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.

1. **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.

1. **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 $$  
 DECLARE  
 \_price INTEGER;  
 \_current\_amount BIGINT;  
 BEGIN  
 IF (*EXISTS*(SELECT \* FROm player where id = \_player\_id) and  
 *exists*(select \* from club where id = \_team\_id))  
 then  
 select *get\_total\_amount\_from\_sponsors*(\_team\_id) into \_current\_amount;  
 select player\_club.\_price into \_price from player\_club where player\_club.id\_player = \_player\_id;  
 if(\_current\_amount > \_price) then  
 update player\_club set id\_team =\_team\_id where id\_player = \_player\_id;  
 call *update\_amount*(\_team\_id := \_team\_id, amount := \_current\_amount - \_price);  
 else  
 RAISE notice 'Amount not enough';  
 end if;  
 else  
 raise notice 'player or club not found';  
 end if;  
 Commit;  
 END;  
 $$;

1. **Sponsor\_new\_club**

create or replace procedure s291193.*sponsor\_new\_club*(\_team\_id INTEGER, \_sponsor\_id INTEGER, amount BIGINT)  
LANGUAGE 'plpgsql'  
AS $$  
 DECLARE  
 new\_amount BIGINT;  
 BEGIN  
 if(*exists*(select \* from club where id = \_team\_id) and  
 *exists*(select \* from sponsor where id = \_sponsor\_id))  
 then  
 insert into sponsor\_club values (\_sponsor\_id, \_team\_id, amount);  
 select club.amount into new\_amount from club where id = \_team\_id;  
 perform *update\_amount*(\_team\_id,new\_amount + amount);  
 end if;  
 end;  
 $$;

1. **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 $$  
 DECLARE  
 \_total\_amount BIGINT;  
 BEGIN  
 SELECT *sum*(sponsor\_amount) into \_total\_amount from sponsor\_club where club\_id = \_team\_id;  
 return \_total\_amount;  
 end;  
 $$;

1. **Update\_amount**

create or replace function s291193.*update\_amount*(\_team\_id INTEGER, new\_amount BIGINT)  
RETURNS VOID  
LANGUAGE 'plpgsql'  
AS $$  
 BEGIN  
 update club set amount = new\_amount where id = \_team\_id;  
 end;  
 $$;

1. **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*();

1. **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();

1. **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.

1. **Анализировать индексацию в базе данных**

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

1. account (account\_food, account\_material, account\_building, inventory):

Каждый раз, когда зарегистрирован новый игрок, эти таблицы должны **INSERT** новую запись, поэтому здесь не стоит применить индексацию здесь;

1. dinosaur:

Сбор динозавров - это главная цель игры, поэтому покупка динозавров используется очень часто. Вот почему в этой таблице будет много **INSERT / UPDATE**. Так что это не подходит для применения индексации;

1. habitat (dinosaur\_habitat):

Пользователи могут купить больше мест обитания для увеличения количества динозавров, которые они могут удержать. Так что здесь еще много операций **INSERT** здесь;

1. island (island\_resource):

Похоже на среду обитания, много операций **INSERT / UPDATE**.

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

1. food:

Есть только 7 видов еды;

1. item (ability, item\_ability):

Есть только 100 видов предмета и 50 способностей;

1. material (build\_requirement, resource\_material, upgrade\_requirement):

Есть только 5 видов материала. И здание, ресурс тоже небольшой;

1. building:

Есть только 5 видов здания;

1. habitat\_level:

Есть только 5 уровней среды обитания.

1. resource:

There are only 50 types of resource.

Итак, наконец, единственным подходящим для индексации является таблица **species** и **species\_environment**.

**species** - это таблица, где содержится все виды видов динозавров. Мы будем использовать индекс в *name*, потому что мы при покупке динозавра нам нужно будет запросить его имя.

**Species\_environment** - это таблица, где содержится среда, в которой мы найдем среду, где эти виды живут по *species\_id*.

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

1. **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

1. **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

1. **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.