МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и вычислительной техники

Кафедра информационной безопасности

Отчёт к курсовому проекту

по дисциплине “Безопасность систем баз данных”

**Разработка базы данных для института**

Выполнили: студенты группы БИ-31

Катягин Д. Д., Мунтян М.М., Лазарев А. В.

Проверил: доцент кафедры

ИБ Сучков Д.С.

Йошкар-Ола

2019 г.

**СОДЕРЖАНИЕ**

**Введение**3

**1. Техническое задание4**

1.1 Требования к курсовой работе4

1.2 Требования к базе данных4

1.3 Требования к API (минимальное количество реализованных методов)2

**2. Порядок выполнения работы5**

2.1 Этапы разработки базы данных5-10

2.2 Этапы разработки API10-31

**3. Доказательство31**

3.1 Первая начальная форма31

3.3 Вторая начальная форма31

3.3 Третья начальная форма31

**4, Приложения32**

3.1 ER-диаграмма32

3.2 Ссылка на github.com32

**5. Вывод32**

**Введение**

В курсовой работе рассматривается создание базы данных, предназначенной для автоматизации работы института. База данных позволяет клиентам просматривать факультеты, специальности и зарплату. Также реализована автоматизация удаление, вставки и изменения которая позволяет пользователям узнать информацию.

## **Техническое задание**

## ***1.1 Требования к курсовой работе:***

* Получить структуру данных из файла, согласно варианта. Привести к 3й нормальной форме. Добавить недостающие таблицы.
* Составить ER-диаграмму
* Разработать API для базы данных на любом языке, выполняющемся на стороне сервера (php, ASP.NET, Java, python, node.js, etc)
* Взаимодействие должно осуществляться по клиент-серверной архитектуре, подключение с клиентской программы недопустимо
* Провести настройку пользователей базы данных для разграничения прав доступа, привести пример конфигурации
* Все документы и исходные коды для курсовой работы должны храниться под контролем системы контроля версий — git или mercurial (https://github.com/, https://bitbucket.org/)
* Во время сдачи курсового проекта необходимо предоставить отчет о проделанной работе в печатном виде (отчет)

## ***1.2 Требования к базе данных***

* Наличие не менее 7 таблиц, в том числе таблицы сессий и пользователей
* Структура таблицы должна содержать не менее 3-х полей, одно из которых ключевое
* Правомерное использование типов данных
* Обязательно использование триггеров и/или хранимых процедур
* Форма нормализации не менее 3NF
* Индексирование по полям поиска

## ***1.3 Требование к API (минимальное количество реализованных методов)***

* аутентификация пользователя (создание сессии);
* добавление/удаление/изменение данных в таблицах;
* выборка данных их ключевых таблиц по запросам;
* выборка данных из таблиц с объединением результатов.

1. **Порядок выполнения работы**
   1. Этапы разработки базы данных

Разработана база данных, содержащая 12 таблиц. Для авторизации пользователя используется таблица - *people,* содержащая информацию об аккаунте пользователя. Для хранения информации о пользователе используются следующие таблицы:

*address* – адрес, время регистрации, емаил;

*company –* дата рождения, округ, номер телефона, компания;

*card\_number –* номер карты;

Для автоматизации оформления пользователя, были созданы следующие таблицы:

*salary –* назначение стипендии/зарплаты;

*faculties* – информация о факультетах;

*specials* – информация о специальностях;

Структуры, реализованных таблиц:

CREATE TABLE `people` (

`id` char(32) NOT NULL,

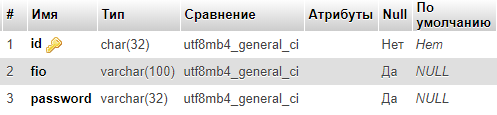
`fio` varchar(100) DEFAULT NULL,

`password` varchar(32) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `people`

ADD PRIMARY KEY (`id`);



*Рис 1. People*

CREATE TABLE `address` (

`id` char(32) NOT NULL,

`address` varchar(100) DEFAULT NULL,

`email` varchar(100) DEFAULT NULL,

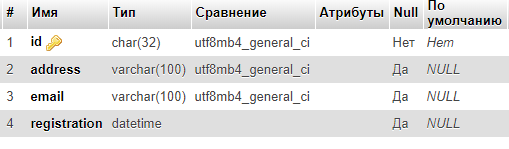
`registration` datetime DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `address`

ADD PRIMARY KEY `id` (`id`),

ADD CONSTRAINT `address\_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



*Рис 2. address*

CREATE TABLE `company` (

`id` char(32) DEFAULT NULL,

`b\_date` datetime DEFAULT NULL,

`about` varchar(100) DEFAULT NULL,

`phoneNumber` varchar(50) DEFAULT NULL,

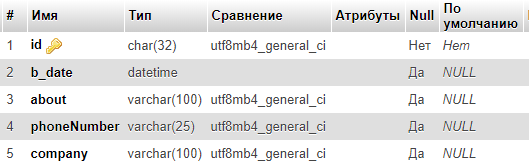
`company` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `company`

ADD PRIMARY KEY `id` (`id`),

ADD CONSTRAINT `company\_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



*Рис 3. company*

CREATE TABLE `card\_number` (

`id` char(32) DEFAULT NULL,

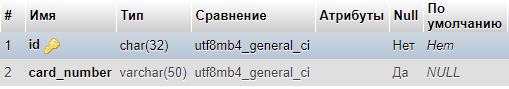
`card\_number` varchar(50) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `card\_number`

ADD PRIMARY KEY `id` (`id`),

ADD CONSTRAINT `card\_number\_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



*Рис 4. card\_number*

CREATE TABLE `students` (

`id` char(32) DEFAULT NULL,

`id\_special` char(32) DEFAULT NULL,

`id\_faculty` char(32) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `students`

ADD PRIMARY KEY `id` (`id`),

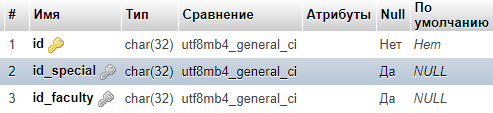
ADD KEY `id\_special` (`id\_special`),

ADD KEY `id\_faculty` (`id\_faculty`),

ADD CONSTRAINT `students\_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `students\_ibfk\_2` FOREIGN KEY (`id\_special`) REFERENCES `specials` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `students\_ibfk\_3` FOREIGN KEY (`id\_faculty`) REFERENCES `faculties` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



*Рис 5. Students*

CREATE TABLE `teachers` (

`id` char(32) DEFAULT NULL,

`id\_special` char(32) DEFAULT NULL,

`id\_faculty` char(32) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `teachers`

ADD PRIMARY KEY `id` (`id`),

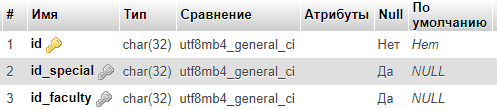
ADD KEY `id\_special` (`id\_special`),

ADD KEY `id\_faculty` (`id\_faculty`),

ADD CONSTRAINT ` teachers \_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT ` teachers \_ibfk\_2` FOREIGN KEY (`id\_special`) REFERENCES `specials` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `teachers\_ibfk\_3` FOREIGN KEY (`id\_faculty`) REFERENCES `faculties` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



*Рис 6. Teachers*

CREATE TABLE `staff` (

`id` char(32) DEFAULT NULL,

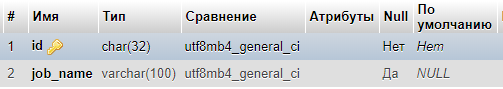
`job\_name` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `staff`

ADD PRIMARY KEY `id` (`id`),

ADD CONSTRAINT `staff\_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



*Рис 7. Staff*

CREATE TABLE `faculties` (

`id` char(32) NOT NULL,

`faculty\_name` varchar(100) DEFAULT NULL,

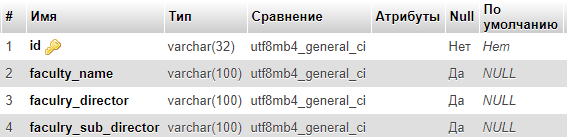
`faculry\_director` varchar(100) DEFAULT NULL,

`faculry\_sub\_director` varchar(100) DEFAULT NULL

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `faculties`

ADD PRIMARY KEY (`id`);



*Рис 8. Faculties*

CREATE TABLE `specials` (

`id` char(32) NOT NULL,

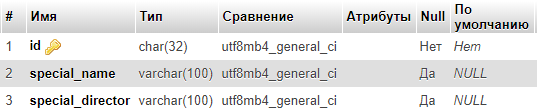
`special\_name` varchar(100) DEFAULT NULL,

`special\_director` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `specials`

ADD PRIMARY KEY (`id`);



*Рис 9. Specials*

CREATE TABLE `salary` (

`id` char(32) DEFAULT NULL,

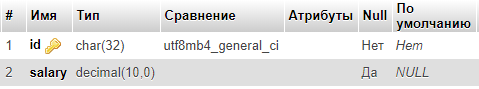
`salary` decimal(10,0) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `salary`

ADD PRIMARY KEY `id` (`id`),

ADD CONSTRAINT `salary\_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



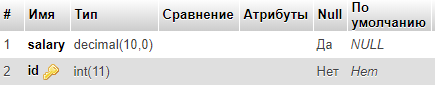
*Рис 10. Salary*

CREATE TABLE `top\_sal` (

`salary` decimal(10,0) DEFAULT NULL,

`id` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;



*Рис 11. top\_sal*

CREATE TABLE `online` (

`id` char(32),

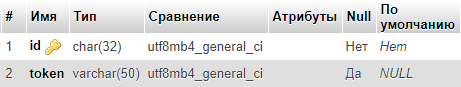
`token` varchar(50) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `online`

ADD PRIMARY KEY `id` (`id`),

ADD CONSTRAINT `online\_ibfk\_1` FOREIGN KEY (`id`) REFERENCES `people` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;



*Рис 12. Online*

Используемые триггеры отвечающие за подсчет всей зарплаты в базе “*kurs*”:

* CREATE TRIGGER `salary1` AFTER INSERT ON `salary`

FOR EACH ROW UPDATE top\_sal set salary = (SELECT SUM(salary) from salary) WHERE id = 1

* CREATE TRIGGER `salary2` AFTER UPDATE ON `salary`

FOR EACH ROW UPDATE top\_sal set salary = (SELECT SUM(salary) from salary) WHERE id = 1

* CREATE TRIGGER `salary3` AFTER DELETE ON `salary`

FOR EACH ROW UPDATE top\_sal set salary = (SELECT SUM(salary) from salary) WHERE id = 1

Проведена настройка пользователей базы данных для разграничения прав доступа:





*Рис 12. пользователи базы данных*

2.2 Этапы разработки API

Было разработано API для аутентификации пользователя, написанных на языке С#.

Разработаны функции авторизации пользователя.

1. **public** **string**[] LoginForm(**string** login, **string** password)
2. {
3. SqlConnection.Open();
4. **string** token;
5. **string**[] str = **new** **string**[3];
6. var role = '0';
7. var rand = **new** Random();
8. MySqlCommand command = **new** MySqlCommand("SELECT id FROM address WHERE email = @uL", SqlConnection);
9. command.Parameters.AddWithValue("@uL", login);
10. MySqlDataReader reader = command.ExecuteReader();
11. **if** (reader.Read())
12. {
13. **string** id = reader["id"].ToString();
14. reader.Close();
15. command = **new** MySqlCommand("SELECT id FROM people WHERE id = @id AND password = MD5(@uP)", SqlConnection);
16. command.Parameters.AddWithValue("@id", id);
17. command.Parameters.AddWithValue("@uP", password);
18. reader = command.ExecuteReader();
19. }
20. **if** (reader.Read())
21. {
22. **using** (SHA1 shaM = **new** SHA1Managed())
23. {
24. **byte**[] hash2 = shaM.ComputeHash(Encoding.UTF8.GetBytes(Convert.ToString(rand.Next(256))));
25. token = BitConverter.ToString(hash2).Replace("-", "").ToLower();
26. }
27. **string** id = Convert.ToString(reader["id"]);
28. reader.Close();
29. var online = Insert\_Online(id, token);
30. **if** (online == '0')
31. {
32. **for** (**int** i = 0; i < str.Length; i++)
33. {
34. str[i] = "0";
35. }
36. **return** str;
37. }
38. role = Choice\_student(token);
39. **if** (role == '0')
40. role = Choice\_teacher(token);
41. **if** (role == '0')
42. role = Choice\_staff(token);
43. **if** (role == '0')
44. {
45. **for**(**int** i = 0; i < str.Length; i++)
46. {
47. str[i] = "-1";
48. }
49. **return** str;
50. }
51. str[0] = id;
52. str[1] = token;
53. str[2] = role.ToString();
54. **return** str;
55. }
56. **else**
57. {
58. reader.Close();
59. **for** (**int** i = 0; i < str.Length; i++)
60. {
61. str[i] = "-1";
62. }
63. **return** str;
64. }
65. }
67. **public** **char** LoginForm\_load(**string** token)
68. {
69. SqlConnection.Open();
70. var id = Select\_Online(token);
71. **if** (id != **null**)
72. {
73. var role = Choice\_student(token);
74. **if** (role == '0')
75. role = Choice\_teacher(token);
76. **if** (role == '0')
77. role = Choice\_staff(token);
78. **return** role;
79. }
80. **else**
81. **return** '0';
82. }

Добавлены функции поиска факультета, специальности и зарплаты.

1. **public** **string** Fac\_find\_stud(**string** token)
2. {
3. SqlConnection.Open();
4. **string** id = Select\_Online(token);
5. **if** (id != **null**)
6. {
7. MySqlCommand command = **new** MySqlCommand("SELECT faculty\_name FROM faculties WHERE id IN (SELECT id\_faculty FROM students WHERE id = @id)", SqlConnection);
8. command.Parameters.AddWithValue("@id", id);
9. MySqlDataReader reader = command.ExecuteReader();
10. **if** (reader.Read())
11. {
12. var str = Convert.ToString(reader["faculty\_name"]);
13. SqlConnection.Close();
14. reader.Close();
15. **return** str;
16. }
17. **else**
18. {
19. SqlConnection.Close();
20. reader.Close();
21. **return** "Не нашло!";
22. }
23. }
24. **else**
25. {
26. SqlConnection.Close();
27. **return** "Ты не авторизирован";
28. }
29. }
31. **public** **string** Spec\_find\_stud(**string** token)
32. {
33. SqlConnection.Open();
34. **string** id = Select\_Online(token);
35. **if** (id != **null**)
36. {
37. MySqlCommand command = **new** MySqlCommand("SELECT special\_name FROM specials WHERE id IN (SELECT id\_special FROM students WHERE id = @id)", SqlConnection);
38. command.Parameters.AddWithValue("@id", id);
39. MySqlDataReader reader = command.ExecuteReader();
40. **if** (reader.Read())
41. {
42. var str = Convert.ToString(reader["special\_name"]);
43. SqlConnection.Close();
44. reader.Close();
45. **return** str;
46. }
47. **else**
48. {
49. SqlConnection.Close();
50. reader.Close();
51. **return** "Не нашло!";
52. }
53. }
54. **else**
55. {
56. SqlConnection.Close();
57. **return** **null**;
58. }
59. }
61. **public** **string** Sal\_find\_stud(**string** token)
62. {
63. SqlConnection.Open();
64. **string** id = Select\_Online(token);
65. **if** (id != **null**)
66. {
67. MySqlCommand command = **new** MySqlCommand("SELECT salary FROM salary WHERE id = @id", SqlConnection);
68. command.Parameters.AddWithValue("@id", id);
69. MySqlDataReader reader = command.ExecuteReader();
70. **if** (reader.Read())
71. {
72. var str = Convert.ToString(reader["salary"]);
73. reader.Close();
74. SqlConnection.Close();
75. **return** str;
76. }
77. **else**
78. {
79. SqlConnection.Close();
80. reader.Close();
81. **return** "Не нашло!";
82. }
83. }
84. **else**
85. {
86. SqlConnection.Close();
87. **return** **null**;
88. }
89. }

Добавлены функции получения таблиц и столбцов таблиц.

1. **public** DataTable Take\_table\_people(**string** token)
2. {
3. SqlConnection.Open();
4. **if** (Select\_Online(token) != **null**)
5. {
6. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM people limit 50", SqlConnection);
7. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
8. DataTable dt = **new** DataTable("people");
9. addapter.Fill(dt);
10. SqlConnection.Close();
11. **return** dt;
12. }
13. **else**
14. {
15. SqlConnection.Close();
16. **return** **null**;
17. }
18. }
20. **public** DataTable Take\_table\_students(**string** token)
21. {
22. SqlConnection.Open();
23. **if** (Select\_Online(token) != **null**)
24. {
25. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM students limit 50", SqlConnection);
26. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
27. DataTable dt = **new** DataTable("students");
28. addapter.Fill(dt);
29. SqlConnection.Close();
30. **return** dt;
31. }
32. **else**
33. {
34. SqlConnection.Close();
35. **return** **null**;
36. }
37. }
39. **public** DataTable Take\_table\_teachers(**string** token)
40. {
41. SqlConnection.Open();
42. **if** (Select\_Online(token) != **null**)
43. {
44. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM teachers limit 50", SqlConnection);
45. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
46. DataTable dt = **new** DataTable("teachers");
47. addapter.Fill(dt);
48. SqlConnection.Close();
49. **return** dt;
50. }
51. **else**
52. {
53. SqlConnection.Close();
54. **return** **null**;
55. }
56. }
58. **public** DataTable Take\_table\_staff(**string** token)
59. {
60. SqlConnection.Open();
61. **if** (Select\_Online(token) != **null**)
62. {
63. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM staff limit 50", SqlConnection);
64. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
65. DataTable dt = **new** DataTable("staff");
66. addapter.Fill(dt);
67. SqlConnection.Close();
68. **return** dt;
69. }
70. **else**
71. {
72. SqlConnection.Close();
73. **return** **null**;
74. }
75. }
77. **public** DataTable Take\_table\_specials(**string** token)
78. {
79. SqlConnection.Open();
80. **if** (Select\_Online(token) != **null**)
81. {
82. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM specials limit 50", SqlConnection);
83. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
84. DataTable dt = **new** DataTable("specials");
85. addapter.Fill(dt);
86. SqlConnection.Close();
87. **return** dt;
88. }
89. **else**
90. {
91. SqlConnection.Close();
92. **return** **null**;
93. }
94. }
96. **public** DataTable Take\_table\_faculties(**string** token)
97. {
98. SqlConnection.Open();
99. **if** (Select\_Online(token) != **null**)
100. {
101. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM faculties limit 50", SqlConnection);
102. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
103. DataTable dt = **new** DataTable("faculties");
104. addapter.Fill(dt);
105. SqlConnection.Close();
106. **return** dt;
107. }
108. **else**
109. {
110. SqlConnection.Close();
111. **return** **null**;
112. }
113. }
115. **public** DataTable Take\_table\_salary(**string** token)
116. {
117. SqlConnection.Open();
118. **if** (Select\_Online(token) != **null**)
119. {
120. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM salary limit 50", SqlConnection);
121. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
122. DataTable dt = **new** DataTable("salary");
123. addapter.Fill(dt);
124. SqlConnection.Close();
125. **return** dt;
126. }
127. **else**
128. {
129. SqlConnection.Close();
130. **return** **null**;
131. }
132. }
134. **public** DataTable Take\_table\_address(**string** token)
135. {
136. SqlConnection.Open();
137. **if** (Select\_Online(token) != **null**)
138. {
139. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM address limit 50", SqlConnection);
140. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
141. DataTable dt = **new** DataTable("address");
142. addapter.Fill(dt);
143. SqlConnection.Close();
144. **return** dt;
145. }
146. **else**
147. {
148. SqlConnection.Close();
149. **return** **null**;
150. }
151. }
153. **public** DataTable Take\_table\_company(**string** token)
154. {
155. SqlConnection.Open();
156. **if** (Select\_Online(token) != **null**)
157. {
158. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM company limit 50", SqlConnection);
159. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
160. DataTable dt = **new** DataTable("company");
161. addapter.Fill(dt);
162. SqlConnection.Close();
163. **return** dt;
164. }
165. **else**
166. {
167. SqlConnection.Close();
168. **return** **null**;
169. }
170. }
172. **public** DataTable Take\_table\_card\_number(**string** token)
173. {
174. SqlConnection.Open();
175. **if** (Select\_Online(token) != **null**)
176. {
177. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM card\_number limit 50", SqlConnection);
178. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
179. DataTable dt = **new** DataTable("card\_number");
180. addapter.Fill(dt);
181. SqlConnection.Close();
182. **return** dt;
183. }
184. **else**
185. {
186. SqlConnection.Close();
187. **return** **null**;
188. }
189. }
191. //---Получить название строк
192. **public** DataTable Take\_rows\_people()
193. {
194. SqlConnection.Open();
195. MySqlCommand command = **new** MySqlCommand("DESCRIBE people", SqlConnection);
196. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
197. DataTable dt = **new** DataTable("people");
198. addapter.Fill(dt);
199. SqlConnection.Close();
200. **return** dt;
201. }
203. **public** DataTable Take\_rows\_students()
204. {
205. SqlConnection.Open();
206. MySqlCommand command = **new** MySqlCommand("DESCRIBE students", SqlConnection);
207. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
208. DataTable dt = **new** DataTable("students");
209. addapter.Fill(dt);
210. SqlConnection.Close();
211. **return** dt;
212. }
214. **public** DataTable Take\_rows\_teachers()
215. {
216. SqlConnection.Open();
217. MySqlCommand command = **new** MySqlCommand("DESCRIBE teachers", SqlConnection);
218. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
219. DataTable dt = **new** DataTable("teachers");
220. addapter.Fill(dt);
221. SqlConnection.Close();
222. **return** dt;
223. }
225. **public** DataTable Take\_rows\_staff()
226. {
227. SqlConnection.Open();
228. MySqlCommand command = **new** MySqlCommand("DESCRIBE staff", SqlConnection);
229. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
230. DataTable dt = **new** DataTable("staff");
231. addapter.Fill(dt);
232. SqlConnection.Close();
233. **return** dt;
234. }
236. **public** DataTable Take\_rows\_specials()
237. {
238. SqlConnection.Open();
239. MySqlCommand command = **new** MySqlCommand("DESCRIBE specials", SqlConnection);
240. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
241. DataTable dt = **new** DataTable("specials");
242. addapter.Fill(dt);
243. SqlConnection.Close();
244. **return** dt;
245. }
247. **public** DataTable Take\_rows\_faculties()
248. {
249. SqlConnection.Open();
250. MySqlCommand command = **new** MySqlCommand("DESCRIBE faculties", SqlConnection);
251. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
252. DataTable dt = **new** DataTable("faculties");
253. addapter.Fill(dt);
254. SqlConnection.Close();
255. **return** dt;
256. }
258. **public** DataTable Take\_rows\_salary()
259. {
260. SqlConnection.Open();
261. MySqlCommand command = **new** MySqlCommand("DESCRIBE salary", SqlConnection);
262. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
263. DataTable dt = **new** DataTable("salary");
264. addapter.Fill(dt);
265. SqlConnection.Close();
266. **return** dt;
267. }
269. **public** DataTable Take\_rows\_address()
270. {
271. SqlConnection.Open();
272. MySqlCommand command = **new** MySqlCommand("DESCRIBE address", SqlConnection);
273. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
274. DataTable dt = **new** DataTable("address");
275. addapter.Fill(dt);
276. SqlConnection.Close();
277. **return** dt;
278. }
280. **public** DataTable Take\_rows\_company()
281. {
282. SqlConnection.Open();
283. MySqlCommand command = **new** MySqlCommand("DESCRIBE company", SqlConnection);
284. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
285. DataTable dt = **new** DataTable("company");
286. addapter.Fill(dt);
287. SqlConnection.Close();
288. **return** dt;
289. }
291. **public** DataTable Take\_rows\_card\_number()
292. {
293. SqlConnection.Open();
294. MySqlCommand command = **new** MySqlCommand("DESCRIBE card\_number", SqlConnection);
295. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
296. DataTable dt = **new** DataTable("card\_number");
297. addapter.Fill(dt);
298. SqlConnection.Close();
299. **return** dt;
300. }

Добавлены функции изменения данных в таблицах.

1. **public** **char** Update\_table\_people(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
2. {
3. SqlConnection.Open();
4. **if** (Select\_Online(token) != **null**)
5. {
6. MySqlCommand command = **new** MySqlCommand("SELECT id FROM people WHERE id = @where", SqlConnection);
7. command.Parameters.AddWithValue("@where", where);
8. MySqlDataReader reader = command.ExecuteReader();
9. **if** (reader.Read())
10. {
11. reader.Close();
12. **char** q;
13. **switch** (name\_set)
14. {
15. **case** "id":
16. q = Update\_people\_id(**set**, where);
17. SqlConnection.Close();
18. **return** q;
19. **case** "password":
20. q = Update\_people\_password(**set**, where);
21. SqlConnection.Close();
22. **return** q;
23. **default**:
24. q = Update\_people\_fio(**set**, where);
25. SqlConnection.Close();
26. **return** q;
28. }
29. }
30. SqlConnection.Close();
31. **return** '0';
32. }
33. **else**
34. {
35. SqlConnection.Close();
36. **return** '0';
37. }
38. }
40. **public** **char** Update\_table\_students(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
41. {
42. SqlConnection.Open();
43. **if** (Select\_Online(token) != **null**)
44. {
45. MySqlCommand command = **new** MySqlCommand("SELECT id FROM students WHERE id = @where", SqlConnection);
46. command.Parameters.AddWithValue("@where", where);
47. MySqlDataReader reader = command.ExecuteReader();
48. **if** (reader.Read())
49. {
50. reader.Close();
51. **char** q;
52. **switch** (name\_set)
53. {
54. **case** "id":
55. q = Update\_students\_id(**set**, where);
56. SqlConnection.Close();
57. **return** q;
58. **case** "id\_special":
59. q = Update\_students\_id\_specials(**set**, where);
60. SqlConnection.Close();
61. **return** q;
62. **default**:
63. q = Update\_students\_id\_faculties(**set**, where);
64. SqlConnection.Close();
65. **return** q;
66. }
67. }
68. SqlConnection.Close();
69. **return** '0';
70. }
71. **else**
72. {
73. SqlConnection.Close();
74. **return** '0';
75. }
76. }
78. **public** **char** Update\_table\_teachers(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
79. {
80. SqlConnection.Open();
81. **if** (Select\_Online(token) != **null**)
82. {
83. MySqlCommand command = **new** MySqlCommand("SELECT id FROM teachers WHERE id = @where", SqlConnection);
84. command.Parameters.AddWithValue("@where", where);
85. MySqlDataReader reader = command.ExecuteReader();
86. **if** (reader.Read())
87. {
88. reader.Close();
89. **char** q;
90. **switch** (name\_set)
91. {
92. **case** "id":
93. q = Update\_teachers\_id(**set**, where);
94. SqlConnection.Close();
95. **return** q;
96. **case** "id\_specials":
97. q = Update\_teachers\_id\_specials(**set**, where);
98. SqlConnection.Close();
99. **return** q;
100. **default**:
101. q = Update\_teachers\_id\_faculties(**set**, where);
102. SqlConnection.Close();
103. **return** q;
104. }
105. }
106. SqlConnection.Close();
107. **return** '0';
108. }
109. **else**
110. {
111. SqlConnection.Close();
112. **return** '0';
113. }
114. }
116. **public** **char** Update\_table\_staff(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
117. {
118. SqlConnection.Open();
119. **if** (Select\_Online(token) != **null**)
120. {
121. MySqlCommand command = **new** MySqlCommand("SELECT id FROM staff WHERE id = @where", SqlConnection);
122. command.Parameters.AddWithValue("@where", where);
123. MySqlDataReader reader = command.ExecuteReader();
124. **if** (reader.Read())
125. {
126. reader.Close();
127. **char** q;
128. **switch** (name\_set)
129. {
130. **case** "id":
131. q = Update\_staff\_id(**set**, where);
132. SqlConnection.Close();
133. **return** q;
134. **default**:
135. q = Update\_staff\_job\_name(**set**, where);
136. SqlConnection.Close();
137. **return** q;
138. }
139. }
140. SqlConnection.Close();
141. **return** '0';
142. }
143. **else**
144. {
145. SqlConnection.Close();
146. **return** '0';
147. }
148. }
150. **public** **char** Update\_table\_specials(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
151. {
152. SqlConnection.Open();
153. **if** (Select\_Online(token) != **null**)
154. {
155. MySqlCommand command = **new** MySqlCommand("SELECT id FROM specials WHERE id = @where", SqlConnection);
156. command.Parameters.AddWithValue("@where", where);
157. MySqlDataReader reader = command.ExecuteReader();
158. **if** (reader.Read())
159. {
160. reader.Close();
161. **char** q;
162. **switch** (name\_set)
163. {
164. **case** "id":
165. q = Update\_specials\_id(**set**, where);
166. SqlConnection.Close();
167. **return** q;
168. **case** "special\_name":
169. q = Update\_specials\_special\_name(**set**, where);
170. SqlConnection.Close();
171. **return** q;
172. **default**:
173. q = Update\_specials\_special\_director(**set**, where);
174. SqlConnection.Close();
175. **return** q;
176. }
177. }
178. SqlConnection.Close();
179. **return** '0';
180. }
181. **else**
182. {
183. SqlConnection.Close();
184. **return** '0';
185. }
186. }
188. **public** **char** Update\_table\_faculties(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
189. {
190. SqlConnection.Open();
191. **if** (Select\_Online(token) != **null**)
192. {
193. MySqlCommand command = **new** MySqlCommand("SELECT id FROM faculties WHERE id = @where", SqlConnection);
194. command.Parameters.AddWithValue("@where", where);
195. MySqlDataReader reader = command.ExecuteReader();
196. **if** (reader.Read())
197. {
198. reader.Close();
199. **char** q;
200. **switch** (name\_set)
201. {
202. **case** "id":
203. q = Update\_faculties\_id(**set**, where);
204. SqlConnection.Close();
205. **return** q;
206. **case** "facilty\_name":
207. q = Update\_faculties\_faculty\_name(**set**, where);
208. SqlConnection.Close();
209. **return** q;
210. **case** "faculty\_director":
211. q = Update\_faculties\_faculty\_director(**set**, where);
212. SqlConnection.Close();
213. **return** q;
214. **default**:
215. q = Update\_faculties\_faculty\_sub\_director(**set**, where);
216. SqlConnection.Close();
217. **return** q;
218. }
219. }
220. SqlConnection.Close();
221. **return** '0';
222. }
223. **else**
224. {
225. SqlConnection.Close();
226. **return** '0';
227. }
228. }
230. **public** **char** Update\_table\_salary(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
231. {
232. SqlConnection.Open();
233. **if** (Select\_Online(token) != **null**)
234. {
235. MySqlCommand command = **new** MySqlCommand("SELECT id FROM salary WHERE id = @where", SqlConnection);
236. command.Parameters.AddWithValue("@where", where);
237. MySqlDataReader reader = command.ExecuteReader();
238. **if** (reader.Read())
239. {
240. reader.Close();
241. **char** q;
242. **switch** (name\_set)
243. {
244. **case** "id":
245. q = Update\_salary\_id(**set**, where);
246. SqlConnection.Close();
247. **return** q;
248. **default**:
249. q = Update\_salary\_salary(**set**, where);
250. SqlConnection.Close();
251. **return** q;
252. }
253. }
254. SqlConnection.Close();
255. **return** '0';
256. }
257. **else**
258. {
259. SqlConnection.Close();
260. **return** '0';
261. }
262. }
264. **public** **char** Update\_table\_address(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
265. {
266. SqlConnection.Open();
267. **if** (Select\_Online(token) != **null**)
268. {
269. MySqlCommand command = **new** MySqlCommand("SELECT id FROM address WHERE id = @where", SqlConnection);
270. command.Parameters.AddWithValue("@where", where);
271. MySqlDataReader reader = command.ExecuteReader();
272. **if** (reader.Read())
273. {
274. reader.Close();
275. **char** q;
276. **switch** (name\_set)
277. {
278. **case** "id":
279. q = Update\_address\_id(**set**, where);
280. SqlConnection.Close();
281. **return** q;
282. **case** "address":
283. q = Update\_address\_address(**set**, where);
284. SqlConnection.Close();
285. **return** q;
286. **case** "email":
287. q = Update\_address\_email(**set**, where);
288. SqlConnection.Close();
289. **return** q;
290. **default**:
291. q = Update\_address\_registration(**set**, where);
292. SqlConnection.Close();
293. **return** q;
294. }
295. }
296. SqlConnection.Close();
297. **return** '0';
298. }
299. **else**
300. {
301. SqlConnection.Close();
302. **return** '0';
303. }
304. }
305. **public** **char** Update\_table\_company(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
306. {
307. SqlConnection.Open();
308. **if** (Select\_Online(token) != **null**)
309. {
310. MySqlCommand command = **new** MySqlCommand("SELECT id FROM company WHERE id = @where", SqlConnection);
311. command.Parameters.AddWithValue("@where", where);
312. MySqlDataReader reader = command.ExecuteReader();
313. **if** (reader.Read())
314. {
315. reader.Close();
316. **char** q;
317. **switch** (name\_set)
318. {
319. **case** "id":
320. q = Update\_company\_id(**set**, where);
321. SqlConnection.Close();
322. **return** q;
323. **case** "b\_date":
324. q = Update\_company\_b\_date(**set**, where);
325. SqlConnection.Close();
326. **return** q;
327. **case** "about":
328. q = Update\_company\_about(**set**, where);
329. SqlConnection.Close();
330. **return** q;
331. **case** "phoneNumber":
332. q = Update\_company\_phoneNumber(**set**, where);
333. SqlConnection.Close();
334. **return** q;
335. **default**:
336. q = Update\_company\_company(**set**, where);
337. SqlConnection.Close();
338. **return** q;
339. }
340. }
341. SqlConnection.Close();
342. **return** '0';
343. }
344. **else**
345. {
346. SqlConnection.Close();
347. **return** '0';
348. }
349. }
350. **public** **char** Update\_table\_card\_number(**string** table, **string** name\_set, **string** **set**, **string** where, **string** token)
351. {
352. SqlConnection.Open();
353. **if** (Select\_Online(token) != **null**)
354. {
355. MySqlCommand command = **new** MySqlCommand("SELECT id FROM card\_number WHERE id = @where", SqlConnection);
356. command.Parameters.AddWithValue("@where", where);
357. MySqlDataReader reader = command.ExecuteReader();
358. **if** (reader.Read())
359. {
360. reader.Close();
361. **char** q;
362. **switch** (name\_set)
363. {
364. **case** "id":
365. q = Update\_card\_number\_id(**set**, where);
366. SqlConnection.Close();
367. **return** q;
368. **default**:
369. q = Update\_card\_number\_card\_number(**set**, where);
370. SqlConnection.Close();
371. **return** q;
372. }
373. }
374. SqlConnection.Close();
375. **return** '0';
376. }
377. **else**
378. {
379. SqlConnection.Close();
380. **return** '0';
381. }
382. }
384. **public** **string** Delete\_table(**string** token)
385. {
386. SqlConnection.Open();
387. **string** id = Select\_Online(token);
388. **if** (id != **null**)
389. {
390. MySqlCommand command = **new** MySqlCommand("SELECT id FROM people WHERE id = @id", SqlConnection);
391. command.Parameters.AddWithValue("@id", id);
392. MySqlDataReader reader = command.ExecuteReader();
393. **if** (reader.Read())
394. {
395. reader.Close();
396. command = **new** MySqlCommand("DELETE FROM people WHERE id = @id", SqlConnection);
397. command.Parameters.AddWithValue("@id", id);
398. command.ExecuteNonQuery();
399. Delete\_Online(token);
400. **return** "1";
401. }
402. **else**
403. {
404. **return** "-1";
405. }
406. }
407. **else**
408. {
409. **return** **null**;
410. }
411. }
412. **private** **char** Update\_people\_id (**string** **set**, **string** where)
413. {
414. **using** (SHA1 shaM = **new** SHA1Managed())
415. {
416. **byte**[] hash2 = shaM.ComputeHash(Encoding.UTF8.GetBytes(**set**));
417. **set** = BitConverter.ToString(hash2).Replace("-", "").ToLower();
418. }
419. MySqlCommand command = **new** MySqlCommand("UPDATE people SET id  = @set  WHERE id = @where", SqlConnection);
420. command.Parameters.AddWithValue("@set", **set**);
421. command.Parameters.AddWithValue("@where", where);
422. command.ExecuteNonQuery();
423. **return** '1';
424. }
426. **private** **char** Update\_students\_id(**string** **set**, **string** where)
427. {
428. MySqlCommand command = **new** MySqlCommand("UPDATE students SET id  = @set  WHERE id = @where", SqlConnection);
429. command.Parameters.AddWithValue("@set", **set**);
430. command.Parameters.AddWithValue("@where", where);
431. command.ExecuteNonQuery();
432. **return** '1';
433. }
435. **private** **char** Update\_students\_id\_specials(**string** **set**, **string** where)
436. {
437. MySqlCommand command = **new** MySqlCommand("UPDATE students SET id\_special  = @set  WHERE id = @where", SqlConnection);
438. command.Parameters.AddWithValue("@set", **set**);
439. command.Parameters.AddWithValue("@where", where);
440. command.ExecuteNonQuery();
441. **return** '1';
442. }
444. **private** **char** Update\_students\_id\_faculties(**string** **set**, **string** where)
445. {
446. MySqlCommand command = **new** MySqlCommand("UPDATE students SET id\_faculty = @set  WHERE id = @where", SqlConnection);
447. command.Parameters.AddWithValue("@set", **set**);
448. command.Parameters.AddWithValue("@where", where);
449. command.ExecuteNonQuery();
450. **return** '1';
451. }
453. **private** **char** Update\_teachers\_id(**string** **set**, **string** where)
454. {
455. MySqlCommand command = **new** MySqlCommand("UPDATE teachers SET id  = @set  WHERE id = @where", SqlConnection);
456. command.Parameters.AddWithValue("@set", **set**);
457. command.Parameters.AddWithValue("@where", where);
458. command.ExecuteNonQuery();
459. **return** '1';
460. }
462. **private** **char** Update\_teachers\_id\_specials(**string** **set**, **string** where)
463. {
464. MySqlCommand command = **new** MySqlCommand("UPDATE teachers SET id\_special  = @set  WHERE id = @where", SqlConnection);
465. command.Parameters.AddWithValue("@set", **set**);
466. command.Parameters.AddWithValue("@where", where);
467. command.ExecuteNonQuery();
468. **return** '1';
469. }
471. **private** **char** Update\_teachers\_id\_faculties(**string** **set**, **string** where)
472. {
473. MySqlCommand command = **new** MySqlCommand("UPDATE teachers SET id\_faculty = @set  WHERE id = @where", SqlConnection);
474. command.Parameters.AddWithValue("@set", **set**);
475. command.Parameters.AddWithValue("@where", where);
476. command.ExecuteNonQuery();
477. **return** '1';
478. }
480. **private** **char** Update\_staff\_id(**string** **set**, **string** where)
481. {
482. MySqlCommand command = **new** MySqlCommand("UPDATE staff SET id = @set  WHERE id = @where", SqlConnection);
483. command.Parameters.AddWithValue("@set", **set**);
484. command.Parameters.AddWithValue("@where", where);
485. command.ExecuteNonQuery();
486. **return** '1';
487. }
489. **private** **char** Update\_specials\_id(**string** **set**, **string** where)
490. {
491. MySqlCommand command = **new** MySqlCommand("UPDATE specials SET id = @set  WHERE id = @where", SqlConnection);
492. command.Parameters.AddWithValue("@set", **set**);
493. command.Parameters.AddWithValue("@where", where);
494. command.ExecuteNonQuery();
495. **return** '1';
496. }
498. **private** **char** Update\_faculties\_id(**string** **set**, **string** where)
499. {
500. MySqlCommand command = **new** MySqlCommand("UPDATE faculties SET id = @set  WHERE id = @where", SqlConnection);
501. command.Parameters.AddWithValue("@set", **set**);
502. command.Parameters.AddWithValue("@where", where);
503. command.ExecuteNonQuery();
504. **return** '1';
505. }
507. **private** **char** Update\_salary\_id(**string** **set**, **string** where)
508. {
509. MySqlCommand command = **new** MySqlCommand("UPDATE salary SET id = @set  WHERE id = @where", SqlConnection);
510. command.Parameters.AddWithValue("@set", **set**);
511. command.Parameters.AddWithValue("@where", where);
512. command.ExecuteNonQuery();
513. **return** '1';
514. }
516. **private** **char** Update\_address\_id(**string** **set**, **string** where)
517. {
518. MySqlCommand command = **new** MySqlCommand("UPDATE address SET id = @set  WHERE id = @where", SqlConnection);
519. command.Parameters.AddWithValue("@set", **set**);
520. command.Parameters.AddWithValue("@where", where);
521. command.ExecuteNonQuery();
522. **return** '1';
523. }
525. **private** **char** Update\_company\_id(**string** **set**, **string** where)
526. {
527. MySqlCommand command = **new** MySqlCommand("UPDATE company SET id = @set  WHERE id = @where", SqlConnection);
528. command.Parameters.AddWithValue("@set", **set**);
529. command.Parameters.AddWithValue("@where", where);
530. command.ExecuteNonQuery();
531. **return** '1';
532. }
534. **private** **char** Update\_card\_number\_id(**string** **set**, **string** where)
535. {
536. MySqlCommand command = **new** MySqlCommand("UPDATE card\_number SET id = @set  WHERE id = @where", SqlConnection);
537. command.Parameters.AddWithValue("@set", **set**);
538. command.Parameters.AddWithValue("@where", where);
539. command.ExecuteNonQuery();
540. **return** '1';
541. }
543. **private** **char** Update\_people\_password(**string** **set**, **string** where)
544. {
545. MySqlCommand command = **new** MySqlCommand("UPDATE people SET password = MD5(@set) WHERE id = @where", SqlConnection);
546. command.Parameters.AddWithValue("@set", **set**);
547. command.Parameters.AddWithValue("@where", where);
548. command.ExecuteNonQuery();
549. **return** '1';
550. }
552. **private** **char** Update\_people\_fio(**string** **set**, **string** where)
553. {
554. MySqlCommand command = **new** MySqlCommand("UPDATE people SET fio = @set  WHERE id = @where", SqlConnection);
555. command.Parameters.AddWithValue("@set", **set**);
556. command.Parameters.AddWithValue("@where", where);
557. command.ExecuteNonQuery();
558. **return** '1';
559. }
561. **private** **char** Update\_staff\_job\_name(**string** **set**, **string** where)
562. {
563. MySqlCommand command = **new** MySqlCommand("UPDATE staff SET job\_name = @set  WHERE id = @where", SqlConnection);
564. command.Parameters.AddWithValue("@set", **set**);
565. command.Parameters.AddWithValue("@where", where);
566. command.ExecuteNonQuery();
567. **return** '1';
568. }
570. **private** **char** Update\_specials\_special\_name(**string** **set**, **string** where)
571. {
572. MySqlCommand command = **new** MySqlCommand("UPDATE specials SET special\_name = @set  WHERE id = @where", SqlConnection);
573. command.Parameters.AddWithValue("@set", **set**);
574. command.Parameters.AddWithValue("@where", where);
575. command.ExecuteNonQuery();
576. **return** '1';
577. }
579. **private** **char** Update\_specials\_special\_director(**string** **set**, **string** where)
580. {
581. MySqlCommand command = **new** MySqlCommand("UPDATE specials SET special\_director = @set  WHERE id = @where", SqlConnection);
582. command.Parameters.AddWithValue("@set", **set**);
583. command.Parameters.AddWithValue("@where", where);
584. command.ExecuteNonQuery();
585. **return** '1';
586. }
588. **private** **char** Update\_faculties\_faculty\_name(**string** **set**, **string** where)
589. {
590. MySqlCommand command = **new** MySqlCommand("UPDATE faculties SET faculty\_name = @set  WHERE id = @where", SqlConnection);
591. command.Parameters.AddWithValue("@set", **set**);
592. command.Parameters.AddWithValue("@where", where);
593. command.ExecuteNonQuery();
594. **return** '1';
595. }
597. **private** **char** Update\_faculties\_faculty\_director(**string** **set**, **string** where)
598. {
599. MySqlCommand command = **new** MySqlCommand("UPDATE faculties SET faculty\_director = @set  WHERE id = @where", SqlConnection);
600. command.Parameters.AddWithValue("@set", **set**);
601. command.Parameters.AddWithValue("@where", where);
602. command.ExecuteNonQuery();
603. **return** '1';
604. }
606. **private** **char** Update\_faculties\_faculty\_sub\_director(**string** **set**, **string** where)
607. {
608. MySqlCommand command = **new** MySqlCommand("UPDATE faculties SET faculty\_sub\_director = @set  WHERE id = @where", SqlConnection);
609. command.Parameters.AddWithValue("@set", **set**);
610. command.Parameters.AddWithValue("@where", where);
611. command.ExecuteNonQuery();
612. **return** '1';
613. }
615. **private** **char** Update\_salary\_salary(**string** **set**, **string** where)
616. {
617. MySqlCommand command = **new** MySqlCommand("UPDATE salary SET salary = @set  WHERE id = @where", SqlConnection);
618. command.Parameters.AddWithValue("@set", **set**);
619. command.Parameters.AddWithValue("@where", where);
620. command.ExecuteNonQuery();
621. **return** '1';
622. }
624. **private** **char** Update\_address\_address(**string** **set**, **string** where)
625. {
626. MySqlCommand command = **new** MySqlCommand("UPDATE address SET address = @set  WHERE id = @where", SqlConnection);
627. command.Parameters.AddWithValue("@set", **set**);
628. command.Parameters.AddWithValue("@where", where);
629. command.ExecuteNonQuery();
630. **return** '1';
631. }
633. **private** **char** Update\_address\_email(**string** **set**, **string** where)
634. {
635. MySqlCommand command = **new** MySqlCommand("UPDATE address SET email = @set  WHERE id = @where", SqlConnection);
636. command.Parameters.AddWithValue("@set", **set**);
637. command.Parameters.AddWithValue("@where", where);
638. command.ExecuteNonQuery();
639. **return** '1';
640. }
642. **private** **char** Update\_address\_registration(**string** **set**, **string** where)
643. {
644. MySqlCommand command = **new** MySqlCommand("UPDATE address SET registration = @set  WHERE id = @where", SqlConnection);
645. command.Parameters.AddWithValue("@set", **set**);
646. command.Parameters.AddWithValue("@where", where);
647. command.ExecuteNonQuery();
648. **return** '1';
649. }
651. **private** **char** Update\_company\_b\_date(**string** **set**, **string** where)
652. {
653. MySqlCommand command = **new** MySqlCommand("UPDATE company SET b\_date = @set  WHERE id = @where", SqlConnection);
654. command.Parameters.AddWithValue("@set", **set**);
655. command.Parameters.AddWithValue("@where", where);
656. command.ExecuteNonQuery();
657. **return** '1';
658. }
660. **private** **char** Update\_company\_about(**string** **set**, **string** where)
661. {
662. MySqlCommand command = **new** MySqlCommand("UPDATE company SET about = @set  WHERE id = @where", SqlConnection);
663. command.Parameters.AddWithValue("@set", **set**);
664. command.Parameters.AddWithValue("@where", where);
665. command.ExecuteNonQuery();
666. **return** '1';
667. }
669. **private** **char** Update\_company\_phoneNumber(**string** **set**, **string** where)
670. {
671. MySqlCommand command = **new** MySqlCommand("UPDATE company SET phoneNumber = @set  WHERE id = @where", SqlConnection);
672. command.Parameters.AddWithValue("@set", **set**);
673. command.Parameters.AddWithValue("@where", where);
674. command.ExecuteNonQuery();
675. **return** '1';
676. }
678. **private** **char** Update\_company\_company(**string** **set**, **string** where)
679. {
680. MySqlCommand command = **new** MySqlCommand("UPDATE company SET company = @set  WHERE id = @where", SqlConnection);
681. command.Parameters.AddWithValue("@set", **set**);
682. command.Parameters.AddWithValue("@where", where);
683. command.ExecuteNonQuery();
684. **return** '1';
685. }
687. **private** **char** Update\_card\_number\_card\_number(**string** **set**, **string** where)
688. {
689. MySqlCommand command = **new** MySqlCommand("UPDATE card\_number SET card\_number = @set  WHERE id = @where", SqlConnection);
690. command.Parameters.AddWithValue("@set", **set**);
691. command.Parameters.AddWithValue("@where", where);
692. command.ExecuteNonQuery();
693. **return** '1';
694. }

Добавлена функция получения всей зарплаты.

1. **public** **string** Top\_Sal(**string** token)
2. {
3. SqlConnection.Open();
4. **if** (token != **null**)
5. {
6. MySqlCommand command = **new** MySqlCommand("SELECT salary FROM top\_sal WHERE id = 1", SqlConnection);
7. MySqlDataReader reader = command.ExecuteReader();
8. **if** (reader.Read())
9. {
10. **string** kek = Convert.ToString(reader["salary"]);
11. SqlConnection.Close();
12. reader.Close();
13. **return** kek;
14. }
15. SqlConnection.Close();
16. reader.Close();
17. **return** **null**;
18. }
19. **else**
20. {
21. SqlConnection.Close();
22. **return** **null**;
23. }
24. }

Добавлены функции создания и обработка сессии.

1. **private** **char** Insert\_Online(**string** id,**string** token)
2. {
3. **if** (Select\_Online(token) == **null**)
4. {
5. **try**
6. {
7. MySqlCommand command = **new** MySqlCommand("INSERT INTO online VALUE(@id,@token)", SqlConnection);
8. command.Parameters.AddWithValue("@id", id);
9. command.Parameters.AddWithValue("@token", token);
10. command.ExecuteNonQuery();
11. **return** '1';
12. }
13. **catch** (Exception)
14. {
15. **return** '0';
16. }
17. }
18. **else**
19. **return** '0';
20. }
22. **public** **void** Delete\_Online(**string** token)
23. {
24. SqlConnection.Open();
25. **string** id = Select\_Online(token);
26. **if** (id != **null**)
27. {
28. MySqlCommand command = **new** MySqlCommand("DELETE FROM online where id = @id", SqlConnection);
29. command.Parameters.AddWithValue("@id", id);
30. command.ExecuteNonQuery();
31. SqlConnection.Close();
32. }
33. }
35. **private** **string** Select\_Online(**string** token)
36. {
37. MySqlCommand command = **new** MySqlCommand("SELECT id FROM online WHERE token = @token", SqlConnection);
38. command.Parameters.AddWithValue("@token", token);
39. MySqlDataReader reader = command.ExecuteReader();
40. **if** (reader.Read())
41. {
42. **string** read = reader["id"].ToString();
43. reader.Close();
44. **return** read;
45. }
46. **else**
47. {
48. reader.Close();
49. **return** **null**;
50. }
51. }

Добавлена функция чтения таблиц.

1. **public** DataTable Plus\_Mun50(**string** table, **int** offset, **string** token)
2. {
3. SqlConnection.Open();
4. **if** (Select\_Online(token) != **null**)
5. {
6. **switch** (table)
7. {
8. **case** "people":
9. **return** Roll\_people(offset);
10. **case** "students":
11. **return** Roll\_students(offset);
12. **case** "teachers":
13. **return** Roll\_teachers(offset);
14. **case** "staff":
15. **return** Roll\_staff(offset);
16. **case** "specials":
17. **return** Roll\_specials(offset);
18. **case** "faculties":
19. **return** Roll\_faculties(offset);
20. **case** "salary":
21. **return** Roll\_salary(offset);
22. **case** "address":
23. **return** Roll\_address(offset);
24. **case** "company":
25. **return** Roll\_company(offset);
26. **case** "card\_number":
27. **return** Roll\_card\_number(offset);
28. **default**:
29. **return** **null**;
30. }
31. }
32. **else**
33. {
34. SqlConnection.Close();
35. **return** **null**;
36. }
37. }
38. **private** DataTable Roll\_people(**int** offset)
39. {
40. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM people limit 50 offset @offset", SqlConnection);
41. command.Parameters.AddWithValue("@offset", offset);
42. command.ExecuteNonQuery();
43. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
44. DataTable dt = **new** DataTable("people");
45. addapter.Fill(dt);
46. SqlConnection.Close();
47. **return** dt;
48. }
50. **private** DataTable Roll\_students(**int** offset)
51. {
52. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM students limit 50 offset @offset", SqlConnection);
53. command.Parameters.AddWithValue("@offset", offset);
54. command.ExecuteNonQuery();
55. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
56. DataTable dt = **new** DataTable("students");
57. addapter.Fill(dt);
58. SqlConnection.Close();
59. **return** dt;
60. }
62. **private** DataTable Roll\_teachers(**int** offset)
63. {
64. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM teachers limit 50 offset @offset", SqlConnection);
65. command.Parameters.AddWithValue("@offset", offset);
66. command.ExecuteNonQuery();
67. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
68. DataTable dt = **new** DataTable("teachers");
69. addapter.Fill(dt);
70. SqlConnection.Close();
71. **return** dt;
72. }
74. **private** DataTable Roll\_staff(**int** offset)
75. {
76. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM staff limit 50 offset @offset", SqlConnection);
77. command.Parameters.AddWithValue("@offset", offset);
78. command.ExecuteNonQuery();
79. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
80. DataTable dt = **new** DataTable("staff");
81. addapter.Fill(dt);
82. SqlConnection.Close();
83. **return** dt;
84. }
86. **private** DataTable Roll\_specials(**int** offset)
87. {
88. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM specials limit 50 offset @offset", SqlConnection);
89. command.Parameters.AddWithValue("@offset", offset);
90. command.ExecuteNonQuery();
91. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
92. DataTable dt = **new** DataTable("specials");
93. addapter.Fill(dt);
94. SqlConnection.Close();
95. **return** dt;
96. }
98. **private** DataTable Roll\_faculties(**int** offset)
99. {
100. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM faculties limit 50 offset @offset", SqlConnection);
101. command.Parameters.AddWithValue("@offset", offset);
102. command.ExecuteNonQuery();
103. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
104. DataTable dt = **new** DataTable("faculties");
105. addapter.Fill(dt);
106. SqlConnection.Close();
107. **return** dt;
108. }
110. **private** DataTable Roll\_salary(**int** offset)
111. {
112. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM salary limit 50 offset @offset", SqlConnection);
113. command.Parameters.AddWithValue("@offset", offset);
114. command.ExecuteNonQuery();
115. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
116. DataTable dt = **new** DataTable("salary");
117. addapter.Fill(dt);
118. SqlConnection.Close();
119. **return** dt;
120. }
122. **private** DataTable Roll\_address(**int** offset)
123. {
124. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM address limit 50 offset @offset", SqlConnection);
125. command.Parameters.AddWithValue("@offset", offset);
126. command.ExecuteNonQuery();
127. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
128. DataTable dt = **new** DataTable("address");
129. addapter.Fill(dt);
130. SqlConnection.Close();
131. **return** dt;
132. }
134. **private** DataTable Roll\_company(**int** offset)
135. {
136. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM company limit 50 offset @offset", SqlConnection);
137. command.Parameters.AddWithValue("@offset", offset);
138. command.ExecuteNonQuery();
139. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
140. DataTable dt = **new** DataTable("company");
141. addapter.Fill(dt);
142. SqlConnection.Close();
143. **return** dt;
144. }
146. **private** DataTable Roll\_card\_number(**int** offset)
147. {
148. MySqlCommand command = **new** MySqlCommand("SELECT \* FROM card\_number limit 50 offset @offset", SqlConnection);
149. command.Parameters.AddWithValue("@offset", offset);
150. command.ExecuteNonQuery();
151. MySqlDataAdapter addapter = **new** MySqlDataAdapter(command);
152. DataTable dt = **new** DataTable("card\_number");
153. addapter.Fill(dt);
154. SqlConnection.Close();
155. **return** dt;
156. }

**Доказательство**

3.1 Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

3.2 Вторая нормальная форма

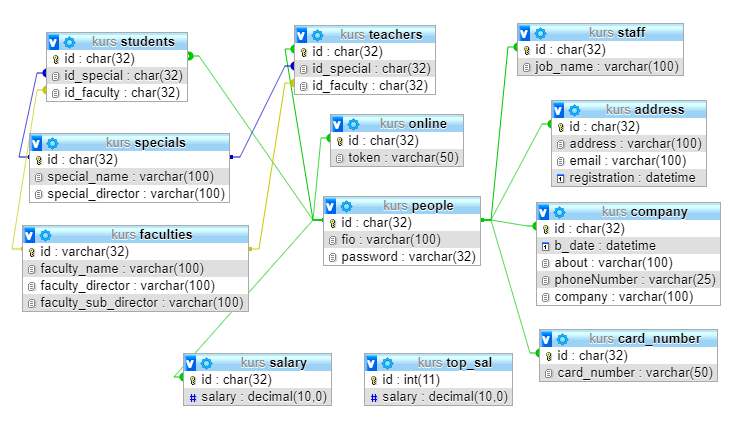
Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК).

3.3 Третья нормальная форма

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

**Приложения**

1. ER-диаграмма



1. Исходные коды и документы:

<https://github.com/T0person/Bdsbd_2019>

**Вывод**

Во время выполнения курсового проекта были изучены методы работы с базами данных, способы управления. Таким образом, в ходе работы была разработана база данных, а также API, для автоматизации работы института.