

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ПЕТРА ВЕЛИКОГО

---

Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе № 4

Дисциплина: «Базы данных»

Тема: «Язык SQL-DML»

Выполнил студент гр. 43501/3

\_\_\_\_\_ А.Ю. Ламтев  
(подпись)

Преподаватель

\_\_\_\_\_ А.В. Мяснов  
(подпись)

«\_\_\_» \_\_\_\_\_ 2019 г.

Санкт-Петербург  
2019

# Содержание

<b>1</b>	<b>Цели работы</b>	<b>3</b>
<b>2</b>	<b>Программа работы</b>	<b>3</b>
<b>3</b>	<b>Стандартные запросы</b>	<b>3</b>
3.1	Выборка данных из одной таблицы с использованием логических операций, LIKE, BETWEEN, IN . . . . .	3
3.2	Запрос с вычисляемым полем . . . . .	4
3.3	Выборка с сортировкой по нескольким полям . . . . .	5
3.4	Запрос с вычислением совокупных характеристик . . . . .	5
3.5	Выборка данных из связанных таблиц . . . . .	6
3.6	Запрос с подзапросами . . . . .	7
3.7	Запрос с ограничением результата группировки . . . . .	7
3.8	Добавление записей в таблицы . . . . .	7
3.9	Изменение значений полей записей, удовлетворяющих условиям	8
3.10	Удаление записей . . . . .	8
<b>4</b>	<b>Запросы в соответствии с индивидуальным заданием</b>	<b>9</b>
<b>5</b>	<b>Сохранение в БД выполненных запросов в виде представлений и ХП</b>	<b>9</b>
<b>6</b>	<b>Выводы</b>	<b>9</b>

# 1. Цели работы

Познакомиться с языком создания запросов управления данными SQL-DML.

## 2. Программа работы

1. Изучение SQL-DML.
2. Выполнение всех запросов из списка стандартных запросов. Демонстрация результатов преподавателю.
3. Получение у преподавателя и реализация SQL-запросов в соответствии с индивидуальным заданием.
4. Демонстрация результатов преподавателю.
5. Сохранение в БД выполненных запросов **SELECT** в виде представлений, запросов **INSERT**, **UPDATE** или **DELETE** — в виде ХП.

## 3. Стандартные запросы

### 3.1. Выборка данных из одной таблицы с использованием логических операций, LIKE, BETWEEN, IN

В листинге 1 представлен запрос, формирующий выборку пользователей мужского пола, родившихся до 30 декабря 2001 года, и длина логина которых находится в диапазоне между 7 и 11.

```
1 SELECT *
2 FROM "user"
3 WHERE sex = '1'
4 AND length(login) BETWEEN 7 AND 11
5 AND birthday < '2001-12-30'
```

Листинг 1: like-between-in-1.sql

Выборка, сформированная данным запросом, представлена на рис. 3.1.

	id	login	password_hash	email	birthday	sex	first_name	last_name
1	6671	Dino.Von901	336491	Dino.Von901@email.com	1967-11-08	1	Dino	Von
2	7358	Olin.Toy549	228c58a1	Olin.Toy549@email.com	1957-03-18	1	Olin	Toy

Рис. 3.1: Выборка, сформированная like-between-in-1.sql

В листинге 2 представлен запрос, формирующий выборку пользователей женского пола, логин которых начинается с «Marie.».

```
1 SELECT id, login, email, birthday, sex, first_name, last_name
2 FROM "user"
3 WHERE sex = '0'
4 AND login LIKE 'Marie.%'
```

Листинг 2: like-between-in-2.sql

Выборка, сформированная данным запросом, представлена на рис. 3.2.

	id	login	email	birthday	sex	first_name	last_name
1	4501	Marie.Kovacek14983	Marie.Kovacek14983@email.com	1995-05-29	0	Marie	Kovacek
2	17162	Marie.Leannon2561	Marie.Leannon2561@email.com	2018-08-03	0	Marie	Leannon
3	12895	Marie.Reilly2214	Marie.Reilly2214@email.com	1990-04-23	0	Marie	Reilly
4	14934	Marie.Zemlak1779	Marie.Zemlak1779@email.com	1979-10-10	0	Marie	Zemlak

Рис. 3.2: Выборка, сформированная like-between-in-2.sql

В листинге 3 представлен запрос, формирующий выборку фильмов, цена которых \$5 или \$7, при этом они не являются эпизодами сериалов, они были выпущены в октябре (не важно какого года), и их IMDB рейтинг лежит в диапазоне между 7.5 и 7.6. Поля выборки следующие: идентификатор, стоимость, возраст в годах и рейтинг IMDB.

```

1 SELECT id ,
2       price ,
3       date_part('years' , age(now() , release_date)) as years_old ,
4       imdb_rating
5 FROM movie
6 WHERE price IN ( '$5' , '$7' )
7 AND imdb_rating BETWEEN 7.5 AND 7.6
8 AND series_season_id is NULL
9 AND date_part('month' , release_date) = 10

```

Листинг 3: like-between-in-3.sql

Выборка, сформированная данным запросом, представлена на рис. 3.3.

	id	price	years_old	imdb_rating
1	435	\$7.00	26	7.560517
2	2520	\$7.00	33	7.5580893
3	2780	\$5.00	31	7.5655518
4	5028	\$5.00	5	7.5696015
5	5395	\$7.00	36	7.561081
6	7757	\$5.00	24	7.5056896

Рис. 3.3: Выборка, сформированная like-between-in-3.sql

### 3.2. Запрос с вычисляемым полем

В листинге 4 представлен запрос, формирующий выборку из 5-ти еще не закончившихся, автоматически возобновляемых подписок, длительность которых равна 30 дням. Поля у выборки следующие: идентификатор подписки, идентификатор пользователя, стоимость и число дней до завершения подписки. При этом последнее поле является вычисляемым.

```

1 SELECT id ,
2       user_id ,
3       payment ,
4       floor(extract(EPOCH FROM age(expires , now())) / 3600 / 24) as expires_in_days
5 FROM subscription
6 WHERE date_part('days' , age(expires , started)) = 30
7 AND expires > now()
8 AND autorenewable
9 LIMIT 5

```

Листинг 4: calculated-field.sql

Выборка, сформированная данным запросом, представлена на рис. 3.4.

	id	user_id	payment	expires_in_days
1	74	19439	\$5.00	25
2	89	4999	\$12.00	15
3	274	18564	\$65.00	176
4	841	2426	\$45.00	176
5	867	18852	\$5.00	16

Рис. 3.4: Выборка, сформированная `calculated-field.sql`

Значения в столбце `expires_in_days`, превосходящие 30, объясняются тем, что эти подписки еще не вступили в силу.

### 3.3. Выборка с сортировкой по нескольким полям

В листинге 5 представлен запрос, формирующий выборку из 5-ти сериалов, при этом сериалы отсортированы по возрастанию числа сезонов и по убыванию стоимости.

```
1 SELECT *
2 FROM series
3 ORDER BY seasons ASC, price DESC
4 LIMIT 5
```

Листинг 5: `sorted.sql`

Выборка, сформированная данным запросом, представлена на рис. 3.5.

	id	seasons	price
1	472	1	\$35.00
2	454	1	\$35.00
3	459	1	\$35.00
4	452	1	\$35.00
5	473	1	\$35.00

Рис. 3.5: Выборка, сформированная `sorted.sql`

### 3.4. Запрос с вычислением совокупных характеристик

В листинге 6 представлен запрос, формирующий одну строку, содержащую общее число пользователей, максимальную длину имени, минимальную длину фамилии, средний возраст пользователей, а также число пользователей мужского пола.

```
1 SELECT count(*)                                as users_count ,
2        max(length(first_name))                  as longest_firstname ,
3        min(length(last_name))                   as shortest_lastname ,
4        round(avg(date_part('years', age(now(), birthday)))) as avg_age ,
5        sum(sex::INT)                             as male_count
6 FROM "user"
```

Листинг 6: `aggregate.sql`

Выборка, сформированная данным запросом, представлена на рис. 3.6.

	users_count ÷	longest_firstname ÷	shortest_lastname ÷	avg_age ÷	male_count ÷
1	20001	11	3	49	9172

Рис. 3.6: Выборка, сформированная aggregate.sql

### 3.5. Выборка данных из связанных таблиц

В листинге 7 представлен запрос, в котором соединяются 2 таблицы — **series** (сериалы) и **series\_translation** (переводы сериалов) для формирования названия сериала с наибольшим числом сезонов и наибольшей стоимостью.

```

1 SELECT st.name,
2       max(s.seasons) as max_seasons,
3       max(s.price)   as max_price
4 FROM series s
5       JOIN series_translation st ON s.id = st.series_id
6 GROUP BY st.name
7 ORDER BY max_seasons DESC, max_price DESC
8 LIMIT 1

```

Листинг 7: join1.sql

Выборка, сформированная данным запросом, представлена на рис. 3.7.

	name ÷	max_seasons ÷	max_price ÷
1	Fear and Trembling	5	\$35.00

Рис. 3.7: Выборка, сформированная join1.sql

В листинге 8 представлен запрос, в котором соединяются 3 таблицы — **movie\_translation** (переводы фильмов), **language** (языки) и **user\_movie** (пользователи – фильмы) для определения наиболее часто приобретаемого фильма, который не является эпизодом сериала. Выборка состоит из 2-х записей, содержащих идентификатор фильма, имя фильма, локаль и частоту его покупки, для 2-х локалей: русской и английской.

```

1 SELECT mt.movie_id,
2       mt.name,
3       l.name           as locale,
4       count(um.movie_id) as frequency
5 FROM movie_translation mt
6       JOIN language l ON mt.language_id = l.id
7       JOIN user_movie um ON mt.movie_id = um.movie_id
8 WHERE mt.movie_id in (SELECT id
9                       FROM movie
10                      WHERE id = mt.movie_id
11                      AND series_season_id IS NULL)
12 GROUP BY mt.movie_id, mt.name, locale
13 ORDER BY frequency DESC
14 LIMIT 2

```

Листинг 8: join2.sql

Выборка, сформированная данным запросом, представлена на рис. 3.8.

	movie_id ÷ name ÷	locale ÷	frequency ÷
1	6285 The Man Within	en-US	7
2	6285 This Lime Tree Bower	ru-RU	7

Рис. 3.8: Выборка, сформированная join2.sql

### 3.6. Запрос с подзапросами

В листинге 9 представлен запрос, который аналогично предыдущему запросу выводит название фильма, который наиболее часто покупается пользователями, но уже с помощью вложенных подзапросов.

```
1 SELECT mt.movie_id, mt.name, l.name as locale
2 FROM movie_translation mt
3 JOIN language l ON mt.language_id = l.id
4 WHERE movie_id in (SELECT movie_id
5                     FROM user_movie
6                     WHERE movie_id in (SELECT id
7                                         FROM movie
8                                         WHERE id = movie_id
9                                         AND series_season_id IS NULL)
10                                GROUP BY movie_id
11                                ORDER BY count(movie_id) DESC
12                                LIMIT 1
13 )
```

Листинг 9: inner.sql

Выборка, сформированная данным запросом, представлена на рис. 3.9.

	movie_id	name	locale
1	6285	The Man Within	en-US
2	6285	This Lime Tree Bower	ru-RU

Рис. 3.9: Выборка, сформированная inner.sql

Выборка получилась такой же, как и при предыдущем запросе.

### 3.7. Запрос с ограничением результата группировки

В листинге 10 представлен запрос, находящий пользователей, у которых больше 42 подписок, с помощью ограничения результата группировки по идентификатору пользователя.

```
1 SELECT user_id
2 FROM subscription
3 GROUP BY user_id
4 HAVING count(user_id) > 42
```

Листинг 10: group.sql

Выборка, сформированная данным запросом, представлена на рис. 3.10.

	user_id
1	18339
2	19345
3	18411
4	10309

Рис. 3.10: Выборка, сформированная group.sql

### 3.8. Добавление записей в таблицы

В листинге 10 представлен запрос, добавляющий записи во все таблицы.

```

1 INSERT INTO language (name)
2 VALUES ('sp-Ar');
3
4 INSERT INTO "user" (login, password_hash, email, birthday, sex, first_name, last_name)
5 VALUES ('null', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ123456', 'email@email.email', '1984-11-12', '0', '
    Firstname', 'Lastname');
6
7 INSERT INTO series (seasons, price)
8 VALUES (1, 25);
9
10 INSERT INTO series_season (series_id, number)
11 VALUES (1, 1);
12
13 INSERT INTO series_translation (series_id, language_id, name, director, description)
14 VALUES (1, 3, 'Silicon Valley', 'Some Director', 'Cool series');
15
16 INSERT INTO series_season_translation (series_season_id, language_id, name, description)
17 VALUES (1, 3, 'First season', 'Very cool season');
18
19 INSERT INTO movie (price, release_date, imdb_rating, series_season_id)
20 VALUES (5, '2018-03-18', 9.0, 1);
21
22 INSERT INTO movie_translation (movie_id, language_id, name, director, description, file_url)
23 VALUES (1, 3, 'Episode 1', 'Some Director', 'First episode', 'https://url.to.video.com/file1')
24 ;
25
26 INSERT INTO user_movie (user_id, movie_id, payment)
27 VALUES (1, 1, 5);
28
29 INSERT INTO subscription (user_id, started, expires, autorenewable, payment)
30 VALUES (1, '2018-12-22', '2019-12-22', FALSE, 15);
31
32 INSERT INTO subscription_movie (subscription_id, movie_id)
33 VALUES (1, 1);
34
35 INSERT INTO subscription_series_season (subscription_id, series_season_id)
36 VALUES (1, 1);
37
38 INSERT INTO category (name, supercategory_id)
39 VALUES ('somegenre', NULL);
40
41 INSERT INTO movie_category (movie_id, category_id)
42 VALUES (1, 1);
43
44 INSERT INTO category_translation (category_id, language_id, translation)
45 VALUES (1, 3, 'somegenre');

```

Листинг 11: insert.sql

### 3.9. Изменение значений полей записей, удовлетворяющих условиям

В листинге 12 представлен запрос, который в таблице "user" изменяет логин, равный 'null', на значение 'notnull'.

```

1 UPDATE "user"
2 SET login = 'notnull'
3 WHERE login = 'null';

```

Листинг 12: update.sql

### 3.10. Удаление записей

В листинге 13 представлен запрос, который в таблице movie удаляет все записи, у которых максимальная цена или максимальный рейтинг.



```
1 DELETE
2 FROM movie
3 WHERE price = (SELECT max(price) FROM movie)
4 OR imdb_rating = (SELECT max(imdb_rating) FROM movie)
```

Листинг 13: delete.sql

В листинге 14 представлен запрос, который в таблице `movie` удаляет все записи, для которых в таблице `movie_translation` нет переводов на языки с идентификаторами 1 и 2.

```
1 DELETE
2 FROM movie
3 WHERE id IN (SELECT movie_id
4             FROM movie_translation
5             WHERE language_id NOT IN (1, 2))
```

Листинг 14: delete-inner-select.sql

## 4. Запросы в соответствии с индивидуальным заданием

### 4.1. Запрос 1

Вывести пользователей, которые смотрят сериалы больше, чем фильмы.

### 4.2. Запрос 2

Вывести сериалы, которые от сезона к сезону увеличивали аудиторию.

## 5. Сохранение в БД выполненных запросов в виде представлений и ХП

## 6. Выводы