Procesarea constrângerilor de securitate pentru controlul inferenţei

În cadrul acestui curs vor fi prezentate abordările privind constrângerile de securitate, din perspectiva problemei inferenţei.

Constrângerile de securitate sunt reguli care atribuie niveluri de securitate datelor.

Dintre aceste constrângeri fac parte cele care clasifică datele în funcţie de conţinut, context, agregare şi timp. Au fost studiate moduri de tratare a constrângerilor de securitate în timpul procesării cererilor astfel încât anumite nerespectări ale securităţii prin inferenţă să nu apară. Un alt studiu a pus accentul pe tratarea constrângerilor în timpul proiectării bazei de date; scopul tratării constrângerilor de securitate în această etapă este să nu poată apărea nerespectarea securităţii. Vom descrie tehnicile de proiectare pentru procesarea constrângerilor de securitate.

Tratarea adecvată a constrângerilor de securitate este esenţială pentru dezvoltarea unui MLS/DBMS util şi sigur.

În urma analizei diferitelor tipuri de constrângeri de securitate, acestea pot fi privite ca o formă de constrângeri de integritate aplicată în MLS/DBMS. Acest lucru are loc deoarece într-o bază de date multinivel, nivelul de securitate al unei entităţi poate fi privit ca o parte a valorii respectivei entităţi. Prin urmare, constrângerile de securitate specifică valori permise pe care le poate lua o entitate.

Deoarece o constrângere de securitate poate fi privită ca o formă de constrângere de integritate, multe dintre tehnicile dezvoltate de către cercetătorii din domeniul programării logice pentru tratarea constrângerilor de integritate în sisteme de baze de date relaţionale non-MLS pot fi utilizate pentru tratarea constrângerilor de securitate într-un MLS/DBMS. În cadrul acestor tehnici:

- unele constrângeri de integritate (denumite reguli de derivare) sunt tratate în timpul procesării cererii ;
- alte constrângeri de integritate (cunoscute ca reguli de integritate) sunt tratate în timpul actualizărilor bazei de date ;
- iar anumite constrângeri (cunoscute ca reguli ale schemei) sunt tratate în timpul proiectării bazei de date.

Un **modul pentru prelucrarea cererilor** trebuie să aibă capacitatea de a procesa şi constrângerile de securitate.

Acest lucru este datorat faptului că majoritatea utilizatorilor construiesc de obicei „rezervorul” de cunoştinţe din răspunsuri pe care le primesc prin interogarea bazei de date. Ulterior, din acest rezervor deduc informaţia secretă. Mai mult, indiferent cât de bine a fost proiectată baza de date în privinţa securităţii, sau dacă datele din baza de date sunt etichetate corespunzător cu etichete de securitate, utilizatorii vor putea, în cele din urmă, să nu respecte securitatea din cauza inferenţei deoarece îşi actualizează continuu rezervorul de cunoştinţe pe măsură ce lucrurile evoluează. Nu este posibil să reproiectăm baza de date sau să reclasificăm datele continuu.

Este de menţionat faptul că procesarea unui număr mare de constrângeri de securitate ar putea avea un impact asupra performanţei algoritmilor de procesare a cererilor. Prin urmare, este de dorit să fie procesate oricâte constrângeri posibile în timpul actualizărilor bazei de date şi al proiectării bazei de date. Acest lucru are loc deoarece, în general, proiectarea bazei de date şi operaţia de actualizare sunt efectuate mai puţin frecvent decât operaţia de interogare.

Prin urmare, atunci când baza de date este proiectată iniţial, ar trebui examinate constrângerile de securitate şi ar trebui generată schema. Atunci când are loc actualizarea datelor, constrângerile sunt examinate şi sunt atribuite sau reatribuite niveluri de securitate datelor afectate. Periodic, responsabilul cu securitatea sistemului (*System Security Officer* – SSO) ar trebui să examineze constrângerile de securitate, să reproiecteze baza de date şi să reclasifice datele. Dacă aplicaţia este statică şi dacă datele din baza de date sunt consistente cu constrângerile de securitate, modulul de procesare a cererilor nu trebuie să examineze constrângerile tratate de către modulul de procesare a actualizărilor şi proiectantul bazei de date. Dacă au apărut modificări în lumea reală care determină ca baza de date sau schema să devină inconsistente, atunci modulul de procesare a interogărilor ar trebui să fie declanşat astfel încât să proceseze constrângerile relevante în timpul operaţiilor sale. În acest mod, o mare parte din sarcinile atribuite procesorului de cereri sunt uşurate.

1. Constrângeri de securitate

Tipurile de constrângeri de securitate care au fost identificate includ următoarele:

1. *Constrângeri simple* – clasifică o bază de date, relaţie sau atribut.
2. *Constrângeri bazate pe conţinut* – clasifică orice parte a bazei de date în funcţie de valoarea anumitor date.

3. *Constrângeri bazate pe eveniment* – clasifică orice parte a bazei de date în funcţie de apariţia unui eveniment din lumea reală.
4. *Constrângeri bazate pe asociere* – clasifică relaţiile dintre date.
5. *Constrângeri bazate pe livrare* – clasifică orice parte a bazei de date în funcţie de informaţia care a fost furnizată anterior. Au fost identificate 2 tipuri de astfel de constrângeri:
    a. constrângeri generale, care clasifică un întreg atribut în funcţie de condiţia că o valoare a unui alt atribut a fost furnizată;
    b. constrângeri individuale, care clasifică valoarea unui atribut în funcţie de condiţia că o valoare a unui alt atribut a fost furnizată.
6. *Constrângeri agregat* – utile pentru clasificarea colecţiilor de date.
7. *Constrângeri logice* – specifică implicaţii.
8. *Constrângeri cu condiţii* – au condiţii asociate lor.
9. *Constrângeri bazate pe nivel* – clasifică orice parte a bazei de date în funcţie de nivelul de securitate al anumitor date.
10. *Constrângeri fuzzy* – atribuie valori *fuzzy* clasificării.

Vom da exemple de constrângeri de securitate de fiecare tip. Exemplele se referă la o bază de date în care există 2 relaţii:

SHIP(S#, SNAME, CAPTAIN, M#)

MISSION(M#, MNAME, LOCATION)

S#, M# sunt chei, iar domeniul lui M# din cele 2 relaţii este acelaşi.

Constrângerile pot fi exprimate sub formă de reguli logice. Vom utiliza clauzele Horn pentru a reprezenta constrângerile. În acest mod se poate beneficia de tehnicile care au fost dezvoltate pentru programele logice.

1. Constrângeri simple

R(A1, A2,…, An) -> Level(Ai1, Ai2,…, Ait) = Secret

Fiecare atribut Ai1, Ai2,…, Ait al relaţiei R este Secret.

Exemplu: SHIP(S#, SNAME, CAPTAIN, M#) -> Level (CAPTAIN) = Secret

2. Constrângeri bazate pe conţinut

R(A1, A2,…, An) AND COND(Value(B1, B2, …, Bm)) -> Level(Ai1, Ai2,…, Ait) = Secret

Fiecare atribut Ai1, Ai2,…, Ait al relaţiei R este Secret dacă este aplicată o condiţie asupra valorilor anumitor date specificate prin B1, B2, …, Bm.

Exemplu: SHIP(S#, SNAME, CAPTAIN, M#) AND (Value(SNAME) = Washington) -> Level (CAPTAIN) = Secret

3. **Constrângeri bazate pe asociere (constrângeri de context)**

R(A1, A2,…, An) -> Level(Together(Ai1, Ai2,…, Ait)) = Secret

Atributele Ai1, Ai2,…, Ait ale relaţiei R, luate împreună, sunt Secret.

Exemplu: SHIP(S#, SNAME, CAPTAIN, M#) -> Level (Together(SNAME, CAPTAIN)) = Secret

4. **Constrângeri bazate pe evenimente**

R(A1, A2,…, An) AND Event(E) -> Level(Ai1, Ai2,…, Ait) = Secret

Fiecare atribut Ai1, Ai2,…, Ait al relaţiei R este Secret dacă a apărut evenimentul E.

Exemplu: SHIP(S#, SNAME, CAPTAIN, M#) AND Event(Change of President) -> Level(CAPTAIN, M#) = Secret

5. **Condiţii generale bazate pe livrare**

R(A1, A2,…, An) AND Release(Ai, Unclassified) -> Level(Aj) = Secret

Atributul Aj al relaţiei R este Secret dacă atributul Ai a fost furnizat la nivelul Unclassified.

Exemplu: SHIP(S#, SNAME, CAPTAIN, M#) AND Release(SNAME, Unclassified) -> Level(CAPTAIN) = Secret

6. **Condiţii individuale bazate pe livrare**

R(A1, A2,…, An) AND Individual-Release(Ai, Unclassified) -> Level(Aj) = Secret

Constrângerile individuale bazate pe livrare clasifică elementele unui atribut la un nivel particular după ce elementele corespunzătoare ale altui atribut au fost livrate. Acestea sunt mai dificil de implementat decât cele generale.

7. Constrângeri agregat

Aceste constrângeri clasifică colecţiile de tupluri luate împreună la un nivel mai înalt decât nivelurile individuale ale tuplurilor din colecţie. Ar putea exista o asociere semantică între tupluri. Aceste tupluri pot fi specificate în forma următoare:

R(A1, A2,…, An) AND Set(S, R) AND Satisfy(S, P) -> Level(S) = Secret

Dacă R este o relaţie şi S este o mulţime ce conţine tuplurile lui R, iar S satisface o proprietate P, atunci S este clasificată la nivelul Secret. P poate fi orice proprietate (exemplu: numărul de elemente este mai mare decât 10).

8. Constrângeri logice

Sunt reguli care sunt folosite pentru a deriva date noi din datele din baza de date. Datele obţinute pot fi clasificate utilizând una dintre celelalte constrângeri. Constrângerile logice sunt de forma :

Ai => Aj

unde Ai, Aj sunt atribute fie ale unei relaţii din baza de date, fie ale uneia din lumea reală.
Constrângerile logice nu sunt chiar constrângeri de securitate, în sensul că nu asociază niveluri de securitate datelor. De fapt, ele sunt constrângeri de integritate. În particular, ele pot fi privite drept constrângeri de integritate care sunt tratate ca reguli de derivare.

9. Constrângeri cu condiţii

Celelalte tipuri de constrângeri pot fi specificate cu condiţii.
Considerăm exemplul: *Ai => Aj dacă este adevarată condiţia C*.

Exemplu: Atributul LOCATION al relaţiei MISSION implică MNAME dacă LOCATION = Atlantic Ocean

10. Constrângeri bazate pe nivel

R(A1, A2,…, An) AND Level(Ai) = Unclassified -> Level(Aj) = Secret
Atributul Aj al relaţiei R este Secret dacă atributul Ai este Unclassified.

Exemplu: SHIP(S#, SNAME, CAPTAIN, M#) AND Level(SNAME) = Unclassified -> Level(CAPTAIN) = Secret.

11. Constrângeri fuzzy

Aceste constrângeri utilizează valori *fuzzy* şi pot fi asociate oricărui alt tip de constrângeri. Un exemplu de constrângere *fuzzy* asociată unei constrângeri bazate pe conţinut este dată mai jos :

R(A1, A2,…, An) AND COND(Value(B1, B2, …, Bm)) -> Level(Ai1, Ai2,…, Ait) = Secret and Fuzzyvalue = r

Fiecare atribut Ai1, Ai2,…, Ait al relaţiei R este Secret cu o valoare *fuzzy* r dacă este aplicată o condiţie asupra valorilor B1, B2, …, Bm.

12. Constrângeri complexe

Exemplele anterioare se referă la o singură relaţie. Constrângerile pot fi aplicate, însă, pe mai multe relaţii. Astfel de constrângeri se numesc complexe. Un exemplu de constrângere complexă este următorul:

R1(A1, A2, … An) & R2(B1, B2,…, Bm) & R1.Ai = R2.Bj -> Level(Together(Ak, Bp)) = Secret

Constrângerea anterioară afirmă că o pereche de valori care include atributul k al relaţiei R1 şi p al relaţiei R2 este Secret cu condiţia ca valorile corespunzătoare (din aceeaşi linie) ale atributului i al relaţiei R1 şi j al relaţiei R2 sunt egale.

Exemplu: SHIP(S#, SNAME, CAPTAIN, M#) & MISSION(M#, MNAME, LOCATION) & SHIP.M# = MISSION.M# -> Level(Together (SNAME, MNAME) = Secret

2. O abordare a procesării constrângerilor de securitate

Constrângerile de securitate conduc la aplicarea unei politici de securitate. Prin urmare, este esenţial ca aceste constrângeri să fie prelucrate doar de către o persoană autorizată (SSO). Preupunem că înseşi constrângerile pot fi clasificate pe mai multe niveluri. Ele sunt stocate la nivelul sistemului. Managerul de constrângeri, care este sigur, va asigura că un utilizator poate citi doar constrângerile clasificate la un nivel inferior sau egal utilizatorului. Abordarea prezentată referitoare la procesarea constrângerilor de securitate este de a trata anumite constrângeri în timpul procesării cererilor, altele în timpul actualizărilor bazei de date, iar anumite constrângeri în timpul proiectării bazei de date.

Vom ilustra pe scurt arhitecturile pentru procesarea constrângerilor în timpul operaţiilor de interogare, actualizare şi proiectare a bazei de date.

Arhitectura pentru procesarea cererilor este prezentată în figura 1. Această arhitectură poate fi privită ca o legătură (slabă) între un MLS/DBMS şi un manager deductiv, care este ceea ce am numit modul de procesare a cererilor. Acesta trebuie să opereze online.

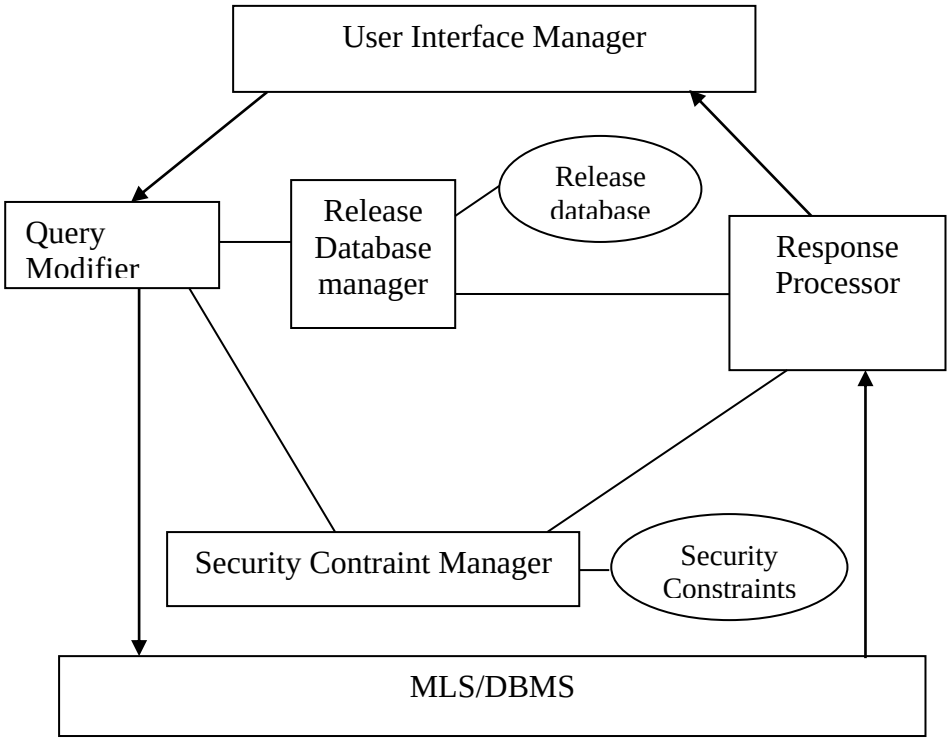

Figura 1. Modulul de procesare a cererilor

Arhitectura pentru procesarea actualizărilor este prezentată în figura 2. Aceasta poate fi, de asemenea, privită ca o legătură între un MLS/DBMS şi un manager deductiv (modulul de procesare a actualizărilor).

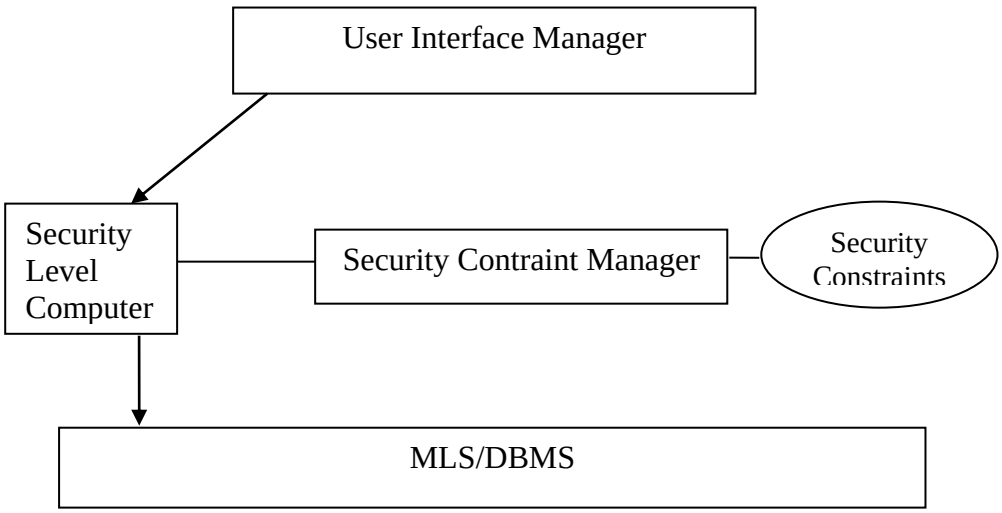

Figura 2. Modulul de procesare a actualizărilor

Acest modul ar putea fi utilizat:

- online acolo unde nivelurile de securitate ale datelor sunt determinate în timpul inserărilor şi actualizărilor în baza de date;
- offline ca un instrument care asigură că datele introduse sau actualizate prin mecanisme bulk data sunt etichetate adecvat.

Dacă acest instrument este utilizat offline, ar putea fi totuşi dificil să recalculeze nivelurile datelor care se află deja în baza de date dacă aceste niveluri sunt afectate de către datele noi care sunt inserate. Instrumentul, care tratează constrângeri de securitate în timpul proiectării bazei de date, ilustrat în figura 3, poate fi utilizat de către SSO pentru a proiecta schema. Intrarea este mulţimea de constrângeri de securitate care ar trebui să fie tratate în timpul proiectării bazei de date şi a schemei. Ieşirea este schema modificată şi constrângerile.

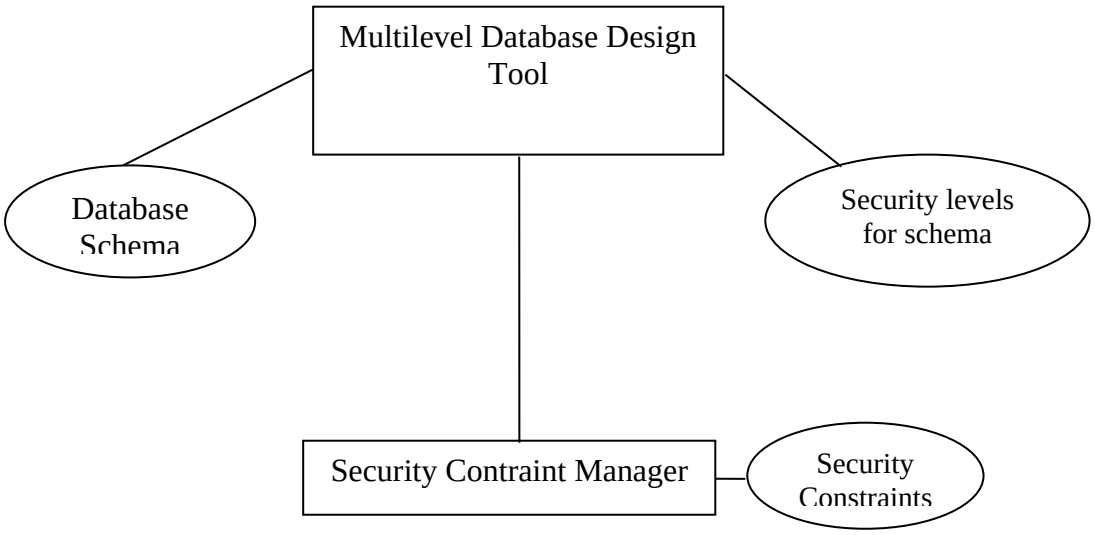

Figura 3. Utilitar pentru proiectarea bazei de date

Deşi modulele de procesare a cererilor, a actualizărilor şi utilitarul de proiectare a bazei de date sunt module separate, toate împreună constituie soluţia la procesarea constrângerilor în baze de date relaţionale multinivel. Aceasta înseamnă că ele furnizează o soluţie integrată la procesarea constrângerior de securitate într-un mediu multinivel.

Figura 4 ilustrează această arhitectură integrată.

- Constrângerile şi schema produsă de către generatorul de constrângeri sunt procesate mai departe de către instrumentul de proiectare a bazei de date.
- Constrângerile modificate sunt date lui *Constraint Updater* pentru a actualiza baza de date de constrângeri.
- Schema este dată lui *Constraint Generator*. Constrângerile din baza de date cu constrângeri sunt utilizate de către modulele de procesare a cererilor şi actualizărilor. Presupunem că există un manager de constrângeri sigur care gestionează aceste constrângeri. Într-un mediu dinamic în care datele şi constrângerile se modifică, modulul de procesare a cererilor va examina toate constrângerile relevante şi va asigura că utilizatorii nu obţin date neautorizate.

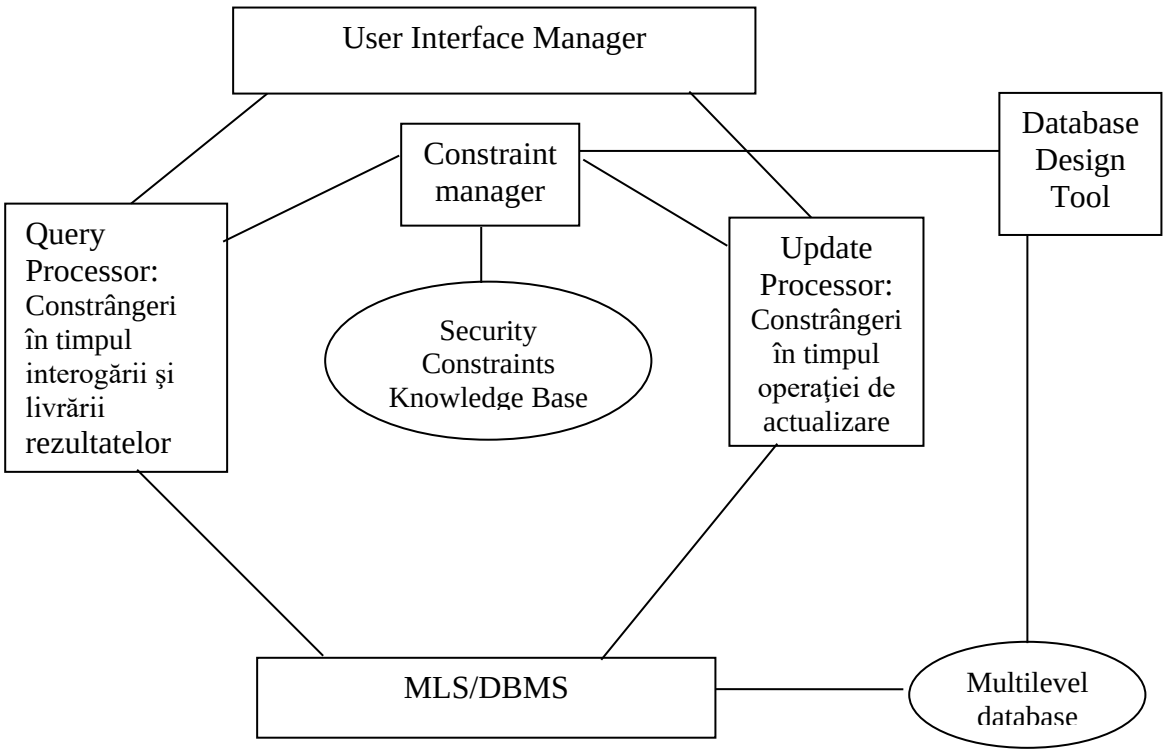

Figura 4. Arhitectura integrată

Consistenţa şi completitudinea constrângerilor

Cele două sarcini impuse de tratarea constrângerilor sunt generarea constrângerilor şi aplicarea acestora. Relaţia dintre acestea este ilustrată în figura 5.

- Generatorul de constrângeri recuperează specificaţia aplicaţiei multinivel şi determină schema iniţială şi constrângerile care trebuie aplicate.
- Instrumentul de proiectare a bazei de date preia acest rezultat şi proiectează baza de date.
- Modulele de procesare a actualizărilor şi cererilor utilizează constrângerile şi schemele produse de către instrumentul de proiectare a bazei de date.

Generarea mulţimii iniţiale de constrângeri rămâne o provocare. Este necesar să analizăm utilizarea structurilor conceptuale şi modelele semantice de date pentru a vedea dacă putem specifica aplicaţia şi, în consecinţă, genera constrângerile de securitate.

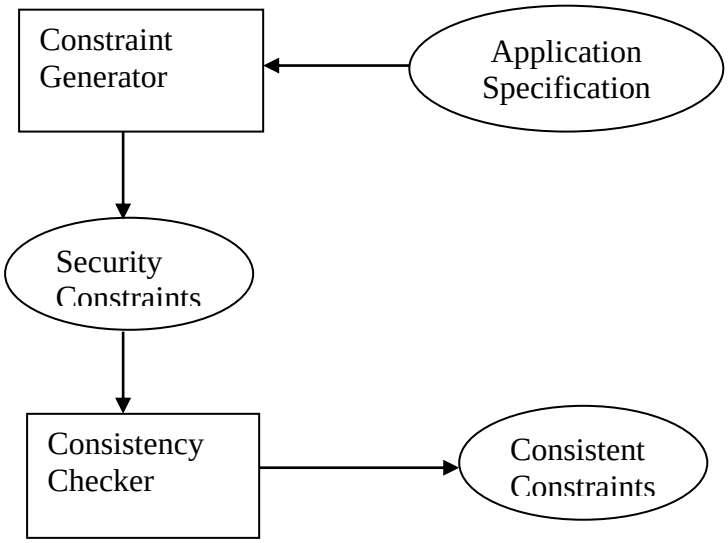

Figura 5. Generarea constrângerilor

Algoritmul următor (A) determină consistenţa şi completitudinea constrângerilor de securitate.

Algoritm A

Intrare : Mulţimea de constrângeri de securitate

Ieşire : Mulţimea de constrângeri de securitate consistente şi complete

Pasul 1. Pentru fiecare relaţie şi atribut, calculează nivelul de securitate asociat.

Pasul 2. Dacă există constrângeri care atribuie niveluri de securitate multiple unei relaţii sau unui atribut, atunci şterge sau actualizează constrângerea care atribuie nivelul de securitate cel mai mic; constrângerea este modificată dacă aceasta atribuie niveluri de securitate altor atribute şi relaţii.

Pasul 3. Dacă există o relaţie sau atribut căruia nu îi este asociat un nivel de securitate de către constrângeri, atunci creează o constrângere care va atribui cel mai mic nivel de securitate relaţiei sau atributului.

Pasul 4. Mulţimea de constrângeri care rezultă constituie ieşirea algoritmului.

3. Proiectarea modulului de procesare a cererilor

Prezentăm o politică de securitate pentru tratarea inferenţei în timpul procesării cererilor, iar apoi introducem o abordare a proiectării modulului.

Politica de securitate

Politica de securitate prezentată pentru procesarea cererilor extinde proprietatea securităţii simple pentru a trata încălcarea inferenţei.

Dat un nivel de securitate L, notăm prin E(L) baza de cunoştinţe asociată lui L. Prin urmare, E(L) va consta din toate răspunsurile care au fost livrate la nivelul de securitate L într-o anumită perioadă de timp şi informaţia din lumea reală la nivelul de securitate L.

Considerăm că un utilizator U cu nivelul de securitate L lansează o cerere. Atunci răspunsul R la cerere va fi livrat utilizatorului dacă este satisfăcută următoarea condiţie :

Pentru toate nivelurile de securitate L^*, unde L^* domină L, dacă ($E(L^*)$ UNION R) ==> X, pentru orice X, atunci L^* domină Level(X).

A ==> B semnifică faptul că B poate fi dedus din A utilizând oricare dintre strategiile de inferenţă şi Level(X) este nivelul de securitate al lui X.

Presupunem că orice răspuns care este livrat într-o bază de cunoştinţe la nivelul L este livrat şi în baza de cunoştinţe la nivelul $L^* >= L$.

Politica afirmă că, oricând un răspuns este livrat unui utilizator la nivelul L, trebuie să ne asigurăm că orice utilizator la nivelul $L^* >= L$ nu poate deduce informaţie clasificată la un nivel L+ >= L din răspuns împreună cu cunoştinţele care au fost deja obţinute.

Considerăm doar niveluri ierarhice în specificarea politicii, însă lucrurile pot fi extinse şi pentru niveluri neierarhice.

Funcţionalitatea modulului de procesare a cererilor

Puterea modulului de procesare a cererilor depinde de tipul strategiilor de inferenţă pe care le poate trata. În proiectarea pe care o prezentăm se consideră doar o mulţime limitată de strategii de inferenţă, cum ar fi inferenţa prin asociere şi deducţie. Vom discuta tehnicile care au fost utilizate pentru implementarea politicii de securitate : modificarea cererii şi procesarea răspunsului.

Trebuie menţionat că cea mai mare parte a codului modulelor de modificare a cererilor şi procesare a răspunsurilor trebuie să fie sigure deoarece efectuează funcţii critice din punct de vedere al securităţii.

Modificarea cererilor

Tehnica de modificare a cererilor a fost folosită în trecut pentru a trata securitatea opţională şi vizualizările. Ulterior, această tehnică a fost extinsă pentru a include securitatea obligatorie. În proiectarea procesorului de cereri, această tehnică este utilizată de către motorul de inferenţă pentru a modifica cererea în funcţie de constrângerile de securitate, răspunsurile livrate anterior şi informaţia din lumea reală. Atunci când va fi lansată cererea modificată, răspunsul generat nu va încălca securitatea.

Considerăm arhitectura pentru procesarea cererilor prezentată în figura 1. Motorul de inferenţă are acces la baza de cunoştinţe care include constrângerile de securitate, răspunsurile furnizate anterior şi informaţiile din lumea reală. Conceptual, putem gândi baza de date ca parte a bazei de cunoştinţe. Ilustrăm tehnica de modificare a cererilor prin exemple.

Fie baza de date ce conţine relaţiile SHIP şi MISSION, descrise anterior. Presupunem că baza de cunoştinţe este formată din următoarele reguli :

1. SHIP(X, Y, Z, A) and Z = Smith -> Level(Y, Secret)
2. SHIP(X, Y, Z, A) and A = 10 -> Level(Y, TopSecret)
3. SHIP(X, Y, Z, A) -> Level(Together(Y, Z), Secret)
4. SHIP(X, Y, Z, A) and Release(Z, Unclassified) -> Level(Y, Secret)
5. SHIP(X, Y, Z, A) and Release(Y, Unclassified) -> Level(Z, Secret)
6. NOT(Level(X, Secret) or Level(X, TopSecret)) -> Level(X, Unclassified)

Prima regula este o constrângere bazată pe conţinut, care clasifică numele unei nave al cărei căpitan este Smith la nivelul Secret. Similar, a doua regulă este o constrângere bazată pe conţinut care clasifică numele unei nave a cărei valoare M# este 10 la nivelul TopSecret. A treia regulă este o constrângere bazată pe asociere, care clasifică numele de nave şi căpitanii, luate împreună, la nivelul Secret. Regulile 4 şi 5 sunt restricţii suplimentare care sunt aplicate ca rezultat al constrângerii bazate pe context specificate în regula 3. Regula 6 afirmă că nivelul implicit de clasificare al unei date este Unclassified.

Presupunem că un utilizator Unclassified interoghează numele navelor din relaţia SHIP. Această cerere este reprezentată astfel:

SHIP(X, Y, Z, D)

Unn nume de navă este clasificat la nivelul Secret în cazul în care căpitanul său este Smith sau dacă numele acestuia a fost deja furnizat la nivelul Unclassified, şi este clasificat la nivelul TopSecret daca valoarea lui M# este 10. Presupunem că numele căpitanilor nu au fost încă furnizate unui utilizator Unclassified, iar cererea este modificată astfel :

SHIP(X, Y, Z, D) and NOT(Z = Smith and D = 10)

Menţionăm că, deoarece modificarea cererilor este efectuată în timp real, va avea impact asupra performanţelor algoritmului de procesare a cererilor.

Procesarea răspunsului

Fie următoarele constrângeri de livrare:

- Toate numele navelor ale căror nume de căpitani au fost deja livrate către utilizatori Unclassified sunt Secret.
- Toate numele de căpitani ale căror nume de nave au fost deja livrate către utilizatori Unclassified sunt Secret.

Presupunem că un utilizator Unclassified cere pentru prima dată numele navelor. Pe baza celorlalte constrângeri impuse, presupunem că doar anumite nume sunt furnizate utilizatorului. Apoi, numele de nave livrate trebuie înregistrate în baza de cunoştinţe. Ulterior, presupunem că un utilizator Unclassified (nu neapărat acelaşi) cere numele căpitanilor. Aceste valori (unele sau toate) sunt apoi asamblate în răspuns. Înainte ca răspunsul să fie furnizat, numele navelor care au fost deja furnizate utilizatorului Unclassified trebuie să fie analizate. Apoi, numele căpitanului care corespunde unui nume de navă care a fost deja furnizat este suprimat din răspuns.

Exemplu: Fie următoarea constrângere agregat :

O colecţie de 10 sau mai multe tupluri din relaţia SHIP este Secret.

Presupunem că un utilizator Unclassified cere tuplurile din SHIP. Răspunsul este asamblat şi apoi analizat în vederea verificării dacă are mai mult de 10 tupluri. În caz afirmativ, este suprimat.

Există nişte probleme asociate cu menţinerea informaţiei livrate. Pe măsură ce este inserată mai multă informaţie relevantă, baza de cunoştinţe poate creşte într-un ritm foarte rapid. Prin urmare, trebuie dezvoltate tehnici eficiente pentru procesarea bazei de cunoştinţe. Acest lucru ar avea impact şi asupra performanţelor algoritmilor de procesare a cererilor. O soluţie ar

consta în includerea doar anumitor informaţii furnizate în baza de cunoştinţe. Restul informaţiilor poate fi stocat cu datele de audit, care pot fi apoi utilizate de către SSO pentru analiză.

4. Proiectarea modulului de procesare a actualizărilor

Politica de securitate

MLS/DBMS-urile asigură atribuirea unui nivel de securitate datelor pe măsură ce acestea sunt inserate sau modificate. Nivelul de securitate atribuit datelor este presupuns a fi, în general, nivelul de securitate la conectare al utilizatorului care introduce date. O abordare mai puternică şi dinamică în atribuirea nivelurilor de securitate pentru date este prin intermediul utilizării constrângerilor de securitate în timpul operaţiilor de actualizare.

Politica de securitate a procesorului de actualizare erse formulată pornind de la proprietatea securităţii simple şi de la o politică de securitate furnizată de către MLS/DBMS-ul de bază. Această politică este următoarea:

1. Tuturor utilizatorilor le este acordat un nivel de autorizare raportat la nivelurile de securitate. Un utilizator se poate conecta la orice nivel care este dominat de nivelul său maxim de autorizare. Subiecţii acţionează în numele utilizatorilor la nivelul de securitate de la log-in.
2. Obiectele sunt liniile, tabelele şi bazele de date, iar fiecărui obiect îi este atribuit un nivel de securitate la creare.
3. Un subiect are acces pentru citire la un obiect dacă nivelul de securitate al subiectului domină nivelul de securitate al obiectului.
4. Un subiect are acces pentru scriere la un obiect dacă nivelul de securitate al obiectului domină nivelul subiectului.

Afirmaţiile 3 şi 4 din politica de mai sus sunt de fapt proprietatea * simplă din politica Bell-La Padula.

Funcţionalitatea modulului de procesare a actualizărilor

Modulul de procesare a actualizărilor utilizează constrângeri de securitate simple şi dependente de conţinut dret ghid pentru determinarea nivelului de securitate al datelor care sunt actualizate. Utilizarea constrângerilor de securitate poate proteja împotriva etichetării incorecte a datelor ca rezultat al conectării utilizatorilor la niveluri greşite sau al importului din sisteme cu

moduri de operare diferite şi împotriva inconsistenţelor bazei de date ca o consecinţă a etichetei de securitate a datelor din baza de date în care sunt introduse date.

Nivelul de securitate al unei cereri de actualizare este determinat de către modulul de procesare a actualizărilor după cum urmează. Constrângerile de securitate simple şi dependente de conţinut asociate cu relaţia care este actualizată şi cu o etichetă de securitate mai mare decât nivelul de securitate la conectare al utilizatorului sunt regăsite îi analizate pentru aplicabilitate. Dacă se aplică mai multe constrângeri, nivelul de securitate este determinat de către constrângerea care specifică cel mai înalt nivel de clasificare ; dacă nu se aplică nicio constrângere, nivelul pentru actualizare este cel de conectare al utilizatorului. Prin urmare, procesorul de actualizare nu determină nivelul de securitate a datelor doar din constrângerile de securitate, dar foloseşte aceste constrângeri pentru a determina nivelul datelor de intrare. Următorul exemplu ilustrează funcţionalitatea procesorului de actualizare.

Considerăm o bază de date care constă din relaţia SHIP(ship_number, name, class, date) şi MISSION (mission_number). Constrângerea bazată pe conţinut care clasifică toate valorile SHIP cu numele Josephine secret este exprimată prin:
SHIP.sname = “Josephine” -> Secret

Un utilizator cu nivelul de securitate la log-in Confidential introduce următoarele date:
INSERT INTO SHIP VALUES(“S123”, “James”, “Thomsen”, “MR2000”)

Procesorul primeşte comanda de inserare şi recuperează constrângerile asociate relaţiei SHIP care specifică un nivel mai mare decât cel al utilizatorului, şi al căror nivel este mai mic sau egal decât cel al utilizatorului. Constrângerea anterioară bazată pe conţinut este regăsită. Deoarece numele introdus nu este “Josephine”, constrângerea de securitate asociată relaţiei SHIP nu va afecta nivelul de clasificare al inserării, iar procesorul de actualizare va determina ca nivelul de inserare să fie cel al utilizatorului, Confidential.

Presupunem că un utilizator cu nivelul de securitate la conectare Confidential introduce următoarea comandă:
INSERT INTO SHIP VALUES(“S124”, “Josephine”, “Jane”, “MR3000”)

Din nou, procesorul va regăsi constrângerea asociată relaţiei SHIP. Având în vedere că numele este Josephine, procesorul va determina ca nivelul la inserare să fie Secret. Dacă utilizatorul se conectează cu nivelul de securitate TopSecret, procesorul va efectua inserarea la acest nivel (pentru că este mai mare decât cel specificat în constrângerea de securitate).

Operaţia de actualizare funcţionează similar celei de inserare. Presupunem că un utilizator cu nivelul Confidential lansează comanda:

UPDATE SHIP
SET name=”Josephine”
WHERE captain = “Thomsen”;

Procesorul va regăsi constrângerea şi va determina ca nivelul la actualizare să fie Secret.

În plus faţă de descrierea funcţionalităţii procesorului de actualizare, exemplele anterioare ilustrează canalele de semnalizare potenţiale care există atunci când operăm cu procesorul de actualizare. Un canal de semnalizare apare atunci când acţiunile unui utilizator cu nivel înalt de securitate interferă cu un utilizator cu nivel scăzut de securitate într-o manieră vizibilă. Canalele de semnalizare apar atunci când datele sunt introduse la un nivel mai mare decât al utilizatorului, iar utilizatorul încearcă să recupereze datele pe care le-a introdus, sau atunci când procesorul de actualizare încearcă să introducă date la un nivel mai înalt dar nu poate deoarece un tuplu cu acea cheie primară există deja la acest nivel.

5. Procesarea constrângerilor de securitate în cadrul proiectării bazei de date

O constrângere bazată pe asociere clasifică o colecţie de atribute, luate împreună, la un anumit nivel de securitate. O astfel de constrângere poate genera relaţii între diferitele atribute.

Exemplu: Dacă avem o relaţie SHIP (S#, SNAME, CAPTAIN) şi o constrângere bazată pe asociere clasifică atributele SNAME şi CAPTAIN luate împreună, la nivelul Secret, atunci una dintre perechile (S#, SNAME), (S#, CAPTAIN) ar trebui să fie, de asemenea, clasificate la nivelul Secret. Altfel, un utilizator Unclassified poate obţine perechile (S#, SNAME) şi (S#, CAPTAIN), iar apoi poate deduce asocierea secretă (SNAME, CAPTAIN).

A fost proiectat un algoritm care procesează o mulţime dată de constrângeri bazate pe asociere şi determină schema pentru baza de date multinivel. Dată mulţimea de constrângeri bazate pe asociere şi o schemă iniţială, algoritmul va determina grupările (*cluster*-ele) de atribute şi nivelul de securitate al fiecărui *cluster*.

Un astfel de algoritm nu trebuie să fie executat doar la momentul proiectării, ci poate fi utilizat şi în timpul procesării cererilor. Astfel, procesorul de cereri poate examina atributele din

diversele *cluster*-e generate de către algoritm, iar apoi poate determina informaţia care urmează să fie livrată utilizatorilor. De exemplu, dacă algoritmul plasează atributele A1, A2 în *cluster*-ul 1 la nivelul *L* şi atributele A3, A4 în *cluster*-ul 2 la nivelul *L*, atunci, după ce un atribut din *cluster*-ul 1 a fost livrat unui utilizator la nivelul *L*, niciun atribut din *cluster*-ul 2 nu poate fi livrat utilizatorilor de la nivelul *L*.

Deoarece o constrângere simplă poate fi privită ca un caz particular de constrângere bazată pe asociere, în cadrul căreia un singur atribut este clasificat, reiese că şi acest tip de constrângeri poate fi tratat la momentul proiectării bazei de date.

O altă constrângere care poate fi tratată la momentul proiectării bazei de date este constrângerea logică. De exemplu, dacă un atribut A implică un atribut B, iar atributul B este clasificat la nivelul Secret, atunci şi atributul A trebuie să fie clasificat cel puţin la nivelul Secret. Dacă o constrângere are condiţii asociate ei, atunci tratarea acesteia la momentul proiectării poate deveni dificilă. De exemplu, considerăm constrângerea: „Atributele SNAME şi LOCATION, luate împreună, sunt la nivelul Secret dacă LOCATION aparţine regiunii X”. O astfel de constrângere depinde de valorile datelor, prin urmare este tratată mai bine în timpul procesării interogării sau actualizării.

Algoritmul de tratare a constrângerilor în etapa de proiectare a bazei de date funcţionează în modul descris în continuare.

- ***Constrângeri bazate pe asociere***

Intrare : Mulţimea de constrângeri bazate pe asociere şi o mulţime de atribute.
Ieşire : Mulţimea de *cluster*-e pentru fiecare nivel de securitate.

Fiecare *cluster* asociat unui nivel de securitate *L* va avea o colecţie de atribute care pot fi clasificate sigur la nivelul *L*. Dacă A1, A2 şi A3 sunt atribute într-un *cluster* C la nivelul Secret, atunci atributele A1, A2 şi A3 pot fi clasificate împreună în siguranţă la nivelul Secret fără a încălca securitatea. După ce *cluster*-ele sunt formate, baza de date poate fi definită conform dependenţelor funcţionale şi multi-valoare care au loc.

- ***Constrângeri simple***

Dacă un atribut A din relaţia R este clasificat la nivelul *L*, atunci toate elementele care aparţin lui A sunt, de asemenea, stocate la nivelul *L*. Prin urmare, putem stoca pe A la nivelul *L*.

Algoritmul care tratează constrângerile simple rezultă direct. Fiecare atribut care este clasificat printr-o constrângere simplă este stocat la nivelul specificat în constrângere. După ce

algoritmul pentru procesarea constrângerilor simple a fost aplicat şi este obţinută schema corespunzătoare, această schemă este dată ca intrare algoritmului de tratare a constrângerilor bazate pe asociere. Aceste constrângeri sunt apoi aplicate şi se obţine schema finală.

- ***Constrângeri logice***

Sunt reguli care pot fi folosite pentru a deduce date noi din cele existente. Dacă o constrângere de securitate clasifică datele noi la un nivel mai înalt decât cel al datelor existente, atunci datele existente trebuie reclasificate. Constrângerile logice pot fi directe, de forma Ai => Aj sau pot fi mai complexe, precum A1 & A2 & A3 & An => Am. Dacă Aj este clasificat la nivelul Secret atunci Ai trebuie clasificat cel puţin la nivelul Secret. Dacă Am este clasificat la nivelul secret, atunci cel puţin unul dintre A1, A2,…, An trebuie să fie clasificat cel puţin la nivelul Secret.

6. Procesarea controlului securităţii şi controlul livrării

Am prezentat o arhitectură integrată pentru un mediu centralizat, în principal, în care constrângerile sunt examinate la momentul cererii, actualizării sau proiectării. Aceasta presupune, în cazul operației de interogare, că o mare parte din activitate este efectuată înainte ca cererea să fie trimisă către MLS/DBMS. După execuția cererii, sunt examinate anumite constrângeri legate de livrarea rezultatului.

Au fost efectuate cercetări asupra procesării constrângerilor numai după ce răspunsul este furnizat de către SGBD, dar înainte să fie livrat utilizatorului. Ideea de la care au pornit aceste cercetări este aceea că interogările nu pot fi modificate în anumite cazuri, mai ales dacă există multe constrângeri.

Această abordare are câteva dezavantaje. După livrarea răspunsului de către sistem, trebuie examinate toate constrângerile de securitate şi determinate informaţiile care vor fi furnizate utilizatorului. Multe dintre operaţiile efectuate de către SGBD va trebui să fie efectuate de către *Release Control Manager* (RCM).

Avantajul abordării este că, dacă există multe constrângeri, procesul complex de modificare a cererii este evitat.

SGBD-ul va produce rezultatul la nivelul de securitate al utilizatorului. RCM va analiza acest rezultat şi constrângerile, iar apoi va determina dacă toate datele obţinute pot fi livrate.

Presupunem că există o constrângere prin care valorile coloanei LOCATION din tabelul MISSION sunt Secret şi nivelul utilizatorului este Unclassified. Datele furnizate vor avea toate

informaţiile din relaţia MISSION. Constrângerea anterioară va fi aplicată de către RCM şi va livra doar valorile permise.

Figura 6 ilustrează modulul RCM.

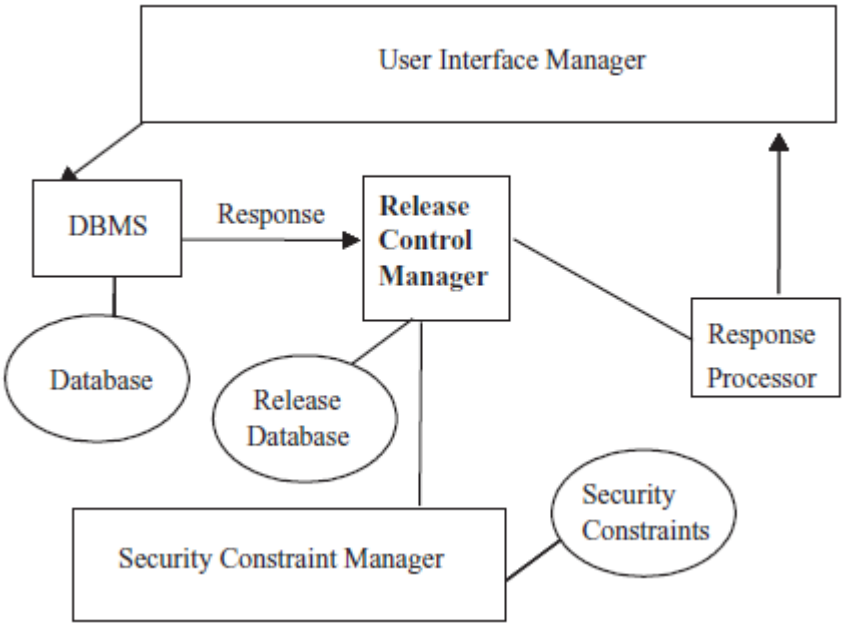

Figura 6. *Release Control Manager*