# Документация по базе данных

## 1. Структура баз данных

### 1.1 Центральная база данных (central)

#### Таблица tenants

CREATE TABLE tenants (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 domain VARCHAR(255) UNIQUE NOT NULL,  
 database VARCHAR(255) UNIQUE NOT NULL,  
 settings JSON,  
 status ENUM('active', 'inactive', 'suspended') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL  
);

#### Таблица users (центральная)

CREATE TABLE users (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 email VARCHAR(255) UNIQUE NOT NULL,  
 password VARCHAR(255) NOT NULL,  
 role ENUM('super\_admin', 'admin') NOT NULL,  
 tenant\_id BIGINT UNSIGNED NULL,  
 status ENUM('active', 'inactive') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL,  
 FOREIGN KEY (tenant\_id) REFERENCES tenants(id)  
);

#### Таблица history

CREATE TABLE history (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 table\_name VARCHAR(255) NOT NULL,  
 record\_id BIGINT UNSIGNED NOT NULL,  
 action ENUM('create', 'update', 'delete') NOT NULL,  
 old\_data JSON NULL,  
 new\_data JSON NULL,  
 user\_id BIGINT UNSIGNED NOT NULL,  
 ip\_address VARCHAR(45) NOT NULL,  
 user\_agent TEXT NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (user\_id) REFERENCES users(id)  
);

### 1.2 База данных клиента (tenant\_\*)

#### Таблица users (клиентская)

CREATE TABLE users (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 email VARCHAR(255) UNIQUE NOT NULL,  
 password VARCHAR(255) NOT NULL,  
 role ENUM('admin', 'manager', 'operator') NOT NULL,  
 status ENUM('active', 'inactive') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL  
);

#### Таблица subscribers

CREATE TABLE subscribers (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 address TEXT NOT NULL,  
 phone VARCHAR(20) NOT NULL,  
 balance DECIMAL(10,2) DEFAULT 0.00,  
 status ENUM('active', 'inactive', 'blocked') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL  
);

#### Таблица services

CREATE TABLE services (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 type VARCHAR(50) NOT NULL,  
 price DECIMAL(10,2) NOT NULL,  
 settings JSON,  
 status ENUM('active', 'inactive') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL  
);

#### Таблица meters

CREATE TABLE meters (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 subscriber\_id BIGINT UNSIGNED NOT NULL,  
 type VARCHAR(50) NOT NULL,  
 number VARCHAR(50) NOT NULL,  
 last\_value DECIMAL(10,2) DEFAULT 0.00,  
 status ENUM('active', 'inactive') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL,  
 FOREIGN KEY (subscriber\_id) REFERENCES subscribers(id)  
);

#### Таблица meter\_readings

CREATE TABLE meter\_readings (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 meter\_id BIGINT UNSIGNED NOT NULL,  
 value DECIMAL(10,2) NOT NULL,  
 reading\_date DATE NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 created\_by BIGINT UNSIGNED NOT NULL,  
 FOREIGN KEY (meter\_id) REFERENCES meters(id),  
 FOREIGN KEY (created\_by) REFERENCES users(id)  
);

#### Таблица bills

CREATE TABLE bills (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 subscriber\_id BIGINT UNSIGNED NOT NULL,  
 number VARCHAR(50) NOT NULL,  
 amount DECIMAL(10,2) NOT NULL,  
 tax\_amount DECIMAL(10,2) NOT NULL DEFAULT 0.00,  
 total\_amount DECIMAL(10,2) NOT NULL,  
 status ENUM('draft', 'sent', 'paid', 'cancelled') DEFAULT 'draft',  
 due\_date DATE NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 created\_by BIGINT UNSIGNED NOT NULL,  
 FOREIGN KEY (subscriber\_id) REFERENCES subscribers(id),  
 FOREIGN KEY (created\_by) REFERENCES users(id)  
);

#### Таблица payments

CREATE TABLE payments (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 bill\_id BIGINT UNSIGNED NOT NULL,  
 amount DECIMAL(10,2) NOT NULL,  
 method VARCHAR(50) NOT NULL,  
 status ENUM('pending', 'completed', 'failed') DEFAULT 'pending',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 created\_by BIGINT UNSIGNED NOT NULL,  
 FOREIGN KEY (bill\_id) REFERENCES bills(id),  
 FOREIGN KEY (created\_by) REFERENCES users(id)  
);

#### Таблица tariffs

CREATE TABLE tariffs (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 type VARCHAR(50) NOT NULL,  
 price DECIMAL(10,2) NOT NULL,  
 settings JSON,  
 status ENUM('active', 'inactive') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL  
);

#### Таблица tax\_rates

CREATE TABLE tax\_rates (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 rate DECIMAL(5,2) NOT NULL,  
 type ENUM('percentage', 'fixed') NOT NULL,  
 is\_default BOOLEAN DEFAULT FALSE,  
 status ENUM('active', 'inactive') DEFAULT 'active',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 deleted\_at TIMESTAMP NULL,  
 deleted\_by BIGINT UNSIGNED NULL,  
 deleted\_reason TEXT NULL  
);

#### Таблица tax\_reports

CREATE TABLE tax\_reports (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 period\_start DATE NOT NULL,  
 period\_end DATE NOT NULL,  
 total\_amount DECIMAL(10,2) NOT NULL,  
 total\_tax DECIMAL(10,2) NOT NULL,  
 status ENUM('draft', 'submitted', 'approved') DEFAULT 'draft',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 created\_by BIGINT UNSIGNED NOT NULL,  
 submitted\_at TIMESTAMP NULL,  
 approved\_at TIMESTAMP NULL,  
 approved\_by BIGINT UNSIGNED NULL,  
 FOREIGN KEY (created\_by) REFERENCES users(id),  
 FOREIGN KEY (approved\_by) REFERENCES users(id)  
);

#### Таблица tickets

CREATE TABLE tickets (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 tenant\_id BIGINT UNSIGNED NOT NULL,  
 subscriber\_id BIGINT UNSIGNED NULL,  
 title VARCHAR(255) NOT NULL,  
 description TEXT NOT NULL,  
 status ENUM('new', 'in\_progress', 'resolved', 'closed') DEFAULT 'new',  
 priority ENUM('low', 'medium', 'high', 'urgent') DEFAULT 'medium',  
 category VARCHAR(50) NOT NULL,  
 created\_by BIGINT UNSIGNED NOT NULL,  
 assigned\_to BIGINT UNSIGNED NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 resolved\_at TIMESTAMP NULL,  
 closed\_at TIMESTAMP NULL,  
 FOREIGN KEY (tenant\_id) REFERENCES tenants(id),  
 FOREIGN KEY (subscriber\_id) REFERENCES subscribers(id),  
 FOREIGN KEY (created\_by) REFERENCES users(id),  
 FOREIGN KEY (assigned\_to) REFERENCES users(id)  
);

#### Таблица ticket\_comments

CREATE TABLE ticket\_comments (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 ticket\_id BIGINT UNSIGNED NOT NULL,  
 user\_id BIGINT UNSIGNED NOT NULL,  
 comment TEXT NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (ticket\_id) REFERENCES tickets(id),  
 FOREIGN KEY (user\_id) REFERENCES users(id)  
);

#### Таблица ticket\_attachments

CREATE TABLE ticket\_attachments (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 ticket\_id BIGINT UNSIGNED NOT NULL,  
 file\_name VARCHAR(255) NOT NULL,  
 file\_path VARCHAR(255) NOT NULL,  
 file\_type VARCHAR(50) NOT NULL,  
 file\_size INT UNSIGNED NOT NULL,  
 uploaded\_by BIGINT UNSIGNED NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (ticket\_id) REFERENCES tickets(id),  
 FOREIGN KEY (uploaded\_by) REFERENCES users(id)  
);

#### Таблица ticket\_status\_history

CREATE TABLE ticket\_status\_history (  
 id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
 ticket\_id BIGINT UNSIGNED NOT NULL,  
 old\_status ENUM('new', 'in\_progress', 'resolved', 'closed') NOT NULL,  
 new\_status ENUM('new', 'in\_progress', 'resolved', 'closed') NOT NULL,  
 changed\_by BIGINT UNSIGNED NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (ticket\_id) REFERENCES tickets(id),  
 FOREIGN KEY (changed\_by) REFERENCES users(id)  
);

## 2. Основные запросы

### 2.1 Запросы для работы с абонентами

#### Получение списка активных абонентов с балансом

SELECT   
 s.id,  
 s.name,  
 s.address,  
 s.phone,  
 s.balance,  
 COUNT(m.id) as meters\_count  
FROM subscribers s  
LEFT JOIN meters m ON s.id = m.subscriber\_id  
WHERE s.status = 'active'  
GROUP BY s.id  
ORDER BY s.name;

#### Получение детальной информации об абоненте

SELECT   
 s.\*,  
 COUNT(m.id) as meters\_count,  
 COUNT(b.id) as bills\_count,  
 SUM(CASE WHEN b.status = 'paid' THEN b.total\_amount ELSE 0 END) as total\_paid  
FROM subscribers s  
LEFT JOIN meters m ON s.id = m.subscriber\_id  
LEFT JOIN bills b ON s.id = b.subscriber\_id  
WHERE s.id = ?  
GROUP BY s.id;

### 2.2 Запросы для работы со счетчиками

#### Получение последних показаний счетчиков абонента

SELECT   
 m.\*,  
 mr.value as last\_reading,  
 mr.reading\_date  
FROM meters m  
LEFT JOIN meter\_readings mr ON m.id = mr.meter\_id  
WHERE m.subscriber\_id = ?  
AND mr.id IN (  
 SELECT MAX(id)  
 FROM meter\_readings  
 GROUP BY meter\_id  
)  
ORDER BY m.type;

#### Получение истории показаний счетчика

SELECT   
 mr.\*,  
 u.name as created\_by\_name  
FROM meter\_readings mr  
JOIN users u ON mr.created\_by = u.id  
WHERE mr.meter\_id = ?  
ORDER BY mr.reading\_date DESC;

### 2.3 Запросы для работы со счетами

#### Получение активных счетов абонента

SELECT   
 b.\*,  
 COUNT(p.id) as payments\_count,  
 SUM(p.amount) as paid\_amount  
FROM bills b  
LEFT JOIN payments p ON b.id = p.bill\_id  
WHERE b.subscriber\_id = ?  
AND b.status IN ('sent', 'partial')  
GROUP BY b.id  
ORDER BY b.due\_date;

#### Получение статистики по счетам

SELECT   
 COUNT(\*) as total\_bills,  
 SUM(CASE WHEN status = 'paid' THEN 1 ELSE 0 END) as paid\_bills,  
 SUM(CASE WHEN status = 'sent' THEN 1 ELSE 0 END) as sent\_bills,  
 SUM(total\_amount) as total\_amount,  
 SUM(CASE WHEN status = 'paid' THEN total\_amount ELSE 0 END) as paid\_amount  
FROM bills  
WHERE created\_at BETWEEN ? AND ?;

### 2.4 Запросы для работы с налогами

#### Получение налоговой отчетности за период

SELECT   
 tr.\*,  
 u.name as created\_by\_name,  
 au.name as approved\_by\_name  
FROM tax\_reports tr  
JOIN users u ON tr.created\_by = u.id  
LEFT JOIN users au ON tr.approved\_by = au.id  
WHERE tr.period\_start BETWEEN ? AND ?  
ORDER BY tr.period\_start DESC;

#### Получение деталей по налогам

SELECT   
 b.id as bill\_id,  
 b.number as bill\_number,  
 b.total\_amount,  
 b.tax\_amount,  
 tr.name as tax\_rate\_name,  
 tr.rate as tax\_rate  
FROM bills b  
JOIN tax\_rates tr ON b.tax\_rate\_id = tr.id  
WHERE b.created\_at BETWEEN ? AND ?  
ORDER BY b.created\_at;

### 2.5 Запросы для работы с тикетами

#### Получение списка тикетов с фильтрацией

SELECT   
 t.\*,  
 s.name as subscriber\_name,  
 c.name as created\_by\_name,  
 a.name as assigned\_to\_name,  
 COUNT(tc.id) as comments\_count,  
 COUNT(ta.id) as attachments\_count  
FROM tickets t  
LEFT JOIN subscribers s ON t.subscriber\_id = s.id  
JOIN users c ON t.created\_by = c.id  
LEFT JOIN users a ON t.assigned\_to = a.id  
LEFT JOIN ticket\_comments tc ON t.id = tc.ticket\_id  
LEFT JOIN ticket\_attachments ta ON t.id = ta.ticket\_id  
WHERE t.tenant\_id = ?  
AND t.status IN (?, ?)  
AND t.priority IN (?, ?)  
GROUP BY t.id  
ORDER BY t.created\_at DESC;

#### Получение детальной информации о тикете

SELECT   
 t.\*,  
 s.name as subscriber\_name,  
 c.name as created\_by\_name,  
 a.name as assigned\_to\_name,  
 GROUP\_CONCAT(DISTINCT tc.comment ORDER BY tc.created\_at) as comments,  
 GROUP\_CONCAT(DISTINCT ta.file\_name) as attachments  
FROM tickets t  
LEFT JOIN subscribers s ON t.subscriber\_id = s.id  
JOIN users c ON t.created\_by = c.id  
LEFT JOIN users a ON t.assigned\_to = a.id  
LEFT JOIN ticket\_comments tc ON t.id = tc.ticket\_id  
LEFT JOIN ticket\_attachments ta ON t.id = ta.ticket\_id  
WHERE t.id = ?  
GROUP BY t.id;

#### Получение статистики по тикетам

SELECT   
 COUNT(\*) as total\_tickets,  
 SUM(CASE WHEN status = 'new' THEN 1 ELSE 0 END) as new\_tickets,  
 SUM(CASE WHEN status = 'in\_progress' THEN 1 ELSE 0 END) as in\_progress\_tickets,  
 SUM(CASE WHEN status = 'resolved' THEN 1 ELSE 0 END) as resolved\_tickets,  
 SUM(CASE WHEN status = 'closed' THEN 1 ELSE 0 END) as closed\_tickets,  
 AVG(TIMESTAMPDIFF(HOUR, created\_at, COALESCE(resolved\_at, NOW()))) as avg\_resolution\_time  
FROM tickets  
WHERE tenant\_id = ?  
AND created\_at BETWEEN ? AND ?;

## 3. Индексы

### 3.1 Центральная база данных

-- Индексы для таблицы tenants  
CREATE INDEX idx\_tenants\_domain ON tenants(domain);  
CREATE INDEX idx\_tenants\_status ON tenants(status);  
  
-- Индексы для таблицы users  
CREATE INDEX idx\_users\_email ON users(email);  
CREATE INDEX idx\_users\_tenant\_id ON users(tenant\_id);  
CREATE INDEX idx\_users\_status ON users(status);  
  
-- Индексы для таблицы history  
CREATE INDEX idx\_history\_table\_record ON history(table\_name, record\_id);  
CREATE INDEX idx\_history\_user\_id ON history(user\_id);  
CREATE INDEX idx\_history\_created\_at ON history(created\_at);

### 3.2 База данных клиента

-- Индексы для таблицы subscribers  
CREATE INDEX idx\_subscribers\_status ON subscribers(status);  
CREATE INDEX idx\_subscribers\_phone ON subscribers(phone);  
  
-- Индексы для таблицы meters  
CREATE INDEX idx\_meters\_subscriber ON meters(subscriber\_id);  
CREATE INDEX idx\_meters\_type ON meters(type);  
CREATE INDEX idx\_meters\_status ON meters(status);  
  
-- Индексы для таблицы meter\_readings  
CREATE INDEX idx\_meter\_readings\_meter ON meter\_readings(meter\_id);  
CREATE INDEX idx\_meter\_readings\_date ON meter\_readings(reading\_date);  
  
-- Индексы для таблицы bills  
CREATE INDEX idx\_bills\_subscriber ON bills(subscriber\_id);  
CREATE INDEX idx\_bills\_status ON bills(status);  
CREATE INDEX idx\_bills\_due\_date ON bills(due\_date);  
  
-- Индексы для таблицы payments  
CREATE INDEX idx\_payments\_bill ON payments(bill\_id);  
CREATE INDEX idx\_payments\_status ON payments(status);  
CREATE INDEX idx\_payments\_created\_at ON payments(created\_at);

## 4. Триггеры

### 4.1 Триггеры для обновления баланса

#### При добавлении платежа

DELIMITER //  
  
CREATE TRIGGER after\_payment\_insert  
AFTER INSERT ON payments  
FOR EACH ROW  
BEGIN  
 UPDATE subscribers s  
 JOIN bills b ON s.id = b.subscriber\_id  
 SET s.balance = s.balance - NEW.amount  
 WHERE b.id = NEW.bill\_id;  
END //  
  
DELIMITER ;

#### При удалении платежа

DELIMITER //  
  
CREATE TRIGGER after\_payment\_delete  
AFTER DELETE ON payments  
FOR EACH ROW  
BEGIN  
 UPDATE subscribers s  
 JOIN bills b ON s.id = b.subscriber\_id  
 SET s.balance = s.balance + OLD.amount  
 WHERE b.id = OLD.bill\_id;  
END //  
  
DELIMITER ;

#### При изменении платежа

DELIMITER //  
  
CREATE TRIGGER after\_payment\_update  
AFTER UPDATE ON payments  
FOR EACH ROW  
BEGIN  
 UPDATE subscribers s  
 JOIN bills b ON s.id = b.subscriber\_id  
 SET s.balance = s.balance + OLD.amount - NEW.amount  
 WHERE b.id = NEW.bill\_id;  
END //  
  
DELIMITER ;

#### При добавлении счета

DELIMITER //  
  
CREATE TRIGGER after\_bill\_insert  
AFTER INSERT ON bills  
FOR EACH ROW  
BEGIN  
 UPDATE subscribers s  
 SET s.balance = s.balance + NEW.total\_amount  
 WHERE s.id = NEW.subscriber\_id;  
END //  
  
DELIMITER ;

#### При удалении счета

DELIMITER //  
  
CREATE TRIGGER after\_bill\_delete  
AFTER DELETE ON bills  
FOR EACH ROW  
BEGIN  
 UPDATE subscribers s  
 SET s.balance = s.balance - OLD.total\_amount  
 WHERE s.id = OLD.subscriber\_id;  
END //  
  
DELIMITER ;

#### При изменении счета

DELIMITER //  
  
CREATE TRIGGER after\_bill\_update  
AFTER UPDATE ON bills  
FOR EACH ROW  
BEGIN  
 UPDATE subscribers s  
 SET s.balance = s.balance - OLD.total\_amount + NEW.total\_amount  
 WHERE s.id = NEW.subscriber\_id;  
END //  
  
DELIMITER ;

### 4.2 Логирование изменений

### 4.3 Триггеры для тикетов  
  
#### Логирование изменения статуса тикета  
```sql  
DELIMITER //  
  
CREATE TRIGGER after\_ticket\_status\_update  
AFTER UPDATE ON tickets  
FOR EACH ROW  
BEGIN  
 IF OLD.status != NEW.status THEN  
 INSERT INTO ticket\_status\_history (  
 ticket\_id,  
 old\_status,  
 new\_status,  
 changed\_by  
 )  
 VALUES (  
 NEW.id,  
 OLD.status,  
 NEW.status,  
 @current\_user\_id  
 );  
 END IF;  
END //  
  
DELIMITER ;