**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе «Выбор предметной области**»**

по дисциплине «**Базы данных**»

Автор: Парфенова Ольга

Факультет: ПИиКТ

Группа: P33201

Преподаватель: Райла Мартин

Санкт-Петербург, 2023

# **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.

# **Код триггеров, функций, процедур**

**Триггер 1: Принудительное назначение VLAN для портов**

Этот триггер гарантирует, что порт не может быть назначен VLAN, который не принадлежит тому же коммутатору.

| CREATE OR REPLACE FUNCTION validate\_vlan\_assignment() RETURNS TRIGGER AS $$ BEGIN  IF NEW.vlan\_id IS NOT NULL THEN  IF (SELECT switch\_id FROM port WHERE id = NEW.id) !=  (SELECT switch\_id FROM vlan WHERE id = NEW.vlan\_id) THEN  RAISE EXCEPTION 'Invalid VLAN assignment for the port.';  END IF;  END IF;  RETURN NEW; END; $$ LANGUAGE plpgsql;  CREATE TRIGGER tr\_validate\_vlan\_assignment BEFORE INSERT OR UPDATE ON port FOR EACH ROW EXECUTE FUNCTION validate\_vlan\_assignment(); |
| --- |

**Триггер 2: Запрет удаления узлов с ассоциированными записями ARP**

Этот триггер предотвращает удаление узлов, у которых все еще есть связанные записи ARP.

| CREATE OR REPLACE FUNCTION prevent\_delete\_node\_with\_arp() RETURNS TRIGGER AS $$ BEGIN  IF EXISTS (SELECT 1 FROM arp WHERE node\_id = OLD.id) THEN  RAISE EXCEPTION 'Cannot delete a node with associated ARP records.';  END IF;  RETURN OLD; END; $$ LANGUAGE plpgsql;  CREATE TRIGGER tr\_prevent\_delete\_node\_with\_arp BEFORE DELETE ON node FOR EACH ROW EXECUTE FUNCTION prevent\_delete\_node\_with\_arp(); |
| --- |

**Триггер 3: Обновление скорости соединений оборудования**

Этот триггер обновляет скорость соединений оборудования в соответствии со скоростью подключенных портов.

| CREATE OR REPLACE FUNCTION update\_commutation\_speed() RETURNS TRIGGER AS $$ BEGIN  UPDATE commutation  SET speed = (SELECT connection\_speed FROM port WHERE id = NEW.equipment1\_id)  WHERE equipment1\_id = NEW.equipment1\_id;   UPDATE commutation  SET speed = (SELECT connection\_speed FROM port WHERE id = NEW.equipment2\_id)  WHERE equipment2\_id = NEW.equipment2\_id;   RETURN NEW; END; $$ LANGUAGE plpgsql;  CREATE TRIGGER tr\_update\_commutation\_speed AFTER UPDATE OF connection\_speed ON port FOR EACH ROW EXECUTE FUNCTION update\_commutation\_speed(); |
| --- |

**Функция добавления нового абонента**

| CREATE OR REPLACE PROCEDURE add\_new\_abonent(  p\_login TEXT,  p\_account\_number INT,  p\_name TEXT,  p\_surname TEXT,  p\_phone\_number TEXT,  p\_email TEXT,  p\_status TEXT,  p\_address\_id INT,  p\_note TEXT,  p\_provider\_id INT ) AS $$ BEGIN   IF EXISTS (SELECT 1 FROM abonent WHERE login = p\_login OR account\_number = p\_account\_number) THEN  RAISE EXCEPTION 'Login or account number already exists.';  END IF;   IF NOT EXISTS (SELECT 1 FROM address WHERE id = p\_address\_id) THEN  RAISE EXCEPTION 'Invalid address\_id.';  END IF;   IF NOT EXISTS (SELECT 1 FROM provider WHERE id = p\_provider\_id) THEN  RAISE EXCEPTION 'Invalid provider\_id.';  END IF;   INSERT INTO abonent (  login, account\_number, name, surname, phone\_number,  email, status, address\_id, note, provider\_id  ) VALUES (  p\_login, p\_account\_number, p\_name, p\_surname, p\_phone\_number,  p\_email, p\_status, p\_address\_id, p\_note, p\_provider\_id  ); END; $$ LANGUAGE plpgsql; |
| --- |

**Функция получения информации по абоненту**

| CREATE OR REPLACE FUNCTION get\_abonent\_info(p\_abonent\_id INT) RETURNS TABLE (  abonent\_id INT,  abonent\_login TEXT,  abonent\_name TEXT,  abonent\_surname TEXT,  abonent\_phone\_number TEXT,  abonent\_email TEXT,  abonent\_status TEXT,  abonent\_address TEXT,  abonent\_note TEXT,  provider\_name TEXT ) AS $$ BEGIN  RETURN QUERY  SELECT  a.id AS abonent\_id,  a.login AS abonent\_login,  a.name AS abonent\_name,  a.surname AS abonent\_surname,  a.phone\_number AS abonent\_phone\_number,  a.email AS abonent\_email,  a.status AS abonent\_status,  CONCAT(h.street, ' ', h.number) AS abonent\_address,  a.note AS abonent\_note,  p.name AS provider\_name  FROM  abonent a  JOIN address ad ON a.address\_id = ad.id  JOIN house h ON ad.house\_id = h.id  JOIN provider p ON a.provider\_id = p.id  WHERE  a.id = p\_abonent\_id; END; $$ LANGUAGE plpgsql; |
| --- |

**Функция очистки таблицы абонентов**

| CREATE OR REPLACE FUNCTION clear\_abonents\_table() RETURNS VOID AS $$ BEGIN  DELETE FROM abonent;  END; $$ LANGUAGE plpgsql; |
| --- |

# **Описание наиболее часто используемых сценариев при работе с базой данных**

1. Получение информации о абонентах:

SELECT \* FROM abonent;

2. Добавление нового запроса на обслуживание:

INSERT INTO request (abonent\_id, author\_id, create\_date, status, type, text) VALUES (abonent\_id\_value, author\_id\_value, current\_timestamp, 'Open', 'Service', 'Request details');

3. Получение информации о коммутации для конкретного абонента:

SELECT \* FROM commutation WHERE abonent\_id = abonent\_id\_value;

4. Обновление статуса порта на коммутаторе:

UPDATE port SET port\_status = 'Link up' WHERE id = port\_id\_value;

5. Получение диагностики соединения:

SELECT \* FROM connection\_diagnostics WHERE abonent\_id = abonent\_id\_value;

6. Добавление нового комментария к запросу:

INSERT INTO comment (request\_id, date, text, employee\_id) VALUES (request\_id\_value, current\_timestamp, 'Comment text', employee\_id\_value);

7. Получение списка абонентов с задолженностью:

SELECT \* FROM billing WHERE balance < 0;

8. Изменение данных о подключении порта:

UPDATE port SET description = 'New description' WHERE id = port\_id\_value;

9. Получение списка запросов с определенным статусом:

SELECT \* FROM request WHERE status = 'Open';

# **Описание индексов и обоснование их использования**

1. Индекс для таблицы `abonent` для улучшения выборки абонентов по статусу и балансу:

CREATE INDEX idx\_abonent\_status\_balance ON abonent (status, balance);

Этот индекс улучшит производительность запросов, которые фильтруют абонентов по их статусу и балансу.

2. Индекс для таблицы `request` для быстрого доступа к запросам по статусу:

CREATE INDEX idx\_request\_status ON request (status);

Этот индекс ускорит запросы, которые фильтруют запросы по их статусу.

3. Индексы для таблицы `commutation` для оптимизации запросов о коммутации абонентов:

CREATE INDEX idx\_comm\_abonent\_id ON commutation (abonent\_id);

CREATE INDEX idx\_comm\_switch\_id ON commutation (switch\_id);

CREATE INDEX idx\_comm\_port\_id ON commutation (port\_id);

Эти индексы помогут ускорить запросы, связанные с коммутацией абонентов.

4. Индексы для таблицы `connection\_diagnostics` для улучшения поиска информации о диагностике соединения:

CREATE INDEX idx\_conn\_diag\_abonent\_id ON connection\_diagnostics (abonent\_id);

CREATE INDEX idx\_conn\_diag\_employee\_id ON connection\_diagnostics (employee\_id);

Эти индексы улучшат производительность запросов, которые фильтруют диагностику соединения по абоненту или сотруднику.

5. Индекс для таблицы `billing` для быстрого доступа к балансам абонентов:

CREATE INDEX idx\_billing\_balance ON billing (balance);

Этот индекс ускорит запросы, связанные с балансами абонентов.

6. Индексы для таблицы `mac\_address` для оптимизации работы с MAC-адресами:

CREATE INDEX idx\_mac\_address\_port\_id ON mac\_address (port\_id);

CREATE INDEX idx\_mac\_address\_mac\_address ON mac\_address (mac\_address);

Эти индексы помогут ускорить запросы, связанные с MAC-адресами.

7. Индексы для таблицы `port` для оптимизации запросов по портам:

CREATE INDEX idx\_port\_switch\_id ON port (switch\_id);

CREATE INDEX idx\_port\_vlan\_id ON port (vlan\_id);

Эти индексы улучшат производительность запросов, связанных с портами коммутаторов.

Время выполнения запросов к таблицам до добавления индексов:

0,15 ms - 0,2 ms

Время выполнения запросов к таблицам после добавления индексов:

0,01 ms - 0,03 ms

Время выполнения уменьшилось в 5-8 раз.

# **Выводы по работе**

За эту лабораторную работу я научилась оптимизировать запросы в бд с помощью использования индексов, создавать ограничения целостности, используя триггеры, создавать функции и процедуры в sql с помощь plpgsql, анализировать запросы на затрату времени и сложность с помощью команды explain analyse.