

Министерство образования и науки РФ
Санкт-Петербургский Политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта
Направление 02.03.01 «Математика и компьютерные науки»
Дисциплина «Методы проектирования баз данных»

Отчёт по лабораторным работам

Студент:

Башарина Е.А., гр. 3530201/90101

Преподаватель:

Попов С.Г.

Санкт-Петербург — 2023

Содержание

Введение	2
1 Постановка задачи	3
2 Схема базы данных	4
2.1 View	6
2.2 Запрос к view	7
3 Триггер	8
4 Процедуры и функции	13
4.1 Процедура	13
4.2 Функция	14
5 Создание двух пользователей	16
6 Транзакции	21
6.1 Lost update	22
6.2 Dirty read	24
6.3 Non-repeatable read	25
6.4 Phantom read	27
Заключение	29

Введение

Данный отчёт содержит описание процесса выполнения лабораторных работ в рамках курса «Методы проектирования баз данных».

Лабораторные работы были направлены на изучение и расширение функциональных возможностей базы данных птиц Ленинградской области, которая была разработана в рамках прохождения курса «Теоретические основы баз данных» в весеннем 2022 года.

Лабораторные работы №1-4 включали в себя изучение и реализацию в СУБД **представления, пользовательских функций и хранимых процедур**, а также процедур на основе триггеров и исследования возможностей и **методов администрирования прав доступа** пользователей базы данных в многопользовательском режиме. В лабораторной работе №5 требовалось изучить применение **транзакций** для синхронизации операций чтения/записи разных пользователей при одновременной работе в базе данных.

1 Постановка задачи

В рамках выполнения лабораторных работ необходимо:

- создать View, сделать к ней запрос и продемонстрировать результаты этого запроса;
- создать триггер и продемонстрировать его работу;
- создать пользовательскую функцию и процедуру, а затем продемонстрировать результаты их вызова;
- создать двух пользователей с разным набором прав, продемонстрировать их возможности. Первый пользователь должен иметь право на чтение view из первой лабораторной работы. Второй пользователь должен иметь право на просмотр view из первой лабораторной работы и на модификацию таблиц, связанных с этой view;
- выбрать уровень изоляции транзакции. Для этого уровня при помощи двух пользователей показать выполнение или невыполнение феноменов параллельного доступа.

2 Схема базы данных

Для данной работы использовалась база данных птиц, разработанная в рамках курсового проекта предыдущего семестра.

База данных реализована на технологии MySQL 8, сервер с базой данных был развернут локально. На рис.1 приведена схема базы птиц на английском языке.

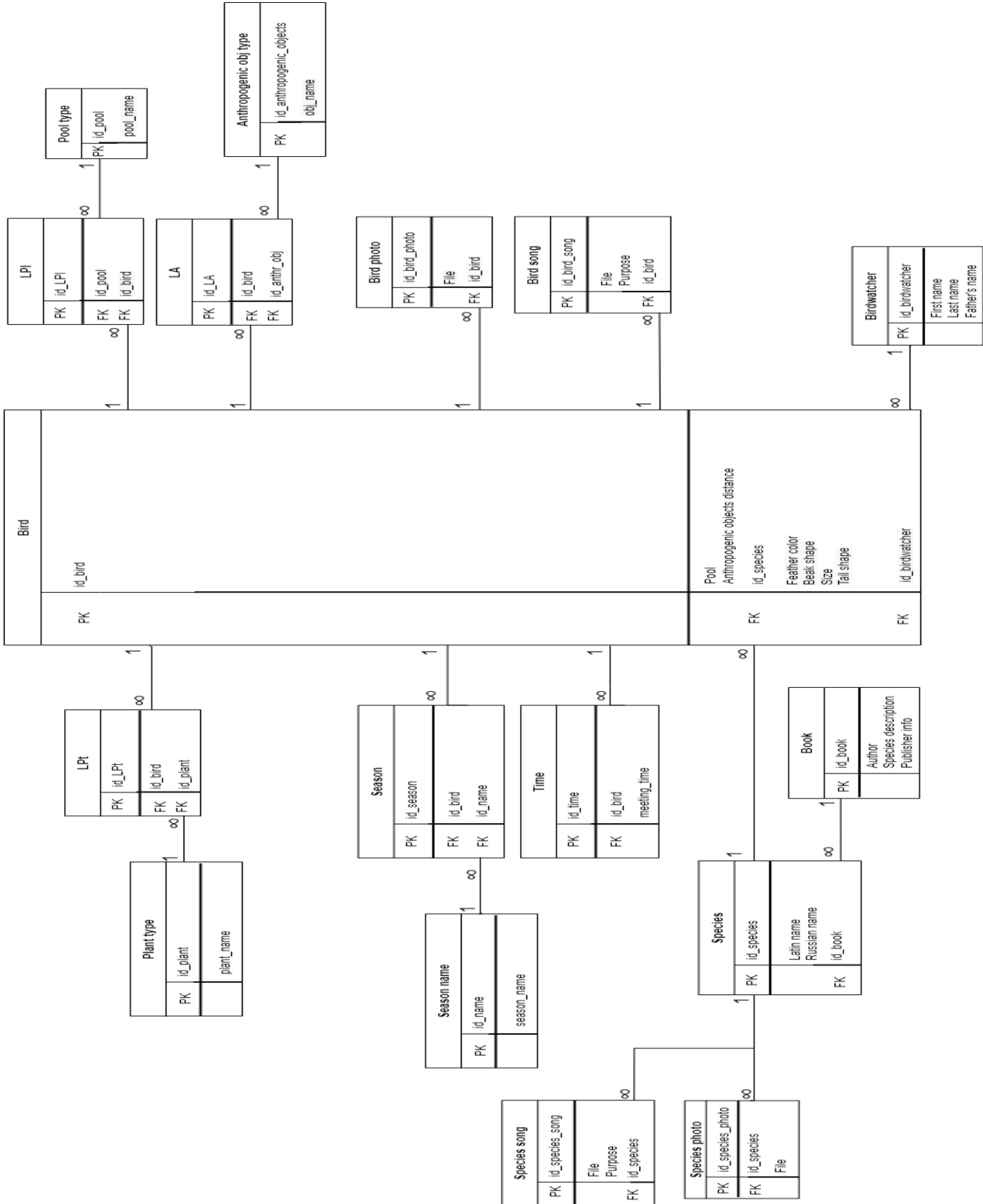


Рис. 1: Схема базы данных на английском языке

2.1 View

Задание: создать view, которая показывает количество наблюдений птицы определённого вида в день, месяц и год.

Реализованный запрос:

```
create view species_count as
select latin_name as 'Latin name',
ifnull(count_year, 0) as 'Count year',
ifnull(count_month, 0) as 'Count month',
ifnull(count_day, 0) as 'Count day'
from species
left join
(select species.latin_name as _species,
count(bird.id_bird) as count_year
from species
inner join bird on bird.id_species = species.id_species
inner join meeting_time on bird.id_bird = meeting_time.id_bird
where time_name BETWEEN '2022-01-01 00:00:00' AND '2022-12-31 23:59:59'
group by species.latin_name) as table_year
on species.latin_name = table_year._species
left join
(select species.latin_name as _species,
count(bird.id_bird) as count_month
from species
inner join bird on bird.id_species = species.id_species
inner join meeting_time on bird.id_bird = meeting_time.id_bird
where time_name BETWEEN '2022-01-01 00:00:00' AND '2022-01-31 23:59:59'
group by species.latin_name) as table_month
on species.latin_name = table_month._species
left join
(select species.latin_name as _species,
count(bird.id_bird) as count_day
from species
inner join bird on bird.id_species = species.id_species
inner join meeting_time on bird.id_bird = meeting_time.id_bird
where time_name BETWEEN '2022-01-01 00:00:00' AND '2022-01-01 23:59:59'
group by species.latin_name) as table_day
on species.latin_name = table_day._species
```

Результат запроса select для данной view представлен на рисунке 2

	Latin name	Count year	Count month	Count day
	Itakxamnibh yccfucykph	5	1	0
	Ctlqbwuivip nkiifuqski	0	0	0
	Fknqgnmkud zuwvyiuchv	9	1	0
	Zxhckuagbtb ycmhtkpgwz	12	1	1
	Kupkmuefju ematijxmp	14	1	0
	Ymlunylcxi phvfnnkyls	11	0	0
	Zkvysnhodr bviafpwmfy	3	0	0
	Sgdfnqbtjuk luqjljbkaq	5	0	0
	Cndoxisvdwr nehuxpijs	16	1	0
	Jotjtnicrgn vgbhykodvb	16	0	0
	Hxsbayfwuhz wzwcnmkvb	10	1	0
	Opaepodmnco lkgcyolqb	11	0	0

Рис. 2: Результат запроса select

2.2 Запрос к view

Формулировка запроса: создать любой запрос, который использует полученную view в качестве таблицы (реализует join с ней).

Реализованный запрос:

Выводит для каждого вида его русское название, а также количество встреч за год.

```

use birds;
select russian_name, count_year
from species
join species_count
order by russian_name;

```

Результат данного запроса представлен на рисунке 3.

	russian_name	count_year
►	Азюкърдззгю	9
	Азюкърдззгю	7
	Азюкърдззгю	12
	Азюкърдззгю	5
	Азюкърдззгю	3
	Азюкърдззгю	0
	Азюкърдззгю	2
	Азюкърдззгю	4
	Азюкърдззгю	3
	Азюкърдззгю	1

Рис. 3: Результат запроса к view

3 Триггер

Триггер — это механизм, который вызывается, когда в указанной таблице происходит определенное действие.

Триггер является указанием, что база данных должна автоматически выполнить заданную функцию, всякий раз, когда выполнен определенный тип операции. Триггеры можно использовать с таблицами, с представлениями и с внешними таблицами. Для обычных и сторонних таблиц можно определять триггеры, которые будут срабатывать до или после любой из команд INSERT, UPDATE, DELETE, либо один раз для каждой модифицируемой строки, либо один раз для оператора SQL.

Триггерная функция должна быть создана до триггера. Она должна быть объявлена без аргументов и возвращать тип trigger. Триггерная функция получает данные на вход посредством специально переданной структуры TriggerData, а не в форме обычных аргументов. После создания триггерной функции создается триггер с помощью CREATE TRIGGER. Одна и та же триггерная функция может быть использована для нескольких триггеров.

Задание:

1. создать таблицу на основе полученной view;
2. при добавлении новой записи в таблицу времени встречи птиц увеличивать значение количества встреч соответствующего вида в год, месяц и день.

При этом, если добавляется птица, относящаяся к новому виду (тому, который не был встречен ранее), этот вид должен быть также добавлен в таблицу, созданную на основе view.

Реализованные запросы

1. Создание таблицы на основе view «species_count».

```
CREATE TABLE if not exists speciescnt (  
  primary key auto_increment(id_species) ) AS  
  SELECT  
    id_species ,  
    species .latin_name ,  
    count_year ,  
    count_month ,  
    count_day  
  FROM species_count  
  join species on species .latin_name = species_count .latin_name ;
```

Созданная таблица представлена на рисунке 4.

	id_species	latin_name	count_year	count_month	count_day
▶	1	Photcgmzieg vsucogzsaq	5	0	0
	2	Xoocbbsawmc ztvpfqwwak	0	0	0
	3	Mkmehwgudud cxgxxhftnd	10	1	0
	4	Dmikxsrvea ejicpxjrb	4	0	0
	5	Wfikuyfkgb mwxdlsvgrv	8	0	0
	6	Johfujcpnd hdlpruits	6	1	0
	7	Uzwbauvat sorynphaho	13	2	0

Рис. 4: Таблица, созданная на основе view

2. Создание триггера.

```

DELIMITER $$
create trigger add_bird
after insert on meeting_time
for each row
begin
    insert ignore into speciescnt(
        speciescnt.id_species ,
        speciescnt.latin_name ,
        speciescnt.count_year ,
        speciescnt.count_month ,
        speciescnt.count_day
    )
    SELECT
        species.id_species ,
        species.latin_name ,
        0, 0, 0
    FROM species where id_species = (
        select id_species
        from bird
        where bird.id_bird = new.id_bird
    );

    if
        new.time_name
        BETWEEN '2022-01-01 00:00:00 ' AND '2022-12-31 23:59:59 ' then
        update speciescnt
        set count_year = count_year + 1
        where speciescnt.id_species = (
            select id_species
            from bird
            where bird.id_bird = new.id_bird
        );
    end if;

    if
        new.time_name
        BETWEEN '2022-01-01 00:00:00 ' AND '2022-01-31 23:59:59 ' then
        update speciescnt
        set count_month = count_month + 1
        where speciescnt.id_species = (

```

```

        select id_species
        from bird
        where bird.id_bird = new.id_bird
    );
end if;

if
new.time_name
BETWEEN '2022-01-01 00:00:00 ' AND '2022-01-01 23:59:59 ' then
    update speciescnt
    set count_day = count_day + 1
    where speciescnt.id_species = (select id_species from bird where bird.id_bird = new.id_bird);
end if;
end;

```

Для того, чтобы проверить работу триггера, выполнить следующие шаги:

1. добавить в таблицу видов новую строку

```

insert into species (
    latin_name,
    russian_name,
    area,
    id_book
) values (
    'Parus major',
    'Большая синица',
    'Ареал',
    1
);

```

2. добавить птицу этого вида в таблицу птиц

```

insert into bird (
    feath_color,
    beak_shape,
    size,
    tail_shape,
    pool,
    anthr_obj_distance,
    id_species,
    id_birdwatcher)
values (
    'синий',
    'загнутый вниз',
    'мелкий',
    'короткий',
    1,
    'рядом с населенным пунктом',
    101,
    8
);

```

3. проверить, что изначально в таблице speciescnt отсутствует запись об этом виде (рис. 5)

```
select * from speciescnt where id_species=101;
```

	id_species	latin_name	count_year	count_month	count_day
*	NULL	NULL	NULL	NULL	NULL

Рис. 5: Запись для вида с ID=101 отсутствует

4. добавить запись с этим видом в таблицу meeting_time

```
insert into meeting_time (  
    time_name,  
    id_bird  
)  
values(  
    '2022-01-01 00:05:00 ',  
    100007  
);
```

5. проверить, что в таблице speciescnt появилась запись об этом виде (рис. 6)

```
select * from speciescnt where id_species=101;
```

	id_species	latin_name	count_year	count_month	count_day
▶	101	Parus major	1	1	1
*	NULL	NULL	NULL	NULL	NULL

Рис. 6: Запись для вида с ID=101

Для проверки корректного увеличения счётчиков был написан следующий запрос.

```
insert into meeting_time (  
    time_name,  
    id_bird  
)  
values(  
    '2022-05-01 00:05:00 ',  
    100007  
);
```

Запись, добавленная по этому запросу, удовлетворяет условию года, однако не удовлетворяет условиям месяца дня. Следовательно, первый счётчик должен увеличиться на единицу, а два других – остаться на значении «1». На рис. 7 представлен результат выполнения этого запроса.

	id_species	latin_name	count_year	count_month	count_day
▶	101	Parus major	2	1	1
✱	NULL	NULL	NULL	NULL	NULL

Рис. 7: Вид с ID=101 после добавления записи

4 Процедуры и функции

4.1 Процедура

Задание: написать процедуру, которая по заданным имени, фамилии и отчеству бёрдвотчера выводит в алфавитном порядке названия всех видов птиц, которых человек с таким именем встретил летом.

Реализованная процедура:

```
create procedure species_for_person(
  fname varchar(30), lname varchar(30), ftname varchar(30)
)
  select birdwatcher.id_birdwatcher, species.latin_name as 'Latin name',
  species.russian_name as 'Russian name'
  from bird
  inner join species on bird.id_species=species.id_species
  inner join birdwatcher on bird.id_birdwatcher=birdwatcher.id_birdwatcher
  inner join season on bird.id_bird=season.id_bird
  inner join season_type on season.id_name=season_type.id_name
  where birdwatcher.last_name = lname
  and birdwatcher.first_name = fname
  and birdwatcher.fathers_name = ftname
  and season_type.season_name = 'лето'
  group by species.latin_name
  order by species.latin_name;
```

Вызов реализованной процедуры осуществляется следующим образом:

```
call species_for_person('Леонид', 'Семёнов', 'Борисович')
```

Результат работы процедуры представлен на рисунке 8

	id_birdwatcher	Latin name	Russian name
▶	1	Ajofpsybfil ysnluhejfr	Неулижгцфэр
	1	Argmxcdftex swzxllzmbw	Фйтъщыдхтжи
	1	Awwijybxuug yyqdlezhfc	Инсшжцркнбл
	1	Beyiptpfcqm jjakchdpqe	Ичжфаъсойир
	1	Bhaargigcta pzqjeyeoba	Ъакущфсеовщ
	1	Bxjunpdwhnk ikgobppsqx	Имаейкръылс
	1	Cjvhxvowixj igigdsbyjx	Дхшштазеаъв
	1	Cndoxisvdwr nehuxpijzs	Трвшфютшюпс
	1	Cprisvywcom mlderzsqhi	Йкльытйжовыр
	1	Crbbuyurxhf olsbupipwe	ьвфрмлийфпят
	1	Ctlqbwiuvip nkiifuqski	Ятылвзпкшян
	1	Cuorskvnpijd znunfimsac	Айизрагщйаз
	1	Cwnpddfasdq cunjswuwlw	Тхюгхуичзщю
	1	Daeltuylwbf cfyhphixbj	Шолаыщщячех
	1	Dmikxsrvea ejicpxjrbc	Ижндъгощпи
	1	Dqwhngpaatj bstmltjqfs	Йрхчлшйщудм
	1	Ecdpmojbwbe flqdmadyrc	Укефшыуефэб

Рис. 8: Результат работы процедуры

4.2 Функция

Задание: написать функцию, которая принимает на вход название сезона и возвращает число птиц, встреченных в этом сезоне за последний год.

Реализованная функция:

```

delimiter //
create function count_birds(ssn varchar(30))
returns int
reads SQL data
deterministic
begin
declare numb int default 0;
select count(*) into numb
from (
select bird.id_bird
from bird
inner join season on bird.id_bird=season.id_bird
inner join season_type on season.id_name=season_type.id_name
inner join meeting_time on meeting_time.id_bird=bird.id_bird
where season_type.season_name = ssn
and year(meeting_time.time_name) = year(curdate())
) table_proc;
return numb;
end //
delimiter ;

```

Вызов реализованной функции осуществляется следующим образом:

```
select count_birds('лето')
```

Результат работы функции представлен на рисунке 9.

	<code>count_birds(лето)</code>
▶	180

Рис. 9: Результат работы функции

5 Создание двух пользователей

Задание: создать двух пользователей с разным набором прав, продемонстрировать их возможности. Первый пользователь должен иметь только права на чтение ранее созданного представления. Второй пользователь должен иметь права на просмотр представления и на модификацию таблиц, связанных с этим представлением.

Для начала создаётся пользователь Limacina, который имеет только права на чтение view.

```
create user 'Limacina'@'localhost' identified by 'qwerty';  
  
grant select on birds.species_count to 'Limacina'@'localhost';
```

Далее создаётся новый пользователь Katrin, который имеет права на чтение view, а также на модификацию таблиц, связанных с ней.

```
create user 'Katrin'@'localhost' identified by '12345';  
  
grant all privileges on birds.species_count to 'Katrin'@'localhost';  
grant all privileges on birds.species to 'Katrin'@'localhost';  
grant all privileges on birds.bird to 'Katrin'@'localhost';  
grant all privileges on birds.meeting_time to 'Katrin'@'localhost';
```

Для проверки наличия доступа к view используется следующий запрос:

```
select * from birds.species_count;
```

Как от имени пользователя Limacina (рис.10), так и от имени пользователя Katrin (рис.11) результат одинаковый, поскольку оба пользователя имеют права на просмотр данной таблицы.

```

C:\Users\Admin>mysql -u Limacina -p
Enter password: ***
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from birds.species_count;

```

latin_name	count_year	count_month	count_day
Photcgmzieg vsucogzsaq	5	0	0
Xoocbbsawmc ztvpfqwwak	0	0	0
Mkmehwgudud cxgxxhftnd	10	1	0
Dmikxsrbea ejicpxjrbc	4	0	0
Wfikuyfkgbg mwxdlsvgrv	8	0	0
Johfujcpwnd hdlpruitls	6	1	0
Uzwbaoonuat sorynphaho	13	2	0
Fggmxjmtuk zzipyompz	12	1	0
Qckytjlbvvl wastucbklp	14	2	0
Srkfakgwanx ribqvzmyox	7	1	0
Vckncvuvhjg njazirutjw	4	1	0
Gnkwilyrynr cuaupzpzwh	8	0	0
Nicatwupbsw iudjfkvbbr	2	0	0
Fiudlnhgqlr jnyoymrgsf	12	1	0
Bxjunpdwhnk ikgobppsqx	4	0	0
Uyibwwncrfl ocdzsojctp	4	1	0
Jrmmoivhgoz lfhsqlnaro	8	0	0
Emotqvkinkp mwjbqxushb	11	1	0
Lrsovrvsbpi bgjyjopqab	13	3	0
Poeitufqpju ptvvyhpcsq	15	1	0
Otkjcsyzspc zxlavmxow	2	0	0
Efvpbamiydp lfslcaybzm	1	0	0
Fxmprnrpuqu lfbvaucjby	13	1	0
Argmxclftex swzxllzmbw	3	0	0
Wblmmbnhykh wcukjnazwh	4	0	0
Jltrphfiwba shqvijbzni	1	0	0
Rsczqdcwgev mhpqxedhuo	2	0	0
Uopbhrcjcxw sieoorhafx	11	0	0

Рис. 10: Результат выполнения запроса select для пользователя Limacina

```

C:\Users\Admin>mysql -u Katrin -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from birds.species_count;
+-----+-----+-----+-----+
| latin_name | count_year | count_month | count_day |
+-----+-----+-----+-----+
| Photcgmzieg vsucogzsaq | 5 | 0 | 0 |
| Xooobbsawmc ztvpfqwwak | 0 | 0 | 0 |
| Mkmehwgudud cxgxxhftnd | 10 | 1 | 0 |
| Dmikxsrvbea ejicpxjrbc | 4 | 0 | 0 |
| Wfikuyfkbgb mwxdlsvgrv | 8 | 0 | 0 |
| Johfujcpxnd hdlpruitls | 6 | 1 | 0 |
| Uzwbaonuvat sorynphaho | 13 | 2 | 0 |
| Fggmxjmxtuk zzipiyompz | 12 | 1 | 0 |
| Qckytjlmbvl wastucbklp | 14 | 2 | 0 |
| Srkfakgwanx ribqvzmyox | 7 | 1 | 0 |
| Vekncvuvhjg njazirutjw | 4 | 1 | 0 |
| Gnkwilyrynr cuaupzpzw | 8 | 0 | 0 |
| Nicatwupbsw iudjfkvbor | 2 | 0 | 0 |
| Fiudlnhgqlr jnyoymrgsf | 12 | 1 | 0 |
| Bxjunpdwhnk ikgobppsqx | 4 | 0 | 0 |
| Uyibwwnrcrfl ocdzsojctp | 4 | 1 | 0 |
| Jrmmoivhgoz lfhsqlnaro | 8 | 0 | 0 |
| Emotqvkinkp mwjbqxushb | 11 | 1 | 0 |
| Lrsovrvsbpi bgjyjopqab | 13 | 3 | 0 |
| Poeitufqpju ptvwyhpcsq | 15 | 1 | 0 |
| Otkjcsyzspc zxlavvmxow | 2 | 0 | 0 |
| Efvpbamiydp lfslcaybzm | 1 | 0 | 0 |
| Fxmprrnrpuqu lfbvaucjby | 13 | 1 | 0 |
| Argmxclftex swzxllzmbw | 3 | 0 | 0 |
| Wblmmbnhykh wcukjnazwh | 4 | 0 | 0 |
| Jltrphfiwba shqvijbzni | 1 | 0 | 0 |
| Rsczodcwev mhpoxedhuo | 2 | 0 | 0 |

```

Рис. 11: Результат выполнения запроса select для пользователя Katrin

При попытке модификации таблицы, связанной с view, от имени пользователя Limacina, происходит ошибка, поскольку у этого пользователя не хватает прав.

```
mysql> insert into birds.meeting_time (time_name, id_bird) values ('2022-05-01 00:05:00', 100007);  
ERROR 1142 (42000): INSERT command denied to user 'Limacina'@'localhost' for table 'meeting_time'  
mysql>
```

Рис. 12: Результат выполнения модификации таблицы, связанной с view, от имени Limacina

Попытка модификации таблицы, связанной с view, от имени пользователя Katrin, оказывается успешной, поскольку у этого пользователя хватает прав.

```
mysql> insert into birds.meeting_time (time_name, id_bird) values ('2022-05-01 00:05:00', 100007);  
Query OK, 1 row affected (0.01 sec)  
  
mysql>
```

Рис. 13: Результат выполнения модификации таблицы, связанной с view, от имени Katrin

Изменение любой другой таблицы базы данных (любой, не связанной с view) не происходит ни для одного из пользователей:

```
mysql> insert into birds.pool_type (id_pool, pooltype) values (1, 'пека');  
ERROR 1142 (42000): INSERT command denied to user 'Limacina'@'localhost' for table 'pool_type'  
mysql>  
  
mysql> insert into birds.pool_type (id_pool, pooltype) values (1, 'пека');  
ERROR 1142 (42000): INSERT command denied to user 'Katrin'@'localhost' for table 'pool_type'  
mysql>
```

Рис. 14: Результат выполнения модификации таблицы для обоих пользователей

В таблице ниже представлены реакции на попытки совершения перечисленных запросов от имени каждого из пользователей.

Действие	Код	Реакция Limacina	Реакция Katrin
Создание Limacina	create user 'Limacina'@'localhost' identified by '123'; grant select on birds.species_count to 'Limacina'@'localhost';}	Query Ok, 0 rows affected	-
Создание Katrin	create user 'Katrin'@'localhost' identified by '1317'; grant all privileges on birds.species_count to 'Katrin'@'localhost'; grant all privileges on birds.species to 'Katrin'@'localhost'; grant all privileges on birds.bird to 'Katrin'@'localhost'; grant all privileges on birds.meeting_time to 'Katrin'@'localhost';	-	Query Ok, 0 rows affected
Просмотреть view	select * from birds.species_count;	100 rows in set	100 rows in set
Редактировать таблицу, связанную с view	insert into birds.meeting_time (time_name, id_bird) values ('2022-05-01 00:05:00', 100007);	ERROR 1142 (42000): Insert comand denied to user 'Limacina'	Query Ok, 1 rows affected
Редактировать таблицу, не связанную с view	insert into birds.pool_type (id_pool, pooltype) values (1, 'пека')	ERROR 1142 (42000): Insert comand denied to user 'Limacina'	ERROR 1142 (42000): Insert comand denied to user 'Katrin'

6 Транзакции

Транзакция — это последовательное выполнение операций чтения и записи. Окончанием транзакции может быть либо сохранение изменений (фиксация, commit), либо отмена изменений (откат, rollback). Транзакция — это несколько запросов, которые трактуются как единый запрос.

Свойства транзакций:

- *атомарность* — транзакция либо выполняется полностью, либо не выполняется вовсе;
- *согласованность* — при завершении транзакции не должны быть нарушены ограничения, накладываемые на данные (например, constraints в БД). Согласованность подразумевает, что система будет переведена из одного корректного состояния в другое корректное;
- *изолированность* — параллельно выполняемые транзакции не должны влиять друг на друга, например, менять данные, которые использует другая транзакция. Результат выполнения параллельных транзакций должен быть таким, как если бы транзакция выполнялась последовательно;
- *устойчивость* — после фиксации изменения не должны быть утеряны.

Уровни изоляции:

- Read Committed;
- Read Uncommitted;
- Repeatable Read;
- Serializable.

Феномены параллельного доступа с использованием транзакций:

- **lost update** — при одновременном изменении одного блока данных разными транзакциями теряются все изменения, кроме последнего;
- **dirty read** — чтение данных, добавленных или измененных транзакцией, которая впоследствии не подтвердится (откатится);
- **non-repeatable read** — при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются измененными;
- **phantom read** — одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям. Другая транзакция в интервалах между этими выборками добавляет строки или изменяет столбцы некоторых строк, используемых в критериях выборки первой транзакции, и успешно заканчивается. В результате получится, что одни и те же выборки в первой транзакции дают разные множества строк.

Задание: необходимо настроить уровень изоляции на READ COMMITTED и проверить работу феноменов при работе с таблицами.

Исходный уровень изоляции: REPEATABLE-READ.

Реализованный запрос для изменения уровня изоляции:

```
set session transaction isolation level read committed;  
select @@transaction_ISOLATION;
```

Новый уровень изоляции: READ-COMMITTED. Для того, чтобы проверить наличие

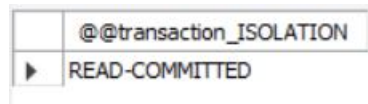


Рис. 15: Текущий уровень изоляции

феноменов, были созданы два пользователя:

- root;
- newguy.

При этом пользователю newguy предоставлен доступ для модификации всех таблиц в базе данных.

6.1 Lost update

Для попытки обнаружения феномена «lost update» были произведены следующие шаги.

1. Пользователь root выполняет запрос.

```
@root  
select id_bird ,  
       feath_color ,  
       beak_shape ,  
       size ,  
       tail_shape ,  
       pool ,  
       anthr_obj_distance  
  
from bird  
  
where id_bird = 100;
```

Результат выполнения запроса представлен на рис.16.

	id_bird	feath_color	beak_shape	size	tail_shape	pool	anthr_obj_distance
►	100	красный	затупенный	мелкий	овальный	1	рядом с населенным пунктом
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 16: Результат выполнения запроса

- Пользователь root начинает транзакцию.

```
@root
start transaction;
Query OK, 0 rows affected (0,00 sec)
```

- Пользователь newguy начинает транзакцию.

```
@newguy
start transaction;
Query OK, 0 rows affected (0,00 sec)
```

- Пользователь root обновляет запись с id=100 в таблице bird, устанавливая размер как «крупный».

```
@root
update birds.bird
set size = 'крупный'
where id_bird = 100
```

- Пользователь newguy пытается выполнить обновление этой же записи в таблице bird.

```
@newguy
update birds.bird
set size = 'мелкий'
where id_bird = 100
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

- Выполнение переходит в ожидание, далее заканчивается ошибкой.

```
@root
commit;
Query OK, 0 rows affected (0,00 sec)
```

```
@newguy
commit;
Query OK, 0 rows affected (0,00 sec)
```

- После завершения всех транзакций пользователь root выводит ту же строку из таблицы bird. В полученной записи стоимость увеличилась только на 100, второе изменение не было учтено и потерялось:

	id_bird	feath_color	beak_shape	size	tail_shape	pool	anthr_obj_distance
▶	100	красный	затупленный	крупный	овальный	1	рядом с населенным пунктом
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 17: Результат выполнения запроса

Результат: феномен Lost update произошел, так как второе обновление не сработало.

6.2 Dirty read

Для попытки обнаружения феномена «Dirty read» были произведены следующие шаги.

1. Пользователь root выполняет запрос.

```
@root
select id_species, latin_name
from species
where id_species = 5;
```

Результат выполнения запроса представлен на рис. 18.

	id_species	latin_name
▶	5	Wfikuyfkgb mwxdisvgrv
*	NULL	NULL

Рис. 18: Результат выполнения запроса

2. Пользователь root начинает транзакцию.

```
@root
start transaction;
Query OK, 0 rows affected (0,01 sec)
```

3. Транзакцию начинает пользователь newguy.

```
@newguy
start transaction;
Query OK, 0 rows affected (0,01 sec)
```

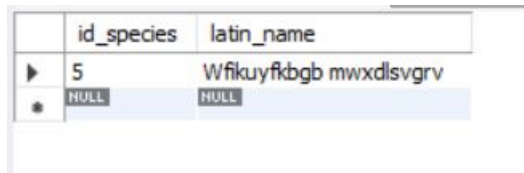
4. Пользователь root обновляет запись с id=5 в таблице species, изменяя латинское название (latin_name).

```
@root
update birds.species
set latin_name = 'Pica pica'
where id_species = 5;
```

5. Пользователь newguy в рамках транзакции пытается считать данные из этой строки.

```
@newguy
select id_species , latin_name
from species
where id_species = 5;
```

Результат выполнения запроса:



	id_species	latin_name
▶	5	Wfikuyfkgb mwxdlsvgrv
•	NULL	NULL

Рис. 19: Результат выполнения запроса

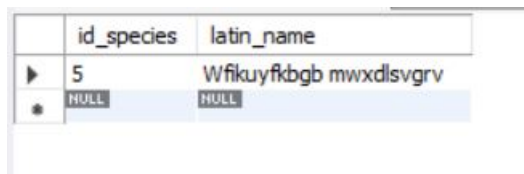
6. Пользователь root отменяет внесенные изменения:

```
@root
rollback;
Query OK, 0 rows affected (0,00 sec)
```

7. Пользователь newguy подтверждает внесенные изменения:

```
@newguy
commit;
Query OK, 0 rows affected (0,00 sec)
```

8. При попытке проверить значение поля latin_name от любого из пользователей будет видно лишь старое значение (рис. 20).



	id_species	latin_name
▶	5	Wfikuyfkgb mwxdlsvgrv
•	NULL	NULL

Рис. 20: Результат выполнения запроса

Результат: феномен Dirty Read не происходит.

6.3 Non-repeatable read

Для попытки обнаружения феномена «Non-repeatable read» были произведены следующие шаги.

1. Пользователь root начинает транзакцию

```
@root
start transaction;
Query OK, 0 rows affected (0,01 sec)
```

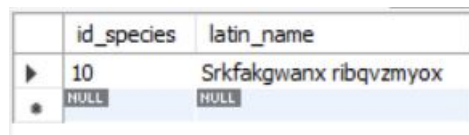
2. Пользователь newguy начинает транзакцию.

```
@newguy
start transaction;
Query OK, 0 rows affected (0,01 sec)
```

3. Пользователь newguy делает запрос.

```
@newguy
select id_species, latin_name
from species
where id_species = 10;
```

Результат выполнения запроса:



	id_species	latin_name
▶	10	Srkfakgwanx ribqvzmyox
•	NULL	NULL

Рис. 21: Результат выполнения запроса

4. Вносятся изменения в таблице species.

```
@root
update birds.species
set latin_name = 'Pica pica'
where id_species = 10;

commit;
```



	id_species	latin_name
▶	10	Pica pica
•	NULL	NULL

Рис. 22: Результат выполнения запроса

5. Пользователь newguy делает запрос (у него транзакция все еще открыта).

```
@newguy
update birds.species
set latin_name = 'Pica pica'
where id_species = 10;

commit;
```

Результат: феномен Non-repeatable read не происходит. Зафиксированные изменения в другой транзакции не отражаются в текущей транзакции.

	id_species	latin_name
▶	10	Srkfakgwanx ribqvzmyox
•	HULL	HULL

Рис. 23: Результат выполнения запроса

6.4 Phantom read

Для попытки обнаружения феномена «Phantom read» были произведены следующие шаги.

1. Пользователь root начинает транзакцию.

```
@root
start transaction;
Query OK, 0 rows affected (0,00 sec)
```

2. Пользователь newguy начинает транзакцию.

```
@newguy
start transaction;
Query OK, 0 rows affected (0,00 sec)
```

3. Пользователь newguy считает количество строк в таблице bird.

```
select count(id_bird) from birds.bird;
```

	count(id_bird)
▶	100008

Рис. 24: Результат выполнения запроса

4. Пользователь root добавляет в таблицу bird запись.

```
@root
insert into bird (
  feath_color ,
  beak_shape ,
  size ,
  tail_shape ,
  pool ,
  anthr_obj_distance ,
  id_species ,
  id_birdwatcher)
values (
  'синий' ,
  'загнутый вниз' ,
  'мелкий' ,
  'короткий' ,
  1 ,
  'рядом с населенным пунктом' ,
  101 ,
```

8
);

5. Пользователь newguу снова считает количество строк в таблице bird.

	count(id_bird)
▶	100008

Рис. 25: Результат выполнения запроса

Результат: феномен Phantom read не происходит.

Заключение

В результате выполнения лабораторных работ был произведён ряд действий с базой данных, перечисленных ниже.

1. Создана view и произведён запрос к ней. На основе этой view создана таблица speciescnt. View в базе данных позволяют создавать представления, которые, например, могут быть доступны пользователям, у которых нет доступа к остальной базе. Таким образом можно реализовать определенный запрос, на который у пользователя не может быть прав. Особенностью использования view является то, что каждое обращение к ней требует выполнения запроса, что увеличивает время работы, но уменьшает количество данных, которые нужно хранить и согласовывать.
2. Создан триггер для обновления счётчика встреченных птиц в таблице speciescnt при добавлении новой записи в таблицу meeting_time. Триггер в базе данных является инструментом событийного программирования и позволяет реализовать некоторую логику прямо в базе данных. Таким образом, в ответ на некоторое событие в базе данных (в данном случае – добавление новой записи) автоматически происходит другое событие (в данном случае – обновление значения счётчика).
3. Создана функция, которая принимает на вход название сезона и возвращает число птиц, встреченных в этом сезоне за последний год.
4. Создана процедура, которая по заданным имени, фамилии и отчеству бёрдвотчера выводит в алфавитном порядке названия всех видов птиц, которых человек с таким именем встретил летом.
5. Создано два пользователя с разными правами доступа и показана работа с ними. Многопользовательская модель позволяет разграничить доступ пользователей к таблицам, выделяя каждому из них отдельную версию схемы базы данных. Таким образом повышается безопасность базы данных.
6. Был проверен уровень изоляции транзакции READ COMMITTED, который допускает феномены, связанные с чтением. Транзакционная модель обеспечивает параллельный доступ к данным для нескольких пользователей. Уровень изоляции транзакции обеспечивает ту или иную степень согласованности данных, которую можно оценить, например, по выполнению феноменов параллельного доступа.