Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина: Базы данных

Отчёт по лабораторной работе № 5

Выполнил: До Зыонг Мань

Ву Минь Хиеу

Студент группы: P33201

Преподаватель: Машина Екатерина Алексеевна

Санкт-Петербург

2022 г.

I. Текст задания

Для выполнения лабораторной работы №5 необходимо:

- 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.

Отчёт по лабораторной работе должен содержать:

- титульный лист;
- текст задания;
- код триггеров, функций, процедур;
- описание наиболее часто используемых сценариев при работе с базой данных;
- описание индексов и обоснование их использования;
- выводы по работе.

Отчёт по лабораторной работе должен содержать:

- PL/pgSQL;
- процедуры, функции;
- триггеры;
- индексы.

II. Код триггеров, функций и процедур

Процедуры

1. Transfer_player

```
Create or replace procedure s291193.tranfer_player(_player_id INTEGER,
_team_id INTEGER)
LANGUAGE 'plpgsql'
AS $$
    DECLARE
        _price INTEGER;
        _current_amount BIGINT;
    BEGIN
        IF (EXISTS(SELECT * FROm player where id = _player_id) and
            exists(select * from club where id = _team_id))
            then
            select get_total_amount_from_sponsors(_team_id) into
_current_amount;
            select player_club._price into _price from player_club where
player_club.id_player = _player_id;
            if(_current_amount > _price) then
                update player_club set id_team =_team_id where id_player =
_player_id;
                call update_amount(_team_id := _team_id, amount :=
_current_amount - _price);
            else
                RAISE notice 'Amount not enough';
            end if;
        else
            raise notice 'player or club not found';
        end if;
        Commit;
    END;
    $$;
```

2. Sponsor_new_club

```
create or replace procedure s291193.sponsor_new_club(_team_id INTEGER,
_sponsor_id INTEGER, amount BIGINT)
LANGUAGE 'plpgsql'
AS $$
    DECLARE
        new_amount BIGINT;
    BEGIN
        if(exists(select * from club where id = _team_id) and
           exists(select * from sponsor where id = _sponsor_id))
            then
                insert into sponsor_club values (_sponsor_id, _team_id,
amount);
                select club.amount into new_amount from club where id =
_team_id;
                perform update_amount(_team_id,new_amount + amount);
        end if;
    end;
    $$;
```

3. Transfer_new_coach

```
create or replace procedure s291193.transfer_new_coach(_coach_id integer,
_team_id integer)
language 'plpgsql'
as $$
    declare
        id_old_club integer;
    begin
        if(exists(select * from coach where id = _coach_id) and
           exists(select * from club where id = _team_id))
            then
            update club set id_coach = _coach_id where id = _team_id;
            if(exists(select id from club where id_coach = _coach_id))
                then
                select id into id_old_club from club where id_coach =
_coach_id;
                update club set id_coach = -1 where id = id_old_club;
            end if;
        end if;
    end;
$$;
```

Функции

1. Get_total_amount_from_sponsors

```
create or replace function s291193.get_total_amount_from_sponsors(_team_id
INTEGER)
RETURNS BIGINT
LANGUAGE 'plpgsql'
AS $$
    DECLARE
        _total_amount BIGINT;
    BEGIN
        SELECT sum(sponsor_amount) into _total_amount from sponsor_club
where club_id = _team_id;
        return _total_amount;
    end;
    $$;
```

2. Update_amount

```
create or replace function s291193.update_amount(_team_id INTEGER,
new_amount BIGINT)
RETURNS VOID
LANGUAGE 'plpgsql'
AS $$
    BEGIN
        update club set amount = new_amount where id = _team_id;
    end;
    $$;
```

3. Get_attribute_player

```
create or replace function s291193.get_attribute_player(_player_id
integer)
returns setof attribute
language 'plpgsql'
as $$
    begin
        return query select * from s291193.attribute where id in (select
id_attribute from player where id = _player_id);
```

```
    end;
    $$;
```

Триггеры

1. insert_player

```
Create or replace function s291193.insert_new_player_1()
returns trigger
language 'plpgsql'
as $$BEGIN
    insert into attribute values (new.id, 60,60,60,60,60,60,60);
    return new;
end;
$$;


create or replace trigger insert_new_player before insert on player
    for each row
    execute function insert_new_player_1();
```

2. insert_new_match_on_league

```
create or replace function insert_new_match_on_league()
    returns trigger
    language 'plpgsql'
as $$BEGIN
    insert into match values
(new.match_id,'Match','01/01/1900','Sunny',1000);
    return new;
end;
$$;
create or replace trigger insert_new_match_on_league after insert on
match_league
    for each row
    execute function insert_new_match_on_league();
```

III. Описание наиболее часто используемых объектов базы данных и видов запросов к ним.

Главное, на чем строится логика информационной системы, — это управление командой. Во время игры наиболее частым взаимодействием является покупка и продажа игроков, а также соответствующая смена тренеров.

- Трансфер игроков

Сначала проверьте правильность id_player и id_team (SELECT). Затем проверим финансовое состояние команды (сумма) по сравнению с ценой игрока на тот момент (цена). При соблюдении финансовых условий перевод будет осуществлен

- Трансфер на автобусе

Процесс передачи похож на передачу игрока.

- Инвестировать в команду

Прежде всего проверьте правильность id_sponsor, id_team (SELECT). После этого мы добавим на счет команды вложенную сумму.

IV. Анализировать индексацию в базе данных

В зависимости от бизнес-логики эти таблицы имеют много **UPDATE** и **INSERT**:

1. Match(match_goal, match_league, match_referee, match_stadium)

Каждый матч имеет место, таблица должна будет INSERT новую записи, поэтому здесь не стоит применить индексацию здесь;

2. Goal

Количество голов точно не меньше количества матчей, потому что в матче может быть много голов, поэтому здесь не стоит применить индексацию здесь;

3. Play_motm

В каждом матче будет отличный игрок, данные будут INSERT, как и в случае с match, поэтому здесь не стоит применить индексацию здесь;

В зависимости от бизнес-логики эти таблицы не имеют большого **UPDATE** и **INSERT**, однако у них очень мало записей:

1. League :

На данный момент в игру добавлено только несколько самых известных турниров, их меньше 10, поэтому нет необходимости использовать индекс

2. Referee :

Как и в случае с лигой, мы также выбираем только самых известных судей в мире для добавления в игру, их число меньше 10, поэтому нет необходимости использовать индекс.

3. Sponsor

Как и в случае с лигой, мы также выбираем только самых известных спонсоров в мире для добавления в игру, их число меньше 10, поэтому нет необходимости использовать индекс.

4. Club

Количество клубов будет меньше 200, поэтому индексировать не нужно

5. Coach

Количество тренеров будет меньше 200, поэтому индексировать не нужно

6. Stadium

Количество стадионов будет меньше 200, поэтому индексировать не нужно

Итак, наконец, единственным подходящим для индексации является таблица **attribute, club, coach, player, stadium**.

Attribute - это таблица, содержащая атаку, защиту и статистику игрока, которые являются большими и неизменными, без повторяющихся значений, поэтому я создам индексы для этой статистики, чтобы игроки могли быстро искать игроков по своей статистике.

Player - Конечно, количество игроков велико, не нужно обновлять и добавлять, нет дублирующихся данных, поэтому индекс необходим для ускорения поиска.

1. **Перед индексацией**

species

```
Index Scan using species_pkey on species  (cost=0.27..8.29 rows=1
width=32) (actual time=0.054..0.055 rows=1 loops=1)
  Index Cond: (id = 250)
Planning Time: 0.133 ms
Execution Time: 0.285 ms
```

species_environment

```
 Seq Scan on species_environment  (cost=0.00..12.75 rows=1 width=8)
(actual time=0.165..0.175 rows=1 loops=1)
  Filter: (species_id = 250)
  Rows Removed by Filter: 699
Planning Time: 0.796 ms
Execution Time: 0.187 ms
```

2. **b-tree**

species

```
CREATE UNIQUE INDEX index_species ON species (name);
```

анализ времени

```
Index Scan using species_id_idx on species  (cost=0.27..8.29 rows=1
width=32) (actual time=0.047..0.048 rows=1 loops=1)
  Index Cond: (id = 250)
Planning Time: 1.706 ms
Execution Time: 0.085 ms
```

species_environment

```
CREATE INDEX ON species_environment (species_id);
```

анализ времени

```
Index Scan using species_environment_species_id_idx on
species_environment  (cost=0.28..8.29 rows=1 width=8) (actual
time=0.043..0.044 rows=1 loops=1)
  Index Cond: (species_id = 250)
Planning Time: 1.146 ms
Execution Time: 0.060 ms
```

3. **hash**

species

```
CREATE UNIQUE INDEX ON species (name);
```

анализ времени

```
Index Scan using index_species on species  (cost=0.00..8.02 rows=1
width=32) (actual time=0.097..0.097 rows=1 loops=1)
  Index Cond: (id = 250)
Planning Time: 1.553 ms
Execution Time: 0.117 ms
```

species_environment

```
CREATE INDEX ON species_environment (species_id);
```

анализ времени

```
Index Scan using index_species_environment on species_environment
(cost=0.00..8.02 rows=1 width=8) (actual time=0.026..0.027 rows=1
loops=1)
  Index Cond: (species_id = 250)
Planning Time: 1.428 ms
Execution Time: 0.057 ms
```

V. Вывод

Мы научились реализовывать сложные ограничения целостности с помощью триггеров, реализовали множество функций, подходящих для бизнес-процессов в нашей информационной системе, а также провели детальный анализ и доказали эффективность добавленных нами индексов.