Java, bazele de date si securitatea pe Internet

Asist. Carmen STANCIU
Catedra de Informatica Economica, A.S.E. Bucuresti

Orice organizatie are una sau mai multe baze de date, deoarece acestea au o aplicabilitate foarte mare: financiar, administrativ, resurse umane etc. Prezenta lor în toate domeniile vietii economio-sociale a dus si la aparitia:
· unor motoare de baze de date, adica un soft optimizat care câteva comenzi asupra unui anumit tip de baze de date , precum si a
· unui standard privind limbajul de interogare al bazelor de date: SQL (Structured Query Language).
Unul dintre standardele motoarelor de baze de date este si specificatia OBDC.
Cuvinte cheie: *Java, baze de date, standard, securitate, Internet.*

Java si bazele de date

ODBC (Open DataBase Connectivity) este conceputa de Microsoft, fiind actualmente acceptata ca un standard de facto în acest domeniu.
Adminstratorul de baze de date pune la punct modelul conceptual al unei baze de date, ajuta la implemntarea acesteia, urmând apoi sa gestioneze integritatea si accesul la datele acumulate. Trebuie sa tina cont de specificul interfetei Web si de problemele de securitate, având în acest domeniu o colaborare cu administratorul Intranet-ului.
Programatorii trebuie sa se ocupe de partea de server SSS (Server Side Script), partea de client fiind de obicei foarte simpla.
Pentru accesul la baze de date se poate folosi o gama variata de produse, de la programarea clasica (C++, C, Pascal), CGI, produse middleware (dbWeb, ColdFusion - returneaza documente în format HTML), scripturi.

JDBC (Java Data Base Connectivity) reprezinta o serie de API specifice Java pentru conectarea la baze de date, realizând astfel apropierea dintre Java si sfera mutimedia din interiorul paginilor HTML.
JDBC este un middleware destul de putin utilizat datorita faptului ca se bazeaza pe programarea clasica.
În cadrul JDBC, un applet Java poate consulta cu usurinta înregistrarile unei baze de date. Are ca dezavantaj, în afara ca se bazeaza pe programare, si faptul ca nu sunt integrate în actualele browsere clasele JDBC. Utilizatorul trebuie deci sa încarce acele clase JDBC înainte de a utiliza apllet-ul. Din aceasta cauza, aceasta solutie se adreseaza în principal Intranet-urilor, unde debitele sunt satisfacatoare.
JDBC-ul are o constructie asemanatoare cu a ODBC-ului, fiind practic varianta ODBC pe obiecte, pentru Java, ambele pornind de la specificatiile X/Open SQL Call Level Interface.
Accesul la o anumita baza de date din Java via JDBC se face pe baza unui driver specific tipului bazei de date pe care dorim s-o utilizam, fie via ODBC, folosind un bridge JDBC-ODBC. În acest caz, apelurile JDBC sunt transformate în apeluri ODBC.
Pe scurt, modul de lucru al JDBC-ului este urmatorul:
- stabileste o conexiune cu baza de date;
- trimite secventele SQL;
- prelucreaza rezultatele.

JDBC-ul este o interfata “low-level”, fiind folosita pentru executia directa a comenzilor SQL si care are implementate doua API-uri mai importante:
- un preprocesor SQL încapsulat pentru Java, care permite folosirea secventelor mixte SQL direct din Java ;
- o mapare directa a tabelelor bazei de date relationale în Java, prin care fiecare

articol al bazei de date devine o instanta a unei clase, si fiecare câmp corespunde unui atribut al instantei respective.
JDBC-ul se afla implementat în doua modele:
- ***modelul cu doua niveluri (two-tier)*** – în care un applet sau o aplicatie Java lucreaza direct cu baza de date;
- ***modelul cu trei nivele (three-tier)*** – în care interogarile se trimit catre un nivel intermediar, numit server de aplicatie, care retrimite cererea SQL catre serverul bazei de date; în acest caz utilizatorul poate utiliza un API de nivel mai înalt, care va fi mai simplu de folosit si care transpune cererile într-un limbaj de baza, înteles de serverul bazei de date.

Comunicarea cu baza de date se poate realiza:
- direct, prin TCP/IP;
- prin intermediul unui protocol nativ de retea, specific bazei de date.

Driverele JDBC pot fi, în functie **de functionalitati**, de doua feluri:
- pe partea de client ;
- mixte, atât pe partea de client cât si pe cea de server.

Firma Sun a stabilit o clasificare a **tipurilor de drivere disponibile**, si anume cea prezentata în figura 1:
- driver-ele de tipul 1 reprezinta singura punte JDBC-ODBC;
- driver-ele de tipul 2 "învelesc" codul nativ, adica driverele existente, în Java;
- driver-ele de tipul 3 sunt scrise complet în Java si utilizeaza o componenta pe server si un protocol de retea pentru comunicare ;
- driver-ele de tipul 4, scrise complet în Java, folosesc un protocol specific bazei de date.

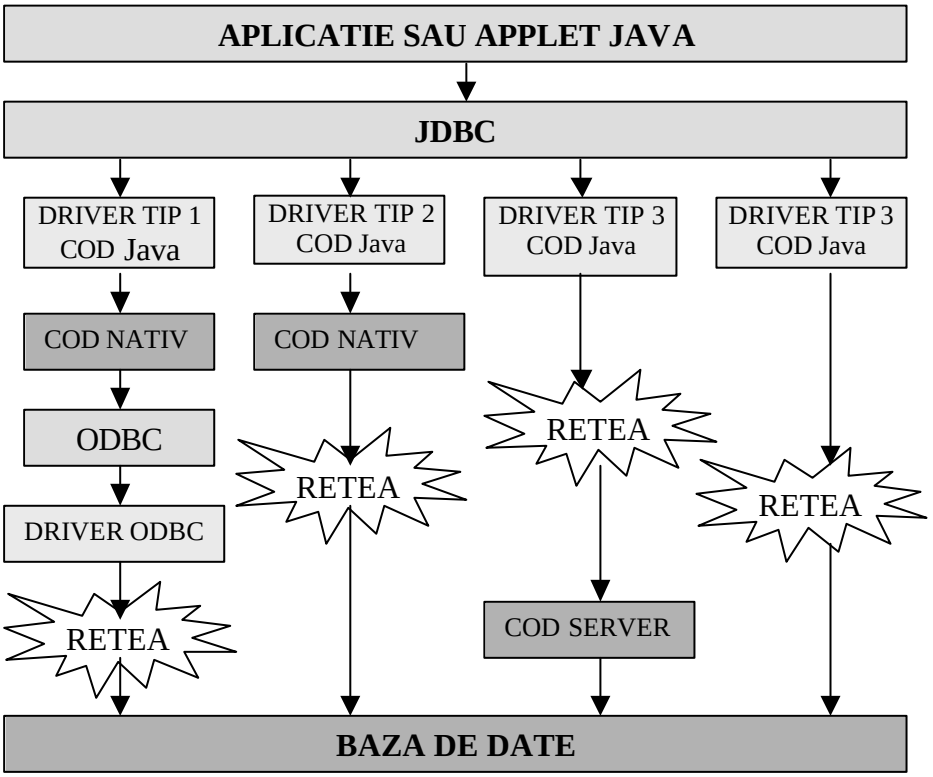

Figura 1

Java Soft furnizeaza JDBC-ul cu trei componente, toate incluse în JDK, si anume:
- *JDBC driver manager* - reprezinta coloana vertebrala a arhitecturii JDBC; este mic, simplu si are functia de a conecta aplicatia Java cu driver-ul JDBC potrivit;

- *JDBC driver test* – asigura ca driver-ul JDBC va functiona cu programul utilizatorului;
- *JDBC- ODBC bridge* – permite driverelor ODBC sa fie folosite ca drivere JDBC.

JDBC este *implementat pe doua nivele*:
- *nivel driver* – care contine implementarile interfetelor specifice lucrului unui anumit tip de baza de date relationala;
- *nivel aplicatie* – constituit din Driver Manager si clasele furnizate de driverul JDBC necesare conectarii la baza de date.

Executia unei comenzi SQL din interiorul unui program Java folosind API-ul JDBC, presupune parcurgerea urmatoarelor etape:
• *încarcarea driver-ului JDBC,*
· *realizarea conexiunii la baza de date prin intermediul driver-ului:*
Connection c=DriverManagement.getConnection()
· *constructia comenzii***(Statement s=c.createStatemant)**,
· *executia comenzii si returnarea rezultatului***(ResultSet r=s.executeQuery)**.
Pentru realizarea acestor functii se utilizeaza urmatoarele **interfete**:
- **java.sql.DriverManager** – încarca un driver JDBC si gestioneaza conexiunea la baza de date;
- **java.sql.Connection** – reprezinta conexiunea la o baza de date;
- **java.sql.Statement**- gestioneaza o conexiune SQL;
- **java.sql.ResultSet** – asigura accesul la rezultatele obtinute în urma executiei unei instrutiuni sql.

Avantajele utilizarii JDBC fata de ODBC, în conditiile în care ODBC este cea mai folosita interfata pentru accesul la aproape orice tip de baze de date relationale, ar fi urmatoarele:
- ODBC nu este potrivit pentru a fi utilizat direct din Java pentru ca foloseste interfata C, iar apelurile Java catre C pot provoca probleme, legate de portabilitate si securitate;
- ODBC-ul este mai greu de învatat, având optiuni complexe chiar si pentru interogari simple;
- JDBC-ul este o solutie pura Java, putând fi deci portabil si securizat pe toate platformele, de la NC (Network Computer) la mainframe-uri.
-

5. Java si securitatea pe Internet

Din punctul de vedere al securitatii, limbajul Java asigura un cadru de securitate care nu poate fi subminat de coduri incorecte sau de erori ale compilatorului. F

Functionând dupa reguli bine stabilite de o politica de securitate la nivel aplicatie, în Java exista totusi anumite clase de probleme de securitate, dupa cum urmeaza:
- *atacuri de refuz al serviciului (denial of service attacs)* – implica asteptare ocupata, consumând ciclii CPU, alocând memorie pâna se blocheaza sistemul, blocând alte fire de executie si procese sistem, blocând browserul pâna la scoaterea sa din uz; nu exista în Java un mecanism care sa împiedice aceste atacuri;
- *atacuri tripartite (three party attacs)* – de exemplu, un atacator C produce un applet "cal troian", un utilizator B pune acest applet, gasit pe Internet în pagina sa de Web, iar un utilizator A vizualizeaza pagina HTML a lui B, applet-ul lui C stabilind un canal ascuns pentru scurgerea informatiilor între de la A la C, fara stirea lui B;
- *canale ascunse (convert channels)* ;
- *informatie disponibila applet-urilor* ;
- *erori de implementare* –reprezinta bug-uri care apar în cadrul subsistemului Java sau a browser-ului; un exemplu este propriul server proxy care poate fi folosit de intrus pentru a vedea traficul bidirectional realizat de browserul HotJava, daca clientul foloseste protocoalele FTP si HTTP necriptate .

Masuri semnificative de protectie se pot adauga printr-un ***manager de securitate,*** introdus prin biblioteca API, care furnizeaza **clasa java.lang.SecurityManager**, ca posibilitate de a defini un set clar de task-uri care pot sau nu sa fie realizate.

Mediul Java contine o clasa globala, numita SecurityManager, ale carei metode sunt utilizate pentru verificari în timpul executiei. Definind o subclasa a acestei clase se pot implementa diferite politici de protectie.
Clasa **SecurityManager** permite stabilirea unei politici de securitate specifice, prin crearea unui obiect care determina daca o operatie pe care programul intentioneaza sa o execute este permisa sau nu. Clasa contine metode pentru atingerea urmatoarelor punte din cadrul politicii de securitate:
- determina daca este permisa conectarea prin retea, de la un host si pe un port specificat;
- determina daca un program poate fi încarcat/ crea o noua clasa într-un pachet Java specificat;
- împiedica crearea unei noi clase ClassLoader;
- îmdiedica un program sa paraseasca JVM fara monitorizare;
- împiedica crearea unui nou SecurityManager, care se poate suprapune peste politica deja existenta;
- verifica daca un fisier poate fi sters/ citit/actualizat/ executat ;
- verifica daca un tread poate fi utilizat de un alt thread sau de un grup de thread-uri;
- verifica daca o biblioteca dinamica poate fi link-editata;
- verifica daca un port de retea sigur poate fi ascultat pentru o conectare la sistemul local;
- verifica daca un program poate sa creeze propria sa implementare pe socket-uri;
- identifica care proprietati pot fi accesate prin metoda System.getProperty();
Alt aspect important relativ la securitatea Java este diferenta dintre applet-uri si aplicatii Java:
- aplicatiile au dreptul de a accesa resursele sistemului, pot deschide si scrie în fisiere, se pot conecta la diferite resurse de retea si pot rula diferite programe de pe sistemul local, deoarece sunt executate direct de interpretorul Java si trebuie instalate si executate manual de catre utilizator;
- applet-uri pot fi rulate dinamic de navigator (prin încarcarea paginilor HTML care contin tag-ul APPLET) si pot proveni si din surse nesigure (de pe retea) având capacitatea de a produce pagube, motiv pentru care sunt rulate într-un mediu de executie foarte controlat.

Pentru a mai rezolva probleme de securitate existente în Java, odata cu lansarea versiunii JDK 1.1 au aparut facilitati pentru verificarea codului de pe alte surse, pentru a vedea daca a fost sau nu alterat de un intrus, introduse prin **Java Security API**.
Bazata pe algoritmi si concepte criptografice, biblioteca API Java Security este construita în jurul **pachetului java.security.pachage** si este proiectata atât pentru aplicatii de nivel scazut, cât si pentru cele de nivel ridicat. Include clase pentru semnaturi digitale, rezumat de mesaje, interfete abstracte pentru managementul cheilor, management de certificate si controlul accesului, respectând standardul X.509. Toate aceste concepte au fost studiate în referatul nr.1.
Arhitectura de clase Java Security API contine:
- **Java Crytography Architecture (JCA)** – reprezinta cadrul de acces si de dezvoltare a functionalitatilor criptografice pentru platforme Java si cuprinde partile interfetei Java Security API, prezentate ca un set de conventii si specificatii criptografice;
- **Java Crytography Extension (JCE)** – reprezinta extinderi ale JCA API, care include criptarea si schimbul de chei; împreuna cu JCA, ofera o platforma independenta de criptografie; JCE este supus interdictiilor de export ale guvernului SUA.
Clasele si interfetele Java Security API sunt urmatoarele:
- *clasa Provider* – este o interfata la un pachet si ofera metode de acces la nume de furnizori, care au implementati diferiti algoritmi de criptare (DES si RSA), de semnatura digitala (DSA, MD5, RSA), pentru

calculul rezumatului mesajului (SHS-1, MD5);

- *clasa Security* – controleaza instalarea furnizorilor si proprietatilor de securitate si contine metode statice (care nu pot fi instantiate);
- *clasa MessageDigest* – ofera algoritmi pentru calculul rezumatului de mesaj (SHS-1, MD5); presupune urmatorii pasi: crearea unui obiect MessageDigest, încarcarea obiectului si prelucrarea rezumatului;
- *clasa Signature* – ofera facilitati de realizare a semnaturilor digitale criptografice, folosind algoritmii DSA, RSA, MD5; pasii necesari sunt: crearea unui obiectg Signature, initializarea obiectului si semnarea unor date sau verificarea semnaturii altor date.

În acest moment, Java nu ofera standarde de criptare pentru transportul fisierelor prin Internet, acestea depinzând de securitatea implementata de mecanismul de transport utilizat.

În forma sa actuala, limbajul Java este departe de a fi securizat în mod eficient, fiind necesara reproiectarea unor elemente din limbaj, a formatului bytecode-urilor si a runtime-system-ului. Prezenta unor astfel de slabiciuni în Java nu implica faptul ca alte sisteme sunt mai sigure iar securitatea pe Web ramâne totusi o problema, prin faptul ca solutia oferita, Java, este înca incompleta.

Bibliografie

*** java.sun.com/products/;
*** Java Development Kit 1.2, documentatie;