Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение

высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе № 5

по дисциплине «Базы данных»

Выполнили: Борисов Георгий

Граник Артем

Факультет: ПИиКТ

Группа: P33212

Цель работы:

- Добавить в ранее созданную базу данных (лр №4) триггеры для обеспечения комплексных ограничений целостности.
- Реализовать функции и процедуры на основе описания бизнес-процессов, определенных при описании предметной области (лр №1). Должна быть обеспечена проверка корректности вводимых данных для созданных функций и процедур.
- Необходимо произвести анализ использования созданной базы данных, выявить наиболее часто используемые объекты базы данных, виды запросов к ним. Результаты должны быть представлены в виде текстового описания.
- На основании полученного описания требуется создать подходящие индексы и доказать, что они будут полезны для представленных в описании случаев использования базы данных.

Ход работы.

Триггеры:

1. Обновление информации в таблице занятости. При добавлении новой записи о сотруднике в таблицу занятости, старая запись обновляется, так как человек там больше не работает, также обновляет размеры соответствующих отделов.

```
 CREATE OR REPLACE FUNCTION hiring_process()
RETURNS trigger as
$$
BEGIN
UPDATE s307353.Division SET size_of_division=size_of_division+1 where division_id=NEW.division_id;
UPDATE s307353.EmploymentHistory SET status='hired';
RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE OR REPLACE FUNCTION dismissal()
RETURNS trigger as
$$
BEGIN
UPDATE s307353.Division SET size_of_division=size_of_division-1 where division_id=NEW.division_id;
UPDATE s307353.EmploymentHistory SET status= 'fired';
RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';


 CREATE TRIGGER EMPLOYMENT_PROCESS
AFTER INSERT ON s307353.EmploymentHistory
FOR EACH ROW
EXECUTE PROCEDURE hiring_process();

CREATE TRIGGER DISCHARGE_PROCESS
studs-> AFTER UPDATE ON s307353.EmploymentHistory
studs-> FOR EACH ROW
studs-> EXECUTE PROCEDURE dismissal1();
CREATE TRIGGER
```

2. Блокируем запуски с неподходящими погодными условиями, при попытке вставить некорректную запись создаем альтернативную запись с запуском, отложенным на неделю (исходим из того, что за неделю погоду гарантированно улучшается)

```
 CREATE OR REPLACE FUNCTION delay_launch_2()
RETURNS trigger as
$$
BEGIN
UPDATE s307353.Launch SET date=NEW.date + INTERVAL '1 week' where launch_id=NEW.launch_id AND weather_conditions='severe conditions';
UPDATE s307353.Launch SET weather_conditions='sunny conditions expected' where launch_id=NEW.launch_id;
RETURN NEW;
END;

$$
LANGUAGE 'plpgsql';

CREATE TRIGGER launch_procedure()
AFTER INSERT ON s307353.Launch
FOR EACH ROW
EXECUTE PROCEDURE delay_launch_2()
```

Функции и процедуры:

1. planMission() - на основе информации о миссии и набора экспериментов с их описаниями(список требуемого оборудования и орбит) добавляет запись в таблицу Mission и соответствующие записи в IntendedExperiment

```
create procedure planMission(myName varchar(64), myGoal text, myDescription text, experiments integer[], info_arr integer[][][])
as $$DECLARE
    experiment int;
    my_id int;
    index int = 1;
    i int = 1;
    len int;
    array2d int[][];
    orbit int;
    BEGIN
    insert into s295964.Mission(name, goal, status, description) values(myName, myGoal, 'PLANNED', myDescription) returning mission_id into my_id;
    foreach experiment in array experiments
    loop
        array2d = ARRAY(SELECT * FROM unnest(info_arr[index:index]));
        len = array_length(array2d, 1);
        while i < len
            loop
            if array2d[i + 1] = 0 then
                orbit = null;
            else
                orbit = array2d[i + 1];
            end if;
            insert into s295964.IntendedExperiments values(experiment, my_id, array2d[i], orbit);
            i = i + 2;
            end loop;
        index = index + 1;
        end loop;
end;
    $$language  plpgsql;
```

2. updateStatus() по номеру миссии проверяет готовность необходимого оборудования в таблице Production, в случае готовности обновляет статус миссии, в противном случае выводит сообщение о том, что не все

оборудование готово;

```
create procedure updateStatus(mission integer)
as $$BEGIN
    if (select bool_and(s295964.Production.status) from s295964.IntendedExperiments
        join s295964.Production on IntendedExperiments.equipment_id = Production.equipment_id
                                            where IntendedExperiments.mission_id = mission) then
            update s295964.Mission set status = 'READY FOR LAUNCH' where mission_id = mission;
        else
            raise notice 'Not all equipment has been produced';
    end if;
end;
    $$language  plpgsql;
```

3. schedule_launch() устанавливает для заданной миссии запуск на определенную дату.

```
 CREATE OR REPLACE FUNCTION schedule_launch(mission_id integer,date DATE)
as $$BEGIN
INSERT INTO s307353.Launch(mission_id,date) VALUES(mission_id,date);
END;
$$
LANGUAGE 'plpgsql';
```

Индексы:

1. Выбор сотрудников из таблицы будет осуществляться по имени и фамилии в большинстве случаев, поэтому целесообразно создать индекс по их комбинации.

```
create index employee_name on s295964.Employee(name, surname);
```

без индекса:

```
studs=> select count(*) from Employee where name ='Ryan' and surname ='Gosling';
 count
-------
   166
(1 строка)

Время: 1,962 мс
```

```
studs=> explain analyze select count(*) from Employee where name = 'Ryan' and surname = 'Gosling';
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=256.43..256.44 rows=1 width=8) (actual time=1.712..1.713 rows=1 loops=1)
   ->  Seq Scan on employee  (cost=0.00..256.00 rows=170 width=0) (actual time=0.033..1.693 rows=166 loops=1)
         Filter: (((name)::text = 'Ryan'::text) AND ((surname)::text = 'Gosling'::text))
         Rows Removed by Filter: 9834
 Planning Time: 0.184 ms
 Execution Time: 1.742 ms
```

с индексом:

```
studs=> create index employee_name on s295964.Employee(name, surname);
CREATE INDEX
Время: 66,420 мс
studs=> select count(*) from Employee where name ='Ryan' and surname ='Gosling';
 count
-------
   166
(1 строка)

Время: 1,059 мс
```

```
studs=> explain analyze select count(*) from Employee where name = 'Ryan' and surname = 'Gosling';
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=8.11..8.12 rows=1 width=8) (actual time=0.094..0.095 rows=1 loops=1)
   ->  Index Only Scan using employee_name on employee  (cost=0.29..7.69 rows=170 width=0) (actual time=0.056..0.074 rows=166 loops=1)
         Index Cond: ((name = 'Ryan'::text) AND (surname = 'Gosling'::text))
         Heap Fetches: 0
 Planning Time: 0.314 ms
 Execution Time: 0.127 ms
(6 строк)
```

В данном случае очевиден выигрыш в скорости выполнения запроса примерно в 2 раза.

2. Выбор сотрудников определенного отдела по id отдела и статусу ‘hired’ требует отдельного триггера

```
create index hired_in_division on s295964.EmploymentHistory(division_id, status) where status = 'hired';
```

без индекса:

```
studs=> select count(*) from EmploymentHistory where division_id = 3 and status = 'hired';
 count
-------
  2531
(1 строка)

Время: 2,849 мс
```

```
studs=> explain analyze select count(*) from EmploymentHistory where division_id = 3 and status = 'hired';
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=327.33..327.34 rows=1 width=8) (actual time=2.201..2.202 rows=1 loops=1)
   ->  Seq Scan on employmenthistory  (cost=0.00..321.00 rows=2531 width=0) (actual time=0.032..2.036 rows=2531 loops=1)
         Filter: ((division_id = 3) AND ((status)::text = 'hired'::text))
         Rows Removed by Filter: 12469
 Planning Time: 0.137 ms
 Execution Time: 2.228 ms
(6 строк)
```

с индексом:

```
studs=> create index hired_in_division on s295964.EmploymentHistory(division_id, status) where status = 'hired';
CREATE INDEX
Время: 137,651 мс
studs=> select count(*) from EmploymentHistory where division_id = 3 and status = 'hired';
 count
-------
  2531
(1 строка)

Время: 1,010 мс
```

```
studs=> explain analyze select count(*) from EmploymentHistory where division_id = 3 and status = 'hired';
                                                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=62.91..62.91 rows=1 width=8) (actual time=0.469..0.470 rows=1 loops=1)
   ->  Index Only Scan using hired_in_division on employmenthistory  (cost=0.29..56.58 rows=2531 width=0) (actual time=0.031..0.303 rows=2531 loops=1)
         Index Cond: (division_id = 3)
         Heap Fetches: 0
 Planning Time: 0.353 ms
 Execution Time: 0.544 ms
(6 строк)
```

Таблица EmploymentHistory содержит 15000 записей, поэтому использование индекса позволяет увеличить скорость выполнения запроса почти в 3 раза.

3. В таблице MissionCharacteristics отсутствует ключ, поэтому для ускорения запросов по выводу информации о конкретной миссии целесообразно добавить индекс для атрибута mission_id

```
create index mission_in_characteristics on s295964.MissionCharacteristics(mission_id);
```

Без индекса:

```
studs=> select * from MissionCharacteristics where mission_id = 1227;
 mission_id | spacecraft_id | rocket_id | orbit_id
------------+---------------+-----------+----------
       1227 |             6 |         3 |       24
       1227 |             4 |         3 |       49
(2 строки)

Время: 1,653 мс
```

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on missioncharacteristics  (cost=0.00..197.50 rows=2 width=16) (actual time=0.094..0.852 rows=2 loops=1)
   Filter: (mission_id = 1227)
   Rows Removed by Filter: 10998
 Planning Time: 0.159 ms
 Execution Time: 0.870 ms
(5 строк)
```

С индексом:

```
studs=> create index mission_in_characteristics on s295964.MissionCharacteristics(mission_id);
CREATE INDEX
Время: 74,056 мс
studs=> select * from MissionCharacteristics where mission_id = 1227;
 mission_id | spacecraft_id | rocket_id | orbit_id
------------+---------------+-----------+----------
       1227 |             6 |         3 |       24
       1227 |             4 |         3 |       49
(2 строки)

Время: 0,876 мс
```

```
studs=> explain analyze select * from MissionCharacteristics where mission_id = 1227;
                                                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on missioncharacteristics  (cost=4.30..11.23 rows=2 width=16) (actual time=0.021..0.024 rows=2 loops=1)
   Recheck Cond: (mission_id = 1227)
   Heap Blocks: exact=2
   ->  Bitmap Index Scan on mission_in_characteristics  (cost=0.00..4.30 rows=2 width=0) (actual time=0.013..0.013 rows=2 loops=1)
         Index Cond: (mission_id = 1227)
 Planning Time: 0.117 ms
 Execution Time: 0.086 ms
```

Использование индекса на таблице MissionCharacteristics, имеющей 11000 записей, позволяет добиться двукратного выигрыша в скорости.

4. Аналогично с таблицей MissionCharacteristics добавляем такой же индекс в таблицу IntendedExperiments

```
create index mission_in_experiments on s295964.IntendedExperiments(mission_id);
```

Без индекса:

```
studs=> explain analyze select count(*) from IntendedExperiments where mission_id=1548;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=181.25..181.26 rows=1 width=8) (actual time=0.859..0.860 rows=1 loops=1)
   ->  Seq Scan on intendedexperiments  (cost=0.00..181.25 rows=1 width=0) (actual time=0.530..0.854 rows=1 loops=1)
         Filter: (mission_id = 1548)
         Rows Removed by Filter: 10099
 Planning Time: 0.098 ms
 Execution Time: 0.894 ms
```

С индексом:

```
studs=> create index mission_in_experiments on s295964.IntendedExperiments(mission_id);
CREATE INDEX
studs=> explain analyze select count(*) from IntendedExperiments where mission_id=1548;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4.31..4.32 rows=1 width=8) (actual time=0.051..0.052 rows=1 loops=1)
   ->  Index Only Scan using mission_in_experiments on intendedexperiments  (cost=0.29..4.30 rows=1 width=0) (actual time=0.042..0.043 rows=1 loops=1)
         Index Cond: (mission_id = 1548)
         Heap Fetches: 0
 Planning Time: 0.267 ms
 Execution Time: 0.101 ms
(6 строк)
```

Использование данного индекса также позволяет увеличить скорость выполнения запроса примерно в 3 раза.

5. Мы часто обращаемся к таблице Production для проверки готовности оборудования, поэтому необходимо добавить индекс

```
create index equipment_readiness on s295964.Production(equipment_id, status) where status = 'true';
```

Без индекса:

```
studs=> explain analyze select * from production where equipment_id = 96 and status = 'true';
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on production  (cost=0.00..180.00 rows=8 width=9) (actual time=0.190..0.949 rows=10 loops=1)
   Filter: (status AND (equipment_id = 96))
   Rows Removed by Filter: 9990
 Planning Time: 0.104 ms
 Execution Time: 0.972 ms
```

С индексом:

```
studs=> create index equipment_readiness on s295964.Production(equipment_id, status) where status = 'true';
CREATE INDEX
studs=> explain analyze select * from production where equipment_id = 96 and status = 'true';
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on production  (cost=4.34..27.29 rows=8 width=9) (actual time=0.055..0.069 rows=10 loops=1)
   Recheck Cond: ((equipment_id = 96) AND status)
   Heap Blocks: exact=9
   ->  Bitmap Index Scan on equipment_readiness  (cost=0.00..4.34 rows=8 width=0) (actual time=0.048..0.048 rows=10 loops=1)
         Index Cond: (equipment_id = 96)
 Planning Time: 0.387 ms
 Execution Time: 0.099 ms
```

Выигрыш в скорость примерно в 2 раза.

Выводы

В ходе выполнения данной лабораторной работы мы завершили работу над созданной ранее базой данных. Нами были добавлены триггеры, обеспечивающие комплексные ограничения целостности данных в нашей БД. Для наиболее часто повторяющихся операций с данными добавлены функции и процедуру, существенно ускоряющие работу. Помимо этого, так как количество записей в таблицах превышает во многих случаях 10000, для ускорения операций с данными были реализованы индексы и проведен анализ их эффективности. В процессе работы нами был получен ценный опыт по реализации сложных баз данных, использующих такой функционал как функции, триггеры и индексы.