UNIVERSITATEA TEHNICĂ "GH ASACHI" IAŞI
FACULTATEA AUTOMATICĂ ŞI CALCULATOARE
SPECIALIZAREA TEHNOLOGIA INFORMAȚIEI

DISCIPLINA PROIECTAREA BAZELOR DE DATE

Gestiunea unei rețele

Coordonator,
Cătălin Mironeanu

Student,
Vrînceanu George

Iaşi, 2021

Titlu proiect : Gestiunea rețelei

Analiza, proiectarea şi implementarea unei baze de date şi a aplicației aferente care să realizeze conectarea & gestiunea la o rețea de internet (în proiectarea acestei teme s-a folosit un singur switch şi un singur router).

Descrierea cerințelor şi modul de organizare al proiectului

Gestiunea unei rețele eficiente de internet poate fi o provocare indiferent de modul în care aceasta a fost gândită şi implementată, din cauza potențialului "rău" ce ar putea fi generat printr-o simplă greşeală. Din acest motiv, acest proiect ar trebui sa "simuleze" mai mult o rețea locala, complexitatea temei fiind astfel drastic micşorată. Astfel, în acest proiect am folosit tabele ce simulează următoarele dispozitive hardware:

- dispozitive, în care se introduce tipul de dispozitiv cu care te-ai conectat la rețea; dispozitive permise: computer, printer, laptop, smarthpone;
- wireless access point – 4 la număr - plasate a.î. un utilizator să se poată conecta la internet de pe laptop / smarthpone din interiorul incintei;
- router cu 4 interfețe IP, cu rolul de a conecta aceasta rețea cu rețele externe, ex.: ISP;
- switch de 48 de porturi, în care se salvează toate dispozitivele conectate – puncte de acces wireless, calculatoare, servere etc.

Informațiile de care ar fi nevoie pentru a crea această rețea sunt legate de:

- dispozitivul conectat: ne interesează să ştim ce fel de dispozitiv se conectează la rețea (de exemplu, să ştim dacă utilizatorul s-a conectat de pe un telefon, sau dacă pur şi simplu s-a conectat cu un calculator la rețea);
- adresa fizica a dispozitivului, numita adesea MAC: această adresă fiind unică fiecărui dispozitiv în parte, munca de identificare a dispozitivului ne este semnificativ uşurată;
- adresa IP a dispozitivului: asemeni adresei MAC, ajută la identificarea dispozitivului;
- sistemul de operare a dispozitivului: în cazul in care se doreşte o anumită deosebire pe baza sistemului de operare folosit;
- wireless-ul la care este conectat user-ul: numai în cazul în care user-ul foloseşte un smarthpone / laptop - acest lucru este total transparent pentru user, SSID-ul fiind comun pentru întreaga rețea;
- port-ul la care este asociat dispozitivul in tabela switch: atât computerele, imprimantele cât şi wireless-urile.

Descrierea funcțională a aplicației

Principalele lucruri ce ar trebui sa ni le ofere rețeaua ar trebui să fie reprezentate de:

- posibilitatea de a vedea ce dispozitiv s-a conectat la rețea;
- tipul dispozitivului – dacă s-a conectat la switch / wireless;

Descrierea detaliată a entităților şi a relațiilor dintre tabele

Tabelele din această aplicație sunt:

- access_point;
- access_point_bridge;
- devices;
- router;
- server;
- switch;

În proiectarea acestui proiect nu s-au folosit relaţii între toate tabelele, interacţiunea unora dintre acestea fiind realizată prin intermediul **triggerelor** pentru o apropiere mai buna față de o rețea adevărata. Astfel, foreign key-urile au fost păstrate doar pentru tabelul de router (o relație de **1:1 switch**) şi pentru cel de server (o relație de **n:1 switch**). Mai departe, vom face abstracție de aceste două tabele, şi vom vorbi doar despre tabele de devices, access_point & access_point_bridge. Odată ce un dispozitiv nou este introdus la rețea prin intermediul tabelei devices, acesta este adăugat în tabela access_point sau switch automat.

PS1: Prin folosirea foreign key-urilor acest lucru era cam dificil, deoarece tabela switch se doreşte a fi tabela părinte, însă, tot odată ea să fie populată ultima.

PS2: Tabela devices ar fi trebuit să aibă ca şi PK un foreign key de la switch, şi un alt UK din tabela access_point. Totodată cheia unică ce aparținea de acess_point şi care era referențiată în tabela devices, era de tipul FK din tabela access_point_bridge care la rândul ei era tot de acest tip, numai că aparținea de tabela switch.

În fiecare tabel sunt folosite constrângeri, prin intermediul triggerelor pentru a se verifica dacă tipul de date introdus este valid (prin folosirea de regex).

- valori verificate prin regex: MAC, IP;

- valori verificate prin trigger asemănător constrângerii CHECK: tipul dispozitivului, sistemul de operare, tipul serverului.

De asemenea, având un switch de 48 de porturi, trebuie avut în considerare o tabela CAM (cu adresele MAC a dispozitivelor, VLAN & port_id), astfel s-au împărțit cele 48 de porturi în următorul mod:

- primele 40 pentru dispozitive ce se conectează direct la internet prin intermediul cablurilor – pentru calculatoare şi imprimante;

- următoarele 4 (41 – 44) pentru cele 4 wireless-uri;

- porturile 45, 46, 47 pentru servere, rămânând ca portul 48 să fie pentru router;

Acestor porturi, VLAN-ul le este asignat automat prin intermediul triggerelor (din 4 în 4, ex.: porturile de la 1 – 4 au VLAN 1, 5 – 8 au VLAN 2 ... 45 – 48 VLAN 12).

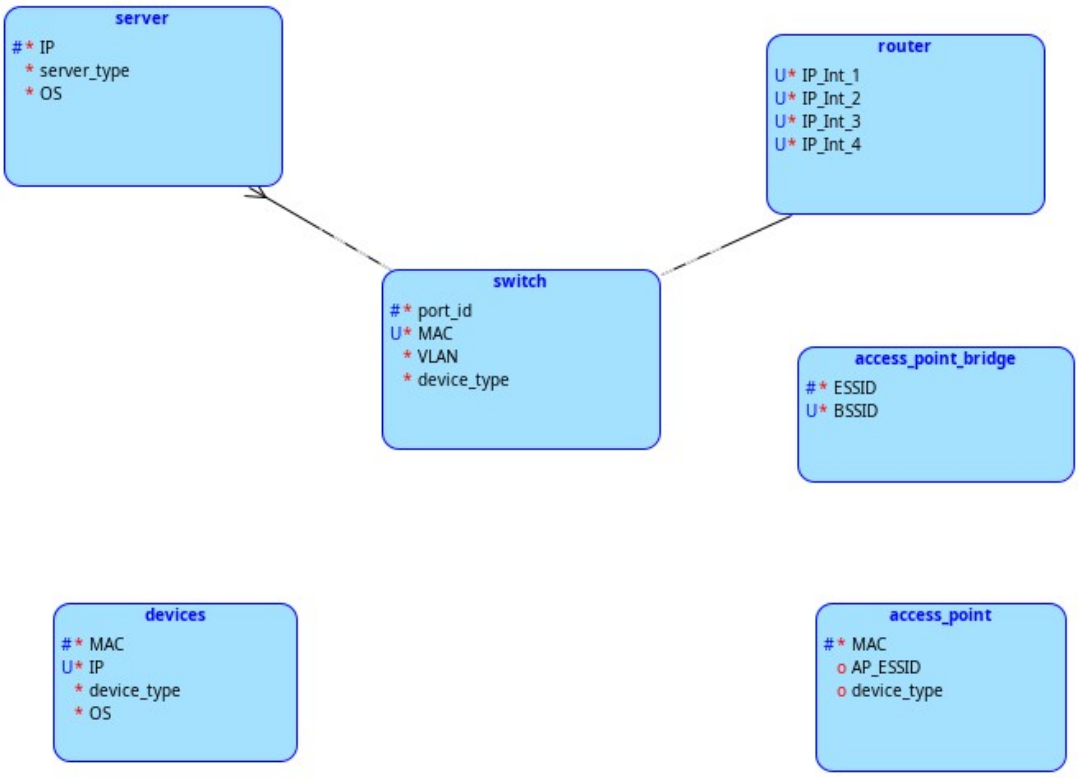
Cu toate că s-au folosit triggere pentru legătura dintre cele 4 tabele, relațiile de 1:n,. ar exista – nu în schemă, ci într-un mod „transparent”. Odată un device nou introdus, înainte de a se adăuga și în switch se verifica tipul acestuia, de aici, împărțindu-se pe cazuri:

- dacă este pe cablu (computer / imprimanta – dacă laptopul este conectat pe cablu se tratează ca și computer) MAC-ul acestui dispozitiv este adăugat automat în tabela switch, astfel legătura dintre **switch** – **devices** ar fi **1** : **n**;
- dacă este pe wireless, traseul este unul mai lung, MAC-ul dispozitivului salvându-se în tabela **access_point;** pe lângă MAC, tabela **access_point** mai are o coloana device_type unde se salvează tipul dispozitivului și **AP_ESSID**; în aceasta coloana se salvează id-ul wireless-ului la care este conectat dispozitivul. Astfel, prin existența tabelului **access_point_bridge** în care se află cele 4 wireless-uri și MAC-urile asociate lor, utilizatorii au acces la internet, această tabela fiind conectată direct la switch. Legăturile dintre tabele ar fi:

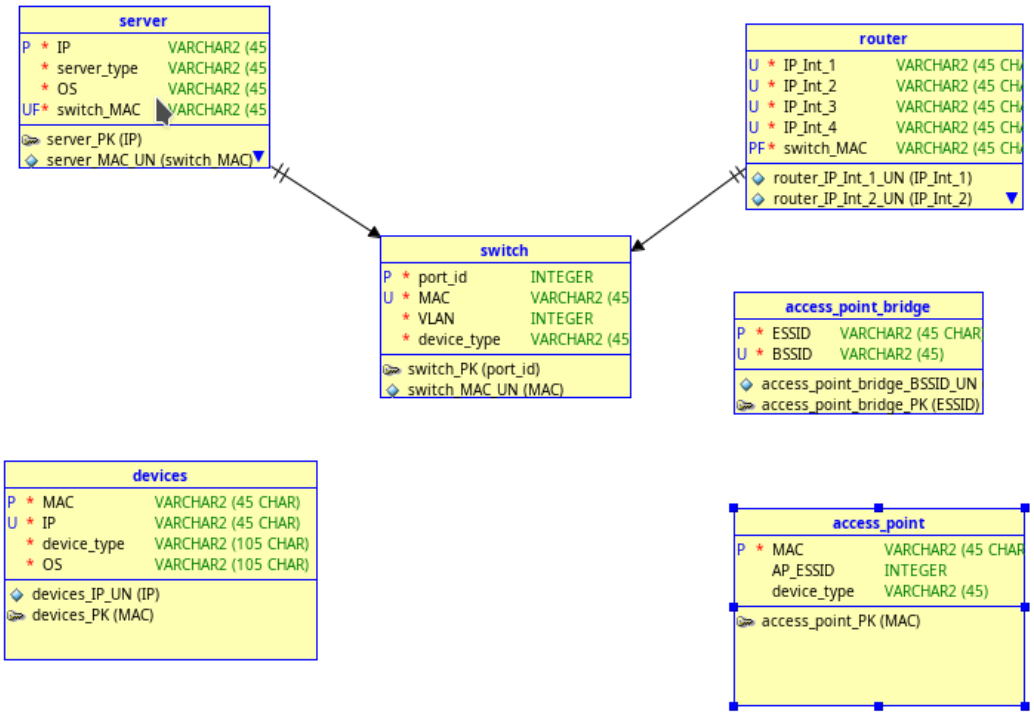
➢ **access_point** – **device** ar fi **1** : **n**

➢ **access_point_bridge – access_point** ar fi **1 : n**

➢ **switch – access_point_bridge** ar fi **1 : n**



Schema logică a tabelelor



Schema relațională a tabelelor

Descrierea logicii stocate

Toată logica prezentată mai sus a fost dezvoltată prin intermediul procedurilor, acestea fiind încapsulate în pachete. De asemenea, aşa cum am precizat adineauri, avem triggere atât pentru verificarea datelor înainte de inserare / actualizare cât şi validarea informaţiei în toate tabelele odată ce aceasta a fost inserată. În cazul în care trigger-ele nu sunt valide, o excepţie va fi apelată.

Tabela access_point

Triggeri

- ACCESS_POINT_VALIDATE_ENTRY **(Before Insert or Update)**
  - Verifică dacă datele din select / update sunt valide (1 <= essid <= 4), mac-ul şi ip-ul respectă structura corectă a unui mac / ip.

Tabela access_point_bridge:

Pachete şi proceduri

- ACCESS_POINT_BRIDGE_PACK
  - ADD_ACCESS_POINT_BRIDGE
  - UPD_ACCESS_POINT_BRIDGE
  - DEL_ACCESS_POINT_BRIDGE

Aşa cum se observă şi din numele procedurilor, acestea urmează paradigma CRUD.

Triggeri

- ACCESS_POINT_BRIDGE_VALIDATE_ENTRY **(Before Insert or Update)**
  - Verifică dacă datele din select / update sunt valide (1 <= essid <= 4), mac-ul şi ip-ul respectă structura corectă a unui mac / ip.
- ACCESS_POINT_BRIDGE_SELECT_PORT_ID **(After Insert)**
  - Introduce datele în switch cu un port_id valid.
- ACCESS_POINT_BRIDGE_UPDATE_BSSID **(After Update)**
  - Odată actualizat MAC-ul dintr-o intrare a tabelei access_point_bridge, acest trigger are rolul de a modifica MAC-ul şi din tabela switch.
- ACCESS_POINT_BRIDGE_DELETE_FROM_SWITCH **(After Delete)**
  - Odată şterse datele din access_point_bridge, acest trigger are rolul de a şterge datele din switch.

Tabela switch:

Pachete şi proceduri

- SWITCH_PACK
  - GET_SWITCH_DATA

Nu avem nevoie de metode precum Create, Update sau Delete deoarece toate se execută automat prin triggere.

Triggeri

- SWITCH_VALIDATE_ENTRY **(Before Insert or Update)**
  - Validează tipul datelor prin regex-uri şi atribuie VLAN-uri în funcţie de disponibilitatea porturilor şi a tipului dispozitivului.

Tabela router:

Pachete şi proceduri

- ROUTER_PACK
  - ADD_ROUTER
  - DEL_ROUTER

Triggeri

- ROUTER_VALIDATE_ENTRY **(Before Insert or Update)**
  - Verifică dacă datele din select / update sunt valide: mac-ul şi ip-ul respectă structura corectă a unui mac / ip;
  - Introduce datele despre router în switch.

- ROUTER_DELETE_FROM_SWITCH **(After Delete)**
  - Odată ştearsă o intrare din router, acest trigger are rolul de a şterge intrarea şi din switch.

Tabela server:

Pachete şi proceduri

- SERVER_PACK
  - ADD_SERVER

Triggeri

- SERVER_VALIDATE_ENTRY **(Before Insert or Update)**
  - Verifică dacă datele din select / update sunt valide: tipul server-ului este valid, mac-ul şi ip-ul respectă structura corectă a unui mac / ip.

Tabela devices:

Pachete şi proceduri

- DEVICE_PACK
  - ADD_ROUTER

Triggeri

- DEVICE_VALIDATE_ENTRY **(Before Insert or Update)**
  - Verifică dacă datele din select / update sunt valide: tipul dispozivitului este valid, mac-ul şi ip-ul respectă structura corectă a unui mac / ip.

- MANAGE_DEVICES **(After Insert or Update)**
  - Are scopul de a intrdouce dispozivitul în tabela corespunzătoare (access_point / switch) şi de selecta PORT_ID-ul corespunzător.

Alt aspect de precizat ar fi faptul că, executarea a mai multor proceduri simultan sau a unor instrucţiuni clasice precum Select / Insert a fost realizată prin **tranzacţii.**