МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и кибербезопасности

Направление 02.03.01 Математика и компьютерные науки

ОТЧЕТ
О ВЫПОЛНЕНИИ ЛАБОРАТОРНЫХ РАБОТ

по дисциплине «Методы проектирования баз данных»

Обучающийся: _____ Шклярова Ксения Алексеевна

Руководитель: _____ Попов Сергей Геннадьевич

«_____»_____20___г.

Санкт-Петербург, 2024

Содержание

Введение

В данном отчёте описан результат выполнения комплекса лабораторных работ, расширяющих функциональные возможности базы данных, реализованной для предметной области «Доставка посылки».

В ходе выполнения лабораторных работ изучены и реализованы в СУБД: представление, событийная модель (триггеры), права доступа, процедура, функция и управление транзакциями.

1 Постановка задачи

В ходе прохождения данного курса необходимо выполнить следующие лабораторные работы:

1. Создать представление, инкапсулирующее запрос.
2. Создать триггеры для таблицы, которая собирает статистические данные.
3. Создать двух пользователей с разными правами доступа.
4. Реализовать процедуру и функцию.
5. Реализовать управление транзакциями.

2 Лабораторные работы

2.1 Создание представлений

Формулировка запроса: для каждого почтового отделения посчитать число посылок, которые были в него поданы, и число посылок, которые были там приняты.

На рис. 2 представлен код создания представления.

```
DROP VIEW IF EXISTS parcel_counts;
CREATE VIEW parcel_counts AS
select post_office.post_office_id, post_office.index,
count(parcel.parcel_id) as receipts_count, t.departure_count
from post_office
left join parcel on post_office.post_office_id = parcel.destination
inner join (select post_office.post_office_id as id_, count(parcel.parcel_id) as departure_count
from post_office
left join application on post_office.post_office_id = application.post_office_id
left join parcel on application.application_id = parcel.application_id
group by id_) as t on t.id_ = post_office.post_office_id
group by post_office.post_office_id;
```

Рис. 1: Создание представления

На рис. 3 представлена «фиктивная таблица», которую создает представление.

po...	index	receipts_count	departure_count
148	330640	2	2
149	683153	2	0
150	660913	2	8
151	214357	1	1
152	177408	0	5
153	407810	1	0
154	250180	0	0
155	806107	2	0
156	495545	1	3
157	716319	1	6
158	725871	3	0
159	812821	0	0
160	397711	3	0
161	943601	1	2
162	917379	2	0
163	072377	2	0
164	343482	2	2
165	730562	0	0
166	085628	2	0
167	876991	3	0

Рис. 2: Созданное представление

Формулировка запроса, который использует представление: вывести почтовые отделения, в которых число посылок, которые были в него поданы, и число посылок, которые были там приняты, больше 5. Код данного запроса представлен на рис. 4, на рис. 5 представлен результат выполнения этого запроса.

```
select parcel_counts.index, receipts_count, departure_count, post_office.address as address
from parcel_counts inner join post_office
on parcel_counts.post_office_id = post_office.post_office_id
where receipts_count > 5 and departure_count > 5;
```

Рис. 3: Пример использования представления в запросе

index	receipts_count	departure_cou...	address
353970	6	9	г. Гагаринул. Дачнаяд. 4
023430	6	7	г. Юровскбул. Стахановскийд. 129
056723	6	6	д. Златоустпр. Огородныйд. 14
377423	7	9	ст. Кетченерыалл. Ростовскаяд. 122
322964	6	6	г. Усть-Катавнаб. Высоцкогод. 115
887079	6	7	г. Магаданпер. Литейныйд. 44
539482	6	8	с. Валаампр. Островскогод. 101
613058	6	6	с. Серафимовичнаб. Космическаяд. 127
540543	7	9	ст. Елабугабул. Абрикосовыйд. 41
952453	7	9	д. Абаканул. Смирновад. 99
030942	6	7	г. Мостовскойалл. Абрикосоваяд. 59
995832	6	7	д. Ачинскш. Вишневоед. 81

Рис. 4: Результат выполнения запроса, использующего представление

Результат попытки добавления и удаления данных в представлении представлен на рис. 6.

```
mysql> update parcel_counts set departure_count = 6 where post_office_id = 1;
ERROR 1288 (HY000): The target table parcel_counts of the UPDATE is not updatable
mysql> delete from parcel_counts where parcel_count.index = 353970;
ERROR 1288 (HY000): The target table parcel_counts of the DELETE is not updatable
mysql> insert into parcel_counts(post_office_id, parcel_counts.index, receipts_count, departure_count) values(50000, 123456, 15, 16);
ERROR 1471 (HY000): The target table parcel_counts of the INSERT is not insertable-into
```

Рис. 5: Результат изменения данных в представлении

2.2 Создание триггеров

Формулировка для создания таблицы: в таблице должны содержаться персоны и количество их отправлений.

Код создания этой таблицы представлен на рис. 7, на рис. 8 представлен код заполнения этой таблицы, на рис. 9 представлена данная таблица.

```
1 create table if not exists person_dep(
2         person_id INT NOT NULL,
3     person_surname VARCHAR(50) DEFAULT NULL,
4     person_name VARCHAR(50) DEFAULT NULL,
5     departure_count INT NOT NULL
6 );
```

Рис. 6: Создание таблицы

```
1 insert into person_dep (person_id, person_surname, person_name, departure_count)
2 select person.person_id, person.surname, person.name, count(application.application_id)
3 from person
4 right join application
5 on application.person_id = person.person_id
6 group by person.person_id;
```

Рис. 7: Заполнение таблицы

person_id	person_surname	person_name	departure_count
1	Селиверстова	Эмилия	2
2	Милица	Алексеевна	4
3	Анжела	Алексеевна	2
4	Лукина	Евпраксия	3
5	Кононова	Октябрина	1
6	Архип	Эдгардович	1
7	Казимир	Данилович	3
8	Демид	Игоревич	1
9	Беляева	Агата	3
10	Оксана	Федоровна	3
11	Элеонора	Евгеньевна	4
12	Журавлев	Кузьма	4
13	Антонина	Вячеславов...	3
14	Мартынова	Фёкла	4
15	Сидоров	Боян	2
16	Лыткин	Демьян	1
17	Кузнецов	Казимир	3
18	Гусева	Ия	3
19	Евдокия	Натановна	1
20	Колобов	Фока	4
21	Рюрик	Гурьевич	1
22	Артемьева	Лора	3
23	Валерия	Анатольевна	1
24	Евфросиния	Викторовна	1
25	Кошелева	Майя	4
26	Ян	Валентино...	2
27	Доброслав	Иосифович	4
28	Эрнест	Григорьевич	2
29	Пестов	Фома	3

Рис. 8: Созданная таблица person_dep

На рис. 10 и рис. 11 представлен код создания триггеров на добавление, удаление и обновление данных в таблицах application и person соответственно. На рис. 12 представлен код для изменения данных в таблицах application и person, которые вызывают срабатывание созданных триггеров. На рис. 13 - рис. 18 представлены результаты выполнения данных запросов.

```
DELIMITER |
create trigger tr_ins_appl
after insert on application
for each row
begin
        update person_dep set departure_count = (person_dep.departure_count + 1)
 where person_id = NEW.person_id;
end|
DELIMITER ;
DELIMITER |
create trigger tr_del_appl
after delete on application
for each row
begin
        update person_dep set departure_count = (person_dep.departure_count - 1)
 where person_id = OLD.person_id;
end|
DELIMITER ;
DELIMITER |
create trigger tr_update_appl
after update on application
for each row
begin
        update person_dep set departure_count = (person_dep.departure_count - 1)
 where person_id = OLD.person_id;
    update person_dep set departure_count = (person_dep.departure_count + 1)
```

```
27      where person_id = NEW.person_id;
28  end|
29  DELIMITER ;
```

Рис. 9: Создание триггеров для таблицы application

```
1   DELIMITER |
2   create trigger tr_ins_person_dep
3   after insert on person
4   for each row
5   begin
6           insert into person_dep values(New.person_id, New.surname, New.name, 0);
7   end|
8   DELIMITER ;
9
10  DELIMITER |
11  create trigger tr_del_person_dep
12  after delete on person
13  for each row
14  begin
15          delete from person_dep where person_id = OLD.person_id;
16  end|
17  DELIMITER ;
18
19  DELIMITER |
20  create trigger tr_update_person_dep
21  after update on person
22  for each row
23  begin
24          update person_dep set person_surname = NEW.surname  WHERE person_id =
25   NEW.person_id;
26      update person_dep set person_name = NEW.name WHERE person_id =
27      NEW.person_id;
28  end|
29  DELIMITER ;
```

Рис. 10: Создания триггеров для таблицы person

```
1   insert into application(`application_id`, `person_id`, `post_office_id`) values(12438, 1, 1);
2
3   update application set person_id = 2 where application_id = 12438;
4
5   delete from application where application_id = 12438;
6
7   insert into person(`person_id`, `surname`, `name`, `patronymic`, `phone_number`)
8   values(5001, 'Shklyarova', 'Ksenia', 'Alekseevna', '89227755320');
9
10  update person set name = 'Juliana' where person_id = 5001;
11
12  delete from person where person_id = 5001;
```

Рис. 11: Запросы для изменения данных в таблицах application и person

person_id	person_surname	person_name	departure_count
1	Селиверстова	Эмилия	3
2	Милица	Алексеевна	4
3	Анжела	Алексеевна	2
4	Лукина	Евпраксия	3
5	Кононова	Октябрина	1
6	Архип	Эдгардович	1
7	Казимир	Данилович	3

Рис. 12: Результат добавления данных в таблицу application

person_id	person_surname	person_name	departure_count
1	Селиверстова	Эмилия	2
2	Милица	Алексеевна	5
3	Анжела	Алексеевна	2
4	Лукина	Евпраксия	3
5	Кононова	Октябрина	1
6	Архип	Эдгардович	1
7	Казимир	Данилович	3

Рис. 13: Результат изменения данных в таблице application

person_id	person_surname	person_name	departure_count
1	Селиверстова	Эмилия	2
2	Милица	Алексеевна	4
3	Анжела	Алексеевна	2
4	Лукина	Евпраксия	3
5	Кононова	Октябрина	1

Рис. 14: Результат удаления данных из таблицы application

person_id	person_surname	person_name	departure_count
4993	Любовь	Робертовна	4
4994	Богдан	Харитонович	4
4995	Попова	Варвара	1
4996	Феликс	Фёдорович	1
4997	Стрелкова	Алина	1
4998	Валентина	Тимуровна	2
4999	Ульяна	Викторовна	2
5000	Марина	Наумовна	3
5001	Шклярова	Ксения	0

Рис. 15: Результат добавления данных в таблицу person

person_id	person_surname	person_name	departure_count
4995	Попова	Варвара	1
4996	Феликс	Фёдорович	1
4997	Стрелкова	Алина	1
4998	Валентина	Тимуровна	2
4999	Ульяна	Викторовна	2
5000	Марина	Наумовна	3
5001	Шклярова	Ульяна	0

Рис. 16: Результат изменения данных в таблице person

person_id	person_surname	person_name	departure_count
4993	Любовь	Робертовна	4
4994	Богдан	Харитонович	4
4995	Попова	Варвара	1
4996	Феликс	Фёдорович	1
4997	Стрелкова	Алина	1
4998	Валентина	Тимуровна	2
4999	Ульяна	Викторовна	2
5000	Марина	Наумовна	3

Рис. 17: Результат удаления данных из таблицы person

2.3 Разграничение прав доступа

Создаются два пользователя - first и second. Первому пользователю даются права на просмотр представления, а второму - на просмотр представления и редактирование таблиц, которые использовались для его создания: post_office, parcel, application. Код создания пользователей и предоставления им прав доступа представлен на рис. 19.

```
1 CREATE USER if not exists 'first'@'localhost' identified by '1111';
2 GRANT SELECT on my_db1.parcel_counts to 'first'@'localhost';
3 CREATE USER if not exists 'second'@'localhost' identified by '2222';
4 GRANT SELECT on my_db1.parcel_counts to 'second'@'localhost';
5 GRANT SELECT, DELETE, UPDATE, INSERT on my_db1.post_office to 'second'@'localhost';
6 GRANT SELECT, DELETE, UPDATE, INSERT on my_db1.parcel to 'second'@'localhost';
7 GRANT SELECT, DELETE, UPDATE, INSERT on my_db1.application to 'second'@'localhost';
```

Рис. 18: Создание аользователей и разграничение их прав доступа

В Таб. 1 представлено сравнение реакций на различные действия пользователей first и second.

Таблица 1: Сравнение прав пользователей

<table>
<tr><th>Номер запроса</th><th>Реакция пользователя first</th><th>Реакция пользователя second</th></tr>
<tr><td rowspan="3">1</td><td colspan="2">Выбираем первые 5 значений из представления parcel_counts</td></tr>
<tr><td colspan="2">select * from parcel_counts limit 5</td></tr>
<tr><td>

post_office_id	index	receipts_count	departure_count
15	878803	1	0
41	999617	1	0
50	399001	1	0
51	027441	2	1
58	373097	2	9

</td><td>

post_office_id	index	receipts_count	departure_count
15	878803	1	0
41	999617	1	0
50	399001	1	0
51	027441	2	1
58	373097	2	9

</td></tr>
<tr><td rowspan="3">2</td><td colspan="2">Выбираем первые 5 значений из таблицы post_office</td></tr>
<tr><td colspan="2">select * from post_office limit 5</td></tr>
<tr><td>

```
ERROR 1142 (42000): SELECT command denied to user
 'first'@'localhost' for table 'post_office'
```

</td><td>

post_office_id	index	address
1	326850	д. Снежинскпр. Карбышевад. 140
2	196740	ст. Ербогаченул. Надеждыд. 23
3	502098	г. Лаганьш. Строительноед. 44
4	684581	ст. Тольяттибул. Северныйд. 97
5	945515	д. Челюскиналл. Автомобилистовд.

</td></tr>
<tr><td rowspan="3">3</td><td colspan="2">Добавляем в таблицу post_office новую запись</td></tr>
<tr><td colspan="2">insert into post_office('post_office_id', 'index', 'address', 'post_office_type_id') values(50001, 123456, "Saint−Petersburg", 1)</td></tr>
<tr><td>

```
ERROR 1142 (42000): INSERT command denied to user
 'first'@'localhost' for table 'post_office'
```

</td><td>

```
Query OK, 1 row affected (0,00 sec)
```

</td></tr>
</table>

<table>
<tr><td rowspan="3"></td><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from post_office where post_office_id > 49997;</td></tr>
<tr><td>

```
ERROR 1142 (42000): SELECT command denied to user
 'first'@'localhost' for table 'post_office'
```

</td><td>

post_office_id	index	address	throughout_c...	post_office_ty...
49998	647046	п. Амурск...	146256	2
49999	169243	г. Теберд...	1296548	2
50000	052370	г. Октябр...	621210	2
50001	123456	Санкт-Пе...	0	1

</td></tr>
<tr><td rowspan="5">4</td><td colspan="2">В таблице post_office обновляем значение post_office_type_id, которому соответствует post_office_id = 50001</td></tr>
<tr><td colspan="2">update post_office set post_office_type_id = 2 where post_office_id = 50001</td></tr>
<tr><td>

```
ERROR 1142 (42000): UPDATE command denied to user
 'first'@'localhost' for table 'post_office'
```

</td><td>

```
Query OK, 1 row affected (0,00 sec)
```

</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from post_office where post_office_id > 49997;</td></tr>
<tr><td>

```
ERROR 1142 (42000): SELECT command denied to user
 'first'@'localhost' for table 'post_office'
```

</td><td>

post_office_id	index	address	throughout_c...	post_office_ty...
49998	647046	п. Амурскш. Детскоед. 15	146256	2
49999	169243	г. Тебердаш. 50 лет Поб...	1296548	2
50000	052370	г. Октябрьский (Башк.)у...	621210	2
50001	123456	Санкт-Петербург	0	2

</td></tr>
<tr><td rowspan="5">5</td><td colspan="2">Из таблицы post_office удаляем запись, которой соответствует post_office_id = 50001</td></tr>
<tr><td colspan="2">delete from post_office where post_office_id = 50001</td></tr>
<tr><td>

```
ERROR 1142 (42000): DELETE command denied to user
 'first'@'localhost' for table 'post_office'
```

</td><td>

```
Query OK, 1 row affected (0,00 sec)
```

</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:
select * from post_office where post_office_id > 49997;</td></tr>
<tr><td>

```
ERROR 1142 (42000): SELECT command denied to user
 'first'@'localhost' for table 'post_office'
```

</td><td>

post_office_id	index	address	throughout_c...	post_office_ty...
49998	647046	п. Амурскш. Детскоед. 15	146256	2
49999	169243	г. Тебердаш. 50 лет Поб...	1296548	2
50000	052370	г. Октябрьский (Башк.)у...	621210	2

</td></tr>
<tr><td rowspan="3">6</td><td colspan="2">Удаляем таблицу post_office</td></tr>
<tr><td colspan="2">drop table post_office</td></tr>
<tr><td>

```
ERROR 1142 (42000): DROP command denied to user '
first'@'localhost' for table 'post_office'
```

</td><td>

```
ERROR 1142 (42000): DROP command denied to user
 'second'@'localhost' for table 'post_office'
```

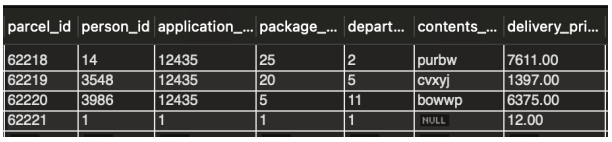
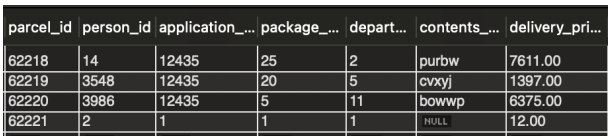
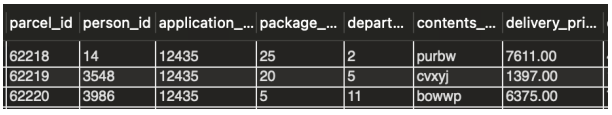
</td></tr>
<tr><td rowspan="3">7</td><td colspan="2">Выбираем первые 5 значений из таблицы parcel</td></tr>
<tr><td colspan="2">select * from parcel limit 5</td></tr>
<tr><td>

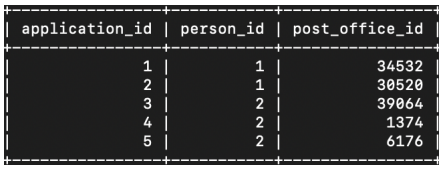
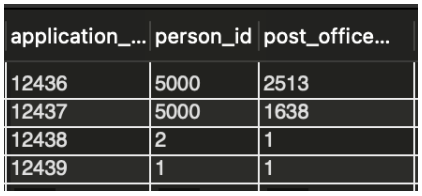
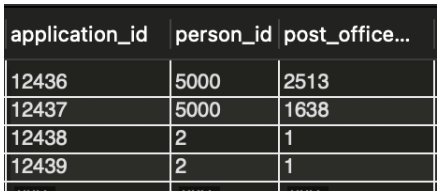
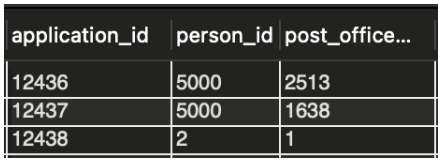
```
ERROR 1142 (42000): SELECT command denied to user
 'first'@'localhost' for table 'parcel'
```

</td><td>

parcel_id	person_id	application_...	package_ty...	depart...	contents_de...	delivery_pri...
1	4613	1	21	5	fixdw	5242.00
2	4842	2	2	14	khvbm	5485.00
3	4022	2	9	10	zvsvf	4261.00
4	3644	2	6	6	xsolp	5883.00
5	275	2	18	1	hzwdo	8269.00

</td></tr>
</table>

<table>
<tr><td rowspan="5">8</td><td colspan="2">Добавляем в таблицу parcel новую запись</td></tr>
<tr><td colspan="2">insert into parcel(`parcel_id`, `person_id`, `application_id`, `package_type_id`
`departure_type_id` , `delivery_price` , `delivery_time`, `weight` ,
`size`, `destination`, `track_number`) values(62221, 1, 1, 1, 1, 12, 12, 12, 123, 1, 987654321);</td></tr>
<tr><td>ERROR 1142 (42000): INSERT command denied to user
'first'@'localhost' for table 'parcel'</td><td>Query OK, 1 row affected (0,00 sec)</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from parcel where parcel_id > 62217;</td></tr>
<tr><td>ERROR 1142 (42000): SELECT command denied to user
'first'@'localhost' for table 'parcel'</td><td>  </td></tr>
<tr><td rowspan="5">9</td><td colspan="2">В таблице parcel обновляем значение person_id, которому
соответствует parcel_id = 62221</td></tr>
<tr><td colspan="2">update parcel set person_id = 2 where parcel_id = 62221</td></tr>
<tr><td>ERROR 1142 (42000): UPDATE command denied to user
'first'@'localhost' for table 'parcel'</td><td>Query OK, 1 row affected (0,00 sec)</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from parcel where parcel_id > 62217;</td></tr>
<tr><td>ERROR 1142 (42000): SELECT command denied to user
'first'@'localhost' for table 'parcel'</td><td>  </td></tr>
<tr><td rowspan="5">10</td><td colspan="2">Из таблицы parcel удаляем запись, которой соответствует parcel_id = 62221</td></tr>
<tr><td colspan="2">delete from parcel where parcel_id = 62221</td></tr>
<tr><td>ERROR 1142 (42000): DELETE command denied to user
'first'@'localhost' for table 'parcel'</td><td>Query OK, 1 row affected (0,00 sec)</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from parcel where parcel_id > 62217;</td></tr>
<tr><td>ERROR 1142 (42000): SELECT command denied to user
'first'@'localhost' for table 'parcel'</td><td>  </td></tr>
<tr><td rowspan="3">11</td><td colspan="2">Удаляем таблицу parcel</td></tr>
<tr><td colspan="2">drop table parcel</td></tr>
<tr><td>ERROR 1142 (42000): DROP command denied to user '
first'@'localhost' for table 'parcel'</td><td>ERROR 1142 (42000): DROP command denied to user
'second'@'localhost' for table 'parcel'</td></tr>
</table>

<table>
<tr><td rowspan="3">12</td><td colspan="2">Выбираем первые 5 значений из таблицы application</td></tr>
<tr><td colspan="2">select * from application limit 5</td></tr>
<tr><td>ERROR 1142 (42000): SELECT command denied to user 'first'@'localhost' for table 'application'</td><td> 
</td></tr>
<tr><td rowspan="5">13</td><td colspan="2">Добавляем в таблицу application новую запись</td></tr>
<tr><td colspan="2">insert into application(`application_id`, `person_id`, `post_office_id`) values(12439, 1, 1)</td></tr>
<tr><td>ERROR 1142 (42000): INSERT command denied to user 'first'@'localhost' for table 'application'</td><td>Query OK, 1 row affected (0,00 sec)</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from application where application_id > 12435;</td></tr>
<tr><td></td><td>ERROR 1142 (42000): SELECT command denied to user 'first'@'localhost' for table 'application'</td><td> 
</td></tr>
<tr><td rowspan="5">14</td><td colspan="2">В таблице application обновляем значение person_id, которому соответствует application_id = 12439</td></tr>
<tr><td colspan="2">update application set person_id = 2 where application_id = 12439</td></tr>
<tr><td>ERROR 1142 (42000): UPDATE command denied to user 'first'@'localhost' for table 'application'</td><td>Query OK, 1 row affected (0,00 sec)</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from application where application_id > 12435;</td></tr>
<tr><td></td><td>ERROR 1142 (42000): SELECT command denied to user 'first'@'localhost' for table 'application'</td><td> 
</td></tr>
<tr><td rowspan="5">15</td><td colspan="2">Из таблицы application удаляем запись, где application_id равно 12439</td></tr>
<tr><td colspan="2">delete from application where application_id = 12439</td></tr>
<tr><td>ERROR 1142 (42000): DELETE command denied to user 'first'@'localhost' for table 'application'</td><td>Query OK, 1 row affected (0,00 sec)</td></tr>
<tr><td colspan="2">Для просмотра внесенных изменений был использован запрос:</td></tr>
<tr><td colspan="2">select * from application where application_id > 12435;</td></tr>
<tr><td></td><td>ERROR 1142 (42000): SELECT command denied to user 'first'@'localhost' for table 'application'</td><td> 
</td></tr>
</table>

16	Удаляем таблицу application	
	drop table application	
	ERROR 1142 (42000): DROP command denied to user 'first'@'localhost' for table 'application'	ERROR 1142 (42000): DROP command denied to user 'second'@'localhost' for table 'application'

2.4 Создание функции

Задача: реализовать функцию, принимающую в качестве аргументов фамилию, имя и отчество и возвращающая строку формата «Фамилия.И.О.», если у человека есть отчество, и строку формата «Фамилия.И.», если нет отчества.

На рис. 20 представлен код создания функции, на рис. 21 и рис. 23 представлен код использования данной функции, а на рис. 22 и рис. 24 представлены соответствующие результаты выполнения данных запросов.

```
delimiter //
create function get_fio(surname1 VARCHAR(50), name1 VARCHAR(50), patronymic1 VARCHAR(50))
returns VARCHAR(55)
deterministic
begin
        if (patronymic1 > '')
                then return concat(left(name1, 1), '.', left(patronymic1, 1), '.', surname1);
        else
                return concat(left(name1, 1), '.', surname1);
        end if;
end//
delimiter ;
```

Рис. 19: Код создания функции

```
select `surname`, `name`, `patronymic`, get_fio(`surname`, `name`, `patronymic`)
from person where person_id > 4996;
```

Рис. 20: Пример использования функции

surname	name	patronymic	get_fio(`surname`, `name`, `patron...
Стрелкова	Алина	Романовна	А.Р.Стрелкова
Валентина	Тимуровна	Киселева	Т.К.Валентина
Ульяна	Викторовна	Новикова	В.Н.Ульяна
Марина	Наумовна	Яковлева	Н.Я.Марина
Богдан	Арсений		А.Богдан
Луговенко	Полина		П.Луговенко
Яшнова	Дарья	Михайловна	Д.М.Яшнова

Рис. 21: Результат использования функции

```
select `surname`, `name`, `patronymic`, get_fio(`surname`, `name`, `patronymic`)
from parcel where parcel_id > 403000;
```

Рис. 22: Пример использования функции в неправильном запросе

Рис. 23: Результат использования функции

2.5 Создание процедуры

Задача: реализовать процедуру, принимающую в качестве аргументов индекс почтового отделения, его адрес, пропускную способность и его тип. Если данное почтовое отделение существует, то ничего не изменяется, но если данного отделения не существует, то в таблице post_office добавляем запись с данным отделением, при этом в тип почтового отделения записываем тот, который указали в качестве входного параметра, но если данного типа не существовало, то добавляем его в таблице post_office_type. Если введенный тип почтового отделения отличается от того, который хранится в базе данных, то мы изменяем его значение, при этом выводя сообщение о том, что мы обновили значение.

На рис. 25 представлен код создания процедуры, на рис. 26 представлен код разных вызовов созданного представления, на рис. 28 - рис. 33 представлены результаты выполнения данных запросов.

```
1  delimiter //
2  create procedure new_post_office_type_(IN p_index varchar(6), IN p_address varchar(150),
3          IN p_throughout int, IN p_post_office_type varchar(45))
4  begin
5          declare p_post_office_type_id int default null;
6          declare p_post_office_id int default null;
7      declare old_post_office_id int default null;
8
9      if (p_post_office_type is null or p_post_office_type = '' or
10     p_index is null or p_index = '' or p_address is null or p_address = '') then
11                 SIGNAL SQLSTATE '23000'
12                         SET MESSAGE_TEXT = 'p_index, p_address, p_post_office_type
13     cannot be NULL';
14         end if;
15
16
17     select post_office_type_id into p_post_office_type_id
18     from post_office_type
19     where `name` = p_post_office_type;
20
21     if p_post_office_type_id is NULL then
22     insert into post_office_type (`name`) values (p_post_office_type);
23         set p_post_office_type_id = LAST_INSERT_ID();
24     end if;
25
26     select post_office_id into p_post_office_id
27     from post_office
28     where `index` = p_index and address = p_address and throughout_capacity = p_throughout;
29
```

```
30      select post_office_id into old_post_office_id
31      from post_office
32      where `index` = p_index and address = p_address and throughout_capacity =
33      p_throughout and post_office_type_id = p_post_office_type_id;
34
35      if p_post_office_id is null then
36          insert into post_office (`index`, address, throughout_capacity, post_office_type_id)
37   values (p_index, p_address, p_throughout, p_post_office_type_id);
38          set p_post_office_id = LAST_INSERT_ID();
39      elseif old_post_office_id is null then
40          update post_office set `post_office_type_id` =
41   p_post_office_type_id where post_office_id = p_post_office_id;
42          select "Тип почтового отделения был обновлен" as message;
43          end if;
44  END //
45  delimiter ;
```

Рис. 24: Код создания процедуры

```
1  call new_post_office_type_('000111', 'Санкт-Петербург', 123, 'склад');
2
3  call new_post_office_type_('000111', 'Санкт-Петербург', 123, 'Сортировочный центр');
4
5  call new_post_office_type_('000111', 'Санкт-Петербург', 123, "");
```

Рис. 25: Пример вызова процедуры

```
1  select(*) from post_office where post_office_id > 49996;
2
3  select(*) from post_office_type;
```

Рис. 26: Запросы для просмотра изменений

post_office_id	index	address	throughout_capacity	post_office_type_id
49997	024601	ст. Печенгапр. Пушкинад. 116	59471	2
49998	647046	п. Амурскш. Детскоед. 15	146256	2
49999	169243	г. Тебердаш. 50 лет Победыд. 121	1296548	2
50000	052370	г. Октябрьский (Башк.)ул. Брянскаяд. 44	621210	2
50001	000111	Санкт-Петербург	123	3

Рис. 27: Результат первого вызова процедуры для таблицы post_office

post_office_type_id	name
1	Почтовое отделение
2	Сортировочный центр
3	склад

Рис. 28: Результат первого вызова процедуры для таблицы post_office_type

message
Тип почтового отделения был обновлен

Рис. 29: Результат второго вызова процедуры

post_office_id	index	address	throughout_capacity	post_office_type_id
49997	024601	ст. Печенгапр. Пушкинад. 116	59471	2
49998	647046	п. Амурскш. Детскоед. 15	146256	2
49999	169243	г. Тебердаш. 50 лет Победыд. 121	1296548	2
50000	052370	г. Октябрьский (Башк.)ул. Брянскаяд. 44	621210	2
50001	000111	Санкт-Петербург	123	2

Рис. 30: Результат второго вызова процедуры для таблицы post_office

post_office_type_id	name
1	Почтовое отделение
2	Сортировочный центр
3	склад

Рис. 31: Результат второго вызова процедуры для таблицы post_office_type

```
ERROR 1644 (23000): p_index, p_address, p_post_office_type cannot be NULL
```

Рис. 32: Результат третьего вызова процедуры

2.6 Управление транзакциями

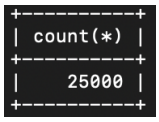
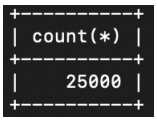
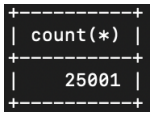
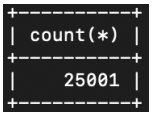
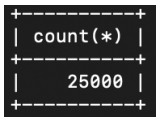
Задача: задать уровень изоляции транзакций READ UNCOMMITTED и проверить выполняет ли этот уровень защиту от «грязного» чтения.

При этом уровне изоляции транзакция в пределах текущей сессии может читать данные, которые модифицируются или удаляются другой транзакцией, но еще не зафиксированы. Этот уровень изоляции накладывает наименьшие ограничения, поскольку ядро базы данных не накладывает никаких разделяемых блокировок.

В таб. 2 представлен пример «грязного» чтения.

Таблица 2: Пример транзакций при выбранном уровне изоляции

№	Первая транзакция	Вторая транзакция
1	Установка уровня изоляции для транзакций SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED	
2	Начало транзакции 1	
	start transaction;	
	Query OK, 0 rows affected (0,00 sec)	
3	Начало транзакции 2	
		start transaction;
		Query OK, 0 rows affected (0,00 sec)

<table>
<tr><td rowspan="4">4</td><td colspan="2">Количество почтовых отделений с типом почтового отделения 2
до внесения изменений транзакциями</td></tr>
<tr><td>select count(*) post_office where
post_office_type_id = 2</td><td>select count(*) post_office where
post_office_type_id = 2</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td rowspan="3">5</td><td colspan="2">Изменение данных в таблице post_office во второй транзакции
без сохранения результата</td></tr>
<tr><td></td><td>update post_office set
post_office_type_id = 2
where post_office_id = 4657</td></tr>
<tr><td></td><td>  </td></tr>
<tr><td rowspan="4">6</td><td colspan="2">Повторный просмотр количества почтовых отделений
с типом почтового отделения 2 после внесения изменений</td></tr>
<tr><td>select count(*) post_office where
post_office_type_id = 2</td><td>select count(*) post_office where
post_office_type_id = 2</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td rowspan="3">7</td><td colspan="2">Отмена изменений во второй транзакции, её завершение</td></tr>
<tr><td></td><td>rollback</td></tr>
<tr><td></td><td>  </td></tr>
<tr><td rowspan="4">8</td><td colspan="2">Повторный просмотр количества почтовых отделений
с типом почтового отделения 2 после завершения второй транзакции</td></tr>
<tr><td>select count(*) post_office where
post_office_type_id = 2</td><td></td></tr>
<tr><td>  </td><td></td></tr>
<tr><td rowspan="3">9</td><td colspan="2">Сохранение результатов первой транзакции, её завершение</td></tr>
<tr><td>commit</td><td></td></tr>
<tr><td>  </td><td></td></tr>
</table>

<table>
<tr><td rowspan="3">10</td><td colspan="2">Повторный просмотр количества почтовых отделений
с типом почтового отделения 2 после завершения первой транзакции</td></tr>
<tr><td><code>select count(*) post_office where
post_office_type_id = 2</code></td><td></td></tr>
<tr><td><pre>+----------+
| count(*) |
+----------+
| 25000 |
+----------+</pre></td><td></td></tr>
</table>

Между шагами 5 - 7 в первой транзакции возникает «грязное чтение»: в 5 шаге во второй транзакции происходит изменение данных, но без сохранения, далее в 6 шаге первая транзакция читает те данные, которые не были сохранены во второй транзакции и это приводит к «грязному чтению», на шаге 7 происходит отмена изменений во второй транзакции и её завершение, далее в шаге 8 в первой транзакции происходит новое чтение данных и теперь уже с другим результатом.

Заключение

В ходе выполнения пяти лабораторных работ были получены различные навыки и знания.

1. Реализовано представление на основе запроса: для каждого почтового отделения посчитать число посылок, которые были в него поданы, и число посылок, которые были там приняты. Для этого было задействовано 3 таблицы, в результате представление содержит 50000 записей. Сформулирован запрос, использующий представление, а также показан результат попытки изменения, добавления и удаления данных в созданное представление.

2. Создано 6 триггеров на добавление, удаление и обновление данных в таблицах person и application, на основе которых была создана таблица для хранения статистических данных: количество отправлений для каждой персоны. Данная таблица содержит 5003 записи. Для проверки триггеров было использовано 6 запросов.

3. Созданы два пользователя с разными правами. Первый может просматривать представление, второй - имеет полный доступ к таблицам, использованным в создании представления. Была сформирована таблица сравнения прав доступа двух пользователей для различных операций: выборка, удаление, вставка, обновление данных. Для этого было использовано 16 запросов.

4. Реализована функция, принимающая на вход принимает фамилию, имя и отчество персоны, а возвращает конкатенацию инициалов и фамилии персоны в формате «И.О. Фамилия». При отсутствии отчества результат возвращается в формате «И. Фамилия». Для проверки функции было выполнено 2 запроса.

5. Создана процедура на основе 2 таблиц post_office и post_office_type, принимающая в качестве аргументов индекс почтового отделения, его адрес, пропускную способность и его тип. При вызове процедуры, если данные в таблицах post_office и post_office_type существовали, то ничего не изменяем, если же в таблице post_office хранится другой тип почтового отделения, то изменяем его на новый, который содержится в вызове, и выводим сообщение об изменении, если таких данных вообще не существовало, то создаем новые записи в таблицах. Для проверки работы данной процедуры было выполнено 3 запроса.

6. Реализовано управление транзакциями. Был задан уровень изоляции транзакций READ UNCOMMITTED и проверено, что данный уровень не выполняет защиту от «грязного» чтения.