DB 구축

PostgreSQL 및 TimescaleDB 설치 상세 가이드

1. PostgreSQL 설치

`sudo apt update` 명령어를 실행하여 서버의 패키지 목록을 최신 상태로 업데이트합니다.
`sudo apt install -y postgresql postgresql-contrib` 명령어를 실행하여 PostgreSQL 및 추가 모듈을 설치합니다.

2. TimescaleDB 설치

TimescaleDB는 PostgreSQL의 확장 기능이므로, TimescaleDB 전용 패키지 저장소를 추가해야 합니다. `sudo sh -c 'echo "deb https://packagecloud.io/timescale/timescaledb/ubuntu/ $(lsb_release -c -s) main" > /etc/apt/sources.list.d/timescaledb.list'` 위 명령어를 실행하여 TimescaleDB 저장소를 추가합니다.

`wget --quiet -O - https://packagecloud.io/timescale/timescaledb/gpgkey | sudo apt-key add -` 위 명령어를 실행하여 GPG 키를 추가합니다.

`sudo apt update` 명령어로 저장소 정보를 업데이트합니다.

`sudo apt install -y timescaledb-2-postgresql-16` 명령어로 TimescaleDB 확장 기능을 설치합니다.

설치 후 TimescaleDB를 PostgreSQL에 적용하기 위해 `sudo service postgresql restart` 명령어를 실행하여 PostgreSQL 서비스를 재시작합니다.

3. PostgreSQL 설정 및 DB 생성

`sudo -i -u postgres` 명령어를 실행하여 `postgres` 사용자(PostgreSQL의 기본 관리자)로 전환합니다. `psql` 명령어를 실행하여 PostgreSQL 명령 프롬프트에 접속합니다.

데이터베이스를 생성합니다. `CREATE DATABASE iot_care;`

생성한 데이터베이스에 접속합니다. `\c iot_care`

TimescaleDB 확장 기능을 활성화합니다. `CREATE EXTENSION IF NOT EXISTS timescaledb;`

4. 스키마 생성

제공된 SQL 스크립트를 `psql` 프롬프트에 복사하여 붙여넣고 실행합니다.

```sql
-- ========================
-- 사용자 및 디바이스 메타데이터
-- ========================
CREATE TABLE IF NOT EXISTS users (
    user_id UUID PRIMARY KEY,
    user_role VARCHAR(20) NOT NULL, -- 'client', 'guardian', 'social_worker'
    user_name TEXT NOT NULL,
    email TEXT UNIQUE,
    phone_number TEXT
);

CREATE TABLE IF NOT EXISTS devices (
    device_id VARCHAR(64) PRIMARY KEY, -- Arduino 칩 ID 또는 고유 ID
    user_id UUID REFERENCES users(user_id) ON DELETE CASCADE,
    location_label TEXT, -- '방', '거실', '주방'
    installed_at TIMESTAMPTZ DEFAULT now()
);

-- ========================
-- Raw 데이터 테이블 (센서별로 분리)
-- ========================
CREATE TABLE IF NOT EXISTS sensor_raw_mq7 (
    time TIMESTAMPTZ NOT NULL,
    device_id VARCHAR(64) REFERENCES devices(device_id) ON DELETE CASCADE,
    adc_value INTEGER NOT NULL,
    PRIMARY KEY (time, device_id)
);
SELECT create_hypertable('sensor_raw_mq7', 'time');

CREATE TABLE IF NOT EXISTS sensor_raw_dht (
```

```
    time TIMESTAMPTZ NOT NULL,
    device_id VARCHAR(64) REFERENCES devices(device_id) ON DELETE CASCADE,
    temperature NUMERIC NOT NULL,
    humidity NUMERIC NOT NULL,
    PRIMARY KEY (time, device_id)
);
SELECT create_hypertable('sensor_raw_dht', 'time');

CREATE TABLE IF NOT EXISTS sensor_raw_door (
    time TIMESTAMPTZ NOT NULL,
    device_id VARCHAR(64) REFERENCES devices(device_id) ON DELETE CASCADE,
    status BOOLEAN NOT NULL, -- true: OPEN, false: CLOSE
    PRIMARY KEY (time, device_id)
);
SELECT create_hypertable('sensor_raw_door', 'time');

CREATE TABLE IF NOT EXISTS sensor_raw_loadcell (
    time TIMESTAMPTZ NOT NULL,
    device_id VARCHAR(64) REFERENCES devices(device_id) ON DELETE CASCADE,
    weight NUMERIC NOT NULL,
    PRIMARY KEY (time, device_id)
);
SELECT create_hypertable('sensor_raw_loadcell', 'time');

CREATE TABLE IF NOT EXISTS sensor_raw_gyro (
    time TIMESTAMPTZ NOT NULL,
    device_id VARCHAR(64) REFERENCES devices(device_id) ON DELETE CASCADE,
    accel_x NUMERIC NOT NULL,
    accel_y NUMERIC NOT NULL,
    accel_z NUMERIC NOT NULL,
    gyro_x NUMERIC NOT NULL,
    gyro_y NUMERIC NOT NULL,
    gyro_z NUMERIC NOT NULL,
    PRIMARY KEY (time, device_id)
);
SELECT create_hypertable('sensor_raw_gyro', 'time');

-- ========================
-- 종합 정보 테이블 (가공된 데이터 저장)
-- ========================
CREATE TABLE IF NOT EXISTS aggregated_data (
    time TIMESTAMPTZ NOT NULL,
    device_id VARCHAR(64) REFERENCES devices(device_id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(user_id),
    co_ppm NUMERIC,
    temp_c NUMERIC,
    humidity_rh NUMERIC,
    motion_detected BOOLEAN,
    door_status BOOLEAN,
    weight_kg NUMERIC,
    fall_detected BOOLEAN,
    PRIMARY KEY (time, device_id)
);
SELECT create_hypertable('aggregated_data', 'time');

-- ========================
-- 알림 및 감사 로그 테이블
-- ========================
```

```sql
CREATE TABLE IF NOT EXISTS alerts (
    alert_id BIGSERIAL PRIMARY KEY,
    device_id VARCHAR(64) REFERENCES devices(device_id) ON DELETE CASCADE,
    rule_code TEXT NOT NULL,
    severity TEXT NOT NULL,
    message TEXT NOT NULL,
    triggered_at TIMESTAMPTZ DEFAULT now()
);

CREATE TABLE IF NOT EXISTS admin_actions (
    action_id BIGSERIAL PRIMARY KEY,
    user_id UUID REFERENCES users(user_id),
    action_type TEXT NOT NULL, -- 'alert_ack', 'device_config_change'
    action_context JSONB,
    action_at TIMESTAMPTZ DEFAULT now()
);
```

PostgreSQL 설정 수정 => TimescaleDB 활성화

1. **PostgreSQL 설정 파일 열기**: `sudo nano /etc/postgresql/16/main/postgresql.conf` 위 명령어를 입력하여 PostgreSQL 설정 파일을 편집기(`nano`)로 엽니다.
2. `shared_preload_libraries` **설정 수정**: 파일 내용에서 `shared_preload_libraries` 부분을 찾습니다. 보통 주석(`#`) 처리되어 있습니다. `#shared_preload_libraries = ''` 이 부분을 다음과 같이 수정합니다. `shared_preload_libraries = 'timescaledb'` 만약 이미 다른 라이브러리가 설정되어 있다면 쉼표(`,`)를 사용하여 `timescaledb` 를 추가합니다. `shared_preload_libraries = 'pg_stat_statements, timescaledb'`
3. **파일 저장 및 편집기 종료**: `Ctrl + O` 를 눌러 파일을 저장하고, `Enter` 를 누른 뒤 `Ctrl + X` 를 눌러 편집기를 종료합니다.
4. **PostgreSQL 서비스 재시작**: `sudo systemctl restart postgresql` 위 명령어를 입력하여 PostgreSQL 서비스를 재시작합니다. 라이브러리가 성공적으로 로드됩니다.
5. `psql` **접속 및 TimescaleDB 활성화**: `sudo -i -u postgres psql` `\c iot_care` `CREATE EXTENSION IF NOT EXISTS timescaledb;` 이제 이 명령어를 다시 실행하면 정상적으로 TimescaleDB가 활성화됩니다.

초기 보안 (로컬 전용 바인딩)

기본적으로 Debian/Ubuntu 패키지는 `listen_addresses = 'localhost'` 로 설정되어 **외부 접속이 차단**됩니다.

경로:

- 메인 설정: `/etc/postgresql/16/main/postgresql.conf`
- 인증 설정: `/etc/postgresql/16/main/pg_hba.conf`

확인:

```
sudo grep -E "^#?listen_addresses|^#?port" /etc/postgresql/16/main/postgresql.conf
sudo systemctl restart postgresql

```

> 2단계에서 **WAS/QnA 서버 IP만 화이트리스트 허용**하도록 `listen_addresses` /UFW/ `pg_hba.conf` 를 업데이트합니다.