Universitatea București
Facultatea de Matematică și Informatică
Departamentul de Informatică
Specializarea Tehnologia Informației

PROIECT
UTILIZAREA SISTEMELOR DE OPERARE

Coordonator Științific:
Drăgan Mihăiță

Studenți:
Bălmău Dragoș-Constantin
Cotescu Victor Andrei
Nicolescu Robert Alexandru
Vărzan Laurențiu-Adrian

București
2020

Universitatea București
Facultatea de Matematică și Informatică
Departamentul de Informatică
Specializarea Tehnologia Informației

LIMBAJ DE INTEROGARE STRUCTURAT
STRUCTURED QUERY LANGUAGE
-SQL-

Coordonator Științific:
Drăgan Mihăiță

Studenți:
Bălmău Dragoș-Constantin
Cotescu Victor Andrei
Nicolescu Robert Alexandru
Vărzan Laurențiu-Adrian

București
2020

Cuprins

1. Introducere

1.1. Scopul proiectului

Proiectul are ca scop simularea unui limbaj SQL. În cadrul proiectului au fost implementate cateva comenzi de baza ale limbajului SQL: CREATE, OPEN, CLOSE, INSERT, SELECT, UPDATE, DELETE, HELP și QUIT. De asemenea, programul a fost transpus pe un server cloud prin intermediul Google Cloud, pentru accesul mai facil și pentru a nu utiliza resursele propriilor calculatoare.

1.2. Motivație

Am ales acest proiect deoarece, pe măsură ce tehnologia evoluează, cresc și nevoile de stocare. Aceasta nu este o problema nouă, ci este o problemă cu care lumea digitală se luptă încă de la început. Stocarea datelor în cloud este viitorul, de aceea trebuie să ajutăm cât mai mult la evoluția acestei tehnologii.

2. Limbajul SQL

Limbajul de interogare structurat - Structured Query Language - SQL este un limbaj de programare specific pentru manipularea datelor în sistemele de manipulare a bazelor de date relaționale (RDBMS), iar la origine este un limbaj bazat pe algebra relațională. Acesta are ca scop inserarea datelor, interogații, actualizare și ștergere, modificarea și crearea schemelor, precum și controlul accesului la date. A devenit un standard în domeniu, fiind cel mai popular limbaj utilizat pentru crearea, modificarea, regăsirea și manipularea datelor de către Sistemelor de Gestiune a Bazelor de Date (SGBD) relaționale. Pe lângă versiunile standardizate ale limbajului, există o mulțime de dialecte și variante, unele proprietare, fiind specifice anumitor SGBD-uri și de asemenea conținând extensii pentru a suporta Sistemele de Baze de Date (SBD) obiectuale (obiectual-relaționale). SQL permite atât accesul la conținutul bazelor de date, cât și la structura acestora.

Este un limbaj neprocedural şi declarativ, deoarece utilizatorul descrie ce date vrea să obţină, fără a fi nevoie să stabilească modalităţile de a ajunge la datele respective. Nu poate fi considerat un limbaj de programare sau unul de sistem, ci mai degrabă face parte din categoria limbajelor de aplicaţii,

fiind orientat pe mulţimi. Foarte frecvent, este utilizat în administrarea bazelor de date client/server, aplicaţia client fiind cea care generează instrucţiunile SQL.

Elementele limbajului SQL

- **Clauze** (cuvinte cheie ex: SELECT, UPDATE etc), care sunt componente ale instrucțiunilor și interogărilor.
- **Expresii**, al căror efect este producerea de valori scalare sau tabele.
- **Predicates**, pot specifica condiții care sunt evaluate de SQL conform logicii ternare sau logicii booleene, în scopul limitării efectelor instrucțiunilor, sau pentru a influența cursul programului.
- **Interogările**, au ca scop regăsirea datelor după criterii specifice.
- **Instrucțiunile**, pot avea un efect persistent asupra datelor sau structurii datelor, sau pot controla tranzacțiile, conexiunile sau cursul programului. În general, instrucțiunile SQL se termină cu caracterul punct-virgulă (";"), deși acest lucru nu este obligatoriu în toate platformele SQL. Spațiile albe suplimentare sunt ignorate, dar ele pot fi folosite pentru lizibilitatea codului SQL.

3. Medii de programare

3.1. BASH

Bourne-again shell-ul (*BASH)* este un interpretor de comenzi dezvoltat inițial de Brian Fox ca înlocuitor pentru Bourne Shell. Este responsabil pentru executarea fișierelor de tip sh, csh și ksh. Un shell linux este un macro procesor capabil de a interpreta și executa comenzi. Bash-ul este un shell specific sistemului de operare Linux. Pentru ca shell-ul să poată executa comenzile bash dintr-un fișier, acesta trebuie sa primeasca drepturi de citire/scriere (comanda **chmod +rwx)**. La începutul fișierelor de tip script este definit shell-ul care este invocat. În cazul bash-ului, în antetul fișierului vom introduce **#!/bin/bash.**

4. Cerințe de sistem

Cerințele minime necesare pentru ca server-ul sa poată funcționa sunt:

- Procesor Intel sau AMD pe 64 de biţi, la 2 GHz, cu două nuclee
- 4 GB RAM
- Spaţiu pe disc 25 GB

Cerința minimă pentru utilizator este de a avea o conexiune stabilă la internet pentru a putea stabili o conexiune cu server-ul.

5. Conectivitate

5.1. Google Cloud

Serverul de pe GoogleCloud este reprezentat de o masina virtuala ce ruleaza Ubuntu 18.04 cu o memorie totala de 10GB. Conexiunea la server se face prin SSH (Secure Shell), nivelul de securitate fiind asigurat de chei asimetrice de tip RSA (Rivest–Shamir–Adleman). Pentru a putea accesa serverul este nevoie de realizarea unei perechi de chei publica-privata. Acest lucru se poate realiza in terminal prin comanda:

ssh-keygen -t rsa

Cand vom fi interogați, vom putea adăuga o parola, parola ce ne va facilita accesul pe server mai târziu. La finalul acestui proces vom avea doua noi fișiere, asemanatoare cu “nume_fisier”.pub și “nume_fisier”, in funcție de cum am setat numele fisierelor. Fisierul “nume_fisier”.pub trebuie sa ajunga pe server in fișierul /.ssh/authorized_keys. Daca acesta nu exista putem forma unul. Putem crea acest fișier folosind comanda “nano”, dar avem nevoie de acces root. Dupa ce avem setat fișierul cu chei publice, ne putem conecta la server in felul urmator din orice terminal pe orice distributie de Linux sau de pe Windows prin terminalul GitBash. Aceasta conexiune se realizeaza prin comanda:

ssh -i /path/to/private/key user@ip

Parametrul **-i** inseamna identify file si primeste ca argument path-ul catre cheia privata generata anterior. Prima data cand ne vom conecta vom primi mesaj de la terminal că nu este cunoscut serverul. Cand vom fi intrebati inseram “Yes” in consola. Dupa acest proces terminalul va cere sa inseram o parola. Aceasta va fi parola pe care am setat-o cand am realizat perechea de chei asimetrice. Avem acum acces pe server.

Pentru a urca scriptul pe server am folosit comanda SCP (Copiere securizată, Secure Copy). Sintaxa acestei comenzi este asemanatoare cu cea a SSH-ului:

scp -i /path/to/private/key file_name user@ip:/home/user

Vom avea iar nevoie de parola introdusă în momentul în care am creat perechea de chei asimetrice. Acum avem scriptul urcat pe server in directory-ul userului ales. Pentru a putea folosi corect scriptul, trebuie sa ii setam drepturi de citire, scriere și executare. Îi putem acorda drepturi prin comanda

chmod +rwx SQL.sh

Parametul ***+rwx*** identifica faptul ca am adaugat aceste drepturi scriptului SQL. Pentru a scoate drepturile acestui fisier putem reapela functia chmod, dar cu parametrul ***-rwx.*** Pentru a adăuga acest script in PATH, avem nevoie de acces root și putem folosi comanda EXPORT. Sintaxa este urmatoarea:

export PATH=/path/to/script:$PATH

Acum putem rula direct scriptul de pe server fara a mai realiza conexiunea “directa”. Folosind iar sintaxa SSH-ului facem urmatorii pasi:

ssh -i /path/to/private/key user@ip SQL.sh

Avem acum acces la scriptul SQL si la toate functiile acestuia prin intermediul server-ului de pe GoogleCloud.

6. Structura

6.1. CREATE DB/TABLE

Sintaxa:

CREATE [file_type]... [file_name] [column_name] [data_type]...

Descriere:

Funcția CREATE (CREARE) are rolul de a genera directoarele și fișierele solicitate de utilizator prin sintaxa(doar dacă acestea nu exista deja, altfel se afișează un mesaj de eroare), astfel directoarele ținând locul bazelor de date, iar fișierele înlocuind tabel in care se stocheaza informatiile. O baza de date poate avea mai multe tabele, iar o tabela mai multe coloane. Toate aceste fișiere sunt create în directorul principal al aplicației (directorul “SQL”), care implicit sa găsește în spațiul alocat fiecărui utilizator al mașinii virtuale.

Sunt doua modalități de a utiliza funcția descrisă anterior cu ajutorul cuvintelor cheie (DB abreviere de la “*database*” - baza de date și TABEL - tabel). În cazul generării tabelei, după precizarea

numelui acesteia, se tastează numele coloanei urmat de tipul de date ce va fi stocat în aceasta (pot exista mai multe coloane, iar în acest caz după fiecare tip de date precizat pentru o coloana va urma denumirea următoarei). Stocarea în fiser sa va face sub forma unui tabel în care pe prima linie sunt afișate tipurile de date în ordinea precizata, pe randul al doilea se afla numele coloanelor create din linia de comanda, iar incepand cu randul al treilea se vor afla datele/informațiile introduse cu ajutorul funcției INSERT(6.3), despărțite de cate un spatiu.

Exemplu de stocare a datelor în tabela/fișier:

datatype1	datatype2	datatype3
columnname1	columnname2	columnname3
data11	data12	data13
data21	data22	data23
data31	data32	data33

6.2. OPEN & CLOSE

Sintaxa:

OPEN [database_name]

CLOSE DB

Descriere:

Cele doua functii OPEN (DESCHIDE) și CLOSE (ÎNCHIDE) sunt complementare, ele fiind folosite la accesarea sau părăsirea unei baze date. Funcția OPEN necesita numele unei baze de date, deoarece utilizatorul are posibilitatea sa creeze mai multe baze de date care sunt identificate după nume, iar funcția CLOSE nu necesita ca parametru numele bazei de date în care se afla, deoarece se poate ieși doar din baza de date în care se afla utilizatorul. Funcțiile folosesc comanda **cd 'databasename'** pentru a accesa directoarele în cadrul funcției **OPEN** și **cd ..** pentru a reveni în directorul părinte "SQL".

La apelarea funcției OPEN tabelel aflate în bazele de date sunt decripatate

6.3. INSERT

Sintaxa:

INSERT INTO [table_name] [cloumn1] [column2] ... VALUES [value1] [value2] ..

Descriere:

Funcția inserează date într-un tabel deja existent, atunci când sintaxa este corectă, fișierul apelat există, iar coloanele în care se încearcă introducerea sunt apelate exact după numele din tabel. Altfel, pentru fiecare lucru în neconcordanță cu sintaxa "ideală" va semnala fiecare eroare în parte, de regulă cu sugestii pentru corectare. De asemenea, apare un mesaj corespunzător și atunci când introducerea se realizează cu succes.

Funcția permite introducerea datelor atât în toate coloanele tabelului, cât și în doar câteva dintre ele. Singura condiție pentru ca inserarea să se realizeze cu succes este ca numele coloanelor sa fie apelate în aceeași ordine în care apar în tabel. În coloanele nemenționate se va introduce un "-" pentru a sugera faptul că acolo este o informație lipsă.

În primul rând, după INSERT INTO funcția primește numele fișierului în care urmează să se realizeze inserarea. Pe urmă sunt introduse numele coloanelor în care dorim să introducem date.

În al doilea rând, după cuvântul VALUES for fi introduse datele pe care dorim sa le introducem în coloanele enumerate anterior, respectând ordinea lor și tipul de date corespunzător fiecăreia.

6.4. SELECT

Funcția SELECT (SELECTARE) selectează și afișează valori dintr-un tabel al unei baze de date.

Sintaxa:

SELECT [column_name]… FROM [table_name]

SELECT * FROM [table_name]

SELECT [column_name]… FROM [table_name] WHERE [condition]

Descriere:

În primul rând, funcția va primi după cuvântul cheie **SELECT** ca prim(e) argument(e) numele coloanei(lor) din care se vor selecta datele dorite. Dacă numele coloanelor nu este corect sau nu

există funcția va returna o eroare .În cazul în care utilizatorul va dori să selecteze toate coloanele unui tabel va trebui să aplice ca prim argument al funcției caracterul ”*”. După ce selecția coloanelor a fost făcută se va scrie cuvântul cheie **FROM** urmat apoi de un argument dat de către utilizator ce va semnifica numele tabelului din care se va face selecția coloanelor. Acum tot ce mai are utilizatorul de făcut, dacă acesta dorește ca elementele selectate să îndeplinească o anumită condiție, este să scrie cuvântul cheie **WHERE** urmat de condiția dorită. Ultimul pas este de a apăsa enter pentru a se executa comanda. Pentru a executa comanda dată de utilizator și pentru a selecta și afișa datele dorite pe ecran funcția copiază conținutul necesar într-un fișier de tipul Valori Separate De Virgulă - Comma Separated Values - CSV și cu ajutorul comenzii “csvcut” funcția selectează coloanele dorite după numele acestora și le va afișa. Am folosit fișiere de tip CSV datorită formatului tabular în care aceste fișiere păstrează informația.

Funcția va evalua comanda și afișa pe ecran coloanele selectate precum în exemplul următor:

columnname1	columnname 2
data11	data12
data21	data22

6.5. UPDATE

Functia UPDATE (ACTUALIZARE) actualizeaza informatii deja existente intr-un table existent in baza de date.

Sintaxa:

UPDATE [table_name] SET [nume_coloana] = [valoare] .. WHERE [nume_coloana] = [condiție] ..

Descriere:

Funcția primește ca parametrii numele table-ului ce urmează să fie actualizat, numele coloanelor ce vor fi actualizate și valorile acestora după cuvântul cheie **SET**, iar după cuvântul cheie **WHERE** coloanele și condițiile după care se va face update-ul. Se vor înlocui, în toate rândurile existente din table-ul selectat, în toate coloanele ce îndeplinesc condițiile selectate cu valorile introduse. Funcția este case sensitive pentru cuvintele cheie. Dacă acestea vor fi scrise altfel decât cu litere mari, funcția va returna o eroare de sintaxă. De asemenea dacă oricare dintre numele coloanelor, fie primite ca parametrii după **SET**, ori după **WHERE** nu există iî table sau sunt scrise greșit, funcția returnează o eroare ce descrie

tipul de greșeală aferent. Este realizată verificarea tipurilor de date din table-ul în care actualizăm informații, iar dacă acestea nu corespund este returnată o eroare care descrie ce coloană nu corespunde tipului de date din table-ul sursă.

6.6. DROP

Sintaxa:

DROP TABLE [tablename]

DROP DB [databasename]

Descriere:

Funcția DROP (RENUNȚARE) are doua modalități de apelare, prima data pentru a șterge o tabela dintr-o baza de date prin folosirea primei variante de sintaxa (apelarea funcției urmată de cuvântul cheie TABLE și de numele tabelei ce va fi ștearsă - **rm table_name**) și cea de-a doua varianta folosită pentru a șterge o baza de date cu tot cu tabelele din ea(apelarea funcției urmată de cuvântul cheie DB și de numele bazei de date ce va fi ștearsă recursiv - **rm -r database_name**)

6.7. HELP

Functia HELP (AJUTOR) returnează descrierea comenzilor existente.

Sintaxa:

HELP [comanda]

Descriere:

Funcția primește ca argument numele unei comenzi și returnează informații pentru utilizarea acesteia. Aceasta poate fi apelată și fără argument și va returna informații despre comenzile existente și modul de utilizare al funcției help.

6.8. QUIT

Funcția QUIT (IEȘIRE) facilitează ieșirea din program.

Sintaxa:

QUIT

Descriere:

Apelarea funcției QUIT va rezulta în terminarea programului. Un dialog este afișat pentru confirmarea acțiunii (Y/N).

6.9. DBS

Funcția DBS afișează bazele de date existente.

Sintaxa:

DBS

Descriere:

Funcția afișează toate bazele de date existente atât timp cât nu există una selectată. Dacă o bază de date existentă este deja selectată, funcția va returna o eroare.

6.10. TABLES

Funcția TABLES (TABELE) afișează tabelele existente

Sintaxa:

TABLES

Descriere:

Funcția afișează toate tabelele existente în baza de date existentă. Dacă nu există o bază de date selectată, funcția va returna o eroare.

7. Criptarea datelor

În acest deceniu, cele mai importante resurse sunt timpul și informațiile. Aplicația descrisă în această documentație are scopul de manipulare a datelor, ce trebuie într-un final protejate de atacurile cibernetice tot mai populare în zilele noastre. Criptarea datelor în acest proiect se realizeaza prin libraria OPENSSL care folosește o cheie de criptare asupra datelor din tabele transformandu-le în caractere indescifrabile. Librăria OPENSSL, implementată în limbajul de programare C, deține funcții de criptografie simple și numeroase unelte de criptare a datelor, care facilitează experiența utilizatorului.

Funcția OPEN (6.2) prevede decriptarea tuturor tabelelor din interiorul bazei de date selectate, iar funcția CLOSE criptează datele astfel incat sa nu poata sa fie folosite de diverși utilizatori. Pentru decriptarea datelor am folosit comandă:

openssl enc -aes-256-cbc 2>/dev/null -d -in "$f" > "$name" -pass pass:"paroleparole"

Instrucțiunea anterioara decripteaza un fișier de forma **”numefisier.enc”** în unul decriptat de forma **”numefisier”** cu ajutorul cheii **paroleparole**. Pentru a evita apariția avertizarilor, am introdus structura **2>/dev/null** în lista parametrilor, alături de **-d**, care precizează funcției ca urmeazî o procedurî de decriptare și desigur -pass care realizează introducerea parolei din linia de comandă .

openssl enc -aes-256-cbc 2>/dev/null -in "$f" -out "$f.enc" -pass pass:"paroleparole"

Fișierul necriptat (fișierul cu care lucreaza algoritmul în momentele în care o baza de date este deschisă) se transforma în unul de forma **“numefisier.enc”** prin decriptarea de către librăria OPENSSL cu cheia precizata anterior, in cadrul functiei CLOSE (6.2).

Bibliografie:

- https://searchsecurity.techtarget.com/definition/RSA
- https://linuxize.com/post/how-to-use-scp-command-to-securely-transfer-files/
- https://profs.info.uaic.ro/~busaco/publications/articles/bash.pdf
- https://www.w3schools.com/sql/sql_intro.asp
- http://vega.unitbv.ro/~cataron/Courses/BD/BD_Cap_5.pdf (Limbajul SQL)
- ANSI/ISO/IEC International Standard (IS). Database Language SQL—Part 2: Foundation (SQL/Foundation). 1999 (Limbajul SQL)