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

Индексы

| CREATE INDEX idx\_abonent\_login ON abonent(login); CREATE INDEX idx\_abonent\_status ON abonent(status); CREATE INDEX idx\_abonent\_address\_id ON abonent(address\_id); CREATE INDEX idx\_abonent\_provider\_id ON abonent(provider\_id); CREATE INDEX idx\_port\_switch\_id ON port(switch\_id); CREATE INDEX idx\_connection\_diagnostics\_status ON connection\_diagnostics(status); |
| --- |