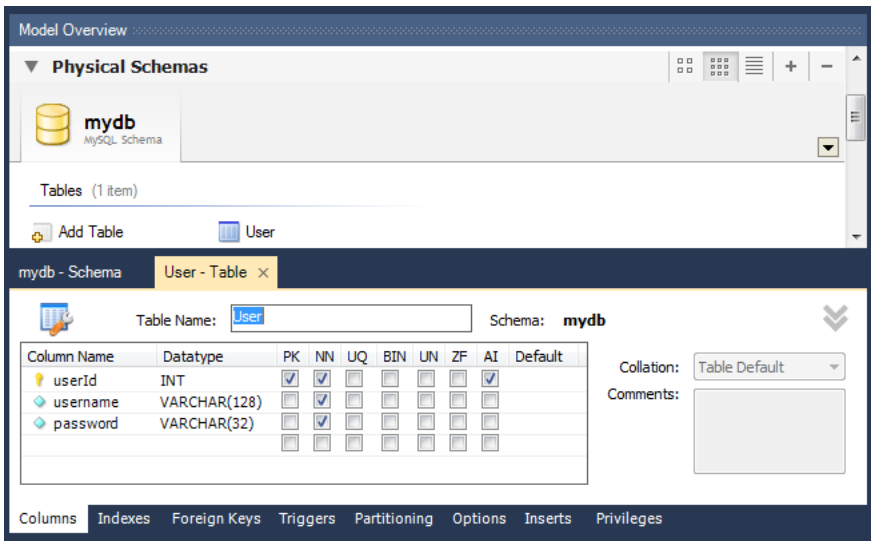
Для добавления модели нажимаем плюсик рядом с заголовком "Models" или выбираем "File → New Model" (Ctrl + N):




На этом экране вводим имя базы данных, выбираем кодировку по умолчанию и, если нужно, заполняем поле комментария. Можно приступать к созданию таблиц.

**Добавление и редактирование таблицы**

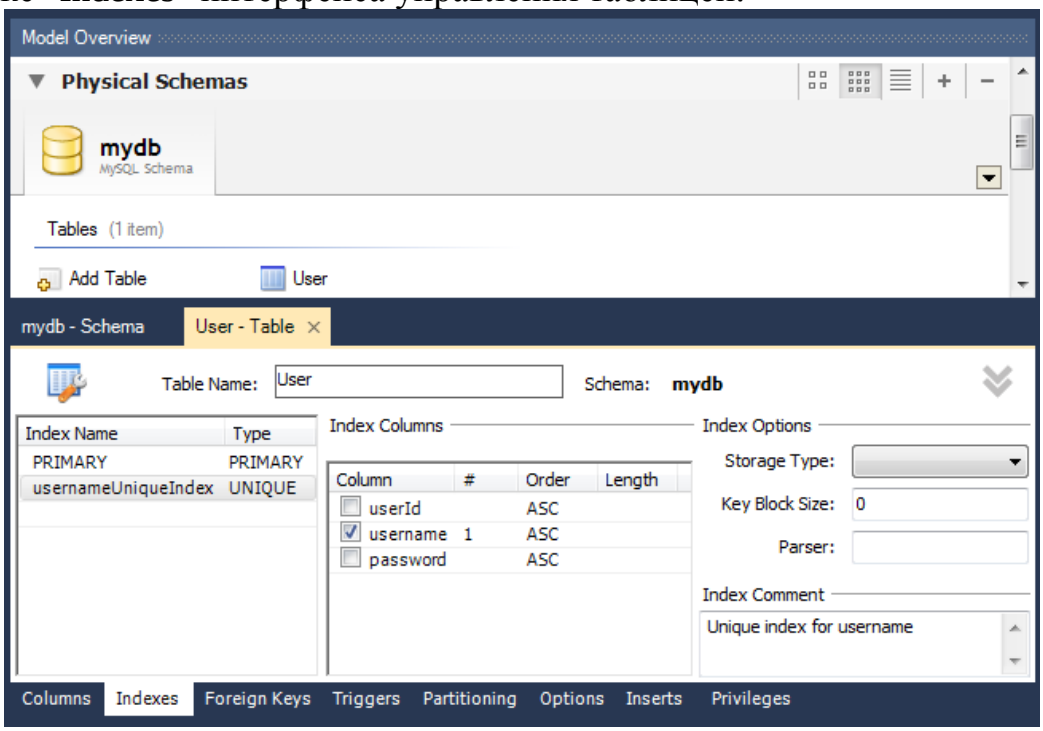
Список баз данных проекта и список таблиц в пределах базы данных будет располагаться во вкладке "Physical Schemas". Чтобы создать таблицу, дважды кликаем на "+Add Table":




Откроется удобный интерфейс для редактирования списка полей и их свойств. Здесь мы можем задать название поля, тип данных, а так же установить для полей различные атрибуты: назначить поле первичным ключом (PK), пометить его Not Null (NN), бинарным (BIN), уникальным (UQ) и другие, установить для поля авто-инкремирование (AI) и значение по умолчанию (Default).

**Управление индексами**

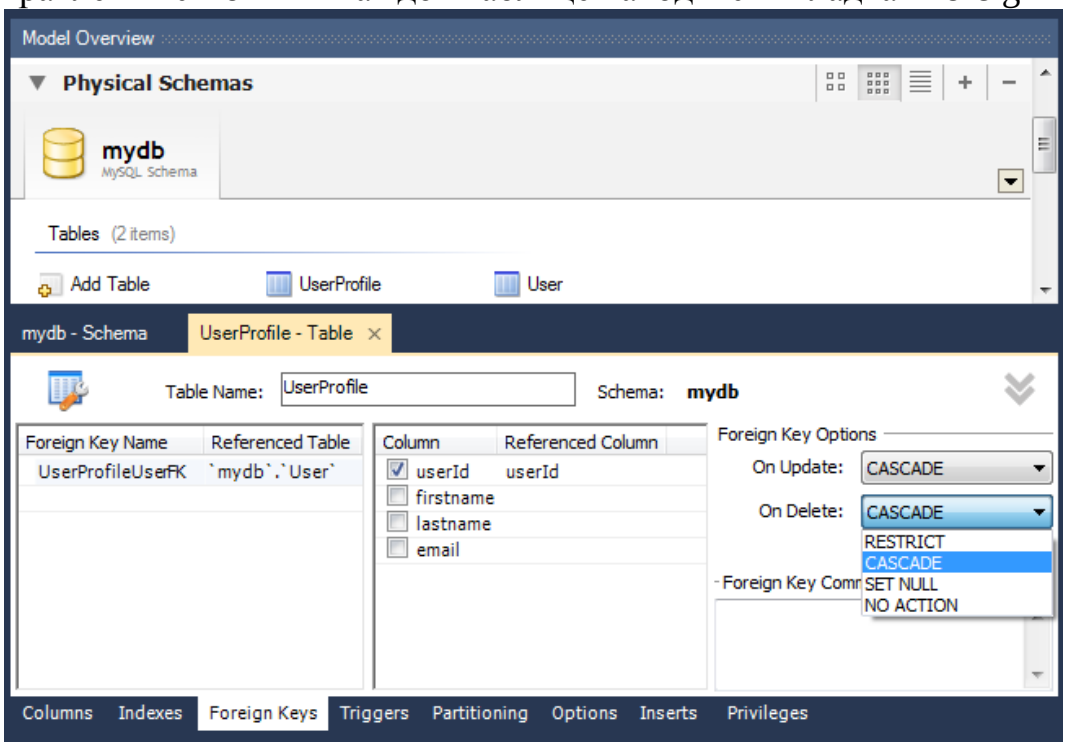
Добавлять, удалять и редактировать индексы таблиц можно во вкладке "Indexes" интерфейса управления таблицей:




Вводим название индекса, выбираем его тип, затем галочками помечаем в нужном порядке список полей, участвующих в данном индексе. Порядок полей будет соответствовать порядку, в котором были проставлены галочки. В данном примере я добавил уникальный индекс к полю username.

**Связи между таблицами**

Установка внешних ключей и связывание таблиц возможно только для таблиц InnoDB (эта система хранения данных выбирается по умолчанию). Для управления связями в каждой таблице находится вкладка "Foreign Keys":




Для добавления связи открываем вкладку "Foreign Keys" дочерней таблицы, вводим имя внешнего ключа и выбираем таблицу-родителя. Далее в средней части вкладки в графе Column выбираем поле-ключ из дочерней таблицы, а в графе Referenced Column - соответствующее поле из родительской таблицы (тип полей должен совпадать). При создании внешних ключей в дочерней таблице автоматически создаются соответствующие индексы.

В разделе "Foreign Key Options" настраиваем поведение внешнего ключа при изменении соответствующего поля (ON UPDATE) и удалении (ON DELETE) родительской записи:

RESTRICT - выдавать ошибку при изменении / удалении родительской записи

CASCADE - обновлять внешний ключ при изменении родительской записи, удалять дочернюю запись при удалении родителя

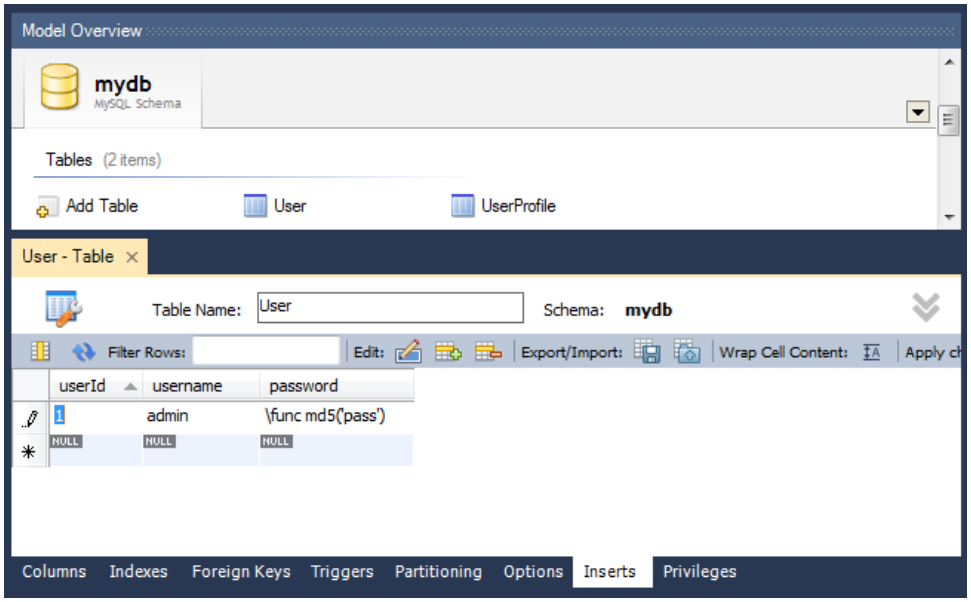
SET NULL - устанавливать значение внешнего ключа NULL при изменении / удалении родителя (неприемлемо для полей, у которых установлен флаг NOT NULL!)

NO ACTION - не делать ничего, однако по факту эффект аналогичен RESTRICT

В приведённом примере я добавил к дочерней таблице UserProfile внешний ключ для связи с родительской таблицей User. При редактировании поля userId и удалении позиций из таблицы User аналогичные изменения будут автоматически происходить и со связанными записями из таблицы UserProfile.

**Наполнение таблицы базовыми данными**

При создании проекта в базу данных часто нужно добавлять стартовые данные. Это могут быть корневые категории, пользователи-администраторы и т.д. В управлении таблицами MySQL Workbench для этого существует вкладка "Inserts":

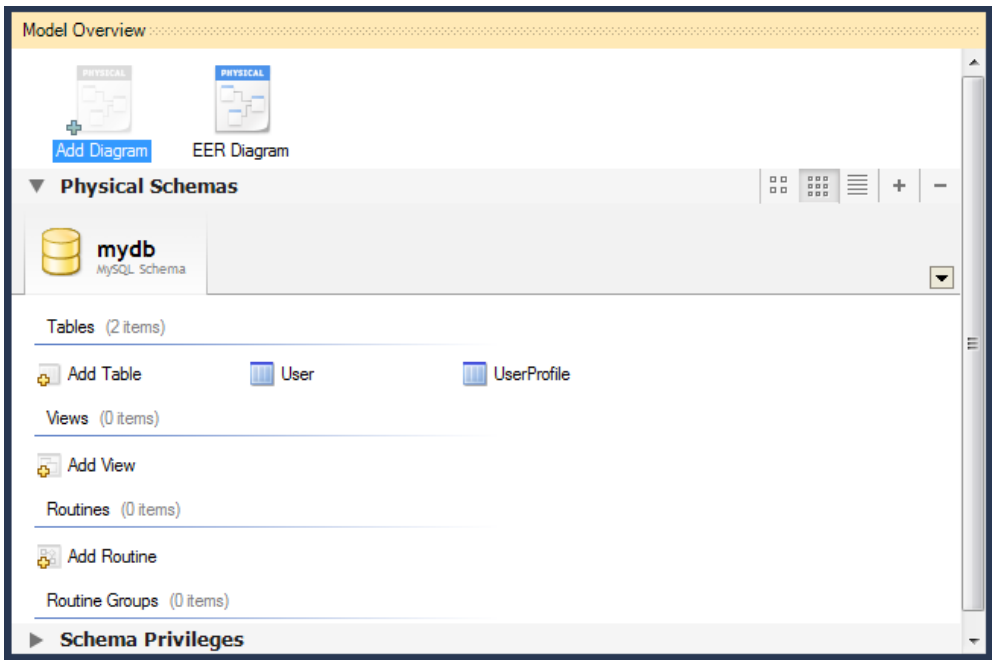



Как видно из примера, в случае, если перед записью в базу данных к данным нужно применить какую-то функцию MySQL, это делается с помощью синтаксиса \func functionName('data'), например, \func md5('password').

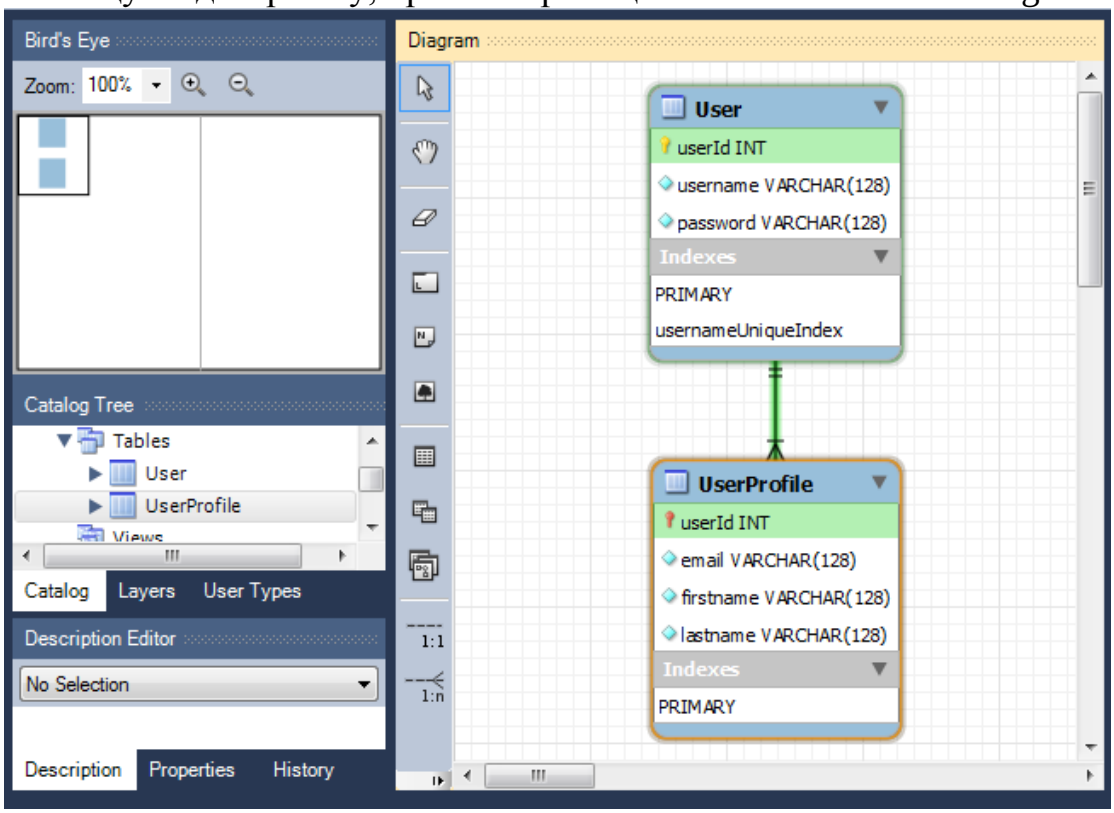
После ввода данных необходимо сохранить их в локальную базу данных нажатием на кнопку "Apply Changes".

**Создание EER диаграммы (диаграммы "сущность-связь")**

Для представления схемы данных, сущностей и их связей в графическом виде в MySQL Workbench существует редактор EER-диаграмм. Для создания диаграммы в верхней части экрана управления базой данных дважды кликаем на иконку "+Add Diagram":



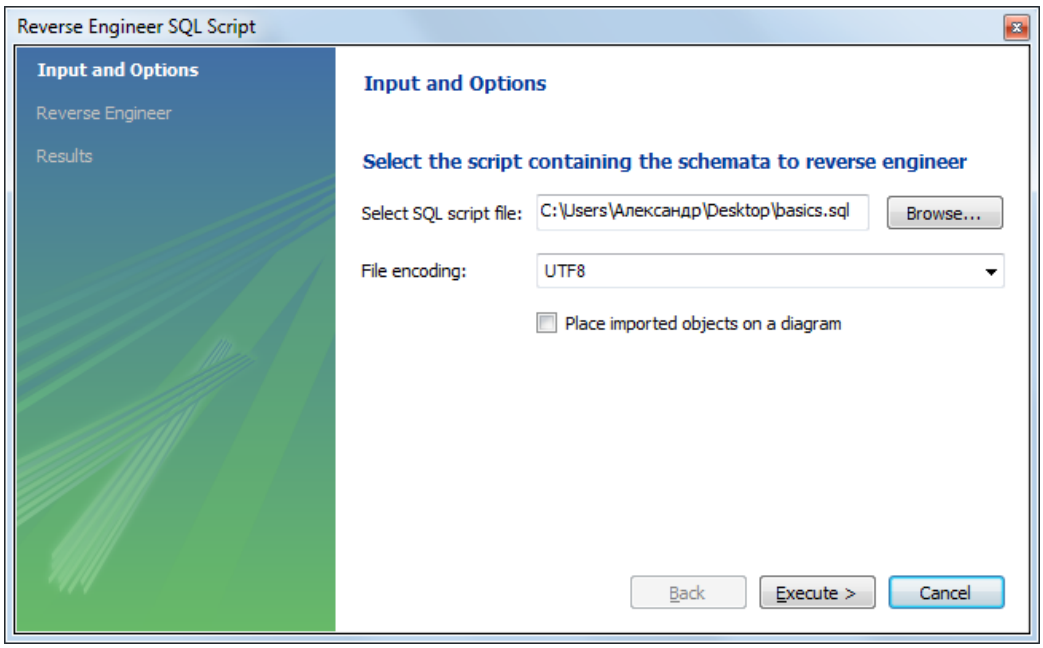

В его интерфейсе можно создавать и редактировать таблицы, добавлять между ними связи различных типов. Чтобы добавить уже существующую в схеме таблицу на диаграмму, просто перетащите её из панели "Catalog Tree".




Для экспорта схемы данных в графический файл выберите "File → Export", а затем один из вариантов (PNG, SVG, PDF, PostScript File).

**Импорт существующей схемы данных (из SQL дампа)**

Если у нас уже есть схема данных, её можно без труда импортировать в MySQL Workbench для дальнейшей работы. Для импорта модели из SQL файла выбираем "File → Import → Reverse Engineer MySQL Create Script...", после чего выбираем нужный SQL файл и жмём "Execute >"




В MySQL Workbench так же предусмотрен импорт и синхронизация модели данных нарямую с удалённым сервером. Для этого потребуется создать подключение удалённого доступа к MySQL.

**Пример программной реализации базы данных проекта из ER-модели**

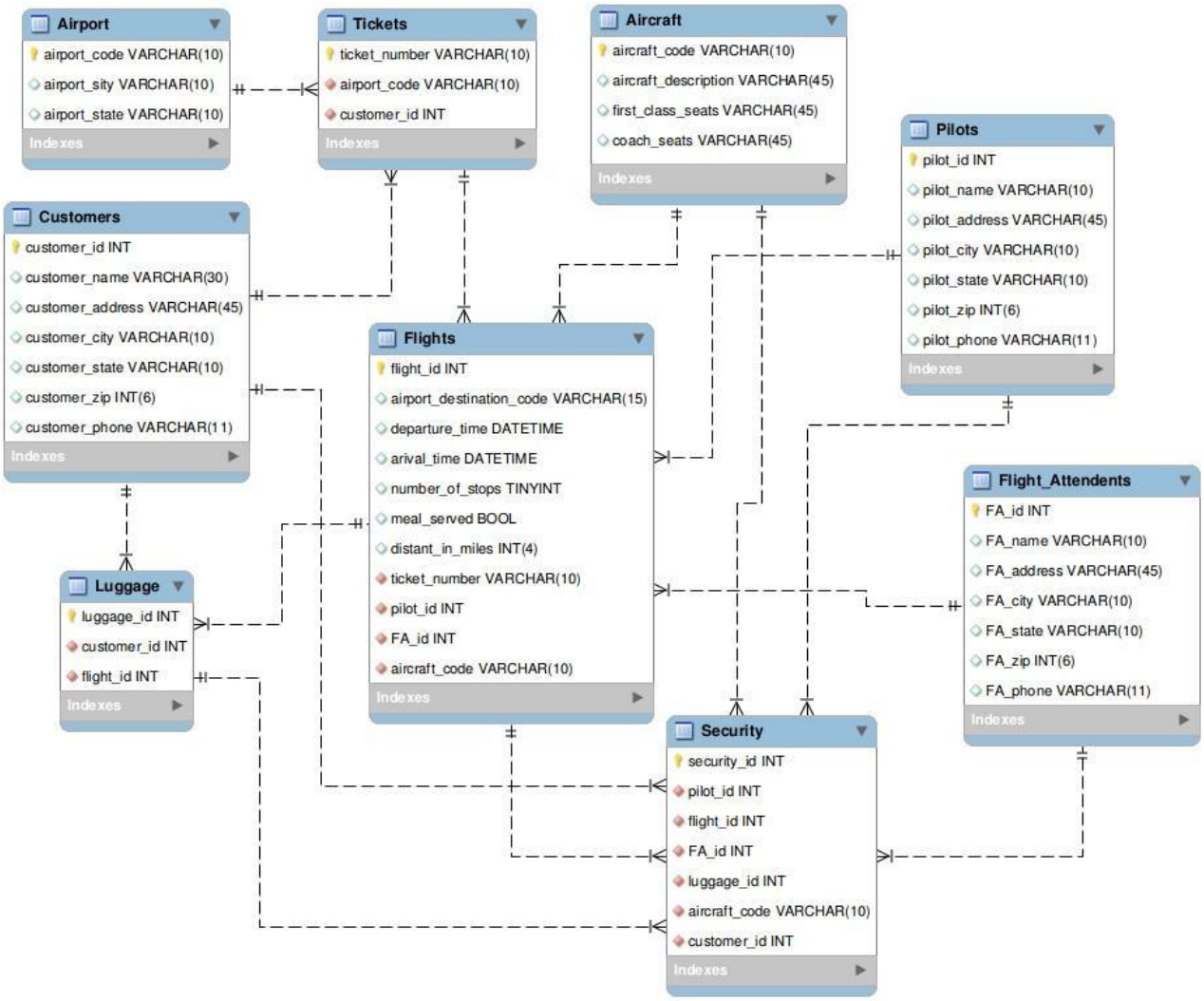
В качестве предметной области проекта выбран аэропорт.

ER-модель базы данных аэропорта выглядит следующим образом:

Данная модель включает следующие сущности (таблицы базы данных), отражающие объекты реального мира и их свойства:

- таблица рейсов (Flights), содержащая данные о всех рейсах аэропорта;
- таблица воздушных судов (Aircraft) содержащая данные об имеющихся в данном аэропорте самолетах;
- таблица пилотов (Pilots), содержащая сведения о пилотах аэропорта;
- таблица членов экипажа (Flight_Attendents), содержащая сведения о бортпроводниках рейсов;
- таблица билетов (Tickets), содержащая информацию о купленных пассажирами билетах;

- таблица аэропортов (Airport), содержащая информацию об аэропортах, с которыми осуществляется воздушное сообщение;
- таблица клиентов (Customers), хранящая данные о клиентах аэропорта;
- таблицы сведений о багаже (Laggage) и данных, необходимых для обеспечения безопасности в аэропорте, являющиеся связующими таблицами в связях многие-ко-многим между другими таблицами базы данных.




Таблицы рассматриваемой базы данных связаны между собой связями один-ко-многим и многие-ко-многим (данный тип связи также реализуется через две связи один-ко-многим).

Для программной реализации данной ER-модели базы данных необходимо написать SQL-запросы, позволяющие создать указанные в модели сущности (таблицы базы данных), их атрибуты (поля или столбцы таблиц базы данных) и связи между сущностями (таблицами базы данных). В качестве языка реализации выберем диалект SQL – язык СУБД MySQL.

Весь программный код (набор SQL-запросов), необходимый для программной реализации рассматриваемой базы данных, выглядит следующим образом:

```
1| CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT
CHARACTER SET utf8 2| COLLATE utf8_general_ci ;

3|              SHOW
WARNINGS;     4|
USE `mydb` ;
5| 6| -- ----------------------------------------
----------
7|  -- Table `mydb`.`Aircraft`
8|  -- -----------------------------------------------
```

```
9|  DROP  TABLE  IF  EXISTS
`mydb`.`Aircraft` ; 10|
11| SHOW WARNINGS;
12|  CREATE  TABLE  IF  NOT  EXISTS
`mydb`.`Aircraft`  (  13|  `aircraft_code`
VARCHAR(10) NOT NULL,
14|  `aircraft_description`
VARCHAR(45)  NULL,  15|
`first_class_seats`  VARCHAR(45)
NULL,  16|  `coach_seats`
VARCHAR(45) NULL,
17|  PRIMARY  KEY
(`aircraft_code`))  18|  ENGINE
= InnoDB;
19|
20|  SHOW
WARNINGS; 21|
22| -- -----------------------------------------------------
23| -- Table `mydb`.`Airport`
24| -- -----------------------------------------------------
25|  DROP  TABLE  IF  EXISTS
`mydb`.`Airport` ; 26|
27| SHOW WARNINGS;
28|  CREATE  TABLE  IF  NOT  EXISTS
`mydb`.`Airport`  (  29|  `airport_code`
VARCHAR(10) NOT NULL,
30|  `airport_sity`  VARCHAR(10)
NULL,  31|  `airport_state`
VARCHAR(10)  NULL,  32|
PRIMARY  KEY
(`airport_code`))
33|  ENGINE  =
InnoDB; 34|
35|  SHOW
WARNINGS; 36|
37| -- -----------------------------------------------------
38| -- Table `mydb`.`Customers`
39| -- -----------------------------------------------------
40|  DROP  TABLE  IF  EXISTS
`mydb`.`Customers` ; 41|
42| SHOW WARNINGS;
43|  CREATE  TABLE  IF  NOT  EXISTS
`mydb`.`Customers`  (  44|  `customer_id`  INT  NOT
NULL,

45| `customer_name` VARCHAR(30) NULL, 46| `customer_address`
VARCHAR(45)  NULL,  47|  `customer_city`  VARCHAR(10)  NULL,
48|  `customer_state`  VARCHAR(10)  NULL,  49|  `customer_zip`
INT(6) NULL,
```

```
50| `customer_phone` VARCHAR(11) NULL, 51| PRIMARY KEY
(`customer_id`))
52| ENGINE = InnoDB; 53|
54| SHOW WARNINGS; 55|
56| -- -----------------------------------------------------
57| -- Table `mydb`.`Tickets`
58| -- -----------------------------------------------------
59| DROP TABLE IF EXISTS `mydb`.`Tickets` ; 60|
61| SHOW WARNINGS;
62| CREATE TABLE IF NOT EXISTS `mydb`.`Tickets` ( 63|
`ticket_number` VARCHAR(10) NOT NULL,
64| `airport_code` VARCHAR(10) NOT NULL, 65| `customer_id`
INT NOT NULL,
66| PRIMARY KEY (`ticket_number`),
67| INDEX `fk_Tickets_Airport_idx` (`airport_code` ASC),
68| INDEX `fk_Tickets_Customers1_idx` (`customer_id` ASC), 69|
CONSTRAINT `fk_Tickets_Airport`
70| FOREIGN KEY (`airport_code`)
71| REFERENCES `mydb`.`Airport` (`airport_code`) 72| ON DELETE NO
ACTION
73| ON UPDATE NO ACTION,
74| CONSTRAINT `fk_Tickets_Customers1` 75| FOREIGN KEY
(`customer_id`)
76| REFERENCES `mydb`.`Customers` (`customer_id`) 78| ON DELETE NO
ACTION
79| ON UPDATE NO ACTION) 80| ENGINE = InnoDB;
81|
82| SHOW WARNINGS; 83|
84| -- -----------------------------------------------------
85| -- Table `mydb`.`Pilots`
86| -- -----------------------------------------------------
87| DROP TABLE IF EXISTS `mydb`.`Pilots` ; 88|
89| SHOW WARNINGS;
90| CREATE TABLE IF NOT EXISTS `mydb`.`Pilots` ( 91| `pilot_id` INT
NOT NULL,
92| `pilot_name` VARCHAR(10) NULL, 93| `pilot_address`
VARCHAR(45) NULL, 94| `pilot_city` VARCHAR(10) NULL,
95| `pilot_state` VARCHAR(10) NULL, 96| `pilot_zip` INT(6)
NULL,
97| `pilot_phone` VARCHAR(11) NULL,

98| PRIMARY KEY (`pilot_id`)) 99| ENGINE =
InnoDB;
100|
```

```
101| SHOW WARNINGS; 102|
103| -- -----------------------------------------------------
104| -- Table `mydb`.`Flight_Attendents`
105| -- -----------------------------------------------------
106| DROP TABLE IF EXISTS `mydb`.`Flight_Attendents` ; 107|
108| SHOW WARNINGS;
109| CREATE TABLE IF NOT EXISTS `mydb`.`Flight_Attendents` ( 110|
`FA_id` INT NOT NULL,
111| `FA_name` VARCHAR(10) NULL,
112| `FA_address` VARCHAR(45) NULL, 113| `FA_city`
VARCHAR(10) NULL, 114| `FA_state` VARCHAR(10)
NULL, 115| `FA_zip` INT(6) NULL,
116| `FA_phone` VARCHAR(11) NULL, 117|
PRIMARY KEY (`FA_id`))
118| ENGINE = InnoDB; 119|
120| SHOW WARNINGS; 121|
122| -- -----------------------------------------------------
123| -- Table `mydb`.`Flights`
124| -- -----------------------------------------------------
125| DROP TABLE IF EXISTS `mydb`.`Flights` ; 126|
127| SHOW WARNINGS;
128| CREATE TABLE IF NOT EXISTS `mydb`.`Flights` ( 129|
`flight_id` INT NOT NULL,
130| `airport_destination_code` VARCHAR(15) NULL, 131|
`departure_time` DATETIME NULL,
132| `arival_time` DATETIME NULL, 133|
`number_of_stops` TINYINT NULL, 134| `meal_served`
TINYINT(1) NULL, 135| `distant_in_miles` INT(4)
NULL,
136| `ticket_number` VARCHAR(10) NOT NULL, 137| `pilot_id`
INT NOT NULL,
138| `FA_id` INT NOT NULL,
139| `aircraft_code` VARCHAR(10) NOT NULL, 140|
PRIMARY KEY (`flight_id`),
141| INDEX `fk_Flights_Tickets1_idx` (`ticket_number` ASC), 142| INDEX
`fk_Flights_Pilots1_idx` (`pilot_id` ASC),
143| INDEX `fk_Flights_Flight_Attendents1_idx` (`FA_id` ASC), 144|
INDEX `fk_Flights_Aircraft1_idx` (`aircraft_code` ASC), 145| CONSTRAINT
`fk_Flights_Tickets1`
146| FOREIGN KEY (`ticket_number`)
147| REFERENCES `mydb`.`Tickets` (`ticket_number`) 148| ON
DELETE NO ACTION
149| ON UPDATE NO ACTION,
```

```
150|  CONSTRAINT  `fk_Flights_Pilots1`  151|  FOREIGN
KEY (`pilot_id`)
152| REFERENCES `mydb`.`Pilots` (`pilot_id`) 153| ON DELETE
NO ACTION
154|  ON UPDATE NO ACTION,
155|  CONSTRAINT  `fk_Flights_Flight_Attendents1`  156|  FOREIGN
KEY (`FA_id`)
157| REFERENCES `mydb`.`Flight_Attendents` (`FA_id`) 158| ON DELETE
NO ACTION
159|  ON UPDATE NO ACTION,
160|  CONSTRAINT  `fk_Flights_Aircraft1`  161|  FOREIGN
KEY (`aircraft_code`)
162| REFERENCES `mydb`.`Aircraft` (`aircraft_code`) 163| ON DELETE
NO ACTION
164| ON UPDATE NO ACTION) 165| ENGINE =
InnoDB;
166|
167| SHOW WARNINGS; 168|
169| -- -----------------------------------------------------
170| -- Table `mydb`.`Luggage`
171| -- -----------------------------------------------------
172| DROP TABLE IF EXISTS `mydb`.`Luggage` ; 173|
174| SHOW WARNINGS;
175| CREATE TABLE IF NOT EXISTS `mydb`.`Luggage` ( 176|
`luggage_id` INT NOT NULL,
177| `customer_id` INT NOT NULL, 178| `flight_id` INT
NOT NULL,
179|  PRIMARY KEY (`luggage_id`),
180| INDEX `fk_Luggage_Customers1_idx` (`customer_id` ASC), 181| INDEX
`fk_Luggage_Flights1_idx` (`flight_id` ASC),
182|  CONSTRAINT  `fk_Luggage_Customers1`  183|  FOREIGN
KEY (`customer_id`)
184| REFERENCES `mydb`.`Customers` (`customer_id`) 185| ON DELETE
NO ACTION
186|  ON UPDATE NO ACTION,
187|  CONSTRAINT  `fk_Luggage_Flights1`  188|  FOREIGN
KEY (`flight_id`)
189| REFERENCES `mydb`.`Flights` (`flight_id`) 190| ON DELETE
NO ACTION
191| ON UPDATE NO ACTION) 192| ENGINE =
InnoDB;
193|
194| SHOW WARNINGS; 195|
196| -- -----------------------------------------------------
```

```
197| -- Table `mydb`.`Security`
198| -- -----------------------------------------------------
199| DROP TABLE IF EXISTS `mydb`.`Security` ; 200|
201| SHOW WARNINGS;

202| CREATE TABLE IF NOT EXISTS
`mydb`.`Security` ( 203| `security_id` INT NOT
NULL,

204| `pilot_id` INT NOT
NULL, 205| `flight_id` INT
NOT NULL, 206| `FA_id`
INT NOT NULL, 207|
`luggage_id` INT NOT
NULL,
208| `aircraft_code` VARCHAR(10)
NOT NULL, 209| `customer_id` INT
NOT NULL,
210| PRIMARY KEY (`security_id`),
211| INDEX `fk_Security_Pilots1_idx`
(`pilot_id` ASC), 212| INDEX
`fk_Security_Flights1_idx` (`flight_id` ASC),
213| INDEX `fk_Security_Flight_Attendents1_idx`
(`FA _id` ASC), 214| INDEX
`fk_Security_Luggage1_idx` (`luggage_id` ASC), 215|
INDEX `fk_Security_Aircraft1_idx` (`aircraft_code`
ASC), 216| INDEX `fk_Security_Customers1_idx`
(`customer_id` ASC), 217| CONSTRAINT
`fk_Security_Pilots1`
218| FOREIGN KEY (`pilot_id`)
219| REFERENCES `mydb`.`Pilots`
(`pilot_id`) 220| ON DELETE NO
ACTION
221| ON UPDATE NO ACTION,
222| CONSTRAINT
`fk_Security_Flights1` 223|
FOREIGN KEY (`flight_id`)

224| REFERENCES `mydb`.`Flights`
(`flight_id`) 225| ON DELETE NO
ACTION
226| ON UPDATE NO ACTION,
227| CONSTRAINT
`fk_Security_Flight_Attendents1` 228|
FOREIGN KEY (`FA_id`)

229| REFERENCES `mydb`.`Flight_Attendents`
(`FA_id`) 230| ON DELETE NO ACTION
231| ON UPDATE NO ACTION,
232| CONSTRAINT
```

`fk_Security_Luggage1` 233|
FOREIGN KEY (`luggage_id`)

234| REFERENCES `mydb`.`Luggage`
(`luggage_id`) 235| ON DELETE *NO ACTION*
236| ON UPDATE *NO ACTION*,
237| CONSTRAINT
`fk_Security_Aircraft1` 238|
FOREIGN KEY (`aircraft_code`)

239| REFERENCES `mydb`.`Aircraft`
(`aircraft_code`) 240| ON DELETE *NO ACTION*
241| ON UPDATE *NO ACTION*,
242| CONSTRAINT
`fk_Security_Customers1` 243|
FOREIGN KEY (`customer_id`)

244| REFERENCES `mydb`.`Customers`
(`customer_id`) 245| ON DELETE *NO ACTION*

246| ON UPDATE *NO ACTION*) 247| ENGINE =
InnoDB;
248|
249| SHOW WARNINGS;

Разберем основные операторы SQL-запросов, которые используются в рамках данного программного кода.

В строках 1-2 проверяется существует ли база данных с именем 'mydb' и, если нет, то такая база создается, при этом для базы выбирается кодировка utf8 и, в часности, диалект utf8_general_ci данной кодировки.

Заданная кодировка указывает обработчику MySQL, обрабатывающему данные в базе данных и отображающему их пользователям базы данных по их запросам, в какой кодировке хранить и представлять текст в таблицах базы данных. Если, например, для базы данных задать кодировку utf8, но при этом поместить в какую-нибудь таблицу этой базы текст в кодировке Windows-1251, то такой текст будет нечитаем при выводе его пользователю, т.е. будет содержать нечитаемые символы.

Поэтому в большинстве приложений, например, в веб-сайтах, работающих с MySQL базой данных, все вводимые пользователем на сайте символы – логин, пароль, ФИО и т.д. сразу автоматически кодируют в нужную кодировку для дальнейшего корректного сохранения текста в базе данных.

В строках 6-8 программного кода записан комментарий для разработчиков, которые, возможно, будут сопровождать данную базу данных и её программный код. Комментарий говорит о том, что далее идет SQL-

запрос, создающий таблицу 'Aircraft' в базе данных 'mydb':

```
-- ------------------------------------------------
-- Table `mydb`.`Aircraft`
-- ------------------------------------------------
```

В строках 9-18 происходит непосредственно создание таблицы '*Aircraft*'. При этом сначала проверяется не существует ли уже данной таблицы (строка 9) – если таблица с таким именем существует, то она удаляется; также включаются предупреждения (строка 11), которые будут выведены программисту, если при создании таблицы что-то пойдет не так или таблица вовсе не создастся. Непосредственно создание таблицы происходит кодом в строках 12-18 – здесь задается имя создаваемой таблицы (строка 12), имена, типы данных их свойства и длина полей (столбцов) таблицы (строки 13-16), столбец, который будет использоваться в качестве первичного уникального ключа (строка 17) и тип таблицы (строка 18).

В качестве типа данных для столбцов указан *VARCHAR* , т.е. текстовое поле переменной длины с минимальным значением для поля, указанным в скобках (может быть разным для каждого поля). В качестве свойства для всех столбцов, кроме первичного ключа, указано *NULL* – это означает что поля могут быть пустыми (не содержать никакого значения), в то время как столбец, который является первичным ключом таблицы, не может иметь пустого значения – для него указано свойство *NOT NULL*, запрещающее ввод и хранение в нем пустого значения. В качестве типа таблицы выбран '*InnoDB*' – данный тип таблиц позволяет в MySQL создавать между ними связи.

В строке 20 снова указан оператор, сигнализирующий программисту в том случае, если предыдущий запрос на создание таблицы завершился некорректно.

Аналогичным образом в строках 22-54 указаны SQL-запросы, создающие в базе '*mydb*' таблицы '*Airport*' и '*Customers*'. Единственным отличием является тип данных в поле `customer_zip` таблицы '*Customers*', в котором хранится почтовый индекс клиента – для данного поля указан тип *INT(6)*, то есть натуральное число, содержащее максимум 6 знаков, так как почтовые индексы РФ содержат не более 6 цифр. Стоит отметить, что если аэропорт планируется использовать для международных рейсов, которыми смогут пользоваться граждане и других стран, то данное поле лучше сделать текстовым полем переменной длины (*VARCHAR*), так как почтовые индексы других стран могут содержать другое количество цифр и буквы – данный момент важно продумать непосредственно на этапе проектирование базы данных (при создании ER-модели), так как впоследствии, когда таблица уже будет заполнена данными, преобразовать тип этого поля из числового в текстовый может быть невозможным.

В строках 56-82 записан SQL-запрос, создающий таблицу '*Tickets*'. Он

аналогичен уже рассмотренным запросам, с тем лишь отличием, что содержит операторы, создающие для таблицы индексы (строки 67-68), служащие для ускорения поиска данных в таблице при выполнении запросов, и внешние ключи (строки 69-79), служащие для связи данной таблицы с двумя другими таблицами рассматриваемой базы данных – таблицами '*Airport*' и '*Customers*'. Оператор создания внешнего ключа содержит следующие директивы: *FOREIGN KEY* – для того чтобы задать название внешнему ключу; *REFERENCES* – для того чтобы указать название другой таблицы, с которой осуществляется связь; *ON DELETE* – для того чтобы задать действие, которое выполняется автоматически над данными в данной таблице (в таблице '*Tickets*') при удалении связанных с ними данными в связуемой (главной) таблице (в данном случае в таблице '*Airport*'); *ON UPDATE* – для того чтобы задать действие, которое выполняется автоматически над данными в данной таблице (в таблице '*Tickets*') при обновлении связанных с ними данными в связуемой (главной) таблице (в данном случае в таблице '*Airport*') – в нашем случае в обеих директивах указано '*NO ACTION*', что означает что никаких действий с данными в рассматриваемой таблице '*Tickets*' при обновлении/удалении данных в таблице '*Airport*' производить не нужно.

Остальные таблицы и связи между ними, указанные в рассматриваемой ER-модели создаются аналогичным образом (строки 84-249 программного кода).

Более подробную документацию по работе в СУБД MySQL с примерами различных запросов (на создание/удаление таблиц и связей, выборку данных и т.п.) вы можете найти на следующих сайтах:

http://dev.mysql.com/doc/
http://www.mysql.ru/docs/

**Варианты для выполнения курсовой работы**

Выбираются по последней цифре в зачетной книжке

**Вариант 1** – Деканат

Задача – информационная поддержка деятельности деканата вуза.

− ведение расписания сессии, хранение результатов сессии;

− составление зачётных и экзаменационных ведомостей;

− составление расписаний экзаменов по группам, кафедрам, для отдельных преподавателей;

− проверка корректности расписания экзаменов (уникальность комбинации "время – дата – аудитория"; между экзаменами в одной группе должно пройти не менее трёх дней);

− подсчёт по результатам зачётов и экзаменов итоговых значений (количество оценок '5', '4', '3', '2', количество неявок, средний балл по группе);

− получение списка экзаменов на текущую дату.

**Вариант 2** – Книжный магазин

Необходимо построить базу данных, располагая которой пользователь может получить справочную информацию о работе книжного магазина:

− список книг, продаваемых в данном книжном магазине;
− количество экземпляров книги;
− жанры книг (поэзия, проза, фантастика, учебная литература и т.д.);
− перечень издательств, поставляющих книги данному магазину;
Пользователю на основе данных из базы данных необходимо:
− сформировать выходной документ "Величина продаж по каждому жанру книг";
− определить самую продаваемую книгу;
− перечень издательств, специализирующихся на конкретных жанрах;
− определить расчетным путем общую выручку магазина за отчетный период и т.д.

**Вариант 3** – Плановый отдел
Задача – информационная поддержка деятельности планового отдела (выбрать конкретное производство).
БД должна осуществлять:
− ведение плановой документации по основному и вспомогательному производствам (план и факт);
− составление заказов на поставку сырья и комплектующих (в соответствии с планом выпуска продукции);
− составление планов работы вспомогательных производств для обеспечения потребностей основного производства;
− подсчёт энергозатрат;
− определение соответствия результатов работы плану (в процентах).

**Вариант 4** – Кафедра
Задача – информационная поддержка учебного процесса и организационной деятельности на кафедре вуза. БД должна содержать учебный план, расписание занятий, списки групп, выпускаемых кафедрой, и списки аспирантов (с руководителями и темами исследований). БД должна обеспечивать составление:
− расписания занятий на семестр (по группам);
− учебного плана (по семестрам) для каждого курса;
− расписания занятий для преподавателей;
− списка телефонов сотрудников;
− нагрузки по часам для преподавателей;
− списка научных кадров по научным направлениям;
− списков студентов-дипломников (по группам и по преподавателям).

**Вариант 5 –** Библиотека
Задача – информационная поддержка деятельности научно-технической библиотеки.
БД должна включать два раздела: "Научная литература" и "Журнальные публикации". БД должна обеспечивать:
− ведение автоматизированного учёта выдачи/приёма литературы;

− ведение очередей на литературу (по заказам);

− учёт рейтинга изданий (количество читателей и дата последней выдачи);

− поиск литературы по требуемым разделу, теме, автору, ключевому слову (с заданием интересующего периода);

− составление списков должников по годам.

**Вариант 6 –** Больница

Задача – информационная поддержка деятельности регистратуры больницы. БД должна осуществлять:

− учёт поступления пациентов (по отделениям);

− учёт проведённого лечения;

− учёт платных услуг с выдачей счетов на оплату;

− ведение архива выписанных пациентов.

Необходимо предусмотреть определение (по отделениям):

− пропускной способности больницы;

− среднего времени пребывания больных в стационаре;

− наличия свободных мест в палатах (отдельно для мужчин и для женщин);

− количества прооперированных пациентов (из них – с осложнениями и умерших);

− смертности.

**Вариант 7 –** Магазин (выбрать конкретный профиль)

Задача – информационная поддержка деятельности магазина выбранного профиля. БД должна осуществлять:

− учёт поставщиков и поставок;

− учёт продаж по отделам;

− подсчёт остатков товаров (по отделам);

− оформление заказов на товары, запасы которых подходят к концу;

− подведение финансовых итогов дня (по отделам и в целом по магазину);

− анализ результативности работы продавцов (для премирования);

− анализ объёмов продаж по дням недели и по месяцам.

**Вариант 8 –** БД адвоката

Задача – информационная поддержка деятельности адвокатской конторы. БД должна осуществлять:

− ведение списка адвокатов;

− ведение списка клиентов;

− ведение архива законченных дел.

Необходимо предусмотреть:

− получение списка текущих клиентов для конкретного адвоката;

− определение эффективности защиты (максимальный срок минус полученный срок) с учётом оправданий, условных сроков и штрафов;

− определение неэффективности защиты (полученный срок минус минимальный срок);

− подсчёт суммы гонораров (по отдельных делам) в текущем году;

− получение для конкретного адвоката списка текущих клиентов, которых он защищал ранее (из архива, с указанием полученных сроков и статей).

**Вариант 9 –** БД риэлторской компании

Задача – информационная поддержка деятельности фирмы, занимающейся продажей и арендой жилых и нежилых помещений. БД должна:

− осуществлять ведение списков жилых и нежилых помещений, предназначенных для аренды и/или продажи;

− поддерживать архив проданных и сданных в аренду помещений;

− производить поиск вариантов в соответствии с требованиями клиента.

Необходимо предусмотреть получение разнообразной статистики:

− наличие помещений разных типов;

− изменение цен на рынке;

− уровни спроса и предложения;

− средние показатели (среднее время нахождения помещения в БД (по типам помещений), среднюю стоимость аренды/продажи помещений и т.п.

**Вариант 10. –** Гостиница

Создаваемая информационная система предназначена для администрации гостиницы, которая на основании информации о номерах занимается размещением клиентов в соответствии с их запросами. При выбытии клиента информация о номере, в котором он проживал, должна обновляться, а информация о клиенте должна удаляться из рабочих таблиц (карточки регистрации клиентов и карточки учета) и помещаться в архивную таблицу.

БД должна осуществлять:

− ведение списка постояльцев;

− учёт забронированных мест (с учетом класса комнаты);

− ведение архива выбывших постояльцев за последний год.

Необходимо предусмотреть:

− получение списка свободных номеров (по количеству мест и классу);

− получение списка номеров (мест), освобождающихся сегодня и завтра;

− выдачу информации по конкретному номеру;

− автоматизацию выдачи счетов на оплату номера и услуг;

− получение списка забронированных номеров;

− проверку наличия брони по имени клиента и/или названию организации.

**Вариант 11 –** Справочная служба аптек

Информационная система предназначена как информационной поддержки работы фармацевтов и для клиентов.

В системе должен вестись учет лекарственных препаратов (цена, наличие как в самой аптеке, так и на складе). Для каждого препарата должны быть указаны: состав, фармакотерапевтическая группа, условия отпуска. Также для лекарственных препаратов должна указываться форма выпуска (таблетки, настойка, мазь, гель и т.п.), условия хранения, срок годности.

Система должна:

− отслеживать выдачу лекарственных препаратов по рецепту;

− отслеживать поступление лекарственных препаратов с указанием цены, наименования производителей;

− выдавать отчет о проданных лекарствах за отчетный период для ведения статистического анализа.

**Вариант 12. –** Спортивный клуб

Информационная система содержит информацию о деятельности спортивного клуба. БД должна осуществлять:

− ведение списков спортсменов и тренеров;

− учёт проводимых соревнований (с ведением их архива);

− учёт травм, полученных спортсменами.

Необходимо предусмотреть:

− возможность перехода спортсмена от одного тренера к другому;

− составление рейтингов спортсменов;

− составление рейтингов тренеров;

− выдачу информации по соревнованиям;

− выдачу информации по конкретному спортсмену;

− подбор возможных кандидатур на участие в соревнованиях (соответствующего уровня мастерства, возраста и без травм).

**Вариант 13 –** Отдел кадров ВУЗа

Задача – информационная поддержка деятельности отдела кадров.

Различают три группы сотрудников: а) администрация; б) преподавательский и инженерно-технический состав (по кафедрам); в) технический персонал. БД должна содержать штатное расписание по отделам (кафедрам) с указанием количества ставок по должностям, включать архив сотрудников и учитывать сотрудников, находящихся в отпуске по уходу за ребенком.

БД должна предоставлять возможность составления должностных (штатных) расписаний по кафедрам и отделам и следующих списков:

− вакансий (с учётом сотрудников, находящихся в отпуске по уходу за ребенком, т.е. с указанием даты, до которой ставка свободна);

− пенсионеров;

− людей предпенсионного возраста (не более 2-х лет до пенсии);

− бездетных сотрудников;

− юбиляров текущего года;
− многодетных сотрудников (трое и более детей);
− ветеранов (работающих в институте не менее тридцати лет);
− сотрудников, работающих более чем на одной ставке.

**Вариант 14 –** Санаторий

Система предназначена для администрации санатория. Система должна содержать информацию о клиентах санатория (как прошлых, так и настоящих): ФИО, дату приезда, дату выезда, процедуры, длительность курсов лечения, курирующем враче.

Также система должна содержать информацию о стоимости процедур, длительности курсов лечения.

Предусмотреть скидки постоянным клиентам, а также сезонные скидки.

**Вариант 15 –** Специализированная научная библиотека

Рассмотрим специализированную библиотеку, которая располагает

книжным фондом определенной тематической направленности. Предполагается, что каждая книга фонда может быть как в одном экземпляре, так и в нескольких. Поэтому каждой книге соответствует уникальный инвентарный номер и библиотечный код. Данные о книге содержатся в библиографической карточке, карточки объединяются в каталоги. Существует два вида каталогов: алфавитный и тематический; в алфавитном каталоге карточки отсортированы по фамилии автора, а в тематическом – сначала по темам, а в пределах каждой темы – по фамилии автора.

Библиотека выдает книги читателям во временное пользование. При записи в библиотеку каждому читателю присваивается порядковый номер, ему выдается читательский билет и для него заводится учетная карточка. Учетная карточка кроме данных о читателе в дальнейшем будет содержать информацию о выданных и возвращенных книгах.

Создаваемая информационная система предназначена, прежде всего, для ведения данных: о книгах (регистрация новых поступлений, списание литературы), о читателях (регистрация новых читателей, удаление информации о выбывших читателях), а также о перемещении книг между библиотекой и читателями, что должно найти отражение в таблицах книжный фонд и выдача книг.

Кроме того, в системе должны быть реализованы возможности просмотра и поиска как среди книг, так и среди читателей.

**Вариант 16 –** Стоматологическая поликлиника

Поликлиника ведет прием и учет пациентов, учет их посещений (визитов) и учет обслуживания пациентов специалистами (врачами) поликлиники.

Существует необходимость в хранении информации обо всех посещениях поликлиники пациентами и о том, на приеме у каких специалистов они находились. Обслуживание пациентов ведется по предварительной записи.

Необходимо обеспечить ввод, хранение и, возможно, редактирование данных. В определенных случаях необходимо выполнять удаление данных. Например, можно удалить информацию обо всех визитах некоторого пациента, если после его последнего визита прошел определенный срок (например, 3 года), а данные о самом пациенте перенести в архив (или также удалить).

Также система должна содержать график работы каждого врача, его специализацию и категорию. Необходимо предусмотреть поиск сведений о пациентах как по фамилии, так и по номеру истории болезни. За каждое посещение пациенту выписывается счет, который он должен оплатить.

Кроме задач, перечисленных выше, информационная система должна решать следующие задачи:

− подсчет выручки каждого специалиста за определенный период (день, месяц, квартал);

− подсчет выручки поликлиники в целом за определенный период (день, месяц, квартал);

− подсчет оплаченной суммы за лекарства за определенный период (день, месяц, квартал);

− подсчет количества посещений поликлиники за месяц в целом и по каждой группе специалистов.

**Вариант 17 –** Ателье мод

Ателье мод выполняет заказы клиентов на индивидуальный пошив одежды. В ателье существует каталог моделей и каталог тканей. По каталогу моделей клиент выбирает модель, а по каталогу тканей – ткань, из которой будет выполнена модель, и заказывает ее пошив в ателье.

Заказ каждого клиента содержит: Ф.И.О. клиента, информацию о модели, информацию о ткани, Ф.И.О. закройщика (исполнителя заказа), дату приема заказа, дату примерки, отметку о выполнении заказа, дату выполнения заказа.

В каталоге моделей каждая модель имеет уникальный номер, для каждой модели указывается рекомендуемая ткань, необходимый расход ткани для данной модели с учетом ширины ткани, цена готовой модели, включающая цену ткани и стоимость пошива изделия.

В каталоге тканей каждая ткань имеет уникальный номер, название, а также указываются ее ширина и цена за 1 метр.

В ателье есть еще и склад тканей. В книге учета тканей на складе для каждой ткани указывается общий метраж, который изменяется, если принимается заказ на изготовление модели из данной ткани.

**Вариант 18 –** Оптовый склад

Склад осуществляет продажу товаров оптом. Любая фирма, занимающаяся продажей товаров в розницу, закупает необходимые ей товары на складе, который служит посредником между производителями и продавцами.

На склад товар поступает от некоторой фирмы-поставщика, в свою очередь склад продает товар фирме-покупателю, заключая с ним сделку о продаже товара.

Необходимо вести учет поставщиков, покупателей, продаж, движения товара на складе. Кроме того, можно делать выводы о работе склада, спросе на определенные товары, выгодности работы с некоторыми поставщиками и покупателями.

**Вариант 19 –** Торгово-закупочное предприятие

Торгово-закупочное предприятие имеет склад, содержащий определенные виды товаров, например, продовольственные товары. Предприятие имеет штат сотрудников, являющихся агентами-реализаторами. Предприятие выдает агенту товар, устанавливая цену его продажи. Агент-реализатор оплачивает выданный товар не сразу, а по мере его реализации, оформляя приходные кассовые ордера. С каждой единицы проданного товара агент получает оплату, установленную предприятием.

Необходимо вести учет движения товаров как на складе, так и у агентов-реализаторов. Кроме того, предприятие проводит операции: по новым поступлениям товара, по выдаче товара агенту, по расчету с агентом за реализованный товар.

**Вариант 20 –** Автосалон

Существует фирма, торгующая автомобилями. Автомобиль выступает в качестве товара и как товар имеет определенные характеристики. Кроме того, на каждый автомобиль имеются исчерпывающие технические данные. Фирма имеет своих клиентов – покупателей автомобилей, сведения о которых хранит.

Необходимо обеспечить ввод, редактирование и просмотр данных в удобной для пользователя форме.

Предполагается также решение следующих задач:

− выдать информацию о наличии автомобилей определенной марки и модели;

− выдать технические данные заданной модели;

− выдать информацию обо всех проданных моделях некоторой марки, значение которой вводится в качестве параметра;

− посчитать сумму продаж моделей каждой марки и общую сумму продаж;

− выдать полную или частичную информацию о клиентах фирмы;

− выдать списки клиентов и автомобилей по виду оплаты.

Возможны постановка и решение других задач.

**Вариант 21 –** Продажа подержанных автомобилей

Фирма по продаже подержанных автомобилей работает с физическими лицами – клиентами фирмы, имеющими подержанный автомобиль или автомобили и желающими продать их через фирму. Непосредственной продажей автомобилей занимаются сотрудники фирмы – дилеры. На каждый предлагаемый в продажу автомобиль фирма заключает с клиентом договор, содержащий данные о клиенте, необходимые сведения об автомобиле, а также данные о дилере, обслуживающем этот договор.

В создаваемой информационной системе необходимо обеспечить ввод и редактирование данных. Кроме того, необходимо выдавать информацию о клиентах и предлагаемых ими автомобилях, а также информацию о деятельности дилеров (с перечислением договоров) и клиентах, которые они обслуживают.

Могут быть выполнены разнообразные запросы, например:

− посчитать количество договоров, заключенных с каждым клиентом;

− посчитать количество договоров, обслуживаемых каждым дилером;

− выдать некоторую информацию (например: данные дилера, дата заключения договора, данные клиента, отметка о продаже) обо всех договорах, договорах за некоторый промежуток времени или договорах, удовлетворяющих определенному условию.

**Вариант 22 –** Ассоциация крестьянских фермерских хозяйств

Предполагается, что существует ассоциацией, которая является организационным объединением крестьянских фермерских хозяйств (КФХ). Ассоциация ведет учет зарегистрированных фермерских хозяйств, собирает информацию о видах их деятельности, а также о предлагаемой хозяйствами продукции и ее цене, ведет статистический учет. Вид деятельности хозяйства определяет его специализацию, например: овощеводство, животноводство, виноградарство и другие. В каждой специализации имеются виды производимых товаров.

Необходимо обеспечить ввод и обновление данных, возможности анализа товаров и цен. Необходимо также предусмотреть возможность получение информации о деятельности конкретных хозяйств: их продукции и ценах, а также получение информации о конкретных видах товаров: их производителях и ценах.

**Вариант 23 –** Пассажирское автопредприятие

Муниципальное автопредприятие осуществляет пассажирские перевозки на внутригородских маршрутах. Автопредприятие имеет парк автобусов, которые работают на определенных маршрутах. Работу автопредприятия обеспечивает персонал предприятия, который можно разделить по категориям занимаемых должностей надминистрацию, инженерно-технический персонал и персонал, обслуживающий маршруты (водители, кондукторы). Выезжая на маршрут, водитель автобуса получает маршрутный лист (или путевой лист), содержащий

данные об автобусе, маршруте, режиме работы, водителе, кондукторе.

В учетных данных персонала должны вестись данные для оплаты труда, если предполагается автоматизация начисления зарплаты. В маршрутных листах можно ввести плановую и фактическую выручки за смену соответственно.

БД должна обеспечивать выполнение запросов:

− выдать список сотрудников администрации с указанием должности;

− на определенную дату для всех номеров маршрутов выдать информацию о количестве автобусов, обслуживающих каждый маршрут;

− по каждому номеру маршрута и дате (параметры запроса) выдать информацию об автобусах, обслуживающих маршрут: бортовой номер, марка, гос. номер автобуса;

− по итогам работы за месяц посчитать количество рейсов, выполненных каждым автобусом или на каждом маршруте;

− по итогам работы за месяц посчитать количество смен, отработанных каждым водителем и кондуктором.

**Вариант 24 –** Междугородные пассажирские перевозки

Рассмотрим автовокзал, который занимается обслуживанием и учетом пассажиров на междугородных автобусных маршрутах. На автовокзале имеется

расписание движения автобусов, содержащее информацию о маршрутах и рейсах. Кроме того, на автовокзале имеется справочное бюро, в котором можно

получить информацию о наличии мест на определенный рейс конкретной даты.

И, наконец, на автовокзале есть кассы, в которых пассажир может приобрести билет. Кассы начинают предварительную продажу билетов за определенный промежуток времени до дня отправления автобуса (например, за 10 дней).

Необходимо построить такую базу данных, в которой хранится информация как о технических характеристиках маршрутов, содержащаяся в расписании, так и нформация о наличии мест на рейсы, и информация о пассажирах, купивших билеты на определенный рейс.

Также система должна выдавать ответы на следующие запросы:

− наличие свободных мест на рейс;

− продажу билетов в оба конца;

− поиск места на рейс в соответствии с требованиями заказчика;

− количество пассажиров уже выполненного рейса, доходность рейса;

− список всех пассажиров определенного рейса (выполненного или того, на который идет продажа билетов);

− определить, покупал ли билет человек с заданной фамилией и, если покупал, то на какой рейс;

− определение количества перевезенных пассажиров и объем перевозок (в денежном выражении) по дням, по месяцам в целом по всем направлениям или по определенному маршруту.

**Вариант 25 –** Агентство по продаже авиабилетов

Агентство занимается продажей авиабилетов на различные рейсы, ведет учет проданных билетов и учет пассажиров, купивших билеты. Особенность данной задачи состоит в том, что информация в базе данных может использоваться как пассажирами (например, для получения сведений о расписании и наличии свободных мест на рейс), так и служащими агентства: кассирами и диспетчерами (администраторами).

В системе должен осуществляться поиск следующей информации:

− расписание рейсов (номер рейса, тип самолета, пункт отправления, пункт назначения, дата вылета, время вылета, время полета, цена билета);

− информация о свободных местах на рейс (номер рейса, дата вылета, общее количество мест, количество свободных мест);

− информация о пассажирах, заказавших билет (фамилия, имя, отчество, предъявленный документ, его серия и номер, номер рейса, дата вылета).

**Вариант 26**

Требуется создать БД, предназначенную для менеджера музыкальных групп. Такая система должна обеспечивать хранение сведений о группах, включающих название группы, год образования и страну, состав исполнителей, положение в последнем хит-параде; репертуар группы. Сведения о каждой песне из репертуара группы - это ее название, композитор, автор текста. Необходимо также хранить данные о последней гастрольной поездке каждой группы: название гастрольной программы, названия населенных пунктов, дата начала и окончания выступлений, средняя цена билета (зависит от места выступления и положения группы в хит-параде). Возможно появление новой группы и изменение состава исполнителей. Каждая песня может быть в репертуаре только одной группы.

Необходимо предусмотреть возможность выдачи отчета о составе групп и их репертуаре, а также отчета о последней гастрольной поездке указанной группы (места и сроки выступлений, цены на билеты, количество проданных билетов, репертуар с указанием авторов песен, общая сумма выручки).