



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра инструментального и прикладного программного обеспечения
(ИиППО)

ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №1 - 4
по дисциплине «Разработка баз данных»

Студент группы *ИКБО 33-22, Шило Юрий Сергеевич*

(подпись)

Преподаватель *Баев Игорь Борисович*

(подпись)

Отчет
представлен

«__» _____ 202__ г.

Москва 2024 г.

СОЗДАНИЕ БАЗЫ ДАННЫХ И ТАБЛИЦ В НЕЙ

Построим физическую модель нашей базы данных. Для этого мы будем использовать инструмент dbForge Studio.

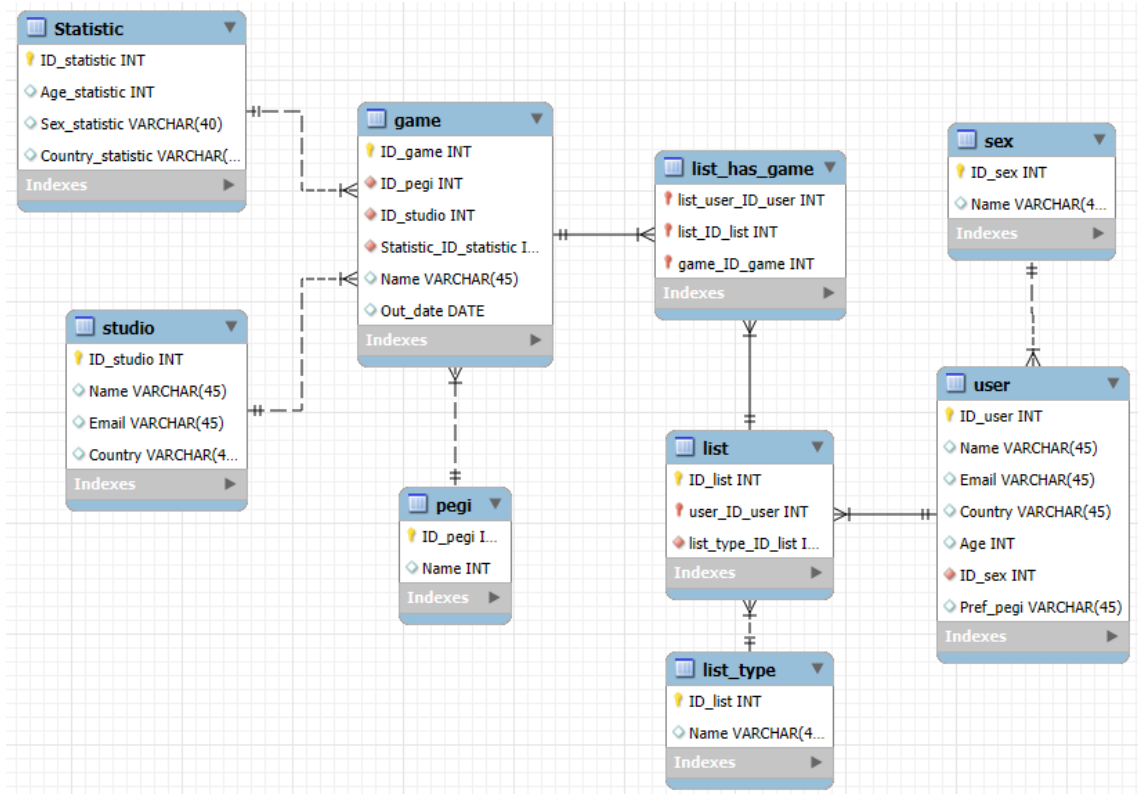


Рисунок 1 – Физическая модель реализуемой базы данных

Нам надо создать базу данных, которую назовем GameCalendar. Для этого в SQL существует оператор `create database`. Перед началом работы с базой данных надо указать серверу в какую именно БД мы создаем таблицы, т.е. надо выбрать БД для работы. Для этого используется оператор `use`.

```
3 CREATE DATABASE GameCalendar;  
4 USE GameCalendar;
```

Рисунок 2 – Процесс создания и выбора базы данных

Для создания таблиц в SQL существует оператор `create table`. Создадим таблицу `Statistic`.

```
6 -- Table for Statistic  
7 CREATE TABLE Statistic (  
8     ID_statistic INT PRIMARY KEY,  
9     Age_statistic INT,  
10    Sex_statistic VARCHAR(64),  
11    Country_statistic VARCHAR(64)  
12 );
```

Рисунок 3 – Процесс создания таблицы «Statistic»

Создадим таблицу `Studio`.

```

14  -- Table for Studio
15  CREATE TABLE studio (
16      ID_studio INT PRIMARY KEY,
17      Name VARCHAR(64),
18      Email VARCHAR(64),
19      Country VARCHAR(64)
20  );

```

Рисунок 4 – Процесс создания таблицы «Studio»

Создадим таблицу PEGI.

```

22  -- Table for PEGI
23  CREATE TABLE pegi (
24      ID_peg_i INT PRIMARY KEY,
25      Name VARCHAR(64)
26  );

```

Рисунок 5 – Процесс создания таблицы «PEGI»

Создадим таблицу Sex.

```

28  -- Table for Sex
29  CREATE TABLE sex (
30      ID_sex INT PRIMARY KEY,
31      Name VARCHAR(64)
32  );

```

Рисунок 6 – Процесс создания таблицы «Sex»

Создадим таблицу Game.

```

34  -- Table for Game
35  CREATE TABLE game (
36      ID_game INT PRIMARY KEY,
37      ID_peg_i INT,
38      ID_studio INT,
39      Statistic_ID_statistic INT,
40      Name VARCHAR(64),
41      Out_date DATE,
42      FOREIGN KEY (ID_peg_i) REFERENCES pegi(ID_peg_i),
43      FOREIGN KEY (ID_studio) REFERENCES studio(ID_studio),
44      FOREIGN KEY (Statistic_ID_statistic) REFERENCES Statistic(ID_statistic)
45  );

```

Рисунок 7 – Процесс создания таблицы «Game»

Создадим таблицу List Type.

```

47  -- Table for List Type
48  CREATE TABLE list_type (
49      ID_list_type INT PRIMARY KEY,
50      Name INT
51  );

```

Рисунок 8 – Процесс создания таблицы «List Type»

Создадим таблицу User.

```

53  -- Table for User
54  CREATE TABLE user (
55      ID_user INT PRIMARY KEY,
56      ID_sex INT,
57      Name VARCHAR(64),
58      Email VARCHAR(64),
59      Country VARCHAR(64),
60      Age INT,
61      Pref_peg_i VARCHAR(64),
62      FOREIGN KEY (ID_sex) REFERENCES sex(ID_sex)
63  );

```

Рисунок 9 – Процесс создания таблицы «User»

Создадим таблицу List.

```

65 -- Table for List
66 CREATE TABLE list (
67     ID_list INT PRIMARY KEY,
68     user_ID_user INT,
69     list_type_ID_list_type INT,
70     FOREIGN KEY (user_ID_user) REFERENCES user(ID_user),
71     FOREIGN KEY (list_type_ID_list_type) REFERENCES list_type(ID_list_type)
72 );

```

Рисунок 10 – Процесс создания таблицы «List»

Создадим таблицу List Has Game.

```

74 -- Table for List Has Game (association between List and Game)
75 CREATE TABLE list_has_game (
76     list_user_ID_user INT,
77     list_ID_list INT,
78     game_ID_game INT,
79     PRIMARY KEY (list_user_ID_user, list_ID_list, game_ID_game),
80     FOREIGN KEY (list_user_ID_user) REFERENCES user(ID_user),
81     FOREIGN KEY (list_ID_list) REFERENCES list(ID_list),
82     FOREIGN KEY (game_ID_game) REFERENCES game(ID_game)
83 );

```

Рисунок 11 – Процесс создания таблицы «List Has Game»

В SQL существует возможность посмотреть какие БД у нас существуют, какие таблицы в них присутствуют, и какие столбцы эти таблицы содержат.

show databases — показать все имеющиеся БД.

	Database
1	gamecalendar
2	information_schema
3	mysql
4	performance_schema
5	sys

Рисунок 12 – Описание всех имеющиеся БД

show tables — показать список таблиц текущей БД (предварительно ее надо выбрать с помощью оператора use).

	Tables_in_gamecalendar
1	game
2	list
3	list_has_game
4	list_type
5	pegi
6	sex
7	statistic
8	studio
9	user

Рисунок 13 – Список таблиц БД «GameCalendar»

describe имя_таблицы — показать описание столбцов указанной таблицы. Просмотрим таблицу statistic.

	Field	Type	Null	Key	Default	Extra
1	ID_statistic	int	NO	PRI	<null>	
2	Age_statistic	int	YES		<null>	
3	Sex_statistic	varchar(64)	YES		<null>	
4	Country_statistic	varchar(64)	YES		<null>	

Рисунок 14 – Описание столбцов в таблице «statistic»

Просмотрим таблицу studio.

	Field	Type	Null	Key	Default	Extra
1	ID_studio	int	NO	PRI	<null>	
2	Name	varchar(64)	YES		<null>	
3	Email	varchar(64)	YES		<null>	
4	Country	varchar(64)	YES		<null>	

Рисунок 15 – Описание столбцов в таблице «studio»

Просмотрим таблицу pegi.

	Field	Type	Null	Key	Default	Extra
1	ID_pegi	int	NO	PRI	<null>	
2	Name	varchar(64)	YES		<null>	

Рисунок 16 – Описание столбцов в таблице «pegi»

Просмотрим таблицу sex.

	Field	Type	Null	Key	Default	Extra
1	ID_sex	int	NO	PRI	<null>	
2	Name	varchar(64)	YES		<null>	

Рисунок 17 – Описание столбцов в таблице «sex»

Просмотрим таблицу game.

	Field ▾		Reload Page Ctrl+F5	Null ▾	Key ▾	Default ▾	Extra ▾
1	ID_game	int	NO	PRI	<null>		
2	ID_pegi	int	YES	MUL	<null>		
3	ID_studio	int	YES	MUL	<null>		
4	Statistic_ID_statistic	int	YES	MUL	<null>		
5	Name	varchar(64)	YES		<null>		
6	Out_date	date	YES		<null>		

Рисунок 18 – Описание столбцов в таблице «game»

Просмотрим таблицу list_type.

	Field ▾	Type ▾	Null ▾	Key ▾	Default ▾	Extra ▾
1	ID_list_type	int	NO	PRI	<null>	
2	Name	int	YES		<null>	

Рисунок 19 – Описание столбцов в таблице «list_type»

Просмотрим таблицу user.

	Field ▾	Type ▾	Null ▾	Key ▾	Default ▾	Extra ▾
1	ID_user	int	NO	PRI	<null>	
2	ID_sex	int	YES	MUL	<null>	
3	Name	varchar(64)	YES		<null>	
4	Email	varchar(64)	YES		<null>	
5	Country	varchar(64)	YES		<null>	
6	Age	int	YES		<null>	
7	Pref_pegi	varchar(64)	YES		<null>	

Рисунок 20 – Описание столбцов в таблице «user»

Просмотрим таблицу list.

	Field ▾	Type ▾	Null ▾	Key ▾	Default ▾	Extra ▾
1	ID_list	int	NO	PRI	<null>	
2	user_ID_user	int	YES	MUL	<null>	
3	list_type_ID_list_type	int	YES	MUL	<null>	

Рисунок 21 – Описание столбцов в таблице «list»

Просмотрим таблицу list_has_game.

	Field ▾	Type ▾	Null ▾	Key ▾	Default ▾	Extra ▾
1	list_user_ID_user	int	NO	PRI	<null>	
2	list_ID_list	int	NO	PRI	<null>	
3	game_ID_game	int	NO	PRI	<null>	

Рисунок 22 – Описание столбцов в таблице «list_has_game»

ДОБАВЛЕНИЕ ДАННЫХ В ТАБЛИЦЫ БАЗЫ ДАННЫХ

Теперь нам необходимо внести данные в наши таблицы. На сайтах, вы обычно вводите информацию в какие-нибудь html-формы, затем сценарий на каком-либо языке (php, java...) извлекает эти данные из формы и заносит их в БД. Делает он это посредством SQL-запроса на внесение данных в базу.

Для этого используется оператор INSERT. Добавим данные в нашу таблицу statistic.

```
100 -- Insert data to Table for statistic
101 ✓ INSERT INTO statistic (ID_statistic, Age_statistic, Sex_statistic, Country_statistic)
102 VALUES ( ID_statistic 1, Age_statistic 25, Sex_statistic 'Male', Country_statistic 'RUS'),
103         ( ID_statistic 2, Age_statistic 30, Sex_statistic 'Female', Country_statistic 'DE'),
104         ( ID_statistic 3, Age_statistic 22, Sex_statistic 'Male', Country_statistic 'RUS'),
105         ( ID_statistic 4, Age_statistic 22, Sex_statistic 'Female', Country_statistic 'RUS'),
106         ( ID_statistic 5, Age_statistic 24, Sex_statistic 'Male', Country_statistic 'RUS'),
107         ( ID_statistic 6, Age_statistic 28, Sex_statistic 'Female', Country_statistic 'RUS'),
108         ( ID_statistic 7, Age_statistic 27, Sex_statistic 'Male', Country_statistic 'RUS'),
109         ( ID_statistic 8, Age_statistic 29, Sex_statistic 'Female', Country_statistic 'RUS'),
110         ( ID_statistic 9, Age_statistic 31, Sex_statistic 'Male', Country_statistic 'RUS'),
111         ( ID_statistic 10, Age_statistic 31, Sex_statistic 'Female', Country_statistic 'RUS');
```

Рисунок 1 – Добавление данных в таблицу «statistic»

Добавим данные в нашу таблицу studio.

```
113 -- Insert data to Table for studio
114 ✓ INSERT INTO studio (ID_studio, Name, Email, Country) VALUES
115     ( ID_studio 1, Name 'Valve', Email 'contact@valve.com', Country 'USA'),
116     ( ID_studio 2, Name 'Bethesda', Email 'contact@bethesda.com', Country 'USA'),
117     ( ID_studio 3, Name 'BioWare', Email 'contact@bioware.com', Country 'Canada'),
118     ( ID_studio 4, Name 'Rockstar Games', Email 'contact@rockstargames.com', Country 'USA'),
119     ( ID_studio 5, Name 'CD Projekt', Email 'contact@cdprojekt.com', Country 'Poland'),
120     ( ID_studio 6, Name 'Epic Games', Email 'contact@epicgames.com', Country 'USA'),
121     ( ID_studio 7, Name 'Ubisoft', Email 'contact@ubisoft.com', Country 'France'),
122     ( ID_studio 8, Name 'Naughty Dog', Email 'support@naughtydog.com', Country 'USA');
```

Рисунок 2 – Добавление данных в таблицу «studio»

Добавим данные в нашу таблицу pegi.

```
125 -- Insert data to Table for pegi
126 ✓ INSERT INTO pegi (ID_pegi, Name)
127 VALUES ( ID_pegi 1, Name 'PEGI 3'),
128         ( ID_pegi 2, Name 'PEGI 7'),
129         ( ID_pegi 3, Name 'PEGI 12'),
130         ( ID_pegi 4, Name 'PEGI 16'),
131         ( ID_pegi 5, Name 'PEGI 18');
```

Рисунок 3 – Добавление данных в таблицу «pegi»

Добавим данные в нашу таблицу sex.

```
134 -- Insert data to Table for sex
135 ✓ INSERT INTO sex (ID_sex, Name)
136 VALUES ( ID_sex 1, Name 'Male'),
137         ( ID_sex 2, Name 'Female');
```

Рисунок 4 – Добавление данных в таблицу «sex»

Добавим данные в нашу таблицу game.

```

139 -- Insert data to Table for studio
140 ✓ INSERT INTO game (ID_game, ID_pegi, ID_studio, Statistic_ID_statistic, Name, Out_date)
141 VALUES ( ID_game 1, ID_pegi 1, ID_studio 6, Statistic_ID_statistic 1, Name 'Fortnite', Out_date '2017-07-25'),
142 ( ID_game 2, ID_pegi 3, ID_studio 7, Statistic_ID_statistic 2, Name 'Assassin's Creed Valhalla', Out_date '2020-11-10'),
143 ( ID_game 3, ID_pegi 3, ID_studio 8, Statistic_ID_statistic 3, Name 'The Last of Us Part II', Out_date '2020-06-19'),
144 ( ID_game 4, ID_pegi 1, ID_studio 1, Statistic_ID_statistic 4, Name 'Half-Life 2', Out_date '2004-11-16'),
145 ( ID_game 5, ID_pegi 1, ID_studio 2, Statistic_ID_statistic 5, Name 'The Elder Scrolls IV: Oblivion', Out_date '2006-03-20'),
146 ( ID_game 6, ID_pegi 1, ID_studio 3, Statistic_ID_statistic 6, Name 'Mass Effect 2', Out_date '2010-01-26'),
147 ( ID_game 7, ID_pegi 1, ID_studio 4, Statistic_ID_statistic 7, Name 'Red Dead Redemption', Out_date '2010-05-18'),
148 ( ID_game 8, ID_pegi 1, ID_studio 5, Statistic_ID_statistic 8, Name 'The Witcher 3: Wild Hunt', Out_date '2015-05-19'),
149 ( ID_game 9, ID_pegi 3, ID_studio 3, Statistic_ID_statistic 9, Name 'The Last of Us Part I', Out_date '2013-06-14'),
150 ( ID_game 10, ID_pegi 3, ID_studio 7, Statistic_ID_statistic 10, Name 'Overwatch', Out_date '2016-05-24');

```

Рисунок 5 – Добавление данных в таблицу «game»

Добавим данные в нашу таблицу list_type.

```

153 -- Insert data to Table for list_type
154 ✓ INSERT INTO list_type (ID_list_type, Name)
155 VALUES ( ID_list_type 1, Name 'Wishlist'),
156 ( ID_list_type 2, Name 'Favorites'),
157 ( ID_list_type 3, Name 'Completed');

```

Рисунок 6 – Добавление данных в таблицу «list_type»

Добавим данные в нашу таблицу user.

```

159 -- Insert data to Table for user
160 ✓ INSERT INTO user (ID_user, ID_sex, Name, Email, Country, Age, Pref_pegi)
161 VALUES ( ID_user 1, ID_sex 1, Name 'John Doe', Email 'john@example.com', Country 'JP', Age 28, Pref_pegi 'PEGI 18'),
162 ( ID_user 2, ID_sex 2, Name 'Jane Smith', Email 'jane@example.com', Country 'UK', Age 35, Pref_pegi 'PEGI 7'),
163 ( ID_user 3, ID_sex 1, Name 'Alex Taylor', Email 'alex@example.com', Country 'RUS', Age 22, Pref_pegi 'PEGI 18'),
164 ( ID_user 4, ID_sex 1, Name 'Michael Johnson', Email 'michael.johnson@example.com', Country 'CH', Age 31, Pref_pegi 'PEGI 18'),
165 ( ID_user 5, ID_sex 2, Name 'Emily Davis', Email 'emily.davis@example.com', Country 'RUS', Age 29, Pref_pegi 'PEGI 12'),
166 ( ID_user 6, ID_sex 1, Name 'David Brown', Email 'david.brown@example.com', Country 'JP', Age 26, Pref_pegi 'PEGI 16'),
167 ( ID_user 7, ID_sex 2, Name 'Olivia Wilson', Email 'olivia.wilson@example.com', Country 'GE', Age 22, Pref_pegi 'PEGI 7'),
168 ( ID_user 8, ID_sex 2, Name 'Taylor Morgan', Email 'taylor.morgan@example.com', Country 'RUS', Age 33, Pref_pegi 'PEGI 18'),
169 ( ID_user 9, ID_sex 1, Name 'Chris Miller', Email 'chris.miller@example.com', Country 'AU', Age 27, Pref_pegi 'PEGI 3'),
170 ( ID_user 10, ID_sex 2, Name 'Sophia Harris', Email 'sophia.harris@example.com', Country 'FR', Age 24, Pref_pegi 'PEGI 12');

```

Рисунок 7 – Добавление данных в таблицу «user»

Добавим данные в нашу таблицу list.

```

172 -- Insert data to Table for list
173 ✓ INSERT INTO list (ID_list, user_ID_user, list_type_ID_list_type)
174 VALUES ( ID_list 1, user_ID_user 1, list_type_ID_list_type 1),
175 ( ID_list 2, user_ID_user 2, list_type_ID_list_type 2),
176 ( ID_list 3, user_ID_user 3, list_type_ID_list_type 3);

```

Рисунок 8 – Добавление данных в таблицу «list»

Добавим данные в нашу таблицу list_has_game.

```

178 -- Insert data to Table for list_has_game
179 ✓ INSERT INTO list_has_game (list_user_ID_user, list_ID_list, game_ID_game)
180 VALUES ( list_user_ID_user 1, list_ID_list 1, game_ID_game 1),
181 ( list_user_ID_user 2, list_ID_list 2, game_ID_game 2),
182 ( list_user_ID_user 3, list_ID_list 3, game_ID_game 3);

```

Рисунок 9 – Добавление данных в таблицу «list_has_game»

Чтобы посмотреть, какие данные у нас содержатся в таблицах. В SQL существует оператор SELECT. Просмотрим данные, которые мы добавили в таблицу statistic.

	ID_statistic ▾	Age_statistic ▾	Sex_statistic ▾	Country_statistic ▾
1	1	25	Male	RUS
2	2	30	Female	DE
3	3	22	Male	RUS
4	4	22	Female	RUS
5	5	24	Male	RUS
6	6	28	Female	RUS
7	7	27	Male	RUS
8	8	29	Female	RUS
9	9	31	Male	RUS
10	10	31	Female	RUS

Рисунок 10 – Добавленные данные в таблицу «statistic»

Просмотрим данные, которые мы добавили в таблицу studio.

	ID_studio ▾	Name ▾	Email ▾	Country ▾
1	1	Valve	contact@valve.com	USA
2	2	Bethesda	contact@bethesda.com	USA
3	3	BioWare	contact@bioware.com	Canada
4	4	Rockstar Games	contact@rockstargames.com	USA
5	5	CD Projekt	contact@cdprojekt.com	Poland
6	6	Epic Games	contact@epicgames.com	USA
7	7	Ubisoft	contact@ubisoft.com	France
8	8	Naughty Dog	support@naughtydog.com	USA

Рисунок 11 – Добавленные данные в таблицу «studio»

Просмотрим данные, которые мы добавили в таблицу pegi.

	ID_pegі ▾	Name ▾
1	1	PEGI 3
2	2	PEGI 7
3	3	PEGI 12
4	4	PEGI 16
5	5	PEGI 18

Рисунок 12 – Добавленные данные в таблицу «pegі»

Просмотрим данные, которые мы добавили в таблицу sex.

	ID_sex ▾	Name ▾
1	1	Male
2	2	Female

Рисунок 13 – Добавленные данные в таблицу «sex»

Просмотрим данные, которые мы добавили в таблицу game.

ID_game ▾	ID_peg1 ▾	ID_studio ▾	Statistic_ID_statistic ▾	Name ▾	Out_date ▾
1	1	1	6	1 Fortnite	2017-07-25
2	2	3	7	2 Assassin's Creed Valhalla	2020-11-10
3	3	3	8	3 The Last of Us Part II	2020-06-19
4	4	1	1	4 Half-Life 2	2004-11-16
5	5	1	2	5 The Elder Scrolls IV: Oblivion	2006-03-20
6	6	1	3	6 Mass Effect 2	2010-01-26
7	7	1	4	7 Red Dead Redemption	2010-05-18
8	8	1	5	8 The Witcher 3: Wild Hunt	2015-05-19
9	9	3	3	9 The Last of Us Part I	2013-06-14
10	10	3	7	10 Overwatch	2016-05-24

Рисунок 14 – Добавленные данные в таблицу «game»

Посмотрим данные, которые мы добавили в таблицу list_type.

ID_list_type ▾	Name ▾
1	1 Wishlist
2	2 Favorites
3	3 Completed

Рисунок 15 – Добавленные данные в таблицу «list_type»

Посмотрим данные, которые мы добавили в таблицу user.

ID_user ▾	ID_sex ▾	Name ▾	Email ▾	Country ▾	Age ▾	Pref_peg1 ▾
1	1	1 John Doe	john@example.com	JP	28	PEGI 18
2	2	2 Jane Smith	jane@example.com	UK	35	PEGI 7
3	3	1 Alex Taylor	alex@example.com	RUS	22	PEGI 18
4	4	1 Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
5	5	2 Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
6	6	1 David Brown	david.brown@example.com	JP	26	PEGI 16
7	7	2 Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7
8	8	2 Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18
9	9	1 Chris Miller	chris.miller@example.com	AU	27	PEGI 3
10	10	2 Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12

Рисунок 16 – Добавленные данные в таблицу «user»

Посмотрим данные, которые мы добавили в таблицу list.

ID_list ▾	user_ID_user ▾	list_type_ID_list_type ▾
1	1	1
2	2	2
3	3	3

Рисунок 17 – Добавленные данные в таблицу «list»

Посмотрим данные, которые мы добавили в таблицу list_has_game.

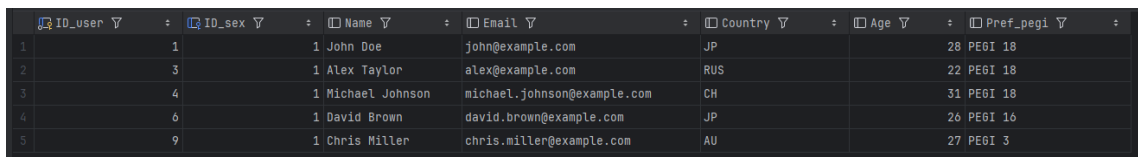
list_user_ID_user ▾	list_ID_list ▾	game_ID_game ▾
1	1	1
2	2	2
3	3	3

Рисунок 18 – Добавленные данные в таблицу «list_has_game»

ВЫБОРКА И СОРТИРОВКА ДАННЫХ

Очень часто бывает, что все информация из таблицы не нужна. Например, необходимо узнать, какие темы были созданы определенным пользователем. Для этого в SQL есть ключевое слово WHERE и специальные операторы.

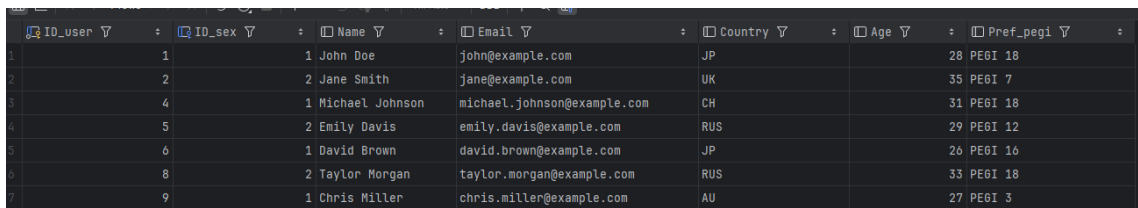
Применим оператор равно. При его использовании отбираются значения равные указанному.



ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	1	John Doe	john@example.com	JP	28	PEGI 18
2	3	Alex Taylor	alex@example.com	RUS	22	PEGI 18
3	4	Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
4	6	David Brown	david.brown@example.com	JP	26	PEGI 16
5	9	Chris Miller	chris.miller@example.com	AU	27	PEGI 3

Рисунок 1 – Просмотр таблицы user с применением оператора ID_sex = 1

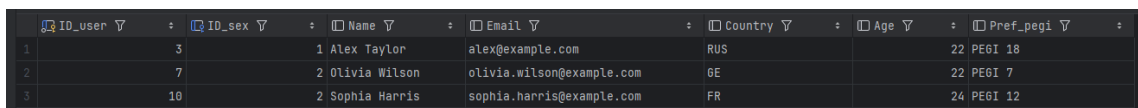
Применим оператор больше. При его использовании отбираются значения больше указанного.



ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	1	John Doe	john@example.com	JP	28	PEGI 18
2	2	Jane Smith	jane@example.com	UK	35	PEGI 7
3	4	Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
4	5	Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
5	6	David Brown	david.brown@example.com	JP	26	PEGI 16
6	8	Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18
7	9	Chris Miller	chris.miller@example.com	AU	27	PEGI 3

Рисунок 2 – Просмотр таблицы user с применением оператора Age > 25

Применим оператор меньше. При его использовании отбираются значения меньше указанного.



ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	3	Alex Taylor	alex@example.com	RUS	22	PEGI 18
2	7	Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7
3	10	Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12

Рисунок 3 – Просмотр таблицы user с применением оператора Age < 25;

Применим оператор больше или равно. При его использовании отбираются значения большие и равные указанному.



ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	1	John Doe	john@example.com	JP	28	PEGI 18
2	2	Jane Smith	jane@example.com	UK	35	PEGI 7
3	4	Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
4	5	Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
5	6	David Brown	david.brown@example.com	JP	26	PEGI 16
6	8	Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18
7	9	Chris Miller	chris.miller@example.com	AU	27	PEGI 3

Рисунок 4 – Просмотр таблицы user с применением оператора Age >= 25;

Применим оператор меньше или равно. При его использовании отбираются значения меньшие и равные указанному.

ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	3	1 Alex Taylor	alex@example.com	RUS	22	PEGI 18
2	7	2 Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7
3	10	2 Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12

Рисунок 5 – Просмотр таблицы user с применением оператора Age <= 25;

Применим оператор IS NOT NULL. При его использовании отбираются строки, имеющие значения в указанном поле.

ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	1	1 John Doe	john@example.com	JP	28	PEGI 18
2	2	2 Jane Smith	jane@example.com	UK	35	PEGI 7
3	3	1 Alex Taylor	alex@example.com	RUS	22	PEGI 18
4	4	1 Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
5	5	2 Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
6	6	1 David Brown	david.brown@example.com	JP	26	PEGI 16
7	7	2 Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7
8	8	2 Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18
9	9	1 Chris Miller	chris.miller@example.com	AU	27	PEGI 3
10	10	2 Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12

Рисунок 6 – Просмотр таблицы user с применением оператора ID_user IS NOT NULL

Применим оператор IS NULL. При его использовании отбираются строки, не имеющие значения в указанном поле.

ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
---------	--------	------	-------	---------	-----	-----------

Рисунок 7 – Просмотр таблицы user с применением оператора ID_user IS NULL;

Применим оператор BETWEEN (между). При его использовании отбираются значения, находящиеся между указанными.

ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	3	1 Alex Taylor	alex@example.com	RUS	22	PEGI 18
2	4	1 Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
3	5	2 Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
4	6	1 David Brown	david.brown@example.com	JP	26	PEGI 16

Рисунок 8 – Просмотр таблицы user с применением оператора ID_user BETWEEN 3 AND 6

Применим оператор IN (значение содержится). При его использовании отбираются значения, соответствующие указанным.

ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	9	1 Chris Miller	chris.miller@example.com	AU	27	PEGI 3
2	10	2 Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12

Рисунок 9 – Просмотр таблицы user с применением оператора Age IN (24, 27)

Применим оператор NOT IN (значение не содержится). При его использовании отбираются значения, кроме указанных.

	ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	1	1	John Doe	john@example.com	JP	28	PEGI 18
2	2	2	Jane Smith	jane@example.com	UK	35	PEGI 7
3	3	1	Alex Taylor	alex@example.com	RUS	22	PEGI 18
4	4	1	Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
5	5	2	Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
6	6	1	David Brown	david.brown@example.com	JP	26	PEGI 16
7	7	2	Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7
8	8	2	Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18

Рисунок 10 – Просмотр таблицы user с применением оператора Age NOT IN (24, 27);

Применим оператор LIKE (соответствие). При его использовании отбираются значения, соответствующие образцу.

	ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	5	2	Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
2	7	2	Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7
3	9	1	Chris Miller	chris.miller@example.com	AU	27	PEGI 3

Рисунок 11 – Просмотр таблицы user с применением оператора Name LIKE '%il%';

Применим оператор NOT LIKE (не соответствие). При его использовании отбираются значения, не соответствующие образцу.

	ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	1	1	John Doe	john@example.com	JP	28	PEGI 18
2	2	2	Jane Smith	jane@example.com	UK	35	PEGI 7
3	3	1	Alex Taylor	alex@example.com	RUS	22	PEGI 18
4	4	1	Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
5	6	1	David Brown	david.brown@example.com	JP	26	PEGI 16
6	8	2	Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18
7	10	2	Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12

Рисунок 12 – Просмотр таблицы user с применением оператора Name NOT LIKE '%il%';

Для сортировки в SQL существует ключевое слово ORDER BY после которого указывается имя столбца, по которому будет происходить сортировка.

	ID_user	ID_sex	Name	Email	Country	Age	Pref_pagi
1	3	1	Alex Taylor	alex@example.com	RUS	22	PEGI 18
2	7	2	Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7
3	10	2	Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12
4	6	1	David Brown	david.brown@example.com	JP	26	PEGI 16
5	9	1	Chris Miller	chris.miller@example.com	AU	27	PEGI 3
6	1	1	John Doe	john@example.com	JP	28	PEGI 18
7	5	2	Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
8	4	1	Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
9	8	2	Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18
10	2	2	Jane Smith	jane@example.com	UK	35	PEGI 7

Рисунок 13 – Отсортированные данные по возрастанию по столбцу Age

По умолчанию сортировка идет по возрастанию, но это можно изменить, добавив ключевое слово DESC.

	ID_user ▾	ID_sex ▾	Name ▾	Email ▾	Country ▾	Age ▾	Pref_peg1 ▾
1	2	2	Jane Smith	jane@example.com	UK	35	PEGI 7
2	8	2	Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18
3	4	1	Michael Johnson	michael.johnson@example.com	CH	31	PEGI 18
4	5	2	Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
5	1	1	John Doe	john@example.com	JP	28	PEGI 18
6	9	1	Chris Miller	chris.miller@example.com	AU	27	PEGI 3
7	6	1	David Brown	david.brown@example.com	JP	26	PEGI 16
8	10	2	Sophia Harris	sophia.harris@example.com	FR	24	PEGI 12
9	3	1	Alex Taylor	alex@example.com	RUS	22	PEGI 18
10	7	2	Olivia Wilson	olivia.wilson@example.com	GE	22	PEGI 7

Рисунок 14 – Отсортированные данные по убыванию по столбцу Age

ИЗМЕНЕНИЕ ДАННЫХ В ТАБЛИЦЕ

Для добавления столбцов в таблицу используется оператор ALTER TABLE — ADD COLUMN.

Для того, чтобы указать местоположение столбца используются ключевые слова: FIRST — новый столбец будет первым, и AFTER — указывает после какого столбца поместить новый.

	Field ▾	Type ▾	Null ▾	Key ▾	Default ▾	Extra ▾
1	ID_list_type	int	NO	PRI	<null>	
2	Name	varchar(64)	YES		<null>	
3	count	int	YES		<null>	

Рисунок 1 – Добавление столбца count в таблицу list_type

Для изменения имени существующего столбца используется оператор CHANGE.

	Field ▾	Type ▾	Null ▾	Key ▾	Default ▾	Extra ▾
1	ID_list_type	int	NO	PRI	<null>	
2	Name	varchar(64)	YES		<null>	
3	mods	int	YES		<null>	

Рисунок 2 – Изменение столбца count на mods в таблице list_type

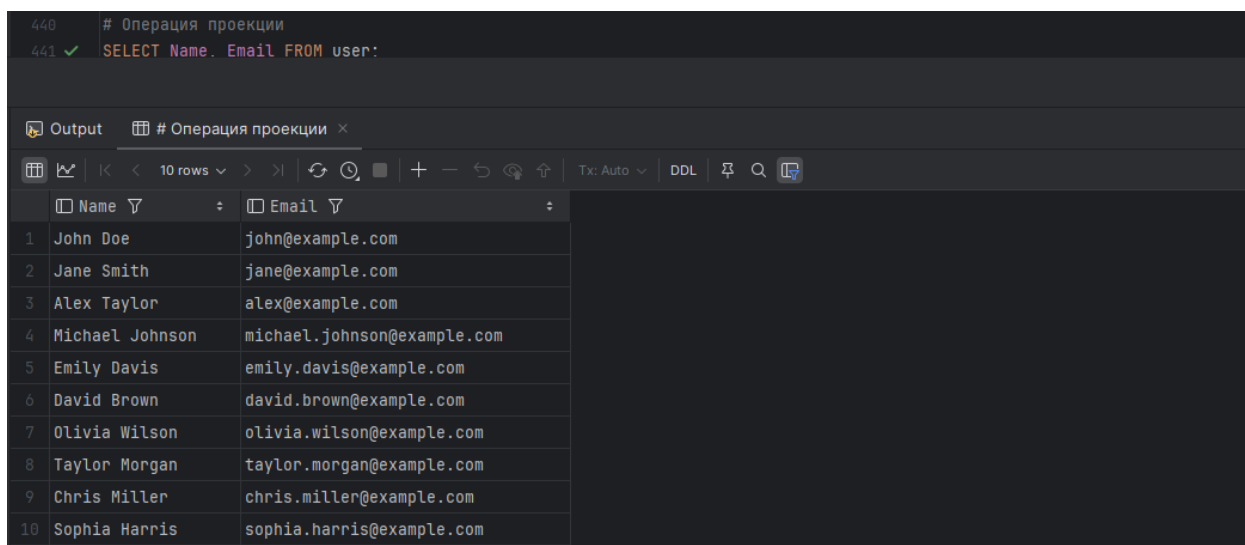
Рассмотрим — оператор DELETE, который позволяет удалять строки из таблицы.

	ID_user ▾	ID_sex ▾	Name ▾	Email ▾	Country ▾	Age ▾	Pref_pagi ▾
1	1	1	John Doe	john@example.com	JP		28 PE6I 18
2	3	1	Alex Taylor	alex@example.com	RUS		22 PE6I 18
3	4	1	Michael Johnson	michael.johnson@example.com	CH		31 PE6I 18
4	6	1	David Brown	david.brown@example.com	JP		26 PE6I 16
5	9	1	Chris Miller	chris.miller@example.com	AU		27 PE6I 3

Рисунок 3 – Удаление всех данных из таблицы user, где ID_sex = 2

РЕАЛИЦИОННАЯ АЛГЕБРА

Операция проекции. Осуществляется выбор только части полей таблицы, т.е. производится вертикальная выборка данных.

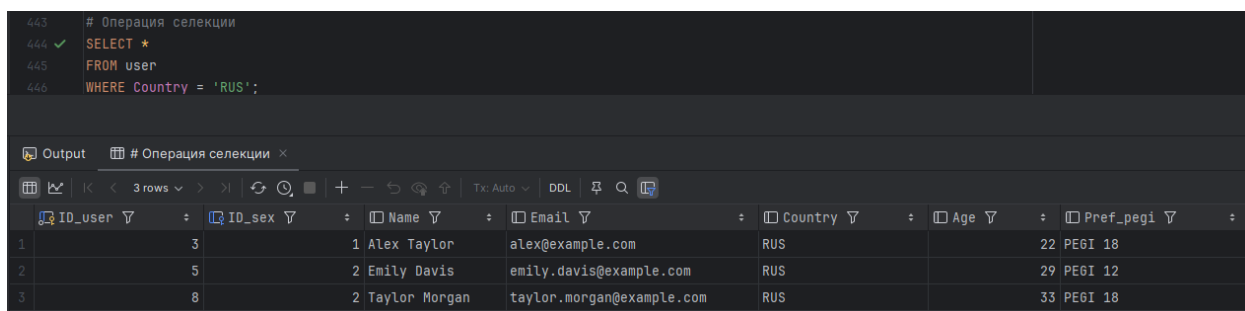


```
440 # Операция проекции
441 ✓ SELECT Name, Email FROM user;
```

	Name	Email
1	John Doe	john@example.com
2	Jane Smith	jane@example.com
3	Alex Taylor	alex@example.com
4	Michael Johnson	michael.johnson@example.com
5	Emily Davis	emily.davis@example.com
6	David Brown	david.brown@example.com
7	Olivia Wilson	olivia.wilson@example.com
8	Taylor Morgan	taylor.morgan@example.com
9	Chris Miller	chris.miller@example.com
10	Sophia Harris	sophia.harris@example.com

Рисунок 1 – Применение операции проекции

Операция селекции. Осуществляется горизонтальная выборка – в результат попадают только записи, удовлетворяющие условию.



```
443 # Операция селекции
444 ✓ SELECT *
445 FROM user
446 WHERE Country = 'RUS';
```

	ID_user	ID_sex	Name	Email	Country	Age	Pref_peg
1	3	1	Alex Taylor	alex@example.com	RUS	22	PEGI 18
2	5	2	Emily Davis	emily.davis@example.com	RUS	29	PEGI 12
3	8	2	Taylor Morgan	taylor.morgan@example.com	RUS	33	PEGI 18

Рисунок 2 – Применение операции селекции

Операции соединения. Здесь следует выделить декартово произведение и на его основе соединение по условию, а также естественное соединение (по одноименным полям или равенству полей с одинаковым смыслом).

448	# Операции соединения
449	✓ <code>SELECT game.ID_game, game.Name, studio.ID_studio, studio.Name</code>
450	<code>FROM game, studio</code>
451	<code>WHERE game.ID_studio = '1' AND studio.Country = 'USA';</code>

Output # Операции соединения					
	ID_game	game.Name	ID_studio	studio.Name	
1	4	Half-Life 2	1	Valve	
2	4	Half-Life 2	2	Bethesda	
3	4	Half-Life 2	4	Rockstar Games	
4	4	Half-Life 2	6	Epic Games	
5	4	Half-Life 2	8	Naughty Dog	

Рисунок 2 – Применение операции соединения

Операция объединения. Теоретико-множественные операции часто можно записать с помощью логических операций, примененных в конструкции WHERE запроса. Например, нужно получить список зачетов и экзаменов, которые сдают студенты 901 или 902 групп в 1 семестре. Таким образом, нужно объединить два множества, соответствующие двум разным группам. Объединение можно задать с помощью логического ИЛИ.

454	# Операция объединения
455	✓ <code>SELECT ID_user, Name, Country</code>
456	<code>FROM user</code>
457	<code>WHERE Country = 'RUS'</code>
458	<code>UNION</code>
459	<code>SELECT ID_user ID_user, Name Name, Country Country</code>
460	<code>FROM user</code>
461	<code>WHERE Country = 'JP';</code>

Output # Операция объединения			
	ID_user	Name	Country
1	3	Alex Taylor	RUS
2	5	Emily Davis	RUS
3	8	Taylor Morgan	RUS
4	1	John Doe	JP
5	6	David Brown	JP

Рисунок 2 – Применение операции объединения

Операция пересечения. В простых случаях эту операцию можно описать с помощью логической операции AND. В более сложных случаях эта операция определяется чаще всего с помощью подзапроса и ключевого слова EXISTS, которое показывает наличие похожего элемента во множестве, которое задается подзапросом.

```
463 # Операция пересечения
464 ✓ SELECT Name, Country
465 FROM user
466 WHERE Country = 'FR'
467 AND EXISTS (
468     SELECT *
469     FROM studio
470     WHERE Country = 'France'
471 );
```

Output # Операция пересечения ×

Name	Country
Sophia Harris	FR

Рисунок 2 – Применение операции пересечения

Операция разности. Эта операция также определяется часто с помощью подзапроса с ключевым словом NOT EXISTS, которое показывает отсутствие элемента во множестве, задаваемом подзапросом.

```
472 # Операция разности
473 ✓ SELECT Name, Country
474 FROM user
475 WHERE Country = 'RUS' OR Country = 'UK'
476 EXCEPT
477 SELECT Name, Country
478 FROM user
479 WHERE Country = 'JP';
```

Output # Операция разности ×

Name	Country
Jane Smith	UK
Alex Taylor	RUS
Emily Davis	RUS
Taylor Morgan	RUS

Рисунок 2 – Применение операции разности

Операция группировки. Эта операция связана со своеобразной сверткой таблицы по полям группировки. Помимо полей группировки результат запроса может содержать итоговые агрегирующие функции по группам (COUNT, SUM, AVG, MAX, MIN).

481	# Операция группировки
482	✓ <code>SELECT Country, COUNT(*) AS UserCount</code>
483	<code>FROM user</code>
484	<code>GROUP BY Country;</code>

Output		# Операция группировки	
7 rows			
	Country		UserCount
1	JP		2
2	UK		1
3	RUS		3
4	CH		1
5	GE		1
6	AU		1
7	FR		1

Рисунок 2 – Применение операции группировки

Операция сортировки. Вывести всех преподавателей, которым сдают студенты зачеты-экзамены в первом семестре, в порядке убывания количества зачетов-экзаменов. Для этого следует сначала выбрать нужные элементы таблицы Sessions, затем осуществить естественное соединение полученной таблицы с таблицей Teachers, после чего производится группировка записей в результате запроса и последующая сортировка.

486	# Операция сортировки
487	✓ <code>SELECT Name, Out_date</code>
488	<code>FROM game</code>
489	<code>ORDER BY Out_date DESC;</code>
490	

Output		# Операция сортировки	
10 rows			
	Name		Out_date
1	Assassin's Creed Valhalla		2020-11-10
2	The Last of Us Part II		2020-06-19
3	Fortnite		2017-07-25
4	Overwatch		2016-05-24
5	The Witcher 3: Wild Hunt		2015-05-19
6	The Last of Us Part I		2013-06-14
7	Red Dead Redemption		2010-05-18
8	Mass Effect 2		2010-01-26
9	The Elder Scrolls IV: Oblivion		2006-03-20
10	Half-Life 2		2004-11-16

Рисунок 2 – Применение операции сортировки

Операция деления. Это самая нетривиальная операция реляционной алгебры, которая обычно применяется тогда, когда требуется найти все записи первой таблицы, которые соединяются естественным образом со всеми записями второй таблицы. Например, нам требуется найти тех преподавателей,

которым должны сдать в первом семестре зачеты-экзамены студенты всех групп факультета. Запрос получается достаточно сложный и он связан с выполнением двух операций разности (первая разность - из всевозможных комбинаций групп и преподавателей вычитаются реальные комбинации этих полей, т.е. 46 результатом становятся всевозможные нереальные пары, вторая разность – выбираются преподаватели, которые в нереальных парах не присутствуют).

ХРАНИМЫЕ ПРОЦЕДУРЫ, ФУНКЦИИ И ТРИГГЕРЫ

Хранимые процедуры, функции и триггеры вводятся в базу данных для обеспечения бизнес-логики приложения на уровне серверной его компоненты. Обычно хранимые процедуры и функции представляют собой утилиты, которые определенным образом обрабатывают данные или реализуют достаточно сложный алгоритм вычисления некоторых показателей.

```
# [ ПРОЦЕДУРА ]
DELIMITER //

CREATE PROCEDURE add_game(
    IN p_ID_pegі INT,
    IN p_ID_studio INT,
    IN p_Statistic_ID_statistic INT,
    IN p_Name VARCHAR(64),
    IN p_Out_date DATE
)
BEGIN
    IF NOT EXISTS (SELECT 1 FROM studio WHERE ID_studio = p_ID_studio) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Studio does not exist!';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM pegі WHERE ID_pegі = p_ID_pegі) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'PEGI category does not exist!';
    END IF;

    INSERT INTO game (ID_pegі, ID_studio, Statistic_ID_statistic, Name, Out_date)
    VALUES ( ID_pegі p_ID_pegі, ID_studio p_ID_studio, Statistic_ID_statistic p_Statistic_ID_statistic, Name p_Name, Out_date p_Out_date);
END;

# ВЫЗОВ ПРОЦЕДУРЫ ДЛЯ ДОБАВЛЕНИЯ ИГРЫ
CALL add_game( p_ID_pegі 5, p_ID_studio 9, p_Statistic_ID_statistic 11, p_Name 'Elden Ring', p_Out_date '2022-02-25');
CALL add_game( p_ID_pegі 1, p_ID_studio 2, p_Statistic_ID_statistic 3, p_Name 'New Game', p_Out_date '2024-02-25');
```

Рисунок 1 – Код процедуры для добавления игры и ее последующий вызов

Вызовем написанную нами процедуру используя оператор CALL.

```
[2024-11-19 15:05:30] completed in 21 ms
gamecalendar> CALL add_game(5, 9, 11, 'Elden Ring', '2022-02-25')
[2024-11-19 15:05:30] [45000][1644] Studio does not exist!
gamecalendar> CALL add_game(1, 2, 3, 'New Game', '2024-02-25')
[2024-11-19 15:06:43] 1 row affected in 16 ms
```

Рисунок 2 – Результат отработки процедуры

```
255 //
256 DELIMITER ;
257 CREATE PROCEDURE create_user_list(
258     IN user_id INT,
259     IN list_type_id INT
260 )
261 BEGIN
262     INSERT INTO list (user_ID_user, list_type_ID_list_type)
263     VALUES ( user_ID_user user_id, list_type_ID_list_type list_type_id);
264 END;
265
266 CALL create_user_list( user_id 1, list_type_id 2);
```

Рисунок 3 – Код процедуры для создания списка пользователя и ее последующий вызов

Вызовем написанную нами процедуру используя оператор CALL.

```
gamecalendar> CALL create_user_list(1, 2)
[2024-11-27 15:25:36] 1 row affected in 19 ms
```

Рисунок 4 – Результат отработки процедуры

```
# [ ФУНКЦИЯ ]
DELIMITER //

CREATE FUNCTION get_game_count_by_studio(p_ID_studio INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE game_count INT;

    SELECT COUNT(*)
    INTO game_count
    FROM game
    WHERE ID_studio = p_ID_studio;

    RETURN game_count;
END;

//
DELIMITER ;

# ВЫЗОВ ФУНКЦИИ
SELECT get_game_count_by_studio(p_ID_studio 2);
```

Рисунок 5 – Код функции для просмотра количества игр у студии и ее вызов
Вызовем написанную нами функцию.

	<code>`get_game_count_by_studio(2)`</code>	
1		2

Рисунок 6 – Результат отработки функции

```
290 //
291 DELIMITER ;
292 CREATE FUNCTION average_age_by_country(country_name VARCHAR(64))
293 RETURNS DECIMAL(5, 2)
294 DETERMINISTIC
295 BEGIN
296     DECLARE avg_age DECIMAL(5, 2);
297     SELECT AVG(Age) INTO avg_age
298     FROM user
299     WHERE Country = country_name;
300     RETURN avg_age;
301 END;
302
303 ✓ SELECT average_age_by_country(country_name 'RUS');
```

Рисунок 7 – Код функции для просмотра среднего возраста по стране и ее
ВЫЗОВ

Вызовем написанную нами функцию.

	<code>`average_age_by_country('RUS')`</code>	
1		28.00

Рисунок 8 – Результат отработки функции

Триггеры – это частный случай хранимой процедуры, который выполняется автоматически при выполнении команд обновления данных (INSERT, DELETE, UPDATE). Триггеры привязываются к конкретным таблицам базы данных. Для каждой команды должны быть свои триггеры.

Создадим таблицу для логирования отработки наших триггеров.

```
# ТАБЛИЦЫ ДЛЯ ЛОГИРОВАНИЯ
CREATE TABLE game_log (
  ID INT AUTO_INCREMENT PRIMARY KEY,
  ID_game INT,
  old_name VARCHAR(64),
  new_name VARCHAR(64),
  operation_date DATETIME
);
```

Рисунок 9 – Создание таблицы для логирования изменений

Напишем триггер, который будет срабатывать при изменении уже существующих данных в таблице game.

```
# ТРИГГЕР ИЗМЕНЕНИЕ
DELIMITER //

CREATE TRIGGER trg_update_game
AFTER UPDATE ON game
FOR EACH ROW
BEGIN
  INSERT INTO game_log (ID_game, operation, old_name, new_name, operation_date)
  VALUES (OLD.ID_game, 'UPDATE', old_name OLD.Name, new_name NEW.Name, operation_date NOW());
END;

//
DELIMITER ;
```

Рисунок 10 – Триггер на изменение игры

При изменении данных в нашей таблице у нас вызывается триггер, который записывает изменения в таблицу логирования, созданную ранее.

ID	ID_game	operation	old_name	new_name	operation_date
1	5	11 UPDATE	New Game	New new name	2024-11-19 15:33:27
2	6	12 UPDATE	New Game	New new name	2024-11-19 15:33:27
3	7	13 UPDATE	New Game	New new name	2024-11-19 15:33:27
4	8	14 UPDATE	New Game	New new name	2024-11-19 15:33:27
5	9	15 UPDATE	New Game	New new name	2024-11-19 15:33:27

Рисунок 11 – Срабатывание триггера на изменение игры

Напишем триггер, который будет срабатывать при обновлении данных пользователя.

```

534 CREATE TRIGGER after_user_age_update
535 AFTER UPDATE ON user
536 FOR EACH ROW
537 BEGIN
538     IF NEW.Age > 30 AND NEW.Pref_pegі != 'PEGI 18' THEN
539         UPDATE user
540             SET Pref_pegі = 'PEGI 18'
541             WHERE ID_user = NEW.ID_user;
542     END IF;
543 END;
544
545 UPDATE user SET Age = 34
546 WHERE ID_user = 1;
547
548 ✓ SELECT * FROM user WHERE ID_user = 1;

```

Рисунок 12 – Триггер на добавления пользователя

При изменении возраста пользователя у нас вызывается триггер, который изменяет предпочтительный PEGI на PEGI-18.

ID_user	ID_sex	Name	Email	Country	Age	Pref_pegі
1	1	John Doe	john@example.com	JP	34	PEGI 18

Рисунок 13 – Срабатывание триггера на попытку добавление не соответствующего пользователя

Напишем триггер, который будет срабатывать при добавлении пользователя и проверять правильно ли написан email.

```

354 DELIMITER $$
355 CREATE TRIGGER validate_email
356 BEFORE INSERT ON user
357 FOR EACH ROW
358 BEGIN
359     IF NEW.Email NOT REGEXP '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$' THEN
360         SIGNAL SQLSTATE '45000'
361         SET MESSAGE_TEXT = 'Invalid email format.';
362     END IF;
363 END$$
364 DELIMITER ;
365
366 ⚠ INSERT INTO user (ID_sex, Name, Email, Country, Age, Pref_pegі)
367 VALUES ( ID_sex 10, Name 'TEST', Email 'KID_MAIL.RU', Country 'RUS', Age 25, Pref_pegі 1);
368

```

Рисунок 14 – Триггер на добавления пользователя

При добавлении пользователя у нас вызывается триггер, который записывает проверяет почту пользователя на соответствие.

```

366 ⚠ INSERT INTO user (ID_sex, Name, Email, Country, Age, Pref_pegі)
367 VALUES ( ID_sex 10, Name 'TEST', Email 'KID_MAIL.RU', Country 'RUS', Age 25, Pref_pegі 1);
368
[45000][1644] Invalid email format.

```

Рисунок 15 – Срабатывание триггера на попытку добавление не правильного email

Напишем триггер, который будет срабатывать при добавлении игры в список.


```

840 CREATE TEMPORARY TABLE temp_list_changes (
841     user_ID INT,
842     list_ID INT,
843     game_ID INT
844 );
845
846 DELIMITER $$
847
848 CREATE TRIGGER after_list_has_game_insert
849 AFTER INSERT ON list_has_game
850 FOR EACH ROW
851 BEGIN
852     DECLARE list_type INT;
853
854     -- Определить тип списка
855     SET list_type = (SELECT list_type_ID_list_type FROM list WHERE ID_list = NEW.list_ID_list);
856
857     -- Если добавление в "Favorites"
858     IF list_type = 2 THEN
859         -- Добавить изменения во временную таблицу
860         INSERT INTO temp_list_changes (user_ID, list_ID, game_ID)
861         VALUES (user_ID NEW.list_user_ID_user, (SELECT list_ID_ID_list FROM list WHERE user_ID_user = NEW.list_user_ID_user AND list_type_ID_list_type = 3), game_ID NEW.game_ID_game);
862     END IF;
863 END $$
864
865 DELIMITER ;
866
867
868 INSERT list_has_game VALUE (list_user_ID_user 1, list_ID_list 2, game_ID_game 3);

```

Рисунок 16 – Триггер на добавление игры в список

При добавлении игры в список Избранное у нас вызывается триггер, который добавляет игру в список Пройдено.

	list_user_ID_user ▾	÷	list_ID_list ▾	÷	game_ID_game ▾	÷
1		1		2		2
2		1		2		3

Рисунок 17 – Срабатывание триггера на попытку добавления игры

Напишем триггер, который будет срабатывать при добавлении игры в список пользователя.

```

181 # ПЯТЫЙ ТРИГГЕР
182 DELIMITER $$
183
184 CREATE TRIGGER update_game_count
185 AFTER INSERT ON list_has_game
186 FOR EACH ROW
187 BEGIN
188     DECLARE total_games INT;
189
190     -- Определить ID списка
191     DECLARE target_list_id INT;
192
193     -- Определить ID списка в зависимости от операции
194     SET target_list_id = NEW.list_ID_list;
195
196     -- Подсчитать количество игр в списке
197     SET total_games = (SELECT COUNT(*) FROM list_has_game WHERE list_ID_list = target_list_id);
198
199     -- Обновить значение количества игр в таблице list
200     UPDATE list
201     SET Game_Count = total_games
202     WHERE ID_list = target_list_id;
203 END $$
204
205 DELIMITER ;

```

Рисунок 18 – Триггер на добавление игры

При добавлении игры у в список пользователя у нас вызывается триггер, который увеличивает количество игр в списке.

	ID_list ▾	÷	user_ID_user ▾	÷	list_type_ID_list_type ▾	÷	Game_Count ▾	÷
1	1		1		1		1	
2	2		1		2		1	
3	3		1		3		0	
4	4		2		1		0	
5	5		2		2		0	

Рисунок 19 – Срабатывание триггера на попытку добавление не вышедшей игры

ОКОННЫЕ ФУНКЦИИ

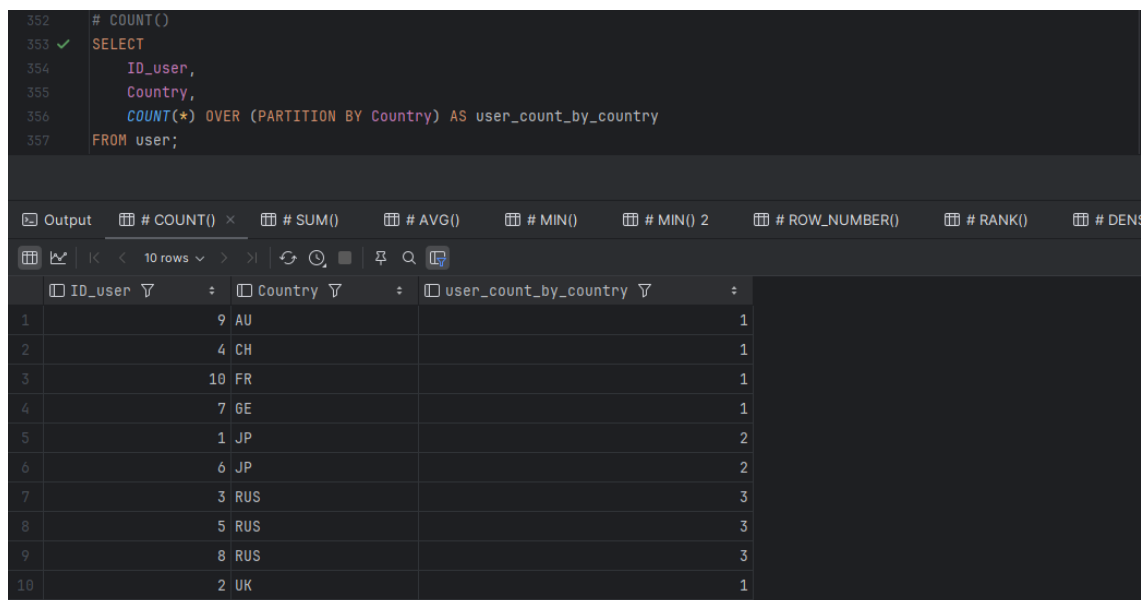
Оконные функции в MySQL — это специальный тип функции, который позволяет выполнять агрегатные и аналитические операции над группами строк, которые определены внутри отдельного окна (или оконного фрейма). Оконные функции представляют собой удобный инструмент по работы с данными и предоставляют более гибкий способ обращения с ними, чем традиционные агрегатные функции за счёт того, что они могут учитывать порядок сортировки данных и разбивать их на группы без фактической группировки.

В рамках данной работы разделим оконные функции на следующие 3 группы:

1. Агрегатные функции:

Агрегатными функциями называются функции, которые выполняют арифметические вычисления на наборе данных и возвращают итоговое значение.

COUNT(*) — вычисляет количество значений в столбце (не учитывает значения NULL)



```
352 # COUNT()
353 ✓ SELECT
354     ID_user,
355     Country,
356     COUNT(*) OVER (PARTITION BY Country) AS user_count_by_country
357 FROM user;
```

	ID_user	Country	user_count_by_country
1	9	AU	1
2	4	CH	1
3	10	FR	1
4	7	GE	1
5	1	JP	2
6	6	JP	2
7	3	RUS	3
8	5	RUS	3
9	8	RUS	3
10	2	UK	1

Рисунок 1 – Код и выполнение агрегатной функции COUNT

SUM(*) — возвращает сумму значений в столбце;

```

359 # SUM()
360 ✓ SELECT
361     ID_user,
362     Country,
363     Age,
364     SUM(Age) OVER (PARTITION BY Country) AS total_age_by_country
365 FROM user;

```

	ID_user	Country	Age	total_age_by_country
1	9	AU	27	27
2	4	CH	31	31
3	10	FR	24	24
4	7	GE	22	22
5	1	JP	28	54
6	6	JP	26	54
7	3	RUS	22	84
8	5	RUS	29	84
9	8	RUS	33	84
10	2	UK	35	35

Рисунок 2 – Код и выполнение агрегатной функции SUM

AVG(*) — определяет среднее значение в столбце

```

367 # AVG()
368 ✓ SELECT
369     ID_user,
370     ID_sex,
371     Age,
372     AVG(Age) OVER (PARTITION BY ID_sex) AS avg_age_by_sex
373 FROM user;

```

	ID_user	ID_sex	Age	avg_age_by_sex
1	1	1	28	26.8000
2	3	1	22	26.8000
3	4	1	31	26.8000
4	6	1	26	26.8000
5	9	1	27	26.8000
6	2	2	35	28.6000
7	5	2	29	28.6000
8	7	2	22	28.6000
9	8	2	33	28.6000
10	10	2	24	28.6000

Рисунок 3 – Код и выполнение агрегатной функции AVG

MIN(*) — определяет минимальное значение в столбце.

```

375 # MIN()
376 ✓ SELECT
377     ID_user,
378     Country,
379     Age,
380     MIN(Age) OVER (PARTITION BY Country) AS total_age_by_country
381 FROM user;

```

	ID_user	Country	Age	total_age_by_country
1	9	AU	27	27
2	4	CH	31	31
3	10	FR	24	24
4	7	GE	22	22
5	1	JP	28	26
6	6	JP	26	26
7	3	RUS	22	22
8	5	RUS	29	22
9	8	RUS	33	22
10	2	UK	35	35

Рисунок 4 – Код и выполнение агрегатной функции MIN

MAX(*) — определяет максимальное значение в столбце

```

383 # MAX()
384 ✓ SELECT
385     ID_user,
386     Country,
387     Age,
388     MAX(Age) OVER (PARTITION BY Country) AS total_age_by_country
389 FROM user;

```

	ID_user	Country	Age	total_age_by_country
1	9	AU	27	27
2	4	CH	31	31
3	10	FR	24	24
4	7	GE	22	22
5	1	JP	28	28
6	6	JP	26	28
7	3	RUS	22	33
8	5	RUS	29	33
9	8	RUS	33	33
10	2	UK	35	35

Рисунок 5 – Код и выполнение агрегатной функции MAX

2. Ранжирующие функции:

Ранжирующие функции — это функции, которые определяют ранг для каждой строки в окне. Например, их можно использовать для присвоения порядковых номеров или для составления рейтинга.

ROW_NUMBER() — функция возвращает номер строки и используется для нумерации.

```

391 #==> Ранжирующие функции
392 # ROW_NUMBER()
393 SELECT
394     ID_user,
395     Age,
396     ROW_NUMBER() OVER (ORDER BY Age DESC) AS row_number_by_age
397 FROM user;

```

ID_user	Age	row_number_by_age
1	2	35
2	8	33
3	4	31
4	5	29
5	1	28
6	9	27
7	6	26
8	10	24
9	3	22
10	7	22

Рисунок 6 – Код и выполнение ранжирующей функции ROW_NUMBER()

RANK() — функция возвращает ранг каждой строки. Данная функция в том числе анализирует данные и, в случае нахождения одинаковых — возвращает одинаковый ранг с пропуском следующего значения (например, два различных товара были проданы на одинаковую сумму по итогам месяца. При использовании данной функции для оценки ранга продаж за месяц обоим товарам будет выставлен ранг 1, а следующий за ними товар получит ранг 3).

```

399 # RANK()
400 SELECT
401     ID_user,
402     Country,
403     Age,
404     RANK() OVER (ORDER BY Country DESC) AS rank_by_age
405 FROM user;

```

ID_user	Country	Age	rank_by_age
1	2 UK	35	1
2	3 RUS	22	2
3	5 RUS	29	2
4	8 RUS	33	2
5	1 JP	28	5
6	6 JP	26	5
7	7 GE	22	7
8	10 FR	24	8
9	4 CH	31	9
10	9 AU	27	10

Рисунок 7 – Код и выполнение ранжирующей функции RANK()

DENSE_RANK() — так же, как и прошлая функция, возвращает ранг каждой строки, но в отличие от функции RANK, следующий ранг пропускаться не будет.

```

407 # DENSE_RANK()
408 ✓ SELECT
409     ID_user,
410     Country,
411     Age,
412     DENSE_RANK() OVER (ORDER BY Country DESC) AS dense_rank_by_age
413 FROM user;

```

ID_user	Country	Age	dense_rank_by_age
2	UK	35	1
3	RUS	22	2
5	RUS	29	2
8	RUS	33	2
1	JP	28	3
6	JP	26	3
7	GE	22	4
10	FR	24	5
4	CH	31	6
9	AU	27	7

Рисунок 8 – Код и выполнение ранжирующей функции DENSE_RANK()

NTILE(*) — это функция, которая позволяет определить, к какой группе относится текущая строка. Количество групп задаётся в скобках.

```

415 # NTILE()
416 ✓ SELECT
417     ID_user,
418     Age,
419     NTILE(4) OVER (ORDER BY Age) AS percentile_by_age
420 FROM user;

```

ID_user	Age	percentile_by_age
3	22	1
7	22	1
10	24	1
6	26	2
9	27	2
1	28	2
5	29	3
4	31	3
8	33	4
2	35	4

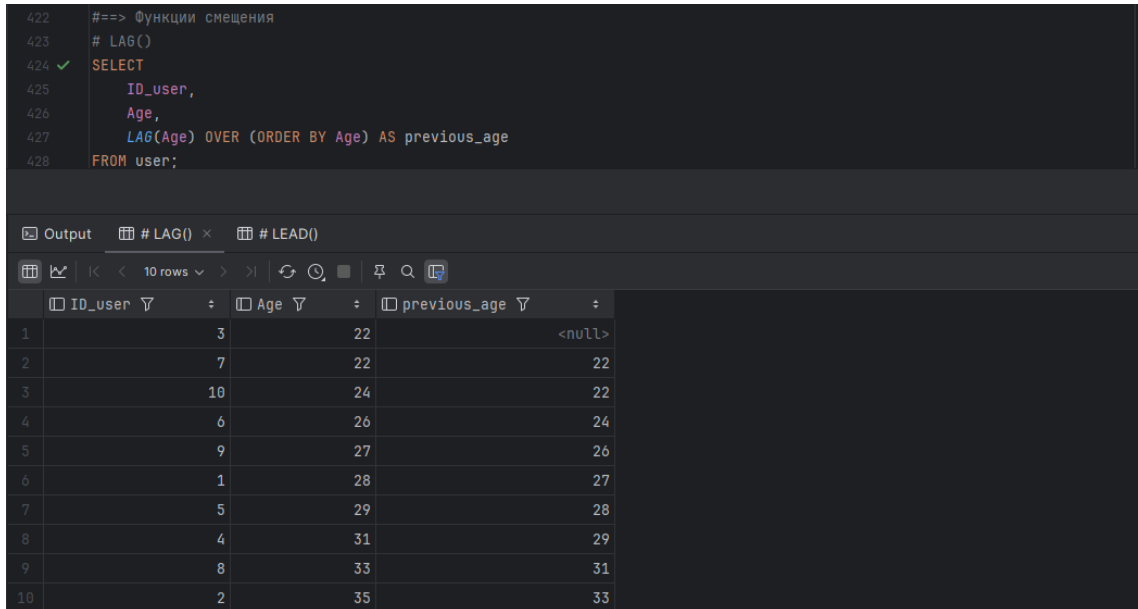
Рисунок 9 – Код и выполнение ранжирующей функции NTILE()

3. Функции смещения:

Функции смещения — это функции, которые позволяют перемещаться и обращаться к разным строкам в окне относительно текущей строки, а также обращаться к значениям в начале или в конце окна.

LAG(*) и LEAD(*) — функция LAG обращается к данным из предыдущей строки окна, а LEAD к данным из следующей строки. Функцию можно использовать для сравнения текущего значения строки с предыдущим

или следующим. Имеет три параметра: столбец, значение которого необходимо вернуть, количество строк для смещения (по умолчанию это 1) и значение, которое необходимо вернуть, если после смещения возвращается значение NULL;



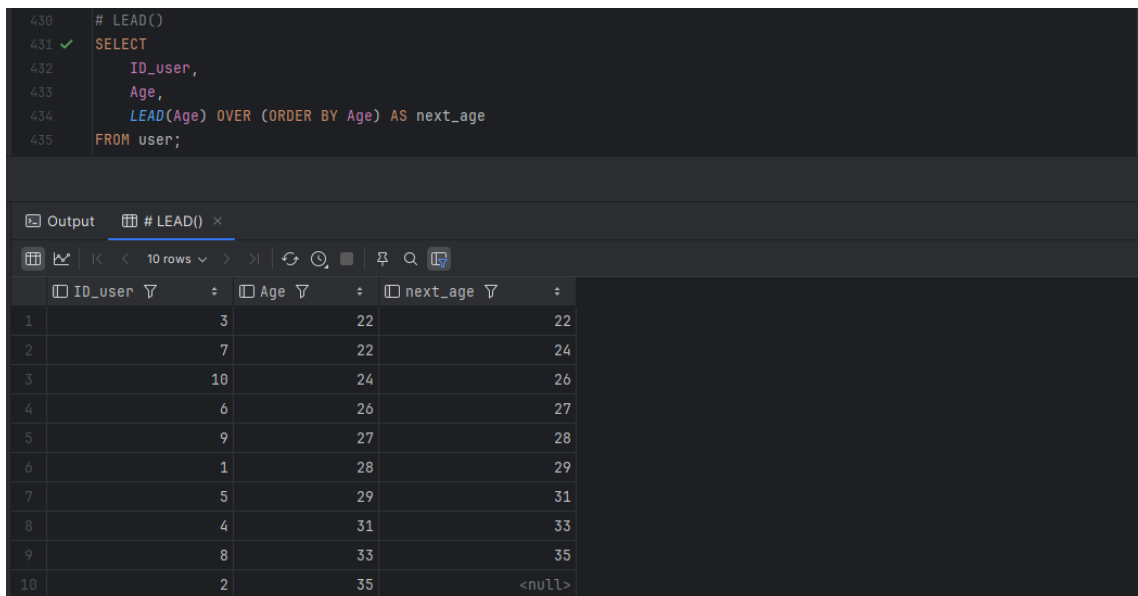
```
422 #==> Функции смещения
423 # LAG()
424 ✓ SELECT
425     ID_user,
426     Age,
427     LAG(Age) OVER (ORDER BY Age) AS previous_age
428 FROM user;
```

Output # LAG() × # LEAD()

	ID_user	Age	previous_age
1	3	22	<null>
2	7	22	22
3	10	24	22
4	6	26	24
5	9	27	26
6	1	28	27
7	5	29	28
8	4	31	29
9	8	33	31
10	2	35	33

Рисунок 10 – Код и выполнение функции смещения LAG()

Теперь отобразим функцию смещения LEAD



```
430 # LEAD()
431 ✓ SELECT
432     ID_user,
433     Age,
434     LEAD(Age) OVER (ORDER BY Age) AS next_age
435 FROM user;
```

Output # LEAD() ×

	ID_user	Age	next_age
1	3	22	22
2	7	22	24
3	10	24	26
4	6	26	27
5	9	27	28
6	1	28	29
7	5	29	31
8	4	31	33
9	8	33	35
10	2	35	<null>

Рисунок 11 – Код и выполнение функции смещения LAG()

FIRST_VALUE(*) и LAST_VALUE(*) — функция FIRST_VALUE позволяет получить первое значение в окне, а LAST_VALUE, соответственно, последнее значение.

438	✓	SELECT
439		ID_game,
440		Name,
441		Out_date,
442		FIRST_VALUE(Name) OVER (ORDER BY Out_date) AS FirstGame
443		FROM game;

Output		Result 49	
ID_game	Name	Out_date	FirstGame
4	Half-Life 2	2004-11-16	Half-Life 2
5	The Elder Scrolls IV: Oblivion	2006-03-20	Half-Life 2
6	Mass Effect 2	2010-01-26	Half-Life 2
7	Red Dead Redemption	2010-05-18	Half-Life 2
9	The Last of Us Part I	2013-06-14	Half-Life 2
8	The Witcher 3: Wild Hunt	2015-05-19	Half-Life 2
10	Overwatch	2016-05-24	Half-Life 2
1	Fortnite	2017-07-25	Half-Life 2
3	The Last of Us Part II	2020-06-19	Half-Life 2
2	Assassin's Creed Valhalla	2020-11-10	Half-Life 2

Рисунок 12 – Код и выполнение функции смещения FIRST_VALUE ()

ВЫВОД

В ходе изучения работы с базами данных на языке MySQL мы освоили множество ключевых аспектов, которые позволяют эффективно управлять данными и их обработкой. Мы научились проектировать структуры данных, создавая базы данных и таблицы с использованием различных типов данных, что дало возможность организовать хранение информации и оптимизировать доступ к ней. При создании таблиц мы учитывали важные элементы, такие как первичные ключи, индексы и связи между таблицами, что является основой нормализации базы данных и позволяет обеспечивать целостность и правильную организацию данных.

Кроме того, были освоены методы добавления данных в таблицы с помощью SQL-запросов, включая как одиночные, так и массовые вставки. Это значительно упрощает процесс наполнения таблиц данными и их дальнейшее управление, что крайне важно для работы с динамично изменяющимися информационными потоками в базе данных.

Хранимые процедуры и функции открыло перед нами возможность создания повторно используемых блоков кода, что значительно упрощает выполнение сложных операций с данными и способствует улучшению производительности базы данных. Триггеры же научили нас автоматически выполнять действия при наступлении определенных событий в базе данных, таких как добавление, обновление или удаление данных, что повысило уровень автоматизации процессов и безопасности при работе с данными.

В результате, эти навыки и знания позволили нам значительно расширить возможности работы с базами данных, улучшив как процесс разработки, так и управление данными на всех этапах их жизненного цикла. Мы научились эффективно проектировать, наполнять, извлекать, изменять и автоматизировать операции с данными, что является основой успешного и безопасного использования реляционных баз данных.