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:

Михайлов Андрей Алексеевич

Факультет: ПИиКТ

Группа: P33212

Преподаватель: Шешуков Д.М.

Санкт-Петербург, 2023г

Задание

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.

Триггеры

1) Триггер для автоматического форматирования колонки капитализация в таблице эмитента, чтобы привести запись в более читаемый вид:

```
CREATE OR REPLACE FUNCTION format_capitalization()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' OR TG_OP = 'UPDATE' THEN
        NEW.capitalization = TO_CHAR(NEW.capitalization::NUMERIC / 1000000000.0, 'FM999999999.0') || ' млрд.руб';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_format_capitalization
BEFORE INSERT OR UPDATE OF capitalization
ON issuer
FOR EACH ROW
EXECUTE FUNCTION format_capitalization();
```

2) Триггер для автоматического подсчета цены транзакции, путем умножения цены одной акции на количество покупаемых акций:

```
CREATE OR REPLACE FUNCTION update_transaction_price()
RETURNS TRIGGER AS $$
BEGIN
    NEW.price = NEW.quantity * (SELECT price FROM tool WHERE id = NEW.tool_id);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_transaction_price_trigger
BEFORE INSERT ON transaction
FOR EACH ROW
EXECUTE FUNCTION update_transaction_price();
```

3) Триггер, который гарантирует невозможность удаления эмитента, если он имеет связь с существующим инструментом:

```
CREATE OR REPLACE FUNCTION check_tool_existence()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM tool WHERE issuer_id = OLD.id) THEN
        RAISE EXCEPTION 'Cannot delete issuer because associated tools exist';
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_tool_trigger
BEFORE DELETE ON issuer
FOR EACH ROW
EXECUTE FUNCTION check_tool_existence();
```

4) Триггер для проверки и выдачи верного налога инвестору:

```
CREATE OR REPLACE FUNCTION insert_tax()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.period < INTERVAL '1 year' OR NEW.turnover = 0 THEN
        NEW.tax := '0%';
    ELSIF NEW.period >= INTERVAL '1 year' AND NEW.turnover > 0 AND NEW.deposits > 0 THEN
        NEW.tax := '13%';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER insert_tax_trigger
BEFORE INSERT ON report
FOR EACH ROW
EXECUTE FUNCTION insert_tax();
```

5) Триггер для автоматического заполнения и обновления таблицы актива на основе сделанной транзакции:

```
CREATE OR REPLACE FUNCTION update_active_info()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.trans_type = 'продажа' THEN
        UPDATE active
        SET quantity = quantity - NEW.quantity,
            total_price = total_price - NEW.price
        WHERE tool_id = NEW.tool_id AND investment_portfolio_id = NEW.investment_portfolio_id;

    ELSIF NEW.trans_type = 'покупка' THEN
        UPDATE active
        SET quantity = quantity + NEW.quantity,
            total_price = total_price + NEW.price
        WHERE tool_id = NEW.tool_id AND investment_portfolio_id = NEW.investment_portfolio_id;
    END IF;

    IF NOT FOUND THEN
        INSERT INTO active (quantity, total_price, tool_id, investment_portfolio_id)
        VALUES (
            COALESCE((SELECT SUM(t.quantity)
                      FROM transaction t
                      WHERE t.tool_id = NEW.tool_id
                        AND t.investment_portfolio_id = NEW.investment_portfolio_id), 0),
            COALESCE((SELECT SUM(t.price)
                      FROM transaction t
                      WHERE t.tool_id = NEW.tool_id
                        AND t.investment_portfolio_id = NEW.investment_portfolio_id), 0),
            NEW.tool_id, NEW.investment_portfolio_id
        );
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_active_info_trigger
AFTER INSERT ON transaction
```

Процедуры

1) Процедура для быстрого добавления инвестора:

```
CREATE PROCEDURE add_investor(
    IN p_type VARCHAR(50),
    IN p_name VARCHAR(50),
    IN p_experience VARCHAR(50))
AS
$$
BEGIN
    INSERT INTO investor (type, name, experience) VALUES (p_type, p_name, p_experience);
    RAISE NOTICE 'Новый инвестор успешно добавлен';
END;
$$ LANGUAGE plpgsql;
```

2) Процедура для получения списка всех инвесторов:

```
CREATE PROCEDURE get_all_investors()
AS
$$
BEGIN
    SELECT * FROM investor;
END;
$$ LANGUAGE plpgsql;
```

3) Процедура для обновления информации об инвесторе:

```
CREATE PROCEDURE update_investor_info(
    IN p_investor_id INT,
    IN p_type VARCHAR(50),
    IN p_name VARCHAR(50),
    IN p_experience VARCHAR(50))
AS
$$
BEGIN
    UPDATE investor SET type = p_type, name = p_name, experience = p_experience WHERE id = p_investor_id;
END;
$$ LANGUAGE plpgsql;
```

Описание наиболее часто используемых сценариев при работе с базой данных

1) Получение информации об инвесторе и его портфеле инвестиций:

```
studs=> SELECT i.name AS investor_name, ip.name AS portfolio_name
FROM investor i
INNER JOIN investment_portfolio ip ON i.id = ip.investor_id;
 investor_name |     portfolio_name
---------------+------------------------
 Андрей        | Большой портфель на год
 Андрей        | Дневной потрфель
 Банк СПБ      | Портфель банка СПБ
 Пётр          | Портфель Петра
 Олег          | Олегс Портфель
(5 строк)
```

2) Получение данных о транзакциях для определенного инструмента и портфеля инвестиций:

```
studs=> SELECT t.id, t.trans_type, t.quantity, t.price, t.date_t
FROM transaction t
INNER JOIN investment_portfolio ip ON t.investment_portfolio_id = ip.id
INNER JOIN tool tl ON t.tool_id = tl.id
WHERE tl.name = 'Yandex' AND ip.name = 'Большой портфель на год';
 id | trans_type | quantity | price |   date_t
----+------------+----------+-------+------------
  7 | покупка    |       10 | 37000 | 2023-01-04
(1 строка)
```

3) Получение информации о дивидендах для конкретного инструмента:

```
studs=> SELECT d.payment_date, d.payment_amount
FROM dividends d
INNER JOIN tool tl ON d.tool_id = tl.id
WHERE tl.name = 'Sberbank';
 payment_date | payment_amount
--------------+----------------
 2023-05-08   |             25
(1 строка)
```

Индексы

1) Индекс для быстрого доступа к транзакциям по идентификатору инструмента:

до создания индекса:

```
studs=> EXPLAIN ANALYZE
SELECT * FROM transaction WHERE tool_id = 1;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on transaction  (cost=0.00..16.00 rows=2 width=142) (actual time=0.019..0.021 rows=1 loops=1)
   Filter: (tool_id = 1)
   Rows Removed by Filter: 9
 Planning Time: 0.096 ms
 Execution Time: 0.044 ms
(5 строк)
```

после:

```
studs=> CREATE INDEX idx_transaction_tool_id ON transaction (tool_id);
CREATE INDEX
```

```
studs=> EXPLAIN ANALYZE
SELECT * FROM transaction WHERE tool_id = 1;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on transaction  (cost=0.00..1.12 rows=1 width=142) (actual time=0.015..0.017 rows=1 loops=1)
   Filter: (tool_id = 1)
   Rows Removed by Filter: 9
 Planning Time: 0.083 ms
 Execution Time: 0.035 ms
(5 строк)
```

2) Индекс для быстрого получения информации об инвесторе и его портфеле инвестиций:

до:

```
studs=> EXPLAIN ANALYZE
SELECT i.name AS investor_name, ip.name AS portfolio_name
FROM investor i
INNER JOIN investment_portfolio ip ON i.id = ip.investor_id;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=14.72..28.40 rows=290 width=236) (actual time=0.040..0.044 rows=5 loops=1)
   Hash Cond: (ip.investor_id = i.id)
   ->  Seq Scan on investment_portfolio ip  (cost=0.00..12.90 rows=290 width=122) (actual time=0.006..0.007 rows=5 loops=1)
   ->  Hash  (cost=12.10..12.10 rows=210 width=122) (actual time=0.014..0.014 rows=4 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Seq Scan on investor i  (cost=0.00..12.10 rows=210 width=122) (actual time=0.005..0.006 rows=4 loops=1)
 Planning Time: 0.790 ms
 Execution Time: 0.102 ms
(8 строк)
```

после:

```
studs=> CREATE INDEX idx_investment_portfolio_investor_id ON investment_portfolio (investor_id);
CREATE INDEX
```

```
studs=> EXPLAIN ANALYZE
SELECT i.name AS investor_name, ip.name AS portfolio_name
FROM investor i
INNER JOIN investment_portfolio ip ON i.id = ip.investor_id;
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=1.11..14.05 rows=5 width=236) (actual time=0.050..0.054 rows=5 loops=1)
   Hash Cond: (i.id = ip.investor_id)
   ->  Seq Scan on investor i  (cost=0.00..12.10 rows=210 width=122) (actual time=0.014..0.015 rows=4 loops=1)
   ->  Hash  (cost=1.05..1.05 rows=5 width=122) (actual time=0.025..0.025 rows=5 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Seq Scan on investment_portfolio ip  (cost=0.00..1.05 rows=5 width=122) (actual time=0.008..0.011 rows=5 loops=1)
 Planning Time: 0.192 ms
 Execution Time: 0.091 ms
(8 строк)
```