UNIVERSITATEA DE STAT DIN MOLDOVA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
DEPARTAMENTUL INFORMATICĂ

Tatiana PAȘA

SECURITATEA BAZELOR DE DATE

Note de curs

Aprobat de
Consiliul Calității al USM

Chișinău 2022
CEP USM

CZU 004.65.056(075.8)
P 58

Recomandat *de Departamentul Informatică al Facultății de Matematică și Informatică a USM*
Aprobat *de Consiliul Facultății de Matematică și Informatică a USM*

Recenzenți: ***Ion ANDRIEȘ****, dr., conf. univ.*
Aureliu ZGUREANU, *dr., conf. univ.*

DESCRIEREA CIP A CAMEREI NAȚIONALE A CĂRȚII
Paşa, Tatiana.

Securitatea bazelor de date: Note de curs / Tatiana Paşa; Universitatea de Stat din Moldova, Facultatea de Matematică și Informatică, Departamentul Informatică. – Chişinău: CEP USM, 2022. – 100 p.: tab.

50 ex.

ISBN 978-9975-159-40-1.

004.65.056(075.8)

P 58

ISBN 978-9975-159-40-1

CUPRINS

INTRODUCERE

Cursul fundamental ***„Securitatea bazelor de date***” este preconizat pentru studenții anului III, specialitatea *Informatică Aplicată*, opțiunea „Securitate cibernetică”. Are drept scop pregătirea studenților interesați în proiectarea, crearea, securizarea și administrarea bazelor de date utilizând sisteme de gestiune a bazelor de date.

Activitatea de predare-învățare la disciplina „Securitatea bazelor de date” este organizată în cadrul orelor de curs și de laborator. În cadrul orelor de laborator studenții realizează sarcini practice create de profesor în baza temelor abordate la orele de curs. Sarcinile sunt formulate în scris, iar studenții le realizează la calculator, apoi explică cum au fost soluționate. Aceste sarcini au rolul de a valorifica și aprofunda cunoștințele și de a stimula activitatea de învățare.

În baza cunoștințelor acumulate la orele de curs și de laborator fiecare student va elabora o lucrare individuală – o bază de date în cadrul căreia sunt implementate metode și practici de securizare a bazelor de date. Studentul va proiecta o bază de date bazându-se pe un plan de securitate elaborat individual în baza cunoștințelor obținute la orele de curs și o va prezenta drept proiect final ce conține managementul utilizatorilor, securitatea internă a datelor, auditul bazei de date, elaborarea unui plan de creare a copiilor de rezervă și restabilire a bazei de date și prevenirea SQL Injection. Studentul va prezenta lucrarea public – colegilor și profesorului. Realizarea lucrărilor practice ale cursului presupune utilizarea MS SQL Server și un sistem de operare.

Cursul este structurat în 7 unități de învățare după cum urmează:

1. **Concepte fundamentale în securitatea bazei de date.** În cadrul unității studentul ia cunoștință de SGBD și elementele de bază care implică securitatea bazelor de date: confidențialitatea, integritatea și accesibilitatea datelor, amenințări, vulnerabilități și riscuri de securitate.
2. **Managementul accesului la baza de date.** Unitatea presupune că studentul va cunoaște cum pot fi implementate politicile de administrare și parolele. Tot aici studentul va cunoaște cum pot fi creați identificatori, utilizatori și roluri ai bazei de date, dar și cum pot fi asignate utilizatorilor anumite privilegii și roluri.
3. **Securitatea internă a datelor.** După studierea acestei unități, studentul are capacitatea de a securiza date prin aplicarea constrângerilor de integritate, crearea vederilor și criptarea datelor cu caracter privat.

4. **Metode de audit al bazei de date.** Cunoștințele acumulate îi permit studentului să cunoască cum pot fi implementate soluții de audit pentru verificarea corectitudinii aplicării metodelor de securitate.

5. **Vulnerabilități ale bazei de date**. În cadrul unității studenții se familiarizează cu vulnerabilitățile bazelor de date accesibile online și cu metodele de prevenire a acestora.

6. **Metode de siguranță a datelor. Riscuri și proceduri pentru restabilirea datelor.** Unitatea are ca scop familiarizarea studentului cu diferite tipuri de copii de rezervă, planuri de creare a copiilor de rezervă și restabilire a bazelor de date. Aceste cunoștințe permit studentului să soluționeze probleme de corupere a datelor.

7. **Problema interferenței și concurenței în baze de date.** Ca urmare a studierii acestei unități, studentul poate gestiona corect tranzacțiile, cunoaște care sunt anomaliile de execuție concurentă a tranzacțiilor și poate utiliza tehnici de control al concurenței.

Cursul permite studentului să cunoască cum poate fi aplicată stocarea, gestionarea și transmiterea corectă și sigură a datelor. Un aspect important este cunoașterea și utilizarea metodelor din matematica aplicată și a softului de sistem, pentru soluționarea problemelor care ar putea apărea în procesul de securizare a bazelor de date, precum și aplicarea metodelor și softului instrumental de criptare și securizare a datelor. Studentul va fi capabil să analizeze, să depisteze și să înlăture posibilele vulnerabilități în procesul transmiterii datelor în rețea, dar și va argumenta eficiența implementării soluțiilor de securitate în cadrul unităților economice în care activează, realizând aplicații și sisteme informatice sigure în cadrul diferitor tipuri de rețele de calculatoare.

Competențele specifice formate în cadrul cursului:

- Cunoașterea arhitecturii calculatorului și a sistemelor de operare;
- Aplicarea rețelelor de calcul, a soft-ului de sistem, a calculatoarelor personale în domeniul de activitate profesională;
- Utilizarea metodelor matematicii aplicate și a softului instrumental la soluționarea problemelor de automatizare a gestiunii întreprinderilor;
- Asigurarea comunicării informaționale în cadrul întreprinderii prin intermediul rețelelor de calculatoare;
- Administrarea eficientă a sistemelor de operare distribuite, a reţelelor de calculatoare, a bazelor de date din cadrul unităţilor economice;
- Dezvoltarea aplicaţiilor WEB.

Aceste competențe pot fi consolidate prin consultarea surselor bibliografice recomandate la finele fiecărei unități de învățate.

I. CONCEPTE FUNDAMENTALE ÎN SECURITATEA BAZELOR DE DATE

1.1. Sisteme de gestiune a bazelor de date. Inițiere în securitatea bazelor de date

Istoria bazelor de date începe în anul 1960 când s-a dezvoltat sistemul de gestiune a fișierelor (SGF). Generația întâi include IMS, DBMS elaborate de IBM, SOCRATE elaborat de CII etc.

Definiția 1.1. O *bază de date* (BD) este o mulțime structurată de elemente, de date și de legături logice între ele ce descriu un univers real sau conceptual (o întreprindere, o societate de asigurări, eveniment etc.) și care pot fi accesate simultan de mai mulți utilizatori.

Definiția 1.2. Un *sistem de gestiune a bazei de date* (SGBD) reprezintă un ansamblu integrat de pachete de aplicații (programe), componente hardware și proceduri prin care sunt create, stocate, modificate, actualizate și securizate BD dintr-o întreprindere.

Deci, o BD este o colecție de date gestionate de un SGBD.

Definiția 1.3. *Baza de date relațională* reprezintă o colecție de date organizată sub forma unor tabele, în care coloanele poartă numele de câmpuri, liniile se numesc înregistrări, capetele de tabel fiind echivalentul structurii BD. Între câmpurile unui tabel există legături de interdependență numite relații.

A doua generație de SGBD, care a apărut după anul 1970, se bazează pe modelul relațional descris de E.Codd care vizează simplificarea înțelegerii, optimizarea structurilor de date și optimizarea accesului la date pentru utilizatori. Acest model de date are un mod eficient de structurare bazat pe teoria normalizării și determină cel mai bun plan de acces la datele pe care dorim să le obținem. Această generație, a cărei fundamentare teoretică se bazează pe teoria relațională, a cunoscut cea mai rapidă dezvoltare. Primele SGBD din această generație au fost comercializate după anul 1980, ca de exemplu DB2, Access, MS SQL, Oracle. Apare limbajul SQL care permite interogarea și manipularea datelor.

O a treia generație care s-a preconizat după anul 1990 este cea de BD obiectuale (orientate obiect). Sistemele de gestiune a bazelor de date obiectuale (SGBDOO) erau concepute pentru a permite administrarea datelor complexe. Neomogenitatea datelor implică o prelucrare anevoioasă a lor și din această cauză astfel de SGBD nu a avut răspândire largă.

Ulterior s-a dezvoltat SGBD bazat pe un model de date nerelațional, în care datele nu sunt stocate în formă tabelară, dar în formă de documente (de exemplu, în format JSON).

SGBD relaționale sunt prevăzute cu posibilitatea realizării de legături între două sau mai multe tabele, legături care permit accesarea simultană a unor date care se află într-o anumită relație, din diferite tabele. Astfel, programatorul nu trebuie să căute în toate tabelele datele ce corespund unor date de referință, această operație fiind realizată automat de SGBD. Programatorul trebuie doar, inițial, să definească relațiile între tabele, urmând ca găsirea anumitor informații într-o BD de referință să determine găsirea automată a informațiilor corespunzatoare din celelalte tabele aflate în relație cu aceasta.

Un SGBD este compus din module, fiecare având în cadrul sistemului sarcini specifice.

SGBD reprezintă ansamblul de programe care gestionează BD și care asigură realizarea următoarelor operații:

- definirea structurii BD;
- încărcarea datelor în BD;
- accesul la date;
- întreținerea BD;
- reorganizarea BD;
- securitatea datelor;
- integritatea datelor.

SGBD apare ca un sistem complex de programe, de aplicație și limbaj propriu, care asigură interfața între o BD și utilizatorii acestuia.

În cadrul cursului vor fi detalizate o serie de obiective care îi revin SGBD:

Implementarea securității datelor împotriva accesului neautorizat la ele. Administratorul BD poate prevedea ca accesul la BD să se realizeze numai prin căile corespunzatoare și să definească niveluri de autorizare pentru accesul la anumite date.

Asigurarea integritatii și securității datelor împotriva unor ștergeri intenționate sau neintenționate, prin intermediul unor proceduri de validare, a unor protocoale de control concurent și a unor proceduri de recuperare a BD după incidente.

În cazul în care la conceperea unei BD securitatea este privită ca un aspect important dar nu ca un adaos la BD, managementul securității va fi minim; în caz contrar, cheltuielile de întreținere cresc foarte mult. Conceperea

efectivă a unei BD este etapa cea mai importantă pentru dezvoltarea și menținerea cu succes a aplicației.

Conceperea unei soluții de securitate presupune înțelegerea necesităților organizaționale ale întreprinderii, iar pentru implementarea efectivă a securității se analizează:

- ***Natura datelor:*** BD pot avea excepții de securitate complet diferite în dependență de specificul datelor stocate în ea, cum ar fi: date personale, salariu, obligațiuni contractuale, secrete tehnologice etc.;

- ***Cerințele de audit existente*****:** În urma analizei se determină care cerință a auditului este luată în considerare în conceperea sistemului și care eveniment poate fi scos dacă conceperea devine prea complexă sau costurile de administrare devin prea mari. Este important să fie clar care evenimente au nevoie să fie auditate înainte de continuarea conceperii sistemului de securitate. Dacă o BD este concepută și implementată, poate fi mult mai greu de implementat strategiile auditului, deoarece multe dintre acestea pot afecta conceptul general al BD (logare (cât de des, la ce oră, numărul de eșecuri), modificarea datelor (nesancționată), vizualizarea rapoartelor).

- ***Administratorul serverului BD:*** Determinarea persoanei care este responsabilă de administrarea serverul BD este critică în crearea infrastructurii securității serverului. Un administrator este cel care creează conturile, acordă privilegii, retrage privilegii și atribuie diferite niveluri de securitate. Ar fi o bună idee minimizarea numărului administratorilor de sistem pentru o singură instanță. Cu cât numărul administratorilor de sistem scade, cu atât responsabilitatea individuală a acțiunilor crește. Este importantă documentarea individuală a persoanelor care administrează fiecare server și documentația trebuie distribuită tuturor persoanelor care au acces la BD, astfel încât fiecare persoană afectată trebuie să știe cine este responsabil de administrare; ca rezultat, crește eficiența și scăde riscul de operațiuni greșite.

Mecanismul de securitate al unui SGBD include modalități de restricționare a accesului la sistem, ceea ce presupune crearea de conturi de utilizator și parole; astfel, SGBD controlează procesul de conectare.

Prin securitatea BD se întelege protejarea datelor împotriva folosirii neautorizate a lor, în special a modificărilor, distrugerilor nedorite și a citirilor nepermise de date. Se recunoaște că, de fapt, nu există protecție absolut sigură, ci doar protecții și măsuri de securitate mai eficiente sau mai puțin eficiente.

BD neprotejate sunt visul criminalității cibernetice. BD care conțin date valoroase ale unei întreprinderi ar putea fi o țintă ușoară a unui atac. Cu toate

acestea, există un șir de recomandări care pot fi urmate pentru a proteja BD din întreprindere și, în același timp, reputația sa.

Securitatea BD este o componentă importantă a planului general de securitate a sistemelor informaționale ale oricărei întreprinderi. Proiectanții de BD au responsabilitatea de a proteja confidențialitatea persoanelor ale căror date sunt stocate. Confidențialitatea este dreptul persoanelor de a avea un anumit control asupra informațiilor despre ele însele. Sunt concepute un șir de legi menite să protejeze confidențialitatea și orice întreprindere care colectează și stochează informațiile despre persoane fizice sunt obligate din punct de vedere legal să adopte politici conforme cu legislația locală privind confidențialitatea. Proiectarea BD trebuie să reflecte angajamentul întreprinderii față de protecția drepturilor persoanelor la confidențialitate. Securitatea informațiilor urmează de obicei modelul CID, unde CID reprezintă *confidențialitate* (numai utilizatorii autorizați au acces la informații pentru a păstra confidențialitatea persoanelor), *integritate* (doar utilizatorii autorizați pot modifica datele pentru a menține coerența și încrederea datelor) și *disponibilitate* (informațiile sunt accesibile utilizatorilor autorizați atunci când este necesar). Odată cu creșterea social media și a afacerilor online datorită internetului, menținerea confidențialității implică utilizarea unor tehnici de criptare adecvate, precum și proceduri de autorizare, identificare și autentificare a utilizatorului. Nerespectarea integrității implică generarea datelor incorecte care pot fi și dăunătoare persoanelor și întreprinderii. Atacurile de securitate împotriva unei întreprinderi pot rezulta cu indisponibilitatea serviciilor întreprinderii, ceea ce duce la încălcarea acordurilor comerciale.

Securitatea este în general asociată cu urmatoarele situații:

- acces fraudulos la date;
- pierderea confidențialității datelor;
- pierderea caracterului privat al datelor;
- pierderea integrității datelor;
- pierderea disponibilității datelor.

Forme de acces intenționat și răuvoitor la datele unei BD:

- citire neautorizată a unor date;
- modificări neautorizate de date;
- distrugeri de date.

Securitatea BD este o zonă largă care acoperă multe subiecte, inclusiv:

- probleme etice și juridice referitoare la dreptul de acces la anumite informații;

- probleme politice la nivel guvernamental, instituțional sau corporativ, legat de tipul de informații care nu ar trebui să fie disponibile publicului;

- probleme legate de sistem și nivelurile de sistem care gestionează diferite funcții de securitate.

Pentru protecția BD se pot lua măsuri de asigurare a securității la mai multe niveluri:

- *la nivel fizic* – încăperile în care se află serverele BD trebuie protejate de accesul persoanelor neautorizate;

- *la nivel uman* – este recomandabil ca autorizațiile de acces să se acorde cu multă grijă și să se țină evidența strictă a persoanelor autorizate;

- *la nivel sistem de operare (SO)* – slăbiciunile protecției la nivel de SO trebuie eliminate sau compensate de alte măsuri;

- *la nivel SGBD* – sistemul oferă anumite facilități care sprijină protecția datelor.

Menținerea securității la toate aceste niveluri ar trebui să consolideze securitatea BD. Slăbiciunea nivelurilor joase de securitate (fizică sau umană) poate ocoli securitatea strictă de măsuri la niveluri superioare (bază de date). Securitatea în cadrul sistemul de operare este aplicată la diferite niveluri: de la parole pentru a accesa sistemul până la izolarea proceselor concurente care rulează pe acesta. De asemenea, sistemul de fișiere oferă un anumit nivel de protecție.

Mecanismele de securitate pot fi orientate spre politici de control al accesului pe baza identificării utilizatorului sau politici care restricționează accesul la informații clasificate confidențial pentru personalul autorizat.

Putem vorbi despre două tipuri de mecanisme de securitate a BD:

- *mecanisme discreționare de securitate* pentru acordarea de privilegii utilizatorilor, inclusiv acces la fișiere, înregistrări sau anumite câmpuri de date într-un anumit mod;

- *mecanisme obligatorii de securitate* pentru gruparea pe mai multe niveluri a datelor și a utilizatorilor în mai multe clase (sau niveluri) și apoi implementarea corespunzătoare a politicii de securitate a întreprinderii.

Deoarece s-ar putea ajunge la date și prin alte mijloace decât prin intermediul SGBD (de exemplu, prin citirea directă a mediului magnetic), se poate face o protecție prin stocarea codificată a datelor pe mediul magnetic. Decodificarea datelor se poate face numai dupa identificarea utilizatorului. Deci, o tehnică de securitate este și criptarea datelor, utilizată pentru a proteja date sensibile transmise printr-o rețea de comunicații. Criptarea oferă suplimentar protecție secțiunii cu date confidențiale a unei BD. Datele sunt

codificate intenționat de un algoritm. Un utilizator neautorizat care are acces la date criptate va încerca să le descifreze, însă doar un utilizator autorizat va avea algoritmul (sau cheia) de criptare sau de decriptare în acest scop. Criptarea este utilizată pentru a stoca informații importante în surse nesigure și a le transfera prin canale de comunicare neprotejate.

Există două tipuri de criptare:

- *simetrică* – se utilizează aceeași cheie pentru criptare și decriptare. Algoritmul și cheia sunt selectate în prealabil și cunoscute de ambele părți. Astfel, șansa de a obține cheia este mai mare, deoarece răspândirea ei poate fi interceptată de persoane nedorite. Menținerea unei chei în secret este o sarcină importantă pentru stabilirea și menținerea unui canal de comunicare securizat;
- *asimetrică* – există o pereche de chei: una publică pentru a cripta informația și alta numită privată, pentru a decripta informația (o are numai cel care citește/decriptează informația).

Criptografia simetrică este mult mai vulnerabilă decât cea asimetrică din cauza utilizării unei singure chei; pe de altă parte, criptarea simetrică este mult mai rapidă decât cea asimetrică din același motiv.

1.2. Confidențialitatea și integritatea datelor

În Comunitatea Europeană există o preocupare serioasă legată de actualizarea legislației la noile nevoi generate de utilizarea intensivă a calculatoarelor. Se încearcă în principal să se adopte legi care să protejeze persoana sau întreprinderea și, respectiv, să protejeze datele cu caracter privat (să nu fie accesibile publicului larg și nici măcar unui grup relativ restrâns, dacă acest fapt ar dăuna proprietarului informațiilor respective). Putem enumera aici o paletă largă de domenii care lucrează cu informații al căror caracter privat trebuie neapărat păstrat: domeniul bancar, domeniul medical, evidențe administrativ-financiare, domeniul producției în majoritatea firmelor de marcă etc. Conducerea întreprinderii, care este proprietara BD, trebuie să ia măsuri de securitate care reduc riscul de a pierde sau de a distruge informații.

Confidențialitatea datelor este una dintre funcțiile SGBD, care este asigurată prin blocarea accesului unor categorii de utilizatori la date pe care nu trebuie să le acceseze și/sau de care nu au nevoie în activitatea lor, fapt ce minimizează riscul distrugerii accidentale a datelor prin operații

necorespunzătoare. Politica de securitate a unei BD presupune definirea de conturi de utilizare și atribuirea de drepturi de acces utilizatorilor. Fiecărui utilizator în parte i se pot acorda drepturi pentru accesul, modificarea, ștergerea datelor în mod individual.

Accesul discreționar este o modalitate de a restricționa accesul la informații bazat pe privilegii. Pot fi definite două niveluri de atribuire a privilegiilor:

- *la nivel de cont*, administratorul specifică privilegii speciale pe care fiecare utilizator al BD le are asupra tabelului (CREATE TABLE, CREATE VIEW, ALTER, MODIFY, SELECT);

- *la nivelul relației,* privilegii de a accesa fiecare tabel al BD și se atribuie unui proprietar de cont, care are toate privilegiile și este responsabil pentru acordarea acestora către alte conturi.

Politica de control al accesului la BD poate fi clasificată în 2 grupuri:

- *închis*, doar autorizarea accesului explicit este permis;

- *deschis*, accesul este permis implicit.

O problemă importantă în gestionarea BD este asigurarea respectării unui set predefinit de reguli de către datele BD. Aceste reguli trebuie să fie în concordanță cu activitățile comerciale pe care le susține sistemul. Integritatea datelor presupune definirea regulilor care restrâng valorile valide pentru o coloană a unui tabel.

Integritatea BD se referă la corectitudinea informațiilor și presupune detectarea, corectarea și prevenirea diferitor erori care pot afecta datele introduse în BD. Când facem referiri la integritatea datelor, înțelegem că datele sunt consistente relativ la toate restricțiile formulate anterior și, ca urmare a acestui fapt, datele sunt considerate valide.

Restricțiile de integritate sunt definite pentru tabele și, prin urmare, toate vederile sunt supuse restricțiilor de integritate. Dacă o instrucțiune DML (Data Manipulation Language) încearcă să efectueze o acțiune care încalcă o restricție de integritate, este generată o eroare și tranzacția este derulată înapoi. Restricțiile de integritate a datelor reprezintă mijloace utilizate pentru a preveni introducerea datelor invalide în tabelele BD și sunt definite pentru a impune respectarea regulilor comerciale care sunt asociate cu informațiile din BD.

Restricțiile de integritate pot fi clasificate în 2 grupuri:

1. *Restricții structurale* sunt exprimate prin dependențe funcționale, multivaloare, joncțiune sau prin declararea unor câmpuri cu valori unice (de

cele mai multe ori aceste câmpuri sunt chei). Acestea presupun respectarea următoarelor ***reguli de integritate***:

- cheia primară a unui tabel trebuie să fie unică și minimală;

- pentru fiecare rând din tabel cheii primare trebuie să-i fie atribuită valoare;

- într-un tabel ***A***, care se referă la un tabel ***B***, valorile cheii externe trebuie să figureze printre valorile cheii primare din ***B*** sau să ia valoarea *null* (neprecizat).

2. *Restricțiile de comportament* sunt cele care definesc comportamentul datelor și țin cont de valori. Restricțiile de comportament fiind foarte generale se gestionează fie la momentul descrierii datelor, fie în afara modelului la momentul execuției:

- *restricția de domeniu* se asociază cu un domeniu corespunzător unui atribut dintr-un tabel ce trebuie să se încadreze între anumite valori;

- *restricții temporale* – valorile atributelor se compară cu niște valori temporale (rezultate din calcule, cum ar fi salariul mediu, stagiul de muncă minim/maxim ș.a.).

1.3. Amenințări, vulnerabilități și riscuri de securitate

BD constituie unul dintre componentele cele mai importante ale oricărui sistem informatic care păstrează și prelucrează diverse date ce pot fi sisteme de gestiune a documentelor și a evidenței contabile, sisteme de gestiune a proceselor tehnologice la producere ș.a. Deoarece BD conține toată informația valoroasă despre întreprindere (clienții săi, activitatea financiară), ea reprezintă unul dintre factorii critici în structura acesteia, fapt ce determină cerințe sporite de confidențialitate, integritate și acces la date.

Riscurile care pot interveni în acest context pot apărea atât din mediul extern, cât și din cel intern. De asemenea, este necesar de a se lua în considerare și fiabilitatea sistemelor, a SGBD, a mijloacelor tehnice care se pot supune unor factori ca incendiile, calamitățile naturale ș.a., cum ar fi:

1. Pierderea consistenței BD:

- structură redundantă a tabelelor BD (grupuri de câmpuri care se repetă în mai multe tabele);

- existența înregistrărilor nule pentru câmpuri care identifică apartenența fenomenului economic (data operației, gestiunea, clientul, furnizorul, secția, grupa de produse);

2. Pierderea integrității BD:

- lipsa cheilor primare pentru câmpurile care necesită restricții de integritate referențială (serie_factura, nr_factura, CIF, nr_inventar, nr_matricol);

- inexistența restricțiilor de integritate referențială (relațiile *cheie primara – cheie externă*);

3. Acces neautorizat la BD (BD pot fi accesate de către orice utilizator cu drept de modificare, stergere, copiere etc.).

4. Pierderea intenționată (fraudă) sau accidentală (eroare) a BD din cauza inexistenței copiilor de rezervă ale BD (backup-uri).

5. Indisponibilitatea server-ului SGBD-ului:

- inexistența protecției contra întreruperilor în alimentarea cu energie electrică;

- blocarea protocoalelor de comunicație;

- blocarea sistemului de operare.

Printre ***amenințările accidentale*** ale securității pot fi următoarele:

- utilizatorul poate solicita neintenționat un obiect sau o operațiune pentru care nu ar trebui să fie autorizat, iar cererea ar putea să fie acceptată din cauza unei autorizări greșite sau din cauza unei erori în SGBD sau în sistemul de operare;

- un utilizator poate primi din greșeală un mesaj care ar trebui să fie direcționat unui alt utilizator, rezultând divulgarea neautorizată a conținutului BD;

- o eroare a sistemului de comunicații ar putea conecta un utilizator la o sesiune care aparține unui alt utilizator cu privilegii de acces diferit;

- sistemul de operare ar putea suprascrie accidental și distruge o parte a BD, pot fi preluate fișiere greșite și apoi din greșeală trimise utilizatorului neautorizat.

Amenințările intenționate (deliberate) de securitate apar atunci când un utilizator obține intenționat acces neautorizat și/sau efectuează operațiuni neautorizate în BD. Un angajat nemulțumit care cunoaște sistemul informatic al întreprinderii reprezintă o amenințare extraordinară pentru securitate. Utilizatorii privilegiați, cum ar fi administratorii BD care accesează datele utilizatorului final, pe care nu ar trebui să i se permită a le vedea, amenință securitatea. Există mai multe modalități de încercare intenționată a securității, care poate fi realizată prin:

- interconectarea liniilor de comunicație pentru interceptarea mesajelor către și din BD;

- ascultarea electronică, pentru a prelua semnale de la stațiile de lucru, imprimante sau alte dispozitive din interiorul unei clădiri;

- citirea ecranelor de afișare și citirea sau copierea tipăriturilor nesupravegheate de către utilizatorii autorizați;

- impersonarea unui utilizator autorizat sau a unui utilizator cu acces mai mare fiind folosită logarea și parola lui;

- scrierea programelor de sistem cu cod ilegal pentru a ocoli SGBD și mecanismul său de autorizare și acces la datele BD direct prin sistemul de operare;

- scrierea de programe de aplicații cu cod care permite operațiuni neautorizate;

- obținerea de informații despre date ascunse prin interogare inteligentă a BD;

- modificarea interogărilor BD prin SQL Injection pentru a obține acces neautorizat la date sau pentru a modifica sau șterge date cu rea intenție;

- scoaterea dispozitivelor de stocare fizică din computer;

- efectuarea de copii fizice ale fișierelor stocate fără a trece prin SGBD, ocolind astfel mecanismele de securitate ale acestuia;

- mituirea, șantajarea sau implicarea în alt mod a utilizatorilor autorizați pentru a-i folosi ca agenți în obținerea de informații sau pentru deteriorarea BD;

- folosirea privilegiilor de sistem pentru a permite accesul unui utilizator care nu ar trebui să-l aibă.

Vulnerabilitatea este o slăbiciune ce permite atacatorului să reducă din siguranța datelor BD. *Vulnerabilitatea* este intersecția a trei elemente: defectul, accesul atacatorului la defect și capacitatea atacatorului de a exploata defectul. Pentru a exploata vulnerabilitatea, atacatorul trebuie să dispună de cel puțin o unealtă aplicabilă sau tehnică pentru a se conecta la slăbiciunea BD.

Pentru a evita apariția unei vulnerabilități, este necesar de a evalua configurarea BD prin verificarea modului de instalare a BD și a sistemului de operare (verificarea fișierului grupurilor de privilegii, lognurilor BD și a tranzacțiilor). De asemenea, este necesar de a verifica dacă nu sunt rulate versiuni ale BD care includ vulnerabilități cunoscute.

1.4. Audit

Termenul vine de la latinescul *auditum* = *ascultare*. Ca demers de ascultare, apoi de anchetă și, în final, de sugerare de soluții, auditul permite aportul unui raționament motivat și independent. Ca examinare în vederea determinării proprietăților unei reprezentări, auditul este aplicat inițial reprezentărilor financiare.

Definiția 1.4.1. Auditul este procesul prin care persoane competente, independente colectează și evaluează probe pentru a-și forma o opinie asupra gradului de corespondență între cele observate și anumite criterii prestabilite.

În cadrul auditului trebuie să fie verificate dacă sunt întrunite condițiile necesare pentru a păstra echilibrul, să fie instrumentată stăpânirea dezordinii și adaptarea la schimbări, să fie evaluate gradul de securitate și riscurile. Auditul este implementat după aplicarea tuturor setărilor și controalelor și presupune efectuarea autoevaluării și monitorizarea recomandărilor de audit pentru a verifica existența abaterilor de la obiectiv (securitate).

Monitorizarea în timp real a activității BD este esențială pentru a limita expunerea detectând intruziunile și utilizarea abuzivă. O alertă cu privire la un acces neobișnuit indică prezența unui atac SQL Injection, modificarea neautorizată a datelor, modificări ale privilegiilor și modificarea configurării contului executând un script SQL. Este foarte importantă monitorizarea utilizatorilor privilegiați necesară pentru respectarea reglementărilor, cum ar fi SOX[1] și reglementările privind confidențialitatea. Monitorizarea permite detectarea intruziunilor, cele mai frecvente atacuri fiind îndeplinite de utilizatori cu mai multe privilegii.

Majoritatea întreprinderilor folosesc astăzi o formă de audit manual sau aplicații native ale SGBD, dar deseori aceste aplicații sunt dezactivate din cauza:

- complexității;
- costurilor de operare ridicate;
- problemelor de performanță;
- lipsei repartizării (segregării) atribuțiilor;
- nevoii de mai mult spațiu de stocare.

Soluțiile trebuie dezvoltate cu un impact minim asupra performanței și costului (bani, timp).

[1] Lege federală apărută în SUA în 2002 ca urmare a unor scandaluri financiare din cadrul unor corporații precum Enron, Tyco International, WordCom etc. Denumirea provine de la numele senatorului Paul Sarbanes și al lui Michael Oxley.

Surse bibliografice:

1. H. Afyouni. *Database Security and Auditing: Protecting Data Integrity and Accessibility*. Cengage Learning, 2006.
2. A. Basta, M. Zgola. *Database Security*. Cengage Learning, 2011.
3. T. Kyte, D. Kuhn. *Oracle Database Transactions and Locking Revealed*. Apress, 2014.

II. MANAGEMENTUL ACCESULUI LA BAZE DE DATE

Pentru a restricționa accesul la resursele întreprinderii, precum sunt datele angajaților și ale clienților, trebuie definite metode de control al accesului. Controlul accesului este o componentă fundamentală în sprijinul confidențialității și integrității.

Un plan de control al accesului ar trebui să înceapă cu măsuri de securitate fizică pentru clădirea în sine, cu precauții speciale pentru dotările informatice. Acestea încep de la ușa din față, unde toți angajații trebuie să fie identificați vizual de pază sau prin utilizarea insignelor, amprentelor sau a altor mecanisme. Ar trebui să fie necesară o identificare suplimentară pentru a avea acces la facilitățile computerului. Măsurile de securitate fizică ar trebui extinse pentru a acoperi orice locație în care sunt stocate date, cum ar fi copiile de rezervă. Împreună cu securitatea fizică, utilizatorii pot avea alte forme de identitate, cum ar fi cardurile inteligente care sunt trecute printr-un cititor de carduri electronic pentru a avea acces la parcări, clădiri și camere care găzduiesc facilități de BD, precum și alte spații generale de lucru. Biometria poate oferi o formă mai sigură de identificare, în special în aplicațiile extrem de confidențiale. Biometria poate include amprentele digitale, amprentele mâinilor, recunoașterea feței, recunoașterea vocii și scanarea retinei.

Pe de altă parte, la nivel de BD, utilizatorilor li se asociază diferite roluri (definite de administratorul BD), deci au acces la anumite date și pot îndeplini doar un set predefinit de operații. SGBD-urile conțin și câteva roluri predefinite, printre care sysadmin sau securityadmin care vor fi studiate în continuare.

2.1. Politici de administrare

În momentul conectării la SQL Server au loc o serie de lucruri care nu ies în evidență la prima vedere. Când se încearcă conectarea la SQL Server, securitatea este validată pe trei niveluri:

1) la nivelul sistemului de operare;

2) la nivelul SQL Server (sub forma unui identificator de conectare SQL Server);

3) la nivelul unei BD individuale (sub forma unui nume de utilizator al BD).

Faptul că există un identificator de conectare nu spune nimic despre bazele de date care pot fi accesate, pentru aceasta trebuie un nume de utilizator al BD.

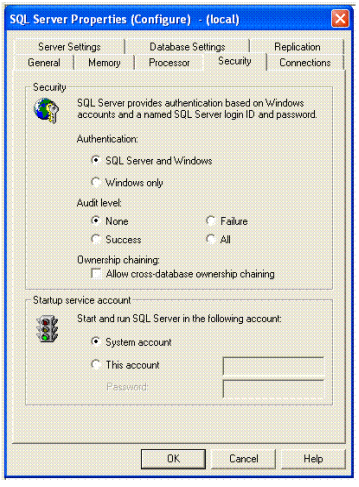
- *Autentificarea SO Windows.* Când se încearcă conectarea de pe un calculator client la un calculator Windows pe care rulează Microsoft SQL Server, s-ar putea ca Windows să solicite validarea conexiunii la rețea.

- *Autentificarea SQL Server.* Pentru a se putea realiza conexiunea la SQL Server, trebuie să se furnizeze un nume de utilizator și o parolă pentru SQL Server, ambele valide. Dacă atributele de identificare sunt valide, conexiunea la SQL Server este efectuată cu succes. În caz contrar accesul este interzis chiar dacă autentificarea de rețea Windows s-a încheiat cu succes.

- *Numele de utilizator al bazei de date SQL Server.* Pentru a utiliza fiecare BD din sistem, trebuie să se permită în mod explicit accesul la ea. Accesul la o BD se obține în mai multe moduri, care vor fi examinate mai târziu. În cazul în care nu există un nume de utilizator pentru o BD, este interzis accesul la aceasta.

- *Permisiuni.* Ultimul nivel de securitate îl constituie permisiunile. După ce se realizează conexiunea la SQL Server și se accesează o BD, se oferă dreptul explicit de a accesa obiectele BD (numai pentru citire sau și pentru modificări).

Pentru a stabili modul de securitate folosit de server, adică *Windows Integrated Mode* sau *Mixed Mode,* se utilizează instrumentul de gestiune din SQL Server numit SQL Server Enterprise Manager (meniul Properties – Security). Pentru a modifica atributul de securitate se bifează opțiunile dorite și se repornește serviciul SQL Server.



Poate fi activată facilitatea de monitorizare (audit) pentru server, în mod prestabilit aceasta este dezactivată. Opțiunile disponibile sunt:

- *Failure* (încercările eșuate de conectare la SQL Server);
- *Success* (conexiunile realizate cu succes)
- *All* (ambele).

Rezultatele procesului de monitorizare se pot examina în jurnalul de erori SQL Server.

Informațiile despre identificatori se

pastrează în SQL Server (în baza de date master, tabelul de sistem *sysxlogins*).

```
select substring (name,1,25) AS name,
substring (password,1,30) AS password
FROM sysxlogins
```

	name	password
1	Admin	0x0100817C372775B10826FB72FC2561C5DD1117934C715D03A1FFAC4CE1CB
2	BUILTIN\Administrators	NULL
3	cezar	0x0100C1459C7C766546638BAFAAAD8CFF52A5ACODF3C415051B3876654663
4	sa	0x0100D37AFF5420AC89ACEACFB03B9D6A950B7B3367421DAA901BF2E94A56
5	teste1	0x0100D67CDA6ACA12FA3CABCD1907AAB492F81617E95E7F0E1B24B768C0AA
6	teste2	0x0100177D92327A04851734282B1F007644DC33392FBF5CAD062189A4F448
7	teste3	0x0100487D5C2829BAC277CEB9328A991B5801501ABF4DDA15EBB8FD92C403

Autorizarea necesită definirea utilizatorilor care au acces la sistem și date specifice. Majoritatea SGBD sunt concepute pentru mai mulți utilizatori și au propriile subsisteme de securitate care prevăd autorizarea utilizatorului, o metodă prin care utilizatorilor le sunt atribuite drepturi de utilizare a obiectelor BD. Administratorul BD specifică drepturile utilizatorilor prin intermediul regulilor de autorizare, declarații care specifică utilizatorii care au acces la date și operații asupra lor. Mecanismul de autorizare este conceput pentru a proteja BD prin prevenire de persoane care intenționează citirea, actualizarea sau distrugerea neautorizată a conținutului BD. Aceste restricții se adaugă mecanismelor de securitate prevăzute de sistemul de operare.

Autentificarea este procesul de verificare a identității unui utilizator – verificare pentru a se asigura că utilizatorul real este cel care pretinde a fi. Autentificarea este implementată inițial la nivel de sistem de operare. Când utilizatorul se conectează, se introduce un ID de utilizator, care este verificat pentru valabilitate. În unele întreprinderi este utilizată o abordare multifactorială a autentificării în care utilizatorii trebuie să ofere două sau mai multe forme de autentificare. În abordarea multifactorială un utilizator ar putea furniza un ID de utilizator și o parolă, precum și un smartcard, o insignă, un simbol sau o formă de biometrie. O procedură de autentificare poate consta, de asemenea, în răspunsuri la o serie de întrebări care ar dura mai mult și ar fi mai complicat decât a reproduce o singură parolă.

O altă măsură de securitate este **blocarea** unui utilizator de la un cont după mai multe încercări de conectare nevalide. Politica de blocare împiedică hackerii de a ghici parola unui utilizator utilizând brutforce.

Componenta finală a controlului accesului la sistemul informațional este **responsabilitatea**. Responsabilitatea se referă la necesitatea de a captura și menține fișierele de jurnal care pot fi utilizate pentru trasabilitate atunci când apar incidente de securitate. De exemplu, sistemul de operare păstrează informațiile de conectare despre utilizatori, precum și fișierele și BD pe care le accesează. În fișierele de logare este stocată informația despre traficul de

rețea care poate fi folosit pentru a urmări accesul de la distanță la un sistem. SGBD mențin fișierele de logare ca parte a sistemului de recuperare a BD, sunt înregistrare informațiile de acces ale utilizatorilor, precum și inserările, actualizările și ștergerile de date care au loc.

2.2. Crearea, modificarea și eliminarea identificatorilor și utilizatorilor bazei de date

2.2.1. Gestionarea identificatorilor

Primul pas în configurarea serverului BD pentru acces este crearea identificatorilor. Se pot adăuga identificatori folosind procedura de sistem stocată *sp_addlogin*:

sp_addlogin [@loginame =] 'identificator'
[,[@passwd =] 'parola'
[,[@defdb=] 'baza_date'
[,[@deflanguage =] 'limba']]]

În *sp_addlogin*:

- *identificator* este numele de conectare al utilizatorului. Acest nume trebuie să fie un identificator valid SQL Server (începe cu o literă sau cu unul din caracterele #, @ sau _ , conține oricare din aceste caractere, litere și cifre – maximum 128 de caractere Unicode);

- *parola* este parola aferentă acestui identificator și este nulă dacă nu este aleasă în momentul creării identificatorului;

- *baza_date* este baza de date prestabilită care este accesată de utilizator la conectare. Dacă nu este specificat acest parametru, baza de date prestabilită va fi master;

- *limba* este limba prestabilită ce va fi folosită pentru acest utilizator când se conectează.

Pentru a adăuga un identificator pe server, se deschide SQL Server Management Studio (instrument de gestionare și de lucru din SQL Server) și după ce se realizează conectarea se execută următoarea comandă *Transact-SQL* (T-SQL):

Exemplul 2.2.1. *EXEC sp_addlogin 'numele', 'parola'*

Se creează un login *numele* cu parola *parola,* care la această etapă nu permite accesul la BD și necesită atribuirea privilegiilor și rolurilor.

Același rezultat îl obținem cu ajutorul instrucțiunii SQL:

CREATE LOGIN *identificator* WITH PASSWORD = *'parola'*

Pentru a putea schimba parola se folosește procedura de sistem stocată *sp_password*:

sp_password [[@old =] 'veche',
[[@new =] 'noua',
[,[@loginame =] 'identificator'

În sintaxa precedentă:

- *veche* este parola cea veche;
- *noua* este parola cea nouă;
- în ceea ce priveste *identificator,* dacă se face conectarea ca membru al rolului *sysadmin* sau *securityadmin*, parola poate fi schimbată oricui fără a cunoaște vechea parolă a identificatorului, ea poate declarată ca fiind NULL.

Exemplul 2.2.2. *EXEC sp_password 'parolaveche', 'parolanoua', 'numele'*

Membrii rolului *securityadmin* pot schimba parola oricui, mai puțin parolele membrilor rolului *sysadmin*.

Același rezultat îl obținem cu ajutorul instrucțiunii SQL:

ALTER LOGIN *numele*
WITH PASSWORD = *'parolanoua'*
[OLD_PASSWORD = *'parolaveche'*]

și se folosește numai pentru utlizatorul logat curent. În cazul în care dorim modificarea parolei pentru un alt utilizator al BD, nu este necesar de a executa secvența [OLD_PASSWORD = *'parolaveche'*].

Un lucru util ar fi schimbarea parolei în mod repetat. Microsoft SQL Server nu oferă nicio metodă de a impune restricții privind parola și alte măsuri de securitate. Acesta este unul dintre motivele pentru care se dorește implementarea securității integrate. În sistemul de operare Windows se poate specifica dimensiunea minimă a parolelor, frecvența schimbării și un minimum de reguli privind complexitatea parolelor.

Alte două proceduri de sistem stocate folosite pentru a gestiona identificatorii utilizatorilor sunt: ***sp_helplogins*** și ***sp_droplogins***.

Procedura stocată *sp_helplogins* oferă un raport cu identificatorii creați pe server. Procedura de sistem stocată *sp_droplogins* elimină un identificator din tabelul *sysxlogins*. După ștergerea identificatorului ***sp_droplogin*** *identificator* utilizatorul respectiv nu se poate conecta la SQL Server.

Același efect de ștergere a unui identificator îl obținem cu ajutorul instrucțiunii SQL:

DROP LOGIN numele

2.2.2. Windows Authentication Mode

O conexiune realizată prin autentificare Windows se numește *conexiune de încredere* (trusted connection). În Windows Authentication Mode, după ce se realizează conectarea prin rețea la SQL Server, trebuie să se prezinte atributele de securitate din Windows. Atributele respective se construiesc în cursul procesului de conectare la o rețea Windows. Acestea sunt verificate în mod automat, fără a se afișa un mesaj. Se poate acorda accesul la SQL Server direct prin conturile de securitate Windows, sau indirect prin grupurile Windows.

Cel mai mare avantaj al opțiunii Windows Authentication Mode este că utilizatorii nu trebuie să-și facă probleme cu conectarea separată la SQL Server. Acest lucru este în concordanță cu conceptul conform căruia utilizatorii trebuie să parcurgă procedura de conectare la rețea o singură dată și să rețină o singură parolă. În plus, se poate profita de faptul că majoritatea utilizatorilor rețelei au conturi de conectare la Windows, ceea ce reduce eforturile de administrare a conturilor pentru SQL Server.

Primul pas în configurarea securității Windows Authentication Mode nu are nicio legătura cu SQL Server, ci implică mai întâi crearea grupurilor Windows pentru utilizatori, crearea conturilor utilizatorilor (dacă nu există deja) și adăugarea lor în grupurile recent create, apoi trebuie să li se atribuie permisiune de conectare la SQL Server.

După definirea utilizatorilor și grupurilor, se acordă acces la SQL Server. Pentru aceasta, se folosesc procedurile de sistem stocate *sp_grantlogin*, *sp_revokelogin* și *sp_denylogin*. Inițial sunt acordate permisiuni de conectare a grupurilor Windows, apoi, conform necesităților, utilizatorilor individuali. Prin această metodă, numărul de comenzi necesare și eforturile de administrare se mențin la un minim, în condițiile unui control individual asupra privilegiilor de conectare.

Pentru a acorda permisiuni de conectare, se folosește procedura de sistem stocată *sp_grantlogin*:

sp_grantlogin [@loginame =] 'identificator'

În această sintaxă:

- *identificator* este numele grupului sau utilizatorului de Windows căruia îi este acordat dreptul să se conecteze la SQL Server.

În așa fel, orice utilizator de Windows care este membru al unui grup căruia i-a fost acordat accesul la SQL Server se poate conecta cu succes în

virtutea apartenenței la grupul respectiv. Dacă se folosește această modalitate de conectare, se va observa că, deși grupul este entitatea căreia i s-au acordat drepturi de conectare la SQL Server, în bara de stare este utilizatorul. Acest tip de conectare va oferi o pistă de audit și un control ulterior mult mai bun asupra permisiunilor.

Se poate ridica dreptul unui utilizator sau al unui grup Windows de a se conecta la SQL Server folosind procedura de sistem stocată *sp_revokelogin*:

sp_revokelogin [@loginame =] 'identificator'

În această procedură:

- *identificator* este numele grupului sau al utilizatorului de Windows al cărui drept de conectare la SQL Server trebuie ridicat;

- *sp_revokelogin* înlătură/anulează un drept acordat anterior.

Dacă se dorește interzicerea dreptului de conectare la SQL Server unui anumit utilizator sau unui grup, se folosește procedura de sistem stocată *sp_denylogin*:

sp_denylogin [@loginame =] 'identificator'

În această procedură:

- *identificator* este numele grupului sau al utilizatorului de Windows căruia îi este interzis dreptul de conectare la SQL Server.

Interzicerea drepturilor are întotdeauna prioritate în raport cu drepturile *acordate*.

2.2.3. Gestionarea utilizatorilor bazei de date

SQL Server conține două tipuri de identificatori:

- *identificatori Windows* pentru grupuri sau utilizatori individuali;

- *identificatori SQL Server* stocați în tabelul de sistem *sysxlogins* din BD master.

După ce se configurează securitatea identificatorilor și se definesc identificatorii, se poate trece la configurarea accesului la BD. Faptul că există un identificator pentru SQL Server nu oferă acces la toate BD de pe server. Pentru aceasta este nevoie de un nume de utilizator al BD.

2.2.3.1. Crearea unui utilizator

Fiecare BD are o cale de acces separată, amplasată în tabelul de sistem *sysusers* al BD respective. În esență, identificatorii sunt asociați cu un nume de utilizator în fiecare BD pe care trebuie să o acceseze utilizatorul.

Se poate realiza această asociere sau se poate crea un utilizator al BD folosind procedura de sistem stocată *sp_grantdbaccess:*

sp_grantdbaccess [@loginame =] 'identificator'
[, [@name_in_db =] 'nume_din_bd]'

În această sintaxă:

- *identificator* este identificatorul care a fost adăugat (utilizator SQL Server sau utilizator/grup Windows);

- *nume_din_bd* este numele ce poate fi utilizat de o persoană în timpul lucrului cu BD (numele de utilizator). Dacă nu este specificat, se va folosi identificatorul.

Exemplul 2.2.3. *EXEC sp_grantdbaccess 'numele', 'nume_bd'*

Același rezultat îl obținem cu ajutorul instrucțiunii SQL:

CREATE USER *nume_bd* FOR LOGIN *numele*

Se recomandă ca ori de câte ori este posibil, numele de utilizator să fie același cu identificatorul, pentru ca securitatea BD să fie mai usor de urmărit/reținut. Dacă numele rămâne același, confuzia dispare.

Dacă a fost adăugat un grup, fiecare membru al acestuia poate accesa BD, dar numai grupul are un nume de utilizator al BD: adică, nu există câte un nume pentru fiecare utilizator al grupului, ci numai pentru grupul în sine, la fel ca la identificatori.

2.2.3.2. Crearea unui alias

Se poate adăuga un *alias* al unui nume de utilizator al BD folosind procedura de sistem stocată **sp_addalias**. Cu ajutorul acestui alias un identificator poate înlocui un alt utilizator al BD, în loc să fie asociat cu propriul său nume de utilizator sau de grup.

sp_addalias *identificator, nume_utilizator*

În sintaxa comenzii sp_addalias:

- *identificator* este identificatorul de conectare al utilizatorului pe care doriți să-l asociați;

- *nume_utilizator* este utilizatorul BD în numele căruia se efectuează accesul.

Tehnica alias a fost creată pentru a fluidiza procesul de securizare a BD. Cu același nume de utilizator din BD pot fi asociați un număr nelimitat de identificatori. Dacă un identificator este deja asociat cu un nume de utilizator

din BD, el nu poate fi asociat cu un alt nume din aceeași BD. Rolurile oferă un model de securitate mult mai corect și mai eficient.

2.2.3.3. Ridicarea dreptului de acces la baza de date

Pentru a ridica dreptul de acces la BD, se va executa procedura de sistem stocată *sp_revokedbaccess:*

sp_revokedbaccess [@name in bd =] 'nume_din_bd']

În această procedură stocată:

- *nume_din_bd* este numele utilizatorului BD care trebuie înlăturat. Numai membrii rolurilor *db_accessadmin* și *db_owner* (sau ai rolului fix de server *sysadmin*) pot executa aceste proceduri stocate.

Exemplul 2.2.4. *EXEC sp_revokedbaccess nume_bd*

Același rezultat îl obținem cu ajutorul instrucțiunii SQL, pentru executarea căreia trebuie să fie activă baza de date *nume_bd*:

DROP USER [IF EXISTS] ***nume_bd***

Pentru a vedea ce utilizatori accesează o anumită BD și ce identificator are fiecare dintre ei, se poate executa procedura de sistem stocată *sp_helpuser*:

sp_helpuser [[@name in bd =] 'nume utilizator']

- *nume_utilizator* este opțional și reprezintă fie un nume de utilizator, fie un nume de rol.

Dacă nu se specifică un nume de utilizator, va fi generat un raport cu toți utilizatorii și cu toate rolurile; în caz contrar, se va obține un raport pentru utilizatorul sau rolul specificat.

Exemplul 2.2.5. *EXEC sp_helpuser 'nume_bd'*

- afișarea utilizatorilor din baza de date *'nume_bd'*.

Exemplul 2.2.6. *EXEC sp_helpuser*

- afișarea utilizatorilor din BD curentă:

	UserName	RoleName	LoginName	DefDBName	DefSchemaName	UserID	SID
1	dbo	db_owner	NULL	NULL	dbo	1	0x0105000000000005150000000026A908E27DEE9530023D7E...
2	guest	public	NULL	NULL	guest	2	0x00
3	INFORMATION_SCHEMA	public	NULL	NULL	NULL	3	NULL
4	nume_bd	public	numele	master	nume_bd	5	0x477C4F8F32E3074E82B060717F80734E
5	sys	public	NULL	NULL	NULL	4	NULL

Exemplul 2.2.7. *EXEC sp_helpuser 'db_securityadmin';*

- afișarea utilizatorilor care fac parte din rolul *db_securityadmin.*

Când este creată o BD, aceasta are deja un utilizator. Unul dintre utilizatori se numește **dbo** (de la database owner - proprietarul BD). Utilizatorul **dbo** este asociat în mod prestabilit cu identificatorul de conectare **sa**. Când se instalează SQL Server, identificatorul **sa** este considerat proprietarul tuturor BD. Dacă altcineva, cu un alt identificator, creează o BD, proprietarul acesteia va fi identificatorul respectiv.

În cadrul unei BD, *proprietarul ei poate face absolut orice*. Are aceeași libertate ca membrii rolului *sysadmin*. Însă numai membrii rolului fix de server *sysadmin* au anumite privilegii la nivel de sistem.

Dacă se dorește renunțarea la alias, se folosește procedura de sistem stocată *sp_dropalias*:

sp_dropalias *identificator*

- *identificator* este identificatorul utilizatorului.

2.2.3.4. Modificarea proprietarului bazei de date

Se poate modifica proprietarul unei BD atunci când se dorește ca responsabilitatea pentru aceasta să fie preluată de un anumit administrator al BD. Când se dorește efectuarea acestei modificări, identificatorul nu trebuie să existe în BD ca nume de utilizator.

Pentru a modifica proprietarul, se execută procedura de sistem stocată *sp_changedbowner*:

sp_changedbowner [@loginame =]'identificator'
[,[@map =] indicator_reasociere_ alias]

În această sintaxă:

- *identificator* este identificatorul SQL Server care va fi asociat;

- *indicator_reasociere_alias* este un parametru opțional care, dacă nu este specificat, determină ștergerea tuturor utilizatorilor asociați cu *dbo* când se modifică proprietarul BD. Dacă se specifică parametrul TRUE, toți utilizatorii asociați cu vechiul proprietar vor fi reasociați cu noul proprietar.

Proprietarul BD poate fi modificat și editând proprietățile BD în setările SQL Server Management Studio.

2.2.3.5. Utilizatorul guest

S-a precizat anterior că un identificator nu permite accesul la o BD dacă nu i-a fost atribuită în mod explicit permisiunea, dar există o exceptie – numele de utilizator **guest**.

Este un utilizator special, care nu respectă regulile obișnuite. Când o BD conține un cont de utilizator guest, orice persoană care solicită acces la BD și nu are un nume de utilizator în BD (un cont individual sau unul creat datorită apartenenței la un grup) poate obține accesul prin intermediul acestui cont. Prin urmare, în cursul procesului de instalare, în fiecare BD prestabilită de pe server se creează un cont *guest*. Numele de utilizator *guest* poate fi eliminat din toate BD existente, cu excepția BD *master*.

Dacă conexiunea se face cu un identificator pentru care nu este creat un nume de utilizator specific în BD și nu există nici contul *guest*, se obține un mesaj de eroare.

Mesajul de eroare precizează că identificatorul nu este asociat corect cu un utilizator din BD și implică faptul că nu există contul *guest*. Pentru a se remedia problema, fie (preferabil) se adaugă în BD un nume de utilizator corespunzator cu identificatorul, fie se adaugă contul *guest* executând comanda:

Exemplul 2.2.8. *EXCEC sp_grantdbaccess 'guest'*

Remarcă: dacă BD *master* conține un cont *guest*, fiecare BD nou-creată va avea un utilizator *guest*. Prin urmare, s-au examinat două modalități de a accesa o BD: definind un *nume de utilizator asociat cu un identificator* sau prin intermediul *contului guest*.

Modalități distincte de a accesa o BD după ce se realizează conectarea la SQL Server sunt:

1. ***sa*** – administratorul (sau orice alt membru al rolului de server *sysadmin*) poate să acceseze întotdeauna BD și să figureze ca *dbo*, chiar dacă proprietarul BD a fost schimbat cu procedura *sp_changedbowner*;

2. ***numele de utilizator al bazei de date*** – modul „normal” de a accesa o BD este de a folosi un nume de utilizator asociat cu un identificator SQL Server (un nume de utilizator sau un nume de grup Windows). Acest lucru este valabil și în cazul grupurilor Windows cărora le-a fost acordat drepturi de acces;

3. ***alias*** – se poate realiza asocierea cu un utilizator valid al BD în cadrul BD, în acest fel se obțin permisiunile și privilegiile utilizatorului respectiv;

4. ***guest*** – dacă toate celelalte metode dau greș, SQL Server verifică dacă în tabelul de sistem *sysusers* al BD există un cont guest. În caz afirmativ, se permite accesul ca guest, altfel accesul este interzis.

2.3. Acordarea și revocarea privilegiilor

Privilegiile reprezintă dreptul de a executa o anumită instrucțiune SQL. Administratorul BD este un utilizator de nivel înalt care are abilitatea de a acorda utilizatorilor dreptul de a accesa BD și obiectele acesteia.

Privilegiile pot include dreptul:

- de a se conecta la BD;
- de a crea și interoga tabele;
- de a executa procedurile stocate, views etc.

Utilizatorii au nevoie de *privilegii* pentru a manipula conținutul obiectelor din BD, ceea ce le permite să execute o anumită acțiune asupra unui obiect specific (privilegiul de a șterge rânduri dintr-un anumit tabel, de a adăuga sau modifica o înregistrare).

Fiecare obiect are un anumit set de privilegii care îi pot fi atașate. Se poate observa că privilegiile variază de la un obiect la altul.

Privilegii obiect	***Tabel***	***Vedere***	***Procedura***
ALTER	*****		
DELETE	*****	*****	
EXECUTE			*****
INSERT	*****	*****	
SELECT	*****	*****	
UPDATE	*****	*****	

2.3.1. Acordarea privilegiilor

Privilegiile sunt acordate utilizatorilor astfel încât aceștia pot accesa și modifica date în BD. Unui utilizator i se pot acorda privilegii în două moduri:

- explicit, direct utilizatorului;
- atribuite unui rol și apoi acesta poate fi asociat unuia sau mai multor utilizatori.

GRANT privilegiu [privilegiu,...] ON
[OBJECT ::] object_name [,...n])]
TO USER [user / rol, PUBLIC...];

Exemplul 2.3.1. *GRANT SELECT, INSERT, UPDATE, DELETE ON Student TO nume_bd*

Userul *nume_bd* poate face selecții, înserări, modificări și ștergeri în BD *univer*.

Exemplul 2.3.2. *GRANT SELECT, INSERT ON Facultate TO nume_bd*

Userul *nume_bd* poate face selecții, înserări în BD *univer*. Deci, putem îndeplini:

INSERT INTO Facultate(name) values('Informatica')

*SELECT * FROM Facultate*

Așa operație ca *UPDATE Facultate set name = 'test'* nu este permisă.

2.3.2. Revocarea privilegiilor

Privilegiile acordate utilizatorilor pot fi înlăturate/revocare prin utilizarea declarației REVOKE:

REVOKE <privilegiu> [,...n] ON
[OBJECT ::] object_name [,...n])]
{ FROM | TO } <database_principal> [,...n]

- *<privilegiu> [,...n]* unul sau mai multe privilegii;
- *[OBJECT ::] object_name [,...n])]* unul sau mai multe obiecte asupra cărora sunt înlăturate privilegiile;
- *<database_principal> [,...n]* unul sau mai mulți useri cărora li se înlătură privilegiul acordat.

Exemplul 2.3.3. *REVOKE UPDATE ON Student TO nume_bd*

Ceea ce semnifică faptul că va fi înlăturată permisiunea de modificare a tabelului *Student*.

2.4. Tipuri de roluri. Crearea, asignarea și revocarea rolurilor

Rolul unui utilizator este o colecție de privilegii care pot fi create și alocate unor utilizatori. Rolurile reprezintă elementul de legatură între toate aspectele sistemului SQL Server. Rolurile pot fi asemănate cu grupurile SQL Server. Rolurile SQL Server permit gruparea numelor utilizatorilor BD. Nu are importanță dacă numele de utilizatori reprezintă grupuri Windows, utilizatori Windows sau identificatori SQL Server. Rolurile pot chiar conține alte roluri ca membri.

2.4.1. Rolul public

În fiecare BD există un rol predefinit numit *public*. Toți utilizatorii, toate grupurile și toate rolurile sunt membri ai rolului public și nu pot fi înlăturați. Este comod de a efectua referiri la toți utilizatorii, fără a-i denumi în mod explicit.

2.4.2. Roluri la nivel de server

Identificatorul **sa** este atotputernic, poate face orice cu o instanță SQL Server. Acest lucru se datorează faptului că **sa** este membru al unui rol la nivel de server, denumit **sysadmin**. SQL Server are opt roluri la nivel de server. În orice moment, un identificator poate fi făcut membru al unuia sau mai multor roluri, însă nu se pot înlătura sau adăuga roluri la nivel de server.

Lista ce descrie setul complet de roluri la nivel de server disponibile în SQL Server este:

1. **sysadmin** – membrii acestui rol pot face orice operație în SQL Server și figurează drept proprietari (dbo) ai BD (chiar atunci când nu sunt). Pot trece peste toate permisiunile și sistemele de securitate;
2. **serveradmin** – membrii acestui rol pot stabili opțiuni de configurare cu procedura de sistem stocată *sp_configure* și pot închide serverul. De asemenea, pot configura serviciul Full-Text. Operatorii serverului sunt niște candidați foarte potriviți pentru acest rol;
3. **setupadmin** – membrii acestui rol pot instala și configura servere legate și pot marca o procedură stocată pentru execuție la momentul pornirii;
4. **securityadmin** – membrii acestui rol pot crea și controla identificatori de conectare la server, precum și permisiuni pentru crearea BD. Pot configura atributele de securitate ale serverelor legate și pot redefini parolele identificatorilor SQL Server (cu excepția membrilor rolului *sysadmin*). Cel mai probabil, membrii acestui rol vor fi operatorii serverului și personalul de asistență;
5. **processadmin** – membrii acestui rol pot controla procese executate pe serverul de BD. Aceste procese implică „anihilarea” interogărilor cu probleme, iar personalul de asistență poate avea nevoie de acest drept;
6. **dbcreator** – membrii acestui rol pot crea, modifica și elimina BD de pe server. De asemenea, pot restabili BD pe server. Administratorii BD pot fi membri ai acestui rol;
7. **bulkadmin** – membrii acestui rol pot executa instrucțiunea BULK INSERT (posibilitatea încărcării datelor direct din fișiere în tabele);
8. **diskadmin** – membrii acestui rol pot gestiona și crea fișierele de pe server.

2.4.3. Roluri la nivelul bazelor de date

Fiecare BD conține roluri. Unele dintre acestea sunt fixe; de asemenea, se pot adăuga propriile roluri (ceea ce nu este posibil la nivel de server).

De reținut că aceste roluri sunt specifice BD, așa că un rol nu poate influența mai multe BD simultan, însă se pot crea aceleași roluri în toate BD.

Fiecare BD conține un ansamblu de roluri predefinite, cărora le poate fi atribuit un nume de utilizator. Există 9 asemenea roluri – un numar fixat pentru totdeauna (nu se poate șterge niciunul dintre ele). La fel ca rolurile la nivel de server, fiecare rol la nivel de BD acordă utilizatorilor anumite permisiuni și capabilități, cum ar fi:

1. **db_owner** – membrii acestui rol pot efectua orice operație, dar numai în cadrul propriei BD. Un utilizator membru al rolului *db_owner* are aproape toate drepturile și permisiunile utilizatorului *dbo* (proprietarul BD). Excepția se referă la comanda *restore*;
2. **db_accessadmin** – membrii acestui rol pot acorda sau anula dreptul de acces al utilizatorilor la BD (ex., prin executarea procedurii de sistem stocate *sp_grantdbaccess*);
3. **db_securityadmin** – membrii acestui rol pot controla toate permisiunile, rolurile, apartenența la roluri și proprietarii obiectelor din BD;
4. **db_ddladmin** – membrii acestui rol pot crea, modifica și înlătura toate obiectele din BD, dar nu pot executa comenzi legate de securitate (cum ar fi *grant, revoke, deny*);
5. **db_backupoperator** – membrii acestui rol pot executa comenzile *checkpoint* și *backup*;
6. **db_datareader** – membrii acestui rol au permisiunea de a selecta date din orice tabel, vedere sau funcție din BD;
7. **db_datawriter** – membrii acestui rol au drepturi de introducere, actualizare sau ștergere pentru orice tabel sau vedere din BD;
8. **db_denydatareader** – membrii acestui rol nu pot selecta date din tabele, vederi sau funcții din BD;
9. **db_denydatawriter** – membrii acestui rol nu pot modifica niciun fel de informații cu comenzile *insert*, *update* sau *delete*, din niciun tabel sau vedere a BD.

2.4.4. Roluri definite de utilizator la nivelul bazelor de date

Pe lângă rolurile predefinite, se pot crea roluri care pot fi atribuite utilizatorilor sau altor roluri. Motivul pentru care se creează roluri la nivelul BD în SQL Server este ca și motivul pentru care se creează grupuri la nivelul sistemului de operare – pentru a-i grupa în mod convenabil pe utilizatorii care dețin funcții similare. Numărul rolurilor create trebuie să corespundă

necesităților. Nu există niciun fel de restricții privind numărul de roluri cărora le poate aparține un utilizator sau un alt rol.

2.4.4.1. Crearea rolurilor

Pentru a crea un rol, se folosește procedura de sistem stocată *sp_addrole*:

sp_addrole [@rolename =] 'rol' [,[@ownername =] 'proprietar']

În această sintaxă:

- *rol* este numele care va fi atribuit noului rol;
- *proprietar* este numele utilizatorului SQL Server care va fi proprietarul rolului (fiecare utilizator poate avea propriile sale roluri).

Exemplul 2.4.1. *EXEC sp_addrole 'rol_student'*

Același rezultat îl obținem cu ajutorul instrucțiunii SQL, pentru executarea căreia trebuie să fie activă baza de date *nume_bd*:

CREATE ROLE *rol [* AUTHORIZATION *proprietar]*

Numai membrii rolului *sysadmin* la nivel de server sau ai rolurilor *db_owner* și *db_securiry* la nivel de BD pot adăuga un rol nou la o BD. Același lucru este valabil și cu privire la înlăturarea rolurilor. Numele rolului trebuie să fie unic în BD respectivă.

2.4.4.2. Revocarea rolurilor

Pentru a înlătura un rol dintr-o BD, se execută procedura de sistem stocată *sp_droprole*:

sp_droprole [@rolename =] 'rol'

- *rol este* numele rolului creat de utilizator ce trebuie a fi înlăturat.

Exemplul 2.4.2. *EXEC sp_droprole 'rol_student'*

Nu se poate șterge un rol dacă acesta are drept membri utilizatori sau alte roluri. De asemenea, nu se pot șterge roluri dacă acestea dețin obiecte.

Același rezultat îl obținem cu ajutorul instrucțiunii SQL, pentru executarea căreia trebuie să fie activă baza de date *nume_bd*:

DROP ROLE *rol [* AUTHORIZATION *proprietar]*

2.4.4.3. Asignarea rolurilor

Pentru a adăuga utilizatori la rolurile fixe sau definite de utilizator, se folosește procedura de sistem stocată *sp_addrolemember*:

sp_addrolemember [@rolename] 'rol' , [@membername =] 'cont_baza_date'

În această sintaxă:

- *rol* este numele rolului la care se adaugă un utilizator;

- *cont_baza_date* este numele utilizatorului, grupului sau rolului ce poate fi adăugat la rolul respectiv.

Exemplul 2.4.3. *EXEC sp_addrolemember 'rol_student', 'nume_bd'*

Ca rezultat este permisă acordarea privilegiilor acestui rol astfel:

GRANT SELECT, UPDATE, INSERT, DELETE ON Examen TO rol_student

Același rezultat îl obținem cu instrucțiunea SQL, pentru executarea căreia trebuie să fie activă baza de date *nume_bd*:

ALTER ROLE *rol* ADD MEMBER *cont_baza_date*

2.4.4.4. Ștergerea rolurilor

Pentru a elimina un membru al unui rol, se execută procedura de sistem stocată *sp_droprolemember*:

sp_droprolemember [@rolename =] 'rol',[@membername=] 'cont_baza_date'

În această sintaxă:

- *rol* este numele rolului din care este eliminat utilizatorul;

- *cont_baza_date* este numele utilizatorului, grupului sau rolului care este eliminat din rolul respectiv.

Exemplul 2.4.5. *EXEC sp_droprolemember 'rol_student', 'nume_bd'*

Poate fi obținut același rezultat și cu instrucțiunea SQL, pentru executarea căreia trebuie să fie activă baza de date *nume_bd*:

ALTER ROLE *rol* DROP MEMBER *cont_baza_date*

Membrii fiecarui rol sunt memorați într-o combinație a tabelelor de sistem *sysusers* și *sysmembers*. Se pot examina rolurile existente cu ajutorul procedurilor de sistem stocate *sp_helprole* sau *sp_helprolemember*. Ambele proceduri primesc un singur parametru: numele rolului între apostrofuri.

2.4.5. Roluri la nivel de aplicație

La nivel de aplicație rolurile constituie o caracteristică extrem de utilă a sistemului SQL Server. Deși la prima vedere par similare cu celelalte roluri prezentate, ele îndeplinesc o altă funcție; ele diferă prin faptul că pot fi

„activate” de o aplicație. După ce o aplicație a activat un rol la nivel de aplicație, toate permisiunile utilizatorului sunt suspendate, fiind impuse numai permisiunile rolului respectiv. Pentru activarea rolului este nevoie de o parolă.

Pentru a crea un rol la nivel de aplicație, se folosește procedura de sistem stocată *sp_addapprole*:

sp_addapprole [@rolename =] 'rol', [@password =]...'parola'

În această sintaxă:

- *rol* este numele rolului care trebuie creat;
- *parola* este parola pe care trebuie să o prezinte aplicația pentru a activa rolul.

Pentru a elimina un rol la nivel de aplicație, se execută procedura de sistem stocată *sp_dropapprole*:

sp_dropapprole [@rolename =] 'rol'

- *rol* este numele rolului care trebuie îndepărtat.

Pentru a utiliza rolul la nivel de aplicație, este executată procedura *sp_setapprole*:

sp_setapprole | 'parola' [,[@encrypt =] 'stil_criptare']

În această sintaxă:

- *rol* este numele rolului care va fi activat;
- *parola* este parola specificată în execuția procedurii *sp_addapprole*;
- *encrypt N 'parola'* cere ca parola să fie criptată când este transmisă prin rețea (dacă specificați numai parola, aceasta este transmisă prin rețea necriptată);
- *stil_criptare* specifică stilul de criptare utilizat: *none* (niciunul) sau *odbc*. Se specifică *odbc* (Open Database Connectivity) când se folosește un client bazat pe ODBC, pentru criptarea parolei se va folosi funcția de criptare standard ODBC.

2.5. Implementarea politicii de parole

Un SGBD permite unui utilizator accesarea BD, fiind utilizată o parolă. Parolele ar trebui ținute secrete și schimbate frecvent. Sistemul nu trebuie niciodată să afișeze parolele la conectare, iar profilurile stocate trebuie să fie păstrate în siguranță în formă criptată. Parolele SQL Server Authentication

Mode sunt amplasate în coloana de parole a tabelului *sysxlogins* din baza de date master.

Aspectele de care trebuie să ținem cont sunt următoarele:

✓ Coloana de parole poate fi vizualizată numai de un membru al rolului *sysadmin* (inclusiv identificatorul său).

✓ Parolele nu se păstrează în formă explicită în baza de date, în locul lor se va plasa rezultatul aplicării unui algoritm de hash criptografic. Acești algoritmi funcționează doar într-o singură direcție: putem genera hash-ul oricărui conținut, procesul nefiind unul inversabil, aidoma unui algoritm de criptare. În momentul conectării, parola furnizată este trecută prin același algoritm hash, apoi comparată cu hash-ul parolei din tabelul *sysxlogins*. Dacă cele doua valori sunt identice, se permite accesul la server; în caz contrar, se obține mesajul de eroare *Login Failed* și nu se poate face conectarea.

✓ Dacă este folosit un identificator de securitate Windows Integrated, în SQL Server nu se stochează niciun fel de parole.

Deoarece parolele sunt nişte chei de acces la valorile proprii, este necesar de a fi protejate cu grijă respectatând câteva reguli de bază la alegerea lor:

✓ parola trebuie să fie complexă: să conțină cifre, litere mari și mici, caractere speciale și o lungime de 12 - 30 simboluri;

✓ schimbarea parolelelor implicite ale utilizatorilor și administratorilor;

✓ parolele trebuie schimbate cel puţin o dată la şase luni, iar pentru datele deosebit de importante şi mai des;

✓ parolele comune (utilizate de mai mulţi) trebuie schimbate imediat ce o persoană părăseşte grupul sau i se retrage dreptul utilizării ei;

✓ parolele se vor schimba imediat ce apare bănuiala cunoaşterii lor de către persoane neautorizate sau dacă din motive de forţă majoră secretul lor a trebuit dezvăluit pentru redresarea unei stări anormale temporare;

✓ parolele trebuie să fie memorate;

✓ dacă parolele duplicat se păstrează prin intermediul calculatorului, astfel de fişiere trebuie să fie protejate împotriva accesului neautorizat şi create copii de siguranţă. Accesul la acest fişier trebuie să fie atribuit doar persoanelor autorizate, respectându-se principiul „niciodată singur”. Listele cu parole vor fi memorate sub formă criptată;

✓ parolele nu vor fi afişate niciodată pe echipamentele din configuraţia sistemului, iar la introducerea lor nu trebuie să se afle persoane străine în preajmă;

✓ pentru blocarea operaţiunilor de încercare repetată, de ordinul miilor, calculatoarele trebuie să permită un număr limitat de încercări de introducere a parolelor, de obicei trei. Dacă limita este depăşită de utilizator, intenţia este raportată conducătorului sistemului sau responsabilului de securitate. În acest timp trebuie să se blocheze terminalul de la care s-au efectuat prea multe încercări nereuşite. În cazul sistemelor speciale se recomandă şi blocarea sălii sau a locului de unde s-a încercat pătrunderea în sistem prin parole eronate repetate, pentru identificarea persoanei respective;

✓ odată pătruns în sistem, utilizatorului nu trebuie să i se permită să-şi schimbe identitatea cu care a deschis sesiunea şi nici să poată pătrunde în partiţiile alocate altor utilizatori;

✓ dacă un terminal funcţionează o perioadă lungă de timp, procesul de autentificare trebuie să aibă loc la intervale regulate de timp pentru a se asigura că nu foloseşte altcineva sistemul. Dacă terminalul rămâne neutilizat dar deschis, el trebuie să se închidă automat după un anumit interval de timp (de exemplu, după 10 minute);

✓ la deschiderea unei noi sesiuni de lucru, utilizatorului trebuie să i se aducă la cunoştinţă ultimul timp de accesare a sistemului cu parola respectivă, pentru a verifica dacă altcineva a folosit-o între timp.

Surse bibliografice:

1. B. Thuraisingham. *Database and Applications Security: Integrating Information Security and Data Management.* Auerbach Publications, 2005.
2. Microsoft SQL Docs, Permissions (Database Engine), 2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/ permissions- database-engine?view=sql-server-ver15 [Accesat: 28.12.2022]

III. SECURITATEA INTERNĂ A DATELOR

O problemă critică în gestionarea BD este asigurarea respectării unui set predefinit de reguli de către datele BD. Aceste reguli trebuie să fie în concordanță cu activitățile comerciale pe care le susține SGBD-ul. Administratorul BD și programatorul aplicației (dar și managerul întreprinderii) sunt responsabili pentru definirea și crearea acestor reguli.

Integritatea datelor presupune definirea regulilor care restrâng valorile valide pentru o coloană a unui tabel. Restricțiile de integritate sunt definite pentru tabele și, prin urmare, toate vederile sunt supuse restricțiilor de integritate. Dacă o instrucțiune DML (Data Manipulation Language) încearcă să efectueze o acțiune care violează o restricție de integritate, este generată o eroare și tranzacția este derulată înapoi.

3.1. Constrângeri de integritate

Pentru garantarea integrității datelor conținute într-o BD, trebuie asigurată:

a) ***integritatea domeniului***, presupune că fiecare valoare individuală este în conformitate cu o regulă specifică.

Pot fi folosite diferite modalități de a valida data introdusă în BD:

- *coloana pentru date este restricționată de valorile care pot fi introduse în această coloană* (aceasta va împiedica introducerea descrierii textuale în coloana de date și a datelor în coloana de note);
- *poate fi restricționată lungimea maximă și lungimea minimă pentru fiecare valoare dată* (de exemplu, se poate determina dacă codul studentului ar trebui să conțină cel puțin cinci caractere lungime și nu mai mult de zece caractere);
- *pot fi restricționate datele care se potrivesc unui anumit format* (acest lucru poate fi foarte folositor pentru a valida codurile poștale sau chiar numerele de telefon);
- *poate fi folosită restricționarea ariei valide de valori* (se poate limita valoarea care este introdusă, de exemplu: ziua de naștere a unui student, până la date cuprinse între 1990-01-01 și ziua curentă), pentru a evita posibilele greșeli;
- *semnificația după o coloană* (de exemplu unui profesor i se pot atribui unul dintre posibilele grade științifice: doctor, doctor habilitat, academician).

b) ***integritatea entității***, presupune că fiecare obiect poate fi identificat în mod unic și fără echivoc.

Orice obiect real ar trebui să fie ușor de identificat într-o BD. Este dificil să ne referim la un client astfel: clientul care locuiește în Chișinău, are 3 copii și 100 de angajați, are vârsta de 40 de ani, primul nume este Ion și numărul său de telefon se termină în 345. Pentru oameni această informație poate fi suficientă pentru căutarea în memoria noastră și identificarea clientului. Cu toate acestea, pentru un program de calculator, cum ar fi SQL Server, această metodă de identificare îl va forța să aplice diferite condiții în secvență, o condiție pentru fiecare atribut. Probabil că ar fi fost mai ușor să identifice fiecare client după o singură valoare unică, de exemplu 256634 stocată într-o coloană de identificare CustomerID. În acest fel, pentru căutarea unui client Serverul SQL va aplica o condiție foarte simplă – CustomerID = 256634.

Importanța acestui lucru este evidentă când se efectuează o legătură între informație și alte entități, deoarece fiecare legătură ar trebui bazată pe cea mai simplă relație.

c) ***integritatea relațională*** presupune că data relatată este corect conectată.

BD relaționale sunt denumite 'relaționale' deoarece unitățile de date stocate în acestea sunt legate între ele prin relații, ca de exemplu:

- clienții au reprezentanți de vânzări care îi servesc;
- clienții înaintează comenzi;
- comenzile au o anumită ordine;
- fiecare obiect într-o ordine face referire la un singur produs;
- produsele sunt organizate pe categorii;
- produsele provin de la producători.

Trebuie de asigurat că toate legăturile sunt bine stabilite și că nu vom avea date fără legături cu restul datelor. În unele situații aplicarea regulilor de integritate complexe sunt imposibil de realizat folosind structuri relaționale standard și în aceste situații vor fi create proceduri stocate, triggere, funcții definite de către utilizator sau componente externe.

Restricția de integritate pentru un tabel este stabilită atunci când tabelul este creat sau modificat, iar pentru a stabili o restricție de integritate vom include clauza *constraint* în comenzile *CREATE TABLE* sau *ALTER TABLE* după cum urmează:

CONSTRAINT nume_restrictie tip_restrictie

Există următoarele tipuri de restricții de integritate a datelor care sunt recomandate a fi aplicate în momentul proiectării BD, dar în cazul

implementării se va ține cont și de așa factori ca performanță (timp de execuție, volum de date, domeniu de aplicare) și scalabilitate (posibilitatea extinderii BD):

1. ***NULL***, în mod prestabilit, toate coloanele unui tabel accepta valori nule, iar stabilind restricția de integritate *NOT NULL* se impune coloanei specificate să posede o valoare pentru fiecare linie a tabelului.

Exemplul 3.1.1.

CREATE TABLE test(
id varchar(8) NOT NULL,
name nvarchar(50) NOT NULL)

2. ***UNIQUE*** impune restricția ca toate valorile dintr-o coloană să fie distincte, iar coloana definită de restricția *unique* este coloană cu cheie unică. Într-un tabel nu trebuie să existe mai multe tupluri cu aceeași valoare în calitate de cheie.

Exemplul 3.1.2.

CREATE TABLE test(
id varchar(8) NOT NULL UNIQUE,
name nvarchar(50) NOT NULL)

Exemplul 3.1.3.

CREATE TABLE test(
id varchar(8) NOT NULL ,
name nvarchar(50) NOT NULL,
CONSTRAINT c1 UNIQUE (id))

Rezultatul generat de exemplul 3.1.3 are același efect ca și cel din exemplul 3.1.2. În acest caz este corectă inserarea următoarelor date:

INSERT INTO test values(1,'Elena')
INSERT INTO test values(2,'Ion')
dar nu se poate efectua:
INSERT INTO test values(1,'Elena')
INSERT INTO test values(1,'Ion')

Exemplul 3.1.4.

CREATE TABLE test(
id varchar(8) NOT NULL ,
name nvarchar(50) NOT NULL,
CONSTRAINT c1 UNIQUE (id,name))
Pentru acest caz este corectă inserarea datelor:

INSERT INTO test values(1,'Elena')
INSERT INTO test values(1,'Ion')
Se va afișa o excepție de validare dacă se va adăuga:
INSERT INTO test values(1,'Elena')

Deoarece toate valorile nule sunt considerate unice și distincte din punctul de vedere al acestei restricții, se recomandă plasarea restricției *NOT NULL* pe coloanele cu valori *UNIQUE*; în acest fel este forțată introducerea unei valori pentru coloana respectivă și este asigurat faptul că valorile sunt distincte.

3. ***Primary key*** desemnează o coloană sau o combinație de coloane drept cheie principală a tabelului care identifică în mod univoc fiecare linie a tabelului și creează în mod implicit restricțiile *NOT NULL* și *UNIQUE* pe coloana respectivă. Deși nu se impune existența unei chei principale pentru fiecare tabel, este bine să fie creată una ale cărei valori nu se modifică niciodată.

Exemplul 3.1.5.

```
CREATE TABLE test(
id  varchar(8) NOT NULL PRIMARY KEY,
name  nvarchar(50) NOT NULL )
```

sau

```
CREATE TABLE test(
id  varchar(8) NOT NULL,
name  nvarchar(50) NOT NULL,
CONSTRAINT pk_test PRIMARY KEY (id) )
```

Putem adăuga și două câmpuri ca chei primare astfel:

CONSTRAINT pk_test PRIMARY KEY (id, name)

Nu poate fi aceeași coloană sau combinație de coloane cheie principală și cheie.

4. Restricția de integritate ***prin referință*** impune regulile comerciale care dictează relațiile dintre coloanele diverselor tabele, iar între tabele se creează o relație de tip părinte-fiu. Tabelul părinte conține cheia referință, iar tabelul fiu conține cheia externă.

Definiția 3.1. *Cheie externă* numim coloana sau setul de coloane incluse în definiția restricției de integritate prin referință.

Definiția 3.2. *Cheie referită* numim cheia unică sau cheia primară a aceluiași tabel sau a unui alt tabel la care face referire cheia externă.

Definiția 3.3. *Tabel dependent* numim tabelul care conține cheia externă.

Definiția 3.4. *Tabel referit* numim tabelul care este referit de cheia externă.

Exemplul 3.1.6. Fie un tabel *Curs* care conține câmpul *id* care este *PRIMARY KEY* pentru acest tabel.

```
CREATE TABLE Curs(
id varchar(8) NOT NULL,
name nvarchar(50) NOT NULL,
id_facult varchar(8) NOT NULL,
nr_credite tinyint NOT NULL,
nr_ore tinyint NOT NULL,
CONSTRAINT pk_Curs PRIMARY KEY(id)  )
```

Referință la acest câmp din tabelul *test* în care adăugam *id_curs* ca *FOREIGN KEY* (cheie externă) se face în felul următor:

```
CREATE TABLE test(
id varchar(8) NOT NULL,
name nvarchar(50) NOT NULL,
id_curs int NOT NULL,
CONSTRAINT pk_test PRIMARY KEY (id),
CONSTRAINT fk_test FOREIGN KEY(id_curs) REFERENCES Curs
(id)
```

5. *CHECK* definește în mod explicit o condiție care trebuie să fie adevărată și va fi evaluată și la valoarea *Unknown* pentru a permite prezența valorii *NULL,* ceea ce presupune că fiecare înregistrare trebuie să respecte această condiție. Pentru a satisface această restricție, fiecare linie a tabelului trebuie să respecte condiția. În cazul când se încearcă executarea unei instrucțiuni care ar duce la nerespectarea condiției, instrucțiunea este derulată înapoi. Restricția *check* nu poate să conțină subinterogări.

Exemplul 3.1.7.

CONSTRAINT CK_Student_an CHECK ([an_nastere] between '1950-01-01' and GETDATE())

Exemplul 3.1.8.

CONSTRAINT CK_Student_sex CHECK ([sex] in ('F','M'))

3.2. Vederi

Vederile sunt tabele logice asemănătoare unor ferestre prin care pot fi privite una sau mai multe tabele sau vederi. O vedere are aspectul unui tabel conținând coloane și linii care pot fi actualizate și în care se pot efectua inserări sau eliminări ca și cum ar fi cu un tabel adevărat (doar dacă vederea se referă numai la un singur tabel). O vedere este, de fapt, un tabel logic care nu stochează date. Ea își preia datele din tabelele sau vederile pe care se bazează. Toate operațiile efectuate asupra unei vederi afectează practic tabelele de bază ale vederii atâta timp cât satisface condițiile de integritate. O vedere este creată folosind o interogare și, prin urmare, poate fi privită ca fiind un tabel virtual sau o interogare stocată, întrucât vederile arată și se comportă ca niște tabele, ele pot fi utilizate în majoritatea situațiilor pe post de tabele.

Vederile sunt dinamice și afișează întotdeauna informațiile curente ale tabelelor. Atunci când tabelele vederii sunt manipulate (când se face o schimbare în tabel aceasta se reflectă și în vedere), aceste modificări sunt reflectate instantaneu în vedere. Vederile permit afișarea datelor într-o formă diferită de cea în care sunt stocate în tabele și permit adaptarea prezentării datelor în conformitate cu cerințele specifice ale diverselor tipuri de utilizatori.

Vederile sunt create în următoarele scopuri:

- Pentru a asigura un nivel mai înalt de securitate a BD prin restrângerea accesului la un număr predeterminat de coloane și linii ale unui tabel. Acest lucru permite utilizatorilor să vizualizeze un subset restrâns al datelor.

- Pentru a îmbunătăți securitatea BD prin mascarea adevăratelor nume și locații ale tabelelor de bază (datele pot fi din diferite tabele, din diferite BD, cu denumirea coloanelor modificată, cu coloane ce conțin date ca rezultat al unor calcule, conținutul modificat (de exemplu, combinarea datelor ca nume și prenume ce se aflau în două coloane, care vor fi prezentate ca o coloană)).

- Pentru simplificarea prezentării datelor prin ascunderea structurilor reuniunilor și tabelelor care stau la baza vederii (cazul când avem nevoie de vizualizat date care fiind conținute în diferite tabele ar necesita, de exemplu, realizarea unui *SELECT* sau *UNION* complex care trebuie executat de fiecare dată).

- Pentru afișarea datelor într-o altă reprezentare decât cea a tabelelor de bază.

- Pentru a simplifica efortul de programare prin posibilitatea de a interoga un tabel virtual în locul interogării fiecăruia dintre tabelele de bază.

3.2.1. Crearea vederilor

Pentru a crea o vedere se poate utiliza comanda SQL ***CREATE VIEW***. O vedere se poate defini cu orice interogare care face referire la tabele sau alte vederi.

Exemplul 3.2.1.

CREATE VIEW vtest

AS SELECT name, surname, an_studii FROM Student

În acest exemplu codul permite crearea unei vederi simple, cu date *name, surname, an_studii* dintr-un singur tabel.

Exemplul 3.2.2.

CREATE VIEW vtest

AS SELECT name, surname, an_studii FROM Student WHERE an_studii = 3

Codul din acest exemplu permite crearea unei vederi simple, cu date *name, surname, an_studii* dintr-un singur tabel cu condiția că *an_studii=3*.

Exemplul 3.2.3.

CREATE VIEW vtest1

AS SELECT s.name, s.surname, s.an_studii, c.name AS curs FROM Student s, Curs_Student cs, Curs c

WHERE s.id =cs.id_student AND cs.id_curs=c.id AND s.an_studii = 3

În acest caz, codul permite crearea unei vederi complexe *vtest1* care face referire numai la acele înregistrări ce conțin *name, surname, an_studii* și *curs* pentru care *an_studii = 3*.

Exemplul 3.2.4.

CREATE VIEW stud_fara_restante AS

SELECT s.name, s.surname

FROM Student s

INNER JOIN (SELECT id_stud FROM Examen GROUP BY id_stud

HAVING MIN(nota) > =5) e ON s.id = e.id_stud

În acest exemplu codul permite crearea unei vederi mai complexe *stud_fara_restante* care face referire numai la înregistrările ce conțin *name* și *surname* cu *nota* minimă la examene − 5.

3.2.2. Modificarea vederilor

Pentru a modifica definiția unei vederi, vederea poate fi înlocuită în două moduri:

1. Vederea poate fi distrusă și apoi recreată cu noua definiție. Atunci când vederea este distrusă, toate privilegiile sunt retrase și trebuie să fie recreate pentru noua vedere.

2. Vederea poate fi recreată prin redefinirea ei cu instrucțiunea *CREATE VIEW* cu clauza opțională *or REPLACE*. Această metodă este folosită pentru înlocuirea definiției curente a unei vederi cu conservarea tuturor privilegiilor curente.

Exemplul 3.2.5.

CREATE or REPLACE VIEW vtest AS SELECT name, an_studii FROM Student

Înlocuirea vederilor are următoarele efecte:

- definiția vederii este actualizată și niciunul dintre obiectele de bază nu este afectat de înlocuirea vederii;
- sunt distruse toate restricțiile care existau în vederea originală dar nu se regăsesc în noua vedere.

Utilizarea vederilor face obiectul următoarelor limitări:

- nu poate fi folosită o vedere pentru a efectua operații *INSERT, UPDATE* sau *DELETE* atunci când interogarea vederii conține o operație *JOIN*, operatorii *SET* sau *DISTINCT*, o clauză *GROUP BY*;
- nu poate fi utilizată o vedere definită cu clauza *WITH CHECK OPTION* pentru a insera sau a actualiza tabelele de bază;
- nu pot fi inserate înregistrări în tabelul de bază dacă o coloană *NOT NULL* a liniei respective este definită fără clauza *DEFAULT VALUE;*
- utilizarea vederilor necesită un timp suplimentar de procesare înainte ca datele să fie citite, reducând puțin viteza de răspuns a serverului de BD.

3.2.3. Recompilarea vederilor

Pentru a recompila în mod explicit o vedere se va utiliza comanda:

ALTER VIEW nume_vedere COMPILE

Recompilarea permite detectarea eventualelor erori referitoare la vederea respectivă înainte de a fi executată vederea în mediul real al întreprinderii. Este recomandabilă recompilarea în mod explicit a vederii după orice modificare a structurii oricăruia dintre tabelele de bază ale vederii.

Comanda *ALTER VIEW* nu modifică definiția vederii sau orice alt obiect care ar putea să depindă de ea.

Exemplul 3.2.6.
ALTER VIEW vtest COMPILE;
Instrucțiunea din acest exemplu va recompila vederea *vtest*.

3.2.4. Distrugerea vederilor

Pentru a distruge o vedere se utilizează comanda ***DROP VIEW***.

Exemplul 3.2.7.
DROP VIEW vtest;
Codul din acest exemplu permite distrugerea vederii *vtest*.
Putem formula următoarele concluzii:
Avantajele utilizării vederilor:
1. *Simplificarea cererilor*, interogările multiple aplicate asupra mai multor BD pot fi aplicate doar asupra unei singure vederi.
2. *Simplicitate structurală*, vederile pot să creeze utilizatorului o viziune personală asupra BD pentru a interpreta rezultatele.
3. *Securitate*, restricționarea accesului la date pentru utilizatori.
Dezavantajele utilizării vederilor:
1. *Performanțe*, interogările asupra vederilor pot să se transforme în interogări asupra tabelelor de bază, lucru care se va reflecta în performanțe.
2. *Restricții la actualizări*, multe dintre vederi pot fi protejate la scriere (read-only), în acest caz nu se pot face actualizări.

3.3. Criptarea datelor

Criptarea este un proces bazat pe un algoritm prin care datele ce vor fi supuse acestui proces, numite și date de intrare, își vor modifica forma inițială astfel încât acestea să devină nedescifrabile fără a efectua algoritmul de decriptare − procedeul efectuat în criptare este reluat în revers.

Știința care studiază algoritmica din spatele procesului de criptare și decriptare a informațiilor poartă denumirea de *criptografie* și se bazează pe noțiuni matematice complexe. Principiul codificării este acela că algoritmul care va prelucra un set de date/informații va ajunge la un rezultat unic pentru informația originală, niciun alt set de date nu va avea acest rezultat final. În informatică, criptarea datelor este folosită oriunde este necesară securitatea transmiterii de informații confidențiale de la un calculator la altul sau stocarea datelor folosind o parolă. Domeniile de aplicabilitate sunt variate, de la

instituții guvernamentale și private (cum sunt băncile) până la datele personale ale utilizatorilor.

Pentru mărirea nivelului de securitate în cadrul procesului de criptare au apărut informații schimbătoare, dar care se bazează pe același algoritm, numite chei. Cu alte cuvinte, algoritmul este același, însă va folosi o cheie de criptare diferită, ceea ce va avea ca rezultat o codificare cu un model de criptare precizat de cheia de criptare. Există două tipuri de criptare: simetrică și asimetrică. Criptarea simetrică va folosi aceeași cheie atât pentru criptarea, cât și pentru decriptarea datelor − cine are cheia poate codifica și decodifica informații în același timp. Al doilea tip este mai complex decât primul, deoarece se folosesc două chei: una publică − de criptare, și alta privată − de decriptare. După cum îi spune și numele, cheia publică este vizibilă oricui și este folosită pentru a codifica datele. Decodificarea datelor va necesita cheia privată care îi corespunde cheii publice în mod unic − dacă se cunosc datele codificate și cheia publică cu care s-a făcut codificarea (fără a cunoaște cheia privată), nu se vor putea decodifica datele. Uzual, cheile publice sunt stocate vizibil și pot fi distribuite public, în timp ce cheile private sunt generate prin intermediul unei parole.

Probleme de securitate pe care criptarea nu le rezolvă

Deși există multe motive bune pentru criptarea datelor, există motive pentru a nu cripta datele.

1. *Criptarea nu rezolvă problemele de control al accesului*

Se cunoaște că atunci când sunt criptate datele, criptarea nu trebuie să interfereze cu modul în care este configurat controlul accesului. Majoritatea întreprinderilor trebuie să limiteze accesul la date pentru utilizatorii care au nevoie să vadă aceste date. De exemplu, un sistem de resurse umane poate limita angajații să-și vadă numai propriile înregistrări ale locurilor de muncă, permițând managerilor angajaților să vadă înregistrările de angajare ale subordonaților. Specialiștii în resurse umane pot, de asemenea, să vadă înregistrări ale mai multor angajați.

De obicei, se utilizează mecanisme de control al accesului pentru a aborda politicile de securitate care limitează accesul la date pentru cei care nu au permisiunea să le vizualizeze. Dacă controalele de acces sunt implementate corect, atunci criptarea adaugă puțină securitate suplimentară în cadrul BD. Un utilizator care are privilegii de acces la date în BD nu are mai multe sau mai puține privilegii ca urmare a criptării. Prin urmare, nu trebuie

utilizată niciodată criptarea pentru a rezolva problemele de control al accesului.

2. ***Criptarea nu protejează împotriva unui administrator rău intenționat***

Unele întreprinderi sunt preocupate de faptul că un utilizator cu intenții rele (răufăcător) încearcă să obțină privilegii ridicate (de exemplu, ca și cele ale administratorului BD). Soluția corectă la această problemă este protejarea contului de administrator BD și schimbarea parolelor implicite pentru alte conturi privilegiate. Cea mai ușoară modalitate de a intra într-o BD este utilizarea unei parole implicite pentru un cont privilegiat pentru care administratorul a permis să rămână neschimbată.

Deși există multe lucruri distrugătoare pe care le poate face un utilizator rău intenționat la o BD după ce a obținut privilegiul administratorului BD, criptarea nu va proteja multe dintre ele. Exemplele includ coruperea sau ștergerea datelor, exportul de date de utilizator către sistemul de fișiere pentru trimiterea de e-mailuri către el însuși pentru a rula un cracker de parole pe el etc.

Întreprinderile sunt preocupate de faptul că administratorii BD, care de obicei au toate privilegiile, pot vedea toate datele din BD. Aceste întreprinderi consideră că administratorii BD ar trebui să administreze BD, dar nu ar trebui să poată vedea datele pe care le conține ea. Concentrarea privilegiilor într-o singură persoană nu li se pare corectă și ar prefera să împartă funcția administratorului BD sau să impună reguli de acces pentru două persoane.

Este tentant să avem încrederea că criptarea tuturor datelor (sau cantități semnificative de date) va rezolva aceste probleme, dar există modalități mai bune de protecție împotriva acestor amenințări. De exemplu, SQL Server suportă partajarea limitată a privilegiilor administratorului BD *SYSADMIN* − pot face orice în SQL Server și figurează drept proprietari (dbo) ai BD (chiar atunci când nu sunt) și pot trece peste toate permisiunile și sistemele de securitate, iar *SETUPADMIN* pot instala și configura servere legate și pot marca o procedură stocată pentru execuție la momentul pornirii. Pot fi create roluri mai mici care să cuprindă câteva privilegii de sistem. Este posibil ca un rol să nu includă toate privilegiile de sistem, ci doar cele potrivite unui administrator BD. Funcția de administrator BD este o poziție de încredere. Chiar și întreprinderile care au cele mai sensibile date, cum ar fi agențiile de informații, nu împart de obicei funcția de administrator BD. Auditul periodic poate ajuta la descoperirea unor activități necorespunzătoare.

Criptarea datelor stocate nu trebuie să interfereze cu administrarea BD, deoarece altfel pot apărea probleme de securitate mai mari. De exemplu, dacă prin criptare sunt corupte datele, atunci se creează o problemă de securitate, deoarece datele înseși nu pot fi interpretate și este posibil să nu fie recuperabile. Poate fi utilizată criptarea pentru a limita capacitatea unui administrator BD sau a unui alt utilizator privilegiat de a vedea date în BD. Cu toate acestea, nu va fi substitut pentru gestionarea corectă a privilegiilor administratorului BD sau pentru controlul utilizării privilegiilor puternice de sistem. Dacă utilizatorii nedemni au privilegii semnificative, atunci pot prezenta mai multe amenințări unei întreprinderi, unele dintre ele mult mai importante decât vizualizarea numerelor cărților de credit necriptate.

3. *Criptarea tuturor lucrurilor nu face ca datele să fie securizate*

O eroare obișnuită este să credem că dacă criptarea unor date întărește securitatea, atunci criptând totul vom avea toate datele în siguranță. Este important ca criptarea să nu interfereze cu controalele de acces obișnuite. În plus, criptarea unei întregi BD de producție înseamnă că toate datele trebuie să fie decriptate pentru a fi citite, actualizate sau șterse. Criptarea tuturor datelor va afecta în mod semnificativ performanța.

Disponibilitatea este un aspect-cheie al securității. Dacă criptarea face ca datele să fie indisponibile sau afectează negativ disponibilitatea prin reducerea performanței, atunci criptarea totală va crea o nouă problemă de securitate. Disponibilitatea este, de asemenea, afectată negativ de faptul că BD este inaccesibilă atunci când cheile de criptare sunt modificate, deoarece practicile bune de securitate necesită modificarea în mod regulat. Când cheile trebuie să fie schimbate, BD este inaccesibilă atât timp cât datele sunt decriptate și recrutate cu o nouă cheie sau chei.

Criptarea este un avantaj pentru datele stocate offline. De exemplu, o întreprindere poate stoca copii de rezervă pentru o perioadă de la 6 luni până la un an offline, într-o locație la distanță. Prima linie de protecție este de a asigura facilitatea de stocare a datelor, prin stabilirea controalelor de acces fizic. Criptarea acestor date înainte de a fi stocate poate oferi beneficii suplimentare, deoarece nu sunt accesate online, iar performanța nu trebuie să fie considerată (deci, nu afectează în niciun fel performanța). Există furnizori care furnizează servicii de criptare. Înainte de a începe criptarea la scară largă a datelor de rezervă, întreprinderile care au în vedere această abordare ar trebui să testeze cu atenție procesul. Este esențial de verificat dacă datele criptate înainte de stocarea offline pot fi decriptate și reimportate cu succes.

Surse bibliografice:

1. A. Watt, N. Eng. *Database Design*, 2014.
2. Beginner SQL Tutorial, SQL Integrity Constraints. https://beginner-sql-tutorial.com/sql-integrity-constraints.htm [Accesat: 28.12.2022]

IV. METODE DE AUDIT AL BAZEI DE DATE

Auditul trebuie să asigure echilibrul, verificând dacă sunt întrunite condițiile necesare pentru a-l păstra, să instrumenteze stăpânirea dezordinii, adaptarea la schimbări, să evalueze gradul de securitate și riscurile.

Definiția 4.1. *Auditul* este procesul prin care persoane competente și independente colectează și evaluează probe pentru a-și forma o opinie asupra gradului de corespondență între cele observate și anumite criterii prestabilite.

Definiția 4.2. *Triggerul* (declanșatorul) este un tip special de procedură stocată care rulează automat atunci când are loc un eveniment în serverul bazei de date.

4.1. Auditul logon la nivel de SQL Server

În cadrul acestui tip de audit este înregistrat un eveniment pentru fiecare conectare (logon). Pentru fiecare eveniment este necesar să salvăm *loginname* și *timestamp*. Pot fi considerate înregistrate și informații adiționale care includ TCP/IP adresa clientului (IP-ul calculatorului de unde are loc conectarea) și programul utilizat pentru inițierea conectării (cu ce aplilcație, de exemplu Management Studio).

Adițional vor fi înregistrate și toate încercările de conectare nereușite care sunt uneori chiar mai importante decât cele reușite și care sunt utilizate ca bază pentru alertă. Logarea excesivă cu eșec este un raport de securitate interesant și deseori se analizează defalcarea (spargerea) acestor încercări de conectare eșuate pe baza următoarelor dimensiuni: *Numele de utilizator, IP-ul clientului de la care conexiunile sunt eșuate, Programul sursă, Ora zilei*.

Activitatea de conectare poate fi auditată:

- utilizând caracteristicile BD;
- utilizând o soluție de securitate a BD externă.

Toate SGBD-urile acceptă această funcție de audit de bază și deoarece numărul acestor evenimente este destul de mic (cel puțin în comparație cu numărul de evenimente care pot fi obținute atunci când este efectuat auditul apelurilor actuale SQL), există o penalitate de performanță redusă în efectuarea acestui nivel de audit al BD.

Exemplul 4.1.1. Pentru a putea fi implementat acest traseu de audit inițial este creat tabelul *LogonAudit* unde este stocată informația necesară:

```
CREATE TABLE LogonAudit
(
   AuditID INT NOT NULL CONSTRAINT PK_LogonAudit_AuditID
         PRIMARY KEY CLUSTERED IDENTITY(1,1)
   , UserName NVARCHAR(255)
   , LogonDate DATETIME
   , spid INT NOT NULL      --SP ID  numărul procesului interior SQL
Server
   , HostName varchar(255)
   , ProgramName varchar(255)
   , DbName varchar(255)
);
```

Apoi, este creat un trigger care urmează să fie activat de o nouă conectare și înregistrează în tabelul de audit LogonAudit cine s-a logat (*SUSER_SNAME()*), când a avut loc logarea (*GETDATE()*), identificatorul sesiunii de conexiune din SQL Server (*@@SPID*), numele calculatorului de rețea (*HOST_NAME()*), ce aplicație a fost utilizată pentru conectare (*PROGRAM_NAME()*) și la care BD are loc conectarea (*DB_NAME()*):

```
CREATE TRIGGER MyLogonTrigger ON ALL SERVER FOR LOGON
   AS
   BEGIN
   INSERT INTO Test.dbo.LogonAudit (UserName, LogonDate, spid,
   HostName, ProgramName, DbName)
   VALUES (SUSER_SNAME(), GETDATE(), @@SPID,
   HOST_NAME(), PROGRAM_NAME(), DB_NAME());
END;
```

4.2. Auditul activității DML (Data Manipulation Language)

Auditarea activității DML (*SELECT, INSERT, UPDATE, DELETE*) este o cerință în special în cazul când vorbim de acuratețea informațiilor financiare. Traseele auditului manipulării datelor sunt comune în aproape toate inițiativele de audit. O cerință de audit aferentă (deși nu este la fel de răspândită ca necesitatea de a controla activitatea DML) este înregistrarea completă a valorii vechi și a noii valori pentru fiecare activitate a DML.

Exemplul 4.2.1. Poate fi creat un traseu de audit pentru coloana unui tabel al angajaților în care sunt stocate bonusurile anuale. În acest caz, pot fi create două cerințe diferite. Prima poate conține înregistrarea completă a

oricărei actualizări a acestor valori și fiecare înregistrare conține utilizatorul care a efectuat actualizarea, clientul utilizat, ce aplicație a fost utilizată, când a fost făcută actualizarea și care este statutul actual al instrucțiunii SQL. O a doua cerință poate fi înregistrarea tuturor informațiilor de mai sus, dar și înregistrarea valorii înaintea actualizării și a valorii după actualizare. Acest fapt nu este întotdeauna același lucru, deoarece îmi pot da (eu fac modificările în favoarea mea) un bonus de 50%, folosind o comandă cum ar fi:

*UPDATE EMP SET BONUS = BONUS * 1.5*

Traseele de audit DML care presupun înregistrarea valorilor vechi și noi reprezintă un tip important de audit care trebuie inclus în cele mai dese cazuri. Cu toate acestea, trebuie atent de lucrat cu această categorie și de realizat un astfel de audit selectiv, deoarece în unele cazuri se exagerează și, din motive de simplitate, activează auditul pentru fiecare operațiune DML. În timp ce acest lucru este posibil din punct de vedere tehnic, cantitatea de date produse poate fi mare și trebuie să se asigure că infrastructura de audit poate gestiona sarcina, mai ales când sunt incluse valorile vechi și noi.

Exemplul 4.2.2. Presupunem că avem 1 milion de tranzacții DML pe zi și fiecare tranzacție actualizează o singură valoare, sunt 100 de tabele în BD, fiecare cu 10 valori care pot fi actualizate, iar auditul începe cu o BD care conține 10.000 de înregistrări în fiecare tabel. Deși calculul este simplist și imprecis, dacă înregistrezi valori vechi și noi, după un an, BD de audit va fi de peste 35 de ori mai mare decât BD în sine.

Prin urmare, atunci când se discută despre trasee de audit DML, ar trebui de ales selectiv ce obiecte și ce comenzi fac parte din obiectul auditului. *De exemplu,* se pot crea trasee de audit pentru un subset de tabele ale BD, pentru un subset de intrări sau conturi și așa mai departe. Chiar mai selectivă este alegerea tabelelor și coloanelor care trebuie să mențină valori vechi și noi.

Auditul DML este susținut prin intermediul a trei metode principale, dar compararea zilnică (sau periodică) nu este o opțiune în acest caz. Cele trei metode utilizează:

1. ***Capacitățile BD***, reprezintă instrumente interne de sistem cu ajutorul cărora putem crea trasee de audit.

2. ***Un sistem de audit extern***, presupune utilizarea sistemelor de audit extern de BD care sprijină auditurile DML pe baza oricăror criterii de filtrare, inclusiv obiectul de BD, utilizatorul, aplicația și așa mai departe. De

asemenea, acestea ajută la captarea și comprimarea acestor informații și la punerea lor la dispoziția cadrelor de raportare chiar și atunci când cantitatea de date este copleșitoare.

3. ***Triggere***, folosirea triggerilor personalizați. Dacă trebuie creat un traseu de audit DML pentru câteva obiecte, adăugarea triggerilor care înscriu informațiile într-un tabel special de audit poate fi cel mai simplu și mai rapid lucru.

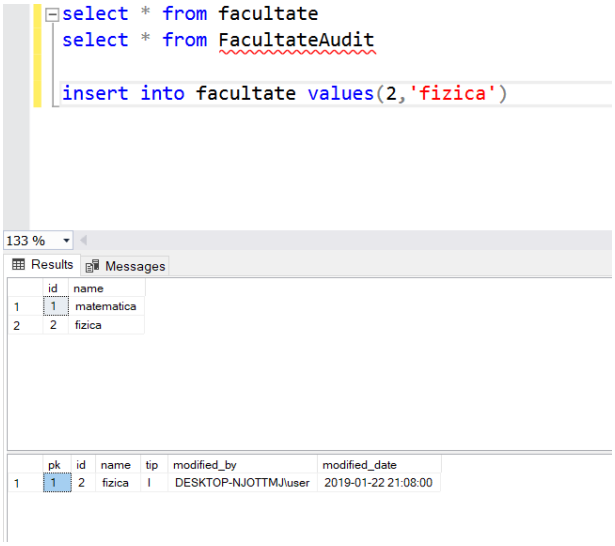
Exemplul 4.2.3. Pentru tabelul *Facultate*, ce conține câmpurile *id* și *name*, se creează tabelul de audit *FacultateAudit*:

```
CREATE TABLE [dbo].[FacultateAudit](
  [pk] [int] IDENTITY(1,1) NOT NULL,
  [id] [int] NULL,
  [name] [nvarchar](50) NULL,
  [tip] [char](1) NOT NULL,
  [modified_by] [varchar](50) NOT NULL,
  [modified_date] [smalldatetime] NULL,
 CONSTRAINT [PK_FacultateAudit] PRIMARY KEY
(   [pk] ASC  ))
```

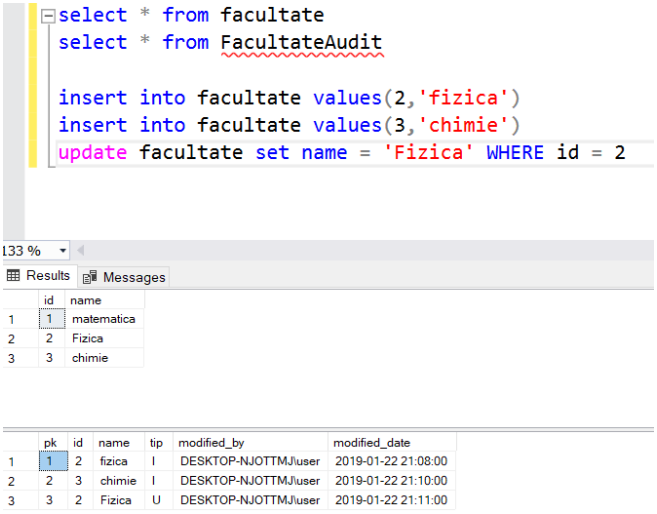
Pentru acest tabel se declanșează triggerul *INSERT UPDATE:*

```
CREATE TRIGGER [dbo].[trg_Facultate_IU]
  ON [dbo].[Facultate]
FOR INSERT, UPDATE
AS
BEGIN
  DECLARE @tip_op char(1) = 'I'
IF EXISTS(SELECT * FROM Deleted)
        SET @tip_op = 'U'
  INSERT INTO FacultateAudit([id], [name], [tip], [modified_by])
  SELECT [id], [name], @tip_op, SUSER_SNAME()
  FROM inserted;
 END
```

În așa fel, dacă în tabelul *Facultate* a fost o înregistrare *matematica*, atunci după adăugarea unei înregistrări *fizica* în tabelul *FacultateAudit* apare o înregistrare nouă *(2, fizica)*.



Fie că este adăugată în tabel încă o facultate, *chimie*, și modificăm *fizica* în *Fizica*, atunci se obține:



Pentru același tabel poate fi declanșat triggerul *INSERT UPDATE DELETE:*

```
CREATE TRIGGER [dbo].[trg_Facultate_IU]
  ON [dbo].[Facultate]
FOR INSERT, UPDATE, DELETE
AS
BEGIN
  DECLARE @tip_op char(1) = 'D'
  IF EXISTS(SELECT * FROM inserted)
  BEGIN
```

```
        SET @tip_op = 'I'
        IF EXISTS(SELECT * FROM Deleted)
            SET @tip_op = 'U'
        INSERT INTO FacultateAudit([id], [name], [tip], [modified_by])
        SELECT [id], [name], @tip_op, SUSER_SNAME()
        FROM inserted;
    END
    ELSE BEGIN
        INSERT INTO FacultateAudit([id], [name], [tip], [modified_by])
        SELECT [id], [name], @tip_op, SUSER_SNAME()
        FROM deleted;
    END
END
```

care generează un rezultat de tipul:

```
select * from facultate
select * from FacultateAudit

insert into facultate values(2,'fizica')
insert into facultate values(3,'chimie')
update facultate set name = 'Fizica' WHERE id = 2
insert into facultate values(4,'chimie astronomica')
DELETE FROM facultate where id = 4
```

33 %

Results | Messages

	id	name
1	1	matematica
2	2	Fizica
3	3	chimie

	pk	id	name	tip	modified_by	modified_date
1	1	2	fizica	I	DESKTOP-NJOTTMJ\user	2019-01-22 21:08:00
2	2	3	chimie	I	DESKTOP-NJOTTMJ\user	2019-01-22 21:10:00
3	3	2	Fizica	U	DESKTOP-NJOTTMJ\user	2019-01-22 21:11:00
4	4	4	chimie astronomica	I	DESKTOP-NJOTTMJ\user	2019-01-22 21:20:00
5	5	4	chimie astronomica	D	DESKTOP-NJOTTMJ\user	2019-01-22 21:20:00

unde I – *INSERT*, U – *UPDATE*, D – *DELETE*. Primul tabel este tabelul asupra căruia se fac modificările, iar al doilea este tabelul de audit. În fiecare din exemplele prezentate poate fi urmărită activitatea DML asupra elementelor tabelului *Facultatea*, rezultatele fiind înscrise în tabelul de audit *FacultateAudit*. În tabelul de audit, pe lângă datele modificate, este înregistrată informația referitor la tipul modificării, cine și când a realizat-o.

4.3. Auditul activităților DDL (Data Definition Language)

Auditul activității DDL a fost întotdeauna important și a devenit unul dintre cele mai implementate trasee de audit, deoarece auditurile de schimbare a schemelor sunt importante din punctul de vedere al securității, al conformității și din punctul de vedere al managementului configurației (adaugă/modifică o coloană, adaugă/modifică o condiție de integritate). Din punctul de vedere al securității, comenzile DDL sunt potențial cele mai dăunătoare comenzi existente și pot fi cu siguranță folosite de un atacator pentru a compromite orice sistem. Chiar furtul de informații poate implica adesea comenzi DDL (de exemplu, prin crearea unui tabel suplimentar în care datele pot fi copiate înainte de extracție). Din punctul de vedere al conformității, multe reglementări cer auditul oricăror modificări ale structurilor de date, cum ar fi tabelele și vederile.

Schimbările de schemă/structură a BD trebuie să fie auditate și salvate pentru referințe ulterioare ca o modalitate de a identifica și rezolva rapid erorile care pot compromite portabilitatea datelor sau care pot provoca coruperea datelor. Auditul activității DDL este implementat și pentru a elimina erorile pe care dezvoltatorii și administratorii BD le pot introduce și care pot avea efecte catastrofale.

Exemplul 4.3.1. Un client poate avea o perioadă de întrerupere a activității din cauza unei schimbări care este făcută de un dezvoltator − o schimbare pe care dezvoltatorul crede că o face pe serverul de dezvoltare, dar greșit este făcută pe serverul de producție. Controlul strict al procesului de gestionare a configurației este important și este unul dintre principalele trasee ale auditului DDL.

Majoritatea SGBD-urilor permit verificarea activității DDL utilizând diferite mecanisme de audit ca, de exemplu, DDL triggere, care se lansează la modificarea structurii schemei de sistem.

Exemplul 4.3.2. Se creează tabelul de audit *DDLEvents*:

```
CREATE TABLE dbo.DDLEvents
(
   EventDate    DATETIME NOT NULL DEFAULT
CURRENT_TIMESTAMP,
   EventType    NVARCHAR(64),
   EventDDL     NVARCHAR(MAX),
   EventXML     XML,
```

```
    DatabaseName NVARCHAR(255),
    SchemaName   NVARCHAR(255),
    ObjectName   NVARCHAR(255),
    HostName     VARCHAR(64),
    IPAddress    VARCHAR(48),
    ProgramName  NVARCHAR(255),
    LoginName    NVARCHAR(255)
);
```

Pentru care se creează un trigger care permite înregistrarea informației referitor la tipul evenimentului (Create Table, Alter Table, Drop Table ș.a.), instrucțiunea SQL executată, când a avut loc evenimentul, identificatorul sesiunii de conexiune din SQL Server, toată informația asupra evenimentului stocată în format XML, BD asupra căreia are loc evenimentul, schema asupra căreia are loc evenimentul, elementul schemei asupra căruia are loc evenimentul (coloană, tabel), numele calculatorului de rețea, adresa IP, ce aplicație a fost utilizată și numele utilizatorlui care a executat:

```
CREATE TRIGGER DDLTrigger_SchemaChange
  ON DATABASE
  FOR CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE,
    CREATE_TABLE, ALTER_TABLE, DROP_TABLE,
    CREATE_FUNCTION, ALTER_FUNCTION, DROP_FUNCTION,
    CREATE_INDEX, ALTER_INDEX, DROP_INDEX,
    CREATE_VIEW, ALTER_VIEW, DROP_VIEW,
    CREATE_TRIGGER, ALTER_TRIGGER, DROP_TRIGGER
AS
BEGIN
  SET NOCOUNT ON;--să nu întoarcă mesaje de sistem (de ex., câte rânduri au fost inserate)
    DECLARE
      @EventData XML = EVENTDATA();
    DECLARE @ip varchar(48) = CONVERT(varchar(48),
      CONNECTIONPROPERTY('client_net_address'));
    INSERT dbo.DDLEvents
    (
EventType, EventDDL, EventXML, DatabaseName, SchemaName, ObjectName, HostName, IPAddress, ProgramName, LoginName
```

```
    )
     SELECT
    @EventData.value('(/EVENT_INSTANCE/EventType)[1]',
'NVARCHAR(100)'),
    @EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
'NVARCHAR(MAX)'),
    @EventData,
    DB_NAME(),
    @EventData.value('(/EVENT_INSTANCE/SchemaName)[1]',
'NVARCHAR(255)'),
    @EventData.value('(/EVENT_INSTANCE/ObjectName)[1]',
'NVARCHAR(255)'),
    HOST_NAME(),
    @ip,
    PROGRAM_NAME(),
    SUSER_SNAME();
    END
```

4.4. Auditul modificării privilegiilor, utilizatorilor (user) / datelor de conectare (login) și a altor atribute de securitate

Această categorie este una obligatorie pentru auditul BD și presupune păstrarea tuturor modificărilor aduse modelului de securitate și privilegiilor BD. Baza de date gestionează o schemă sofisticată de securitate, permisiuni și modificări, dar regula numărul unu în securitate este că modificările aduse modelului de securitate trebuie auditate. Ar trebui luată în considerare auditarea următoarelor modificări:

- adăugarea și ștergerea utilizatorilor, a logărilor și a rolurilor;
- modificări ale mapărilor (relațiilor) dintre logări și utilizatori sau roluri;
- modificările privilegiilor prin intermediul unui utilizator sau rol;
- schimbarea parolei;
- modificarea atributelor de securitate la nivel de server, BD sau obiect (tabel, coloană, vedere).

Deoarece modelul de securitate a BD este poarta de acces la BD, orice modificări ale privilegiilor trebuie să fie auditate. Atacatorii vor încerca să-și ridice nivelul privilegiilor, iar greșelile sunt deseori făcute atunci când privilegiile sunt oferite necorespunzător. Auditul tuturor modificărilor care ar

putea afecta securitatea BD poate fi asociată cu plasarea unei camere de supraveghere care urmărește ușa din față a clădirii, locul unde se schimbă codul de intrare și locul unde sunt emise insignele.

Deoarece modificările permisiunilor de securitate pot fi periculoase pentru BD (în cazul unui scenariu de atac), nu trebuie de bazat pe un tip de comparație efectuată la o anumită perioadă de timp (o dată pe zi/săptămână), dar ar trebui de optat pentru notificarea în timp real a modificărilor care nu sunt planificate într-un mediu de producție. Aceasta înseamnă că trebuie utilizată o BD externă de securitate și de audit sau construite alerte în timp real bazate pe trasee de audit.

Dacă se intenționează implementarea acestui traseu de audit, trebuie capturate evenimente relevante și apoi construit cadrul de alertare. Dacă este utilizat un sistem extern care acceptă atât audit, cât și alerte în timp real, se pot adăuga reguli la politica de alertă care vor informa când se utilizează proceduri de securitate sau comenzi.

Exemplul 4.4.1. Pentru implementarea unui traseu de audit DDL este creat tabelul *DDLUserAudit*:

```
CREATE TABLE dbo.DDLUserAudit
(
   Id           INT IDENTITY(1,1) NOT NULL,
   CreateDate   DATETIME NULL,
   LoginName    SYSNAME NULL,
   ComputerName SYSNAME NULL,
   ProgramName  NVARCHAR(255),
   DBName       SYSNAME NOT NULL,
   SQLEvent     SYSNAME NOT NULL,
   SchemaName   SYSNAME NULL,
   ObjectName   SYSNAME NULL,
   SQLCmd       NVARCHAR(max) NULL
);
```

Pentru care se va crea un trigger:

```
CREATE TRIGGER [DDLTrigger_UserAudit] ON ALL SERVER
   FOR CREATE_LOGIN, ALTER_LOGIN, DROP_LOGIN,
     CREATE_USER, ALTER_USER, DROP_USER,
         CREATE_ROLE, ALTER_ROLE, DROP_ROLE,
         ADD_ROLE_MEMBER, DROP_ROLE_MEMBER
AS
```

```
BEGIN
   DECLARE   @EventDataXML   XML
   SET @EventDataXML = EVENTDATA();
   INSERT univer.dbo.DDLUserAudit
   (
      CreateDate, LoginName, ComputerName,
ProgramName, DBName, SQLEvent, SchemaName,
 ObjectName, SQLCmd
   )
      SELECT
GETDATE(),
SUSER_NAME(),
HOST_NAME(),
PROGRAM_NAME(),
DB_NAME(),
   @EventDataXML.value('(/EVENT_INSTANCE/EventType)[1]',
'SYSNAME'),

@EventDataXML.value('(/EVENT_INSTANCE/SchemaName)[1]',
'SYSNAME'),

@EventDataXML.value('(/EVENT_INSTANCE/ObjectName)[1]',
'SYSNAME'),

@EventDataXML.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
'NVARCHAR(MAX)');
END;
```

Structura tabelului de audit obținut în urma creării a două loginuri și a unui user este:

	Id	CreateDate	LoginName	ComputerName	ProgramName	DBName	SQLEvent	SchemaName	ObjectName	SQLCmd
1	4	2020-01-20 18:57:32.603	DESKTOP-NJOTTMJ\user	DESKTOP-NJOTTMJ	Microsoft SQL Server Management Studio - Query	master	CREATE_LOGIN	NULL	i2	create login i2 with PASSWORD = ********
2	5	2020-01-20 18:58:02.600	DESKTOP-NJOTTMJ\user	DESKTOP-NJOTTMJ	Microsoft SQL Server Management Studio - Query	master	CREATE_USER	NULL	i2	create user i2 for login i2
3	3	2020-01-20 18:21:59.050	DESKTOP-NJOTTMJ\user	DESKTOP-NJOTTMJ	Microsoft SQL Server Management Studio - Query	master	CREATE_LOGIN	NULL	i1	CREATE LOGIN i1 WITH PASSWORD = ********

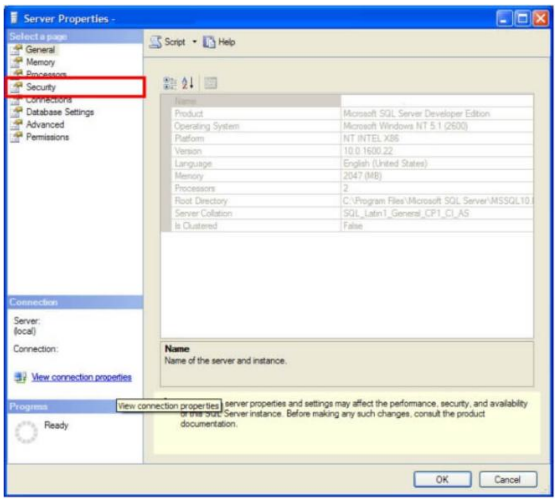
4.5. Auditul erorilor bazei de date la nivel de aplicație și SQL Server

Auditul erorilor returnate de BD are o mare importanță din punctul de vedere al securității și este un traseu de audit care trebuie implementat. Când discutăm despre atacurile SQL Injection, spunem că, în multe cazuri, atacatorii vor face multe încercări până când vor reuși.

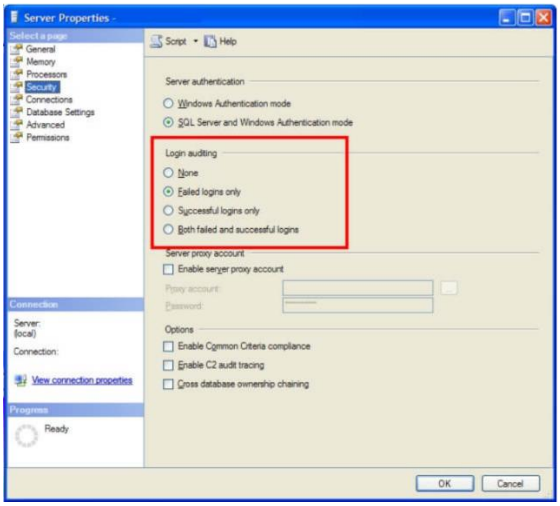
În cazul unui atac bazat pe UNION, în care atacatorii trebuie să ghicească numărul corect de coloane, BD va întoarce continuu un cod de eroare cu mesajul: *coloanele selectate de cele două instrucțiuni SELECT nu corespund*. Fiind înregistrate toate erorile, se poate identifica această situație și reacționa în mod adecvat. Înregistrarea nereușită la server este un alt exemplu al unei erori care trebuie înregistrată și monitorizată, chiar dacă nu se face audit la BD. În cele din urmă, orice încercare eșuată de a ridica privilegiile este un indicator puternic că un atac poate fi în desfășurare.

Auditul erorilor este important și din perspectiva calității. Aplicațiile de producție care cauzează erori din cauza problemelor legate de erorile din cadrul aplicațiilor ar trebui identificate și reparate. Logarea erorilor SQL este o modalitate de a identifica aceste probleme și erorile pot să orienteze în direcția unor probleme care afectează timpul de răspuns și disponibilitatea. Erorile pot fi raportate folosind orice set de criterii, iar informațiile sunt disponibile pentru construirea unui traseu de bază. În majoritatea mediilor unele aplicații generează erori de bază chiar și în producție. Cu toate acestea, erorile generate de aplicații sunt repetitive: aceleași erori apar în aproximativ același loc, deoarece erorile rezultă de obicei din bug-uri, și acestea nu se schimbă. Dacă apar brusc erori care vin din diferite locuri sau este depistat un cod de eroare complet diferit, atunci ar trebui de investigat ce se întâmplă.

SQL Server permite verificarea atât a succeselor, cât și a eșecurilor de conectare. În funcție de situație, sunt diferite moduri de activare a acestui audit folosind SQL Server Management Studio. După conectarea la SQL Server în Object Explorer se face clic dreapta pe SQL Server și se alege opțiunea *Properties* din meniul pop-up. Proprietățile serverului se pot vedea astfel:



Se face clic pe pagina *Securitate* (din imaginea anterioară), după care se poate seta auditul de autentificare:



Există 4 opțiuni disponibile:

- ***None*** − nu vor fi auditate nici autentificările de succes și nici cele eșuate.

- ***Failed logins only*** − autentificările eșuate vor fi auditate, iar autentificările de succes vor fi ignorate.

- ***Successful logins only*** − autentificările de succes vor fi auditate, iar autentificările eșuate vor fi ignorate.

- ***Both failed and successful logins*** − autentificările vor fi auditate indiferent de succes și eșec.

După alegerea opțiunii se face clic pe butonul *OK*. Setarea pentru auditul de autentificare este, de fapt, o intrare de registru care diferă în funcție de versiunea SQL Server și numele instanței SQL Server. SQL Server citește

această setare doar atunci când serviciul SQL Server pornește. Pentru ca această setare să intre în vigoare este necesar de a reporni serviciul SQL Server.

Pentr vizualizarea conținutul jurnalului SQL Server folosind T-SQL există o procedură de stocare extinsă *xp_readerrorlog*. Ea poate fi utilizată pentru a arunca rezultatele jurnalului de erori într-un set de înregistrări.

Exemplul 4.5.1. *EXEC xp_readerrorlog*

Pentru a căuta în jurnalul de erori curent și pentru a returna autentificările eșuate se poate utiliza comanda *xp_readerrorlog* cu doi parametri: primul specifică jurnalul de erori current (0 = curent), al doilea parametru specifică tipul de jurnal (1 = Jurnal de eroare SQL), iar al treilea parametru specifică mesajul pe care trebuie să îl caute.

Exemplul 4.5.2. *EXEC sp_readerrorlog 0, 1, 'Login failed'*

Se obține un rezultat de tipul:

	LogDate	ProcessInfo	Text
1	2020-01-21 20:31:43.720	Logon	Login failed for user 'octav'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]
2	2020-01-21 20:31:51.660	Logon	Login failed for user '9i8u7'. Reason: Could not find a login matching the name provided. [CLIENT: <local machine>]
3	2020-01-21 20:32:07.860	Logon	Login failed for user 'numele'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]
4	2020-01-21 20:32:18.750	Logon	Login failed for user 'numele'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]

Ca recomandare, ar trebui verificate cel puțin conectările eșuate pe sistemele de producție. Pot exista cazuri în care este necesar să se auditeze și autentificările reușite, dar trebuie de realizat faptul că se pot genera o mulțime de evenimente care vor fi cu greu analizate. Este important să existe acest nivel de control pe sisteme extrem de sensibile. Cu toate acestea, pe sisteme mai puțin critice s-ar putea aplica doar verificarea autentificărilor eșuate.

4.6. Auditul instrucțiunilor SELECT pentru date confidențiale

Instrucțiunile SELECT nu au fost în centrul atenției în trecut, dar accentul recent asupra confidențialității datelor a schimbat totul. Dacă trebuie asigurată conformitatea cerințelor de confidențialitate (de tip GLBA) sau trebuie doar să fie asigurați clienții, partenerii și angajații că informațiile lor confidențiale nu se scurg din BD, urmează să fie audiate instrucțiunile SELECT. În mod specific, trebuie de afișat de unde a venit instrucțiunea SELECT (adresa IP, aplicația), cine a selectat datele (numele de utilizator) și ce date au fost selectate. Ca și în cazul traseelor de audit ale DML, auditarea instrucțiunilor SELECT este impracticabilă pentru întreaga BD și trebuie de concentrat pe subgrupe care sunt semnificative și necesare.

Primul pas în ceea ce privește un traseu de audit SELECT este o clasificare a datelor importante. Le vom numi *set de confidențialitate*,

deoarece, în viața reală, colecțiile de valori ale datelor împreună sunt importante din perspectiva confidențialității.

Exemplul 4.6.1. Numele de familie nu este confidențial, dar numele unui președinte, împreună cu numărul de permis al conducătorului auto și numărul de securitate socială sunt confidențiale.

La etapa de clasificare trebuie de definit unde se află informațiile confidențiale (ce nume de obiecte și ce nume de coloane) și care combinație de date este confidențială. Un set de confidențialitate este, prin urmare, o colecție de 2 tupluri, fiecare tuplu constând dintr-un nume de obiect și un nume de coloană.

4.7. Auditul utilizării bazei de date în afara programului normal de lucru

Auditul activităților care se desfășoară în afara programului normal de lucru (de obicei, se realizează numai la tabele cu date confidențiale) este o cerință intuitivă și care este adesea cerută de o afacere din punctul de vedere al conformității. Cerința intuitivă de utilizare a BD în afara orelor normale de funcționare este necesară deoarece activitățile desfășurate în afara orelor de lucru sunt adesea suspecte și pot fi rezultatul unui utilizator neautorizat care încearcă să acceseze sau să modifice date. Desigur, un bun hacker va încerca să spargă BD în timpul unei perioade „camuflate”. Este mult mai bine să încerci când există mult „zgomot” care servește ca o deviere.

Când este efectuat auditul activității în afara orelor de lucru, de obicei nu este suficient să fie urmărite numai datele de conectare și înregistrările care au loc în afara orelor. În general, se dorește și capturarea activităților care sunt efectuate − de obicei, la un nivel SQL. Dacă astfel de conectare este suspectă, atunci este important de înțeles ce se obișnuiește să se facă în cadrul BD. Având un traseu de audit complet al tuturor activităților efectuate de orice utilizator în afara orelor normale de funcționare, este un criteriu bun de implementat și satisface multe cerințe de reglementare internă.

La nivel tehnic trebuie să fie clar că, deoarece majoritatea SGBD funcționează 24/7, nu este dorită generarea unei mulțimi de alarme false la prelucrarea controlată a datelor în afara orelor normale de funcționare (de exemplu, noaptea deseori se fac calcule grele ale datelor, ceea ce este o prelucrare controlată a datelor și nu necesită alarme). Prin urmare, cheia pentru buna punere în aplicare a acestui tip de audit nu trebuie să includă

activitățile care sunt programate să funcționeze întotdeauna în afara orelor de lucru.

Dacă se vede că un tip de activitate apare în fiecare seară, atunci traseul de audit în afara orelor de lucru ar trebui să excludă orice activitate efectuată de aplicațiile respective utilizând anumite nume de conectare și provenind de la adrese IP cunoscute din timp (sau, așa cum se întâmplă adesea, de la localhost). Auditarea numai a ceea ce diferă de activitatea de bază a SGBD reduce mărimea traseelor de audit care ar trebui inspectate, deoarece activitățile care vor fi înregistrate sunt doar acele activități care se produc în afara normei.

4.8. Auditul modificărilor codului sursă a procedurilor stocate și triggerelor

Deoarece procedurile stocate și triggerele BD folosesc limbaje de programare flexibile și pline de trăsături procedurale, este ușor de ascuns codul dăunător intenționat care altfel ar fi nedetectabil. Prin urmare, trebuie adoptată o bună practică de revizuire a tuturor modificărilor aduse acestor construcții.

Această categorie poate fi auditată în mai multe moduri:

- Modul cel mai simplu se bazează pe controlul structurii codului și poate fi implementat periodic (de exemplu, zilnic) pentru preluarea codului din BD și compararea lui cu codul extras din perioada anterioară. Această metodă este relativ simplu de implementat folosind un set de instrumente și scripturi (cum ar fi *diff*, sunt diferite repozitorii și codul din repozitoriu este versionat și se compară o versiune a codului mai vechi cu una mai nouă).

- O altă opțiune constă în utilizarea unui sistem de securitate și de audit extern care poate avertiza asupra oricărei comenzi de creare sau modificare a procedurilor și triggherelor în timp real și poate produce un set de rapoarte care detaliază modificările.

Surse bibliografice:

1. H. Afyouni. *Database Security and Auditing: Protecting Data Integrity and Accessibility*. Cengage Learning, 2006.
2. R. Natan. *Implementing Database Security and Auditing*. Elsevier, 2005.

3. Microsoft SQL Docs, SQL Server Audit (Database Engine) 2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/auditing/sql-server-audit-database-engine?view=sql-server-ver15 [Accesat: 28.12.2022]
4. Microsoft SQL Docs, Create a server audit and database audit specification, 2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/auditing/ create-a-server-audit-and-database-audit-specification?view=sql-server-2017 [Accesat: 28.12.2022]
5. CREATE TRIGGER (Transact-SQL), 2021 https://docs.microsoft.com/en-us/sql/t-sql/statements/ create-trigger-transact-sql?view=sql-server-ver15 [Accesat: 28.12.2022]

V. VULNERABILITĂȚI ALE BAZEI DE DATE

5.1. Vulnerabilitățile bazei de date în Internet

Cu excepția cazului în care se utilizează software de securitate, toate mesajele transmise prin Internet sunt în text simplu, pot fi detectate de intruși folosind software-ul *packet sniffing*. Atât emițătorii, cât și receptorii de mesaje trebuie să fie convinși că convorbirile (comunicațiile) lor sunt păstrate private. Evident, clienții care doresc achiziționarea produselor online trebuie să fie asigurați că informațiile cardului lor de credit sunt în siguranță atunci când le trimit prin Internet. Întreprinderile care permit conexiunile Web la rețelele lor interne pentru acces la baza lor de date trebuie să o poată proteja de atac. Receptorii de mesaje trebuie să aibă modalități de a asigura că acele mesaje sunt autentice și demne de încredere și că nu au fost manipulate. Expeditorii de mesaje nu ar trebui să le poată respinge, negând că le-au trimis. Pentru a rezolva aceste probleme sunt folosite mai multe tehnici.

5.1.1. Servere Proxy

Un server proxy este un computer sau un program care acționează ca intermediar între un client și un alt server, gestionând mesajele în ambele direcții. Când clientul solicită un serviciu, cum ar fi o conexiune sau o pagină Web, proxy îl evaluează și determină dacă poate îndeplini cererea în sine. Poate stoca în cache un răspuns al serverului (memorează o copie a răspunsului) astfel încât o nouă solicitare poate fi îndeplinită utilizând conținutul stocat fără a apela din nou serverul. Serverul proxy poate fi folosit pentru a menține securitatea prin ascunderea adresei IP reale a serverului, pentru a îmbunătăți performanța prin memorarea în cache, pentru a preveni accesul la site-uri la care o întreprindere dorește să blocheze accesul, pentru a proteja serverul de malware și pentru a proteja datele prin scanarea mesajelor de ieșire.

5.1.2. Firewall-uri

Un firewall este o barieră hardware și/sau software care este folosită pentru a proteja rețeaua internă a întreprinderii (intranet) de la acces neautorizat. Sunt folosite diverse tehnici pentru a se asigura că mesajele care intră sau ies din intranet respectă standardele întreprinderii. De exemplu, un server proxy poate fi folosit pentru a ascunde adresa reală de rețea. O altă tehnică este un *packet filter*, care examinează fiecare pachet de informații înainte ca acesta să intre sau să iasă din intranet, asigurându-se că respectă un

set de reguli. Pot fi aplicate câteva mecanisme de securitate a aplicațiilor sau conexiunilor.

5.1.3. Semnături digitale

Semnăturile digitale sunt algoritmi criptografici cu cheie publică, care se utilizează pentru demonstrarea autenticității mesajului și a sursei, precum și la asigurarea integrității mesajelor și a non-repudierii originii.

Semnătura digitală este, de fapt, rezultatul criptării *esenței mesajului* cu cheia privată a expeditorului. Această esență a mesajului reprezintă valoarea rezultantă din aplicarea unui algoritm de hash criptografic asupra mesajului care trebuie semnat. Pentru a verifica semnătura, receptorul va compara valoarea hash, obținută prin două metode: una dintre acestea o va obține la fel cum a obținut-o expeditorul înainte de semnare, utilizând același algoritm hash, a doua valoare o va obține prin decriptarea semnăturii cu cheia publică a expeditorului. Semnătura se consideră validă doar în cazul în care aceste două valori sunt egale.

5.1.4. Autorități de certificare

Clienții care doresc să cumpere bunuri de pe un site de comerț electronic trebuie să facă acest lucru fiind convinși că site-ul cu care comunică este autentic și că informațiile lor despre comandă sunt transmise în mod privat. O metodă utilizată pe scară largă pentru verificarea faptului că un site este autentic este prin intermediul autorităților de certificare (precum *Verisign*). Procesul utilizează criptarea cu cheie publică. Site-ul începe procesul de certificare prin generarea unei chei publice și a unei chei private și trimiterea unei cereri către autoritatea de certificare, împreună cu cheia publică a site-ului. Autoritatea de certificare generează un certificat criptat al site-ului, care este stocat pentru utilizarea ulterioară. Când clientul dorește să plaseze o comandă folosind o conexiune securizată la site, browserul solicită site-ului certificatul autorității, pe care îl primește formă criptată. Browserul decriptează certificatul folosind cheia publică a autorității și verifică dacă acesta este într-adevăr un certificat al autorității de certificare și că URL-ul site-ului este cel corect. Certificatul conține și cheia publică a site-ului. Browserul creează o cheie de sesiune – pe care o criptează folosind cheia publică a site-ului din certificat − și trimite cheia de sesiune către site. Pe parcursul sesiunii cheia este criptată cu cheia publică a site-ului, numai site-ul real o poate decripta folosind cheia sa privată. Atât browserul, cât și site-ul sunt singurii deținători ai cheii de sesiune și pot face schimb de mesaje

criptate cu cheia de sesiune, folosind un protocol. Procesul descris este cel utilizat în protocolul Secure Sockets Layer (SSL) și este de obicei folosit pentru mesaje către și de la un client în procesul unei comenzi. O măsură suplimentară de securitate este de obicei utilizată pentru a transmite numărul de card de credit. În timp ce utilizatorul browserului trimite site-ului vânzătorului mai multe informații despre comandă codificate cu cheia sa publică, atunci când clientul este gata să transmită informații despre cardul de credit la sfârșitul comenzii, aceste informații, împreună cu suma care trebuie percepută, sunt trimise direct către site-ul companiei de carduri pentru autorizare și aprobare.

Kerberos este un protocol de autentificare pentru rețele care permite autentificare reciprocă, în care atât clientul, cât și serverul pot verifica identitatea. Un Server Kerberos de încredere este folosit ca autoritate de certificare. Are un centru de distribuție a cheilor care stochează cheile secrete ale fiecărui client și server din rețea și le folosește ca intrare pentru a genera bilete marcate de timp atunci când clientul solicită serviciul. Un bilet este apoi folosit pentru a demonstra serverului că clientul este aprobat pentru serviciu. Mesajele pot fi criptate folosind oricare cheie simetrică sau protocoale cu cheie publică.

5.2. SQL Injection

SQL Injection este o vulnerabilitate foarte frecventă în apicațiile web care folosesc BD. Foarte mulți programatori nu conștientizează faptul că interogarile SQL pot fi modificate din exterior și presupun că aceste comenzi sunt sigure. Aceasta presupune că orice atac SQL (Structured Query Language) va trece de orice măsură de control și de securitate existentă pe site, precum și de orice metodă de autentificare folosită. În plus, mulți dintre ei se întreabă: Cine va incerca să atace site-ul? Ce motiv ar avea să facă această faptă?

Direct SQL Command Injection este o tehnică prin care un atacator creează sau modifică o interogare SQL existentă pentru a afla informații confidențiale, pentru a suprascrie informații sau chiar pentru a crea noi utilizatori. Interogăriile SQL se realizează prin introducerea în input-urile de pe site a comenzilor SQL pentru a realiza scopul propus. Majoritatea aplicațiilor-web păstrează datele în BD, deoarece acest fapt permite generarea paginilor dinamice. Aplicația-web primește de la utilizatori date care sunt folosite de aplicație/script pentru generarea unei cereri la BD.

Definiția 5.1. *SQL Injection* este o vulnerabilitate ce apare în cazurile când datele primite de la utilizatori nu se prelucrează corect. Prin urmare, răufăcătorul potențial poate schimba interpelarea la BD, astfel fiind posibil furtul datelor private.

Tehnicile care sunt folosite de răufăcători în procesul de exploatare a vulnerabilității SQL Injection demonstrează cum pot fi obținute cu ușurință conținutul BD, datele private, realizat atacul DoS, obținute privilegiile maxime etc. SQL Injection profită de faptul că interogările SQL pot fi construite dinamic în codul aplicației.

Exemplul 5.2.1. Fie un formular Web care permite unui student să-și introducă identificatorul și parola în variabilele *user_id* și *password*. Site-ul web utilizează apoi aceste informații pentru a prelua informațiile confidențiale ale studentului din tabelul *Student*. În codul aplicației interogarea poate fi construită dinamic prin următoarea instrucțiune:

*Student_query = "SELECT * FROM Student*
WHERE user_id = ' " + user_id + " ' AND password = ' " + password + " ' ; "

Caracterul + reprezintă operatorul de concatenare a șirurilor. Dacă *user_id* conține valoarea *Andrei* și *password* conține valoarea *pass123*, apoi *Student_query* ar conține următoarea instrucțiune:

*SELECT * FROM Student*
WHERE user_id = 'Andrei' AND password = 'pass123'

Student_query este trimisă la BD pentru preluarea informațiilor. Această interogare funcționează conform intenției numai dacă valoarea introdusă pentru *user_id* și/sau *password* nu conține un singur caracter de ghilimele. De exemplu, dacă utilizatorul introduce *pass'123* din greșeală, interogarea devine:

*SELECT * FROM Student*
WHERE user_id = 'Andrei' AND password = 'pass'123'

Ghilimelele străine din parolă vor determina generarea de către SQL a unui mesaj de eroare de tipul *'pass'123'; is invalid syntax in the SQL parse*. Atacatorul poate folosi această tehnică pentru a descoperi dacă o BD este vulnerabilă la un atac SQL Injection, unde mesajul de eroare de sintaxă oferă atacatorului un indiciu că interogarea este construită dinamic fără nicio validare a datelor de intrare. O interogare ulterioară construită, rău intenționată, oferă atacatorului acces la informațiile lui *Andrei*, precum și la informațiile tuturor celorlalți studenți.

Dacă presupunem că atacatorul nu cunoaște niciun *user_id* de utilizator sau *password* valid și atacatorul introduce valoarea *XXX* ca *user_id* și *YYY' OR '1'='1* ca *password*, interogarea devine:

*SELECT * FROM Student*
WHERE user_id = 'XXX' AND password = 'YYY' OR '1'='1';

Chiar și cu un *user_id* și un *password* incorecte, această interogare este întotdeauna evaluată ca adevărată, deoarece condiția va fi întotdeauna satisfăcută. Interogarea este evaluată ca interogarea ce urmează, care returnează informații despre toți studenții:

*SELECT * FROM Student WHERE '1'='1';*

Exemplul 5.2.2. Deoarece unele SGBD-uri permit executarea mai multor instrucțiuni SQL separate prin punct și virgulă într-o singură interogare, atacatorii pot profita și de această vulnerabilitate împreună cu ghilimele simple în valorile de intrare pentru a introduce instrucțiuni rău intenționate suplimentare. Pentru a ilustra acest tip de atac, presupunem că atacatorul introduce valoarea *XXX* ca *user_id* și valoarea *YYY' OR '1'='1'; DELETE * FROM Student; --* ca *password*. În acest caz, interogarea devine:

*SELECT * FROM Student*
WHERE user_id = 'XXX' AND password = 'YYY' OR '1'='1';
*DELETE * FROM Student;*
--' ;

Interogarea preia toate informațiile despre student și apoi șterge toate informațiile din tabelul *Student*. Caracterele de comentariu „ -- ” sunt folosite la sfârșitul șirului de parolă, astfel încât orice caractere străine vor fi comentate în afara execuției interogării pentru a evita o eroare de sintaxă.

Exemplul 5.2.3. O interogare ce extrage datele din BD poate avea forma:

*SELECT * FROM* Student *WHERE name = '$name'*

În acest caz valorile din câmpul “*name*” sunt comparate cu valoarea “*$name*”. Dacă valoarea este obținută din parametrii URL, câmp de tip *imput* al unei forme sau cookie și ***nu se prelucrează la simboluri speciale,*** atunci interogarea la BD este vulnerabilă și *răufăcătorul poate modifica interogarea.* Fie dintr-o pagină web (aplicație) se cere determinarea datelor despre student după un oarecare *id,* unde id-ul se introduce de utilizator. În acest scop este declarată variabila de tip *string* care conține instrucțiunea SELECT:

*sql = 'SELECT * FROM student WHERE id =' + Text1.Value*

În caz că răufăcătorul adaugă în caseta input pe lângă *id* și codul de tipul *1;SELECT name FROM sys.tables*, atunci la server vine cererea de executare a secvenței:

*SELECT * FROM Student WHERE id = 1;SELECT name FROM sys.tables*

Ca rezultat, sunt furnizate datele despre student, dar este afișată și lista de tabele ale BD.

Exemplul 5.2.4. În caz că răufăcătorul adaugă în caseta input pe lângă *id* și codul *1;Select * FROM Facultate*, atunci la server vine cererea de executare a secvenței:

*SELECT * FROM Student WHERE id =*1;Select * FROM Facultate

Ca rezultat, vor fi furnizate datele despre student și vor fi afișate datele din tabelul Facultate.

Exemplul 5.2.5. În caz că răufăcătorul adaugă în caseta input pe lângă *id* și codul *1;DELETE FROM Facultate*, atunci la server vine cererea de executare a secvenței:

*SELECT * FROM Student WHERE id = 1;DELETE FROM Facultate*

Ca rezultat, sunt furnizate datele despre student și sunt șterse datele din tabelul Facultate.

Exemplul 5.2.6. Fie dintr-o pagină web se cere determinarea datelor despre student după *nume*, unde numele se introduce de utilizator. În acest scop este declarată variabila de tip *string* care conține instrucțiunea SELECT:

*sql = 'SELECT * FROM Student WHERE name =' ' + Text2.Value + ' ' '*

În acest caz, răufăcătorul adăugând în caseta input pe lângă *id* și un cod de tipul *Ion' ';SELECT name FROM sys.tables WHERE ' '1' '=' '1*, atunci la server vine cererea de executare a secvenței:

*SELECT * FROM Student WHERE name = ' Ion' ;SELECT name FROM sys.tables WHERE '1'='1'*

Ca rezultat, sunt furnizate datele despre student dar este afișată și lista de tabele ale BD. În acest acest caz avem un SQL Injection la câmp de tip *string*. Aplicând aceleași manipulări putem obține, modifica sau șterge date din tabele.

Procedurile de testare a aplicațiilor web la SQL Injection se reduc la formarea unei liste de parametri cu care lucrează aplicația (atât parametrii *GET*, cât și *POST*), incluzând și parametrii cookie. Apoi acești parametri se

testează individual la prelucrarea simbolurilor speciale sau a cuvintelor-cheie (de genul *WHERE*) care ajută la exploatarea vulnerabilității.

Pașii unui atac al aplicației web cu scopul obținerii informației neautorizate a datelor din BD:

1) *identificarea parametrilor vulnerabili*, în caz că aplicația web este configurată astfel, încât în cazul apariției unei erori SQL în browser apare textul erorii și posibil o porțiune din interogare;

2) *identificarea tipului atributelor* și a *numărului de atribute* (coloanele tabelelor atacate);

3) *atacul propriu-zis* prin aplicarea unei metode cunoscute.

Din moment ce răufăcătorul găsește un parametru vulnerabil pentru a exploata vulnerabilitatea, ar trebui aproximativ să cunoască tipul cererii SQL în care se încearcă injectarea codului malicios. Cel mai des în aplicațiile-web sunt utilizate 4 tipuri de cereri SQL: *SELECT*, *INSERT*, *UPDATE*, *DELETE*. Din analiza logică și semantică a scriptului vulnerabil poate fi determinat tipul de cerere aplicat într-un caz concret.

5.2.1. Metode de atac SQL Injection

- Dacă scriptul afișează date ce corespund unui identificator anumit, atunci cu o mare probabilitate cererea este de tipul *SELECT*.

- Dacă scriptul adaugă unele date în BD (spre exemplu, adăugarea unui comentariu sau a unui post în forum), cererea este de tipul *INSERT*.

- Dacă scriptul modifică informația (spre exemplu, schimbarea parolei, redactarea postului în forum), cererea este de tipul *UPDATE*.

- Dacă are loc ștergerea informației (spre exemplu, anularea accountului), posibil că cererea este sau de tipul *DELETE*, sau de tipul *UPDATE*.

Cel mai des sunt întâlnite vulnerabilități în cereri *SELECT*.

Pot fi aplicate următoarele metode pentru a realiza SQL Injection:

• *operatorul SQL Union*, permite reunirea a două interogări într-un singur rezultat fiind utilizat în operatorul SELECT;

• *boolean*, utilizarea condițiilor logice pentru a verifica dacă careva condiții sunt adevărate sau false;

• *erori*, metoda obligă BD să genereze o eroare, ceea ce poate oferi atacatorului informații suplimentare pentru a perfecta interogarea pentru atac;

• *timpul*, utilizarea comenzilor BD pentru a reține răspunsurile în interogări.

5.2.2. Recomandări de prevenire SQL Injection

✓ Securizarea cu atenție a datelor introduse de utilizator. Orice date primite de la un utilizator trebuie considerate nesigure. Eliminarea tuturor ghilimelelor simple și duble poate preveni cele mai multe tipuri de acest tip de atac.

✓ Scrie toate interogările SQL folosind interogări parametrizate, astfel încât interogarea este predefinită și datele introduse sunt verificate și transmise ulterior.

✓ Limitează privilegiile userului pe care îl folosești la conectare la minimum posibil.

✓ Verifică dacă input-urile de la utilizatori sunt ceea ce te aștepți să fie. Folosește expresii regulate (*regexp*).

✓ Transformă orice date introduse de utilizator în tipul pe care îl aștepți (în cazul în care în cererea SQL se utilizează date numerice primite de la utilizatori, înainte de a le plasa în cererea SQL acestea ar trebui aduse la tipul numeric: *$id=(int)$id;* în cazul în care în cererea SQL se utilizează date de tip șir de caractere primite de la utilizatori, înainte de a le plasa în cererea SQL acestea trebuie prelucrate la simboluri speciale, cea mai bună practică fiind formarea expresiilor regulate).

✓ Limitează numărul de caractere în input-urile userilor; deși nu va împiedica toate atacurile SQL, va oprii o mică parte din ele.

✓ Printarea pe ecran a mesajelor de eroare SQL nu este recomandată (există riscul ca atacatorul să afle structura BD din aceste mesaje).

✓ O modalitate de a preveni injectările este de a scapă de caracterele care au o semnificație specială în SQL. Manualul pentru SGBD SQL explică ce caractere au o semnificație specială, ceea ce permite crearea unei liste negre de caractere ce au nevoie de traducere. De exemplu, fiecare apariție a unei ghilimele simple **‘** într-un parametru trebuie înlocuită cu doua ghilimele simple **‘’** pentru a forma un șir valid SQL.

- în majoritatea cazurilor injectarea codului SQL se efectuează cu ajutorul ghilimelelor atât simple, cât și duble (‘, “, `);
- cu ajutorul simbolurilor de comentarii specifice SGBD (/*, --) poate fi omisă o parte din interogare;
- simbolurile ce împart instrucțiunile SQL (;) permit formarea mai multor cereri la BD;
- trebuie sa fie verificate datele introduse de utilizator și la prezența altor simboluri, precum _, %, *.

De menționat că vulnerabilitățile de tipul SQL Injection sunt foarte răspândite, deoarece permit răufăcătorului să afle/extragă informații despre server, să obțină un interpretator de comenzi sau chiar să realizeze un atac DoS.

Pentru a evita prezența acestor vulnerabilități este necesar de a prelucra la simboluri speciale absolut toate datele ce parvin de la utilizatori. În această categorie intră parametrii GET, POST și chiar cookie. Fiecare SGBD are nuanțele sale pe care răufăcătorul potențial poate să le folosească în interesul său.

Surse bibliografice:

1. A. Watt, N. Eng. *Database Design*, 2014.
2. Microsoft SQL Docs, Vulnerability assessment for SQL Server2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-vulnerability-assessment?view=sql-server-2017 [Accesat: 28.12.2022]

VI. METODE DE SIGURANȚĂ A DATELOR. RISCURI ȘI PROCEDURI PENTRU RESTABILIREA DATELOR

Una dintre provocările la care trebuie să răspundă întreprinderile este creșterea volumului de date care necesită să fie salvate și/sau recuperate în cazul unui incident provocat din cauze interne sau externe. Este cunoscut faptul că planificarea continuității activității unei întreprinderi, în caz de dezastru, este o activitate vitală. Înainte de crearea planului de continuitate, este esențial să se ia în considerare efectele potențiale ale dezastrelor și riscurile asociate. Planul în sine trebuie să fie întreținut, testat și verificat, pentru a avea certitudinea că acesta rămâne adecvat nevoilor întreprinderii. Disponibilitatea datelor trebuie să fie cât mai ridicată și să asigure continuitatea desfășurării afacerii, chiar dacă datele din producție sunt deteriorate sau pierdute.

Pentru orice plan de restabilire în caz de dezastru trebuie să fie precizate cele două valori de referință:

1. *RPO - Recovery Point Objective*

- reprezintă cantitatea maximă de informație care este acceptabil să fie pierdută, exprimată în unități de timp (dacă pentru BD se face o copie la ora 10:00, din două în două ore este acceptabilă pierderea informației acumulate în decurs de 2 ore);

2. *RTO - Recovery Time Objective*

- reprezintă intervalul maxim de timp scurs de la declanșarea incidentului până la momentul în care sistemul redevine funcțional (din momentul în care cade BD în cât timp trebuie să fie ridicată);
- dacă acest obiectiv este depășit, activitatea începe să fie grav afectată.

6.1. Copii de rezervă ale bazei de date

Un rol important în activitatea unui administrator BD este elaborarea și implementarea unei strategii care să permită salvarea, arhivarea și restabilirea cât mai eficientă a datelor.

O strategie bună de creare a copiilor de rezervă (backup) și de restabilire (restore) a BD poate preveni:

- pierderea de date în cazul în care apare imposibilitatea accesării BD în urma coruperii unui fișier sau a unui grup de fișiere;
- pierderea anumitor funcționalități specifice în cazul trecerii la versiuni noi ale aplicațiilor care folosesc BD sau al unei versiuni noi de BD;

- pierderea de date în cazul următoarelor operațiuni: import de date, funcționalități noi apărute în aplicație, greșeli de operare ale utilizatorilor;
- pierderea BD în cazul în care serverul este indisponibil din cauza anumitor factori externi: întreruperi de curent, incendii, inundații.

6.1.1. Elaborarea unui plan de creare a copiilor de rezervă și de restabilire a bazei de date

În acest scop se specifică care dintre BD este necesar să se păstreze în siguranță, cât de des trebuie realizate copiile de rezervă/siguranță etc. La crearea unui astfel de plan se iau în considerare următoarele:

1. ***Tipul de BD.*** BD *sistem* și *utilizator* au, de obicei, cerințe diverse atât la realizarea copiei de rezervă, cât și la restabilire. De exemplu, BD *master* este esențială pentru efectuarea tuturor operațiilor în SQL Server. În caz că se defectează sau devine indisponibilă, întregul server nu va funcționa. Însă nu este nevoie să se realizeze copii de rezervă ale acestei BD la fiecare oră sau jumătate de oră, așa cum se procedează în cazul BD care gestionează tranzacții complicate ale clienților în timp real. Copia de rezervă a BD master se realizează (de obicei, o dată în săptămână - backup complet) doar atunci când se creează o BD nouă, când se creează conturi noi de utilizator sau se efectuează operații care produc schimbări majore în BD.

2. ***Importanța BD.*** Importanța BD conduce la alegerea tipului de copie de rezervă.

3. ***Periodicitatea efectuării modificărilor într-o BD.*** O BD care poate fi doar vizualizată necesită mai puține copii de rezervă decât o BD care poate fi modificată.

4. ***Rapiditatea refacerii datelor.*** Copiile de rezervă pentru BD mari (care au modificări masive) se vor realiza incremental într-un interval de timp mai mic. De exemplu, la fiecare oră, astfel încât, în cazul unei pene, este necesară restabilirea unei cantități mai mici de informație și, respectiv, mai repede are loc restabilirea BD.

5. ***Existența unui echipament adecvat pentru efectuarea copiilor de rezervă.*** Hardware-ul necesar pentru realizarea copiilor de rezervă poate conține: unități de stocare optică, discuri de stocare externă, HDD, SSD.

6. ***Planificarea executării copiilor de rezervă.*** De obicei, executarea copiilor de rezervă se realizează atunci când încărcarea sistemului este cea mai scăzută, ceea ce va conduce la creșterea rapidității realizării copiilor de rezervă.

7. ***Stocarea copiilor de rezervă în alte locații decât cele de producție*.** O astfel de politică este de dorit pentru a evita pierderea datelor în cazul unor dezastre naturale (serverele cu datele BD pot fi stocate în alt oraș, țară).

O copie de rezervă (un backup) este o copie a datelor din BD care pot fi folosite pentru a restabili datele respective. Backup-urile pot fi divizate în:

- *copii de rezervă fizice* − sunt copii de rezervă ale fișierelor fizice utilizate pentru stocarea și restabilirea BD, cum ar fi fișierele de date, fișierele de control și jurnalele arhivate (loguri), care stochează informațiile BD în altă locație, fie pe disc, fie pe o unitate de stocare offline;

- *copii de rezervă logice* − conțin date logice (de exemplu, tabele sau proceduri stocate) exportate dintr-o BD cu un utilit de export și stocat într-un fișier binar.

Restabilirea fizică reprezintă fundamentul oricărei strategii de salvare și de restabilire a datelor. Copiile de rezervă logice reprezintă un supliment util pentru copiile de rezervă fizice în multe situații, dar nu sunt suficiente pentru protecția împotriva pierderii datelor fără copiile de rezervă fizice. Pentru a face o copie de rezervă a unei părți sau a întregii BD, este necesar de făcut o copie fizică.

Tehnologiile care trebuie luate în calcul la fundamentarea unei strategii de backup sunt:

- *On-Premise*, salvarea datelor se face pe echipamente aflate local sau la distanță (offsite). Trebuie considerate costurile din cauza volumelor tot mai mari necesare de achiziționare/gestionare, scalabilitatea echipamentelor/ istemelor, întreținerea/mentenanța software și hardware, precum și costurile cu energia electrică;

- *Cloud Backup*, salvarea și, respectiv, arhivarea se face direct în cloud în mod automat și programat. Aceste servicii ar trebui să funcționeze în background, existând posibilitatea ca spațiul de stocare din cloud să reprezinte o extensie a celui situat On-Premise, cele două fiind gestionate împreună. Este de cele mai multe ori mai avantajos din punct de vedere economic decât backup-ul On-Premise.

Pentru a restabili datele, în funcție de tipul de spațiu de stocare și locația sa, multe întreprinderi le stochează offsite sau în Cloud, din simplul motiv că restabilirea durează mult mai puțin decât în cazul unei copii de rezervă On-Premise. Cu cât timpul de restabilire a datelor și/sau readucerea activității la un punct anterior este mai redus, cu atât costurile cresc.

Transpunerea în practică a strategiei și adaptarea acesteia cerințelor de business (cum ar fi automatizarea procesului de backup sau a procesului de

recuperare a datelor folosind un centru de date secundar) necesită testarea și verificarea corectitudinii proceselor de backup. Pentru a avea un backup 100% sigur și disponibil oricând, este necesară verificarea automată a acestuia, astfel încât să se asigure că orice bloc de date salvat este o replică exactă a originalului, chiar dacă acest lucru presupune ridicarea costurilor. Testarea proceselor presupune simularea unor situații de downtime chiar în condițiile în care, pe durata testelor, datele sunt predispuse la un anumit grad de risc. Având o strategie doar teoretică, neverificată în mediul real, în momentul apariției unui incident întreprinderea se poate confrunta cu o situație inedită, iar modalitatea de reacție în urma unui incident poate fi uneori deficitară.

6.1.2. Tipuri de copii de rezervă

O copie de rezervă a BD poate fi realizată atât în cazul unei BD pornite, cât și în cazul în care este oprită. Fiecare situație își are avantajele și dezavantajele sale.

1. Copie de rezervă cu BD pornită:

- este recomandat să fie folosită în mediul de producție, asigură o pierdere de date cât mai mică;
- nu necesită oprirea BD, utilizatorii nu sunt afectați de această operațiune;
- trebuie monitorizat spațiul disponibil pe disc; în cazul în care spațiul rezervat pentru operațiunea de backup nu este suficient de mare, umplerea acestuia poate duce la oprirea serverului.

2. Copie de rezervă cu BD oprită:

- BD trebuie să fie oprită;
- utilizatorii nu se pot conecta la BD;
- toate conexiunile active sunt întrerupte în timpul acestei operațiuni.

Există o gamă largă de metode de creare a copiilor de rezervă, cum sunt *complet, incremental, diferențial* (și opțional complet sintetic, incremental definitiv sau incremental inversat), toate având avantaje și dezavantaje. Cele mai cunoscute metode de protejare a datelor rămân protecția în timp real, replicarea și realizarea de instanțe în timp real. Totuși, deși există multe soluții inovative, rămâne foarte utilizată soluția prin care datele sunt pur și simplu copiate la anumite intervale de timp în alte locații decât cea principală (se poate realiza și o oglindă a BD în timp real pe alt disc).

Backup-ul complet este tehnica fundamentală de salvare a datelor, unde acestea sunt copiate în totalitate, inclusiv obiectele BD, tabelele de sistem și

date, inclusiv porțiuni din jurnalul tranzacțiilor necesare procesului de refacere, într-un set de date specificat (ca fâșiere de date). Este modul de backup care necesită cel mai mult timp de stocare a datelor, dar care necesită și o capacitate mare de stocare. Avantajul acestei tehnici este restaurarea rapidă și ușoară a datelor dintr-o copie realizată prin backup complet.

Backup-ul incremental presupune existența unui backup complet realizat, de exemplu, o dată pe săptămână, ulterior modul incremental realizând doar copierea datelor schimbate după momentul backup-ului complet. Avantajul este consumul mic de spațiu și timp, dar există dezavantajul unei complexități a restabilirii datelor salvate în caz de necesitate: faptul că trebuie unite toate copiile incrementale cu copia completă existentă.

Exemplul 6.1.1. Fie că este creat un backup complet duminica. Luni se păstrează modificările de luni, marți doar cele de marți ș.a.m.d. Astfel se obțin fișiere pentru fiecare zi. Duminica următoare se face alt backup complet și cele din timpul săptămânii se șterg. Pentru restabilirea BD deteriorate miercuri, până a fi creat backup-ul incremental de miercuri, se restabilește backupul complet, apoi incrementul de luni și de marți.

Backup-ul diferențial este asemănător cu cel incremental, realizând copii zilnice ale datelor schimbate după ziua realizării unui backup complet. Diferența față de cel incremental este că datele sunt actualizate zilnic, existând în caz de necesitate completarea backup-ului complet cu un singur backup diferențial. Ca avantaj se menționează rapiditatea cu care se pot completa cele două tipuri, dar timpul și volumul de lucru pot fi crescute față de modul incremental.

Exemplul 6.1.2. Fie că este creat un backup complet duminica. Luni se păstreează modificările de luni, marți la modificările de luni se adaogă cele de marți ș.a.m.d. și se obține un fișier. Duminica se face alt backup complet și cel diferențial din timpul săptămânii se șterge. Pentru restabilirea BD care a fost deteriorată miercuri, până a fi creat backup-ul diferențial de miercuri, se restabilește backupul complet, apoi cel diferențial din timpul săptămânii obținut marți.

Backup-ul jurnalului tranzacțiilor. Jurnalul tranzacțiilor reprezintă înregistrări seriale ale tuturor modificărilor efectuate în BD și se folosește atunci când se execută operații de restabilire a datelor pentru a încheia o tranzacție sau pentru a o anula.

Jurnalele de tranzacții sunt esențiale pe parcursul restabilirii unei BD în SQL Server. Spre deosebire de BD la care se pot face copii de rezervă

complete sau diferențiale, copiile de rezervă ale jurnalelor de tranzacții sunt, de obicei, incrementale, ceea ce înseamnă că fiecare copie a jurnalului unei tranzacții are o înregistrare a tranzacțiilor doar într-un anumit interval de timp. Jurnalele tranzacțiilor sunt întotdeauna aplicate secvențial începând după momentul de încheiere a ultimei copii de siguranță complete sau diferențiale a tranzacției precedente. De asemenea, dacă se dorește restabilirea unei BD, trebuie folosită fiecare secvență după momentul defectării.

Exemplul 6.1.3. Dacă se face o copie de rezervă completă la ora 1:00, iar BD se defectează la ora 1:46, atunci BD este restabilită pe baza ultimei copii de rezervă complete, după care se folosesc copiile de rezervă ale jurnalelor create la 1:15, 1:30 și 1:45. Deci, dacă nu s-ar folosi copiile de rezervă secvențiale ale jurnalelor de tranzacții, s-ar pierde toate tranzacțiile efectuate după ora 1, când s-a făcut ultima copie de rezervă completă a BD. La restabilirea unei BD, SQL Server oferă anumite opțiuni care preîntâmpină utilizarea secvențială a jurnalelor de tranzacții. Momentul de încheiere a ultimei copii de rezervă complete sau diferențiale marchează momentul începerii utilizării secvenței de copii ale jurnalelor de tranzacții. Dacă se invalidează folosirea secvenței de copii de rezervă ale jurnalelor de tranzacții, trebuie efectuată o copie completă sau diferențială a BD pentru a începe o nouă secvență.

Backup-ul fișierelor și grupurilor de fișiere. În acest caz se copiază doar fișierele BD și nu toată BD. Varianta este utilă atunci când se lucrează cu BD de dimensiuni foarte mari. În acest caz trebuie însă realizată o copie de rezervă și a jurnalului de tranzacții (dacă datele sunt multe, atunci pot fi stocate fiind grupate după ani sau luni și în acest caz se fac copii de rezervă doar pentru date dintr-o anumită perioadă).

6.1.3. Tipuri de modele de restabilire a bazei de date

1. ***Simplu.*** Este destinat BD care trebuie restabilite de la momentul ultimei copii de rezervă. Strategia de realizare a copiei de rezervă în cazul acestui model presupune realizarea de copii de rezervă complete și diferențiale. Nu se realizează copii de rezervă ale jurnalului de tranzacții.

2. ***Complet.*** Se recomandă pentru BD care trebuie restabilite de la momentul ultimei defectări sau de la un anumit moment de timp. Se jurnalizează toate operațiile, inclusiv cele de încărcare cu date. Strategia va include efectuarea de copii de rezervă complete, diferențiale și ale jurnalului de tranzacții sau doar a jurnalului de tranzacții.

3. ***Jurnalizarea încărcării cu date.*** Acest model reduce spațiul ocupat de utilizarea jurnalului. Operațiile de încărcare cu date sunt jurnalizate la minim și este nevoie de o reluare a operațiilor de încărcare dacă BD se defectează înainte de a efectua o copie de rezervă completă sau diferențială.

Fiecare BD poate avea propriul model de restabilire. Implicit, BD *master*, *msdb* şi *tempdb* folosesc modelul simplu, iar BD *model* (care reprezintă un șablon al tuturor BD noi) folosește modelul complet de restabilire.

6.2. Coruperea bazei de date

Pentru a evita coruperea/deteriorarea BD ar trebui ca administratorul BD să asigure o exploatare optimă a acesteia, ceea ce presupune îndeplinirea următoarelor atribuții:

Să asigure buna funcționare a acesteia, mai exact:

- încărcare optimă a resurselor pe server: procesorul să nu fie ocupat 80 - 90% din timp, memoria să nu fie ocupată peste 90%, să fie suficient spațiu, astfel încât să permită creșterea BD cu cel puțin 25%;
- un timp cât mai bun de răspuns al BD pentru activitățile curente: introducere de documente, rapoarte financiare, balanțe contabile etc.;
- diagnosticarea și rezolvarea cât mai rapidă a incidentelor apărute în BD, astfel încât utilizatorii să fie afectați cât mai puțin.

Să creeze și să implementeze o strategie de backup cât mai robustă:

- să repună în funcțiune cât mai repede și cu pierderi minime BD în caz de corupere de fișiere, ștergere accidentală de date, pierderea BD etc.;
- să programeze execuția job-urilor de backup și mentenanță a BD, astfel încăt utilizatorii să fie cât mai puțin afectați de execuția acestora;
- să se asigure, în mod recurent, că backup-ul BD poate fi restabilit fără erori pe un mediu de test;
- să se asigure că mediul de stocare este unul sigur, este asigurată redundanța datelor și are disponibil un spațiu suficient.

Să verifice periodic dacă în logurile (tot ce se face cu BD, toate operațiile) BD nu au apărut incidente sau erori:

- dacă infrastructura permite, să creeze notificări pe mail în cazul în care apar erori la nivelul BD;
- să verifice periodic logul BD (ce evenimente au avut loc, de exemplu: nu este accesibil un disc);

- în cazul în care apar mesaje de eroare, să trimită aceste incidente către *suport* și să urmeze indicațiile primite.

Să monitorizeze în mod constant indicii de performanță ai BD:

- să monitorizeze alertele primite (de exemplu, referitor la spațiul disponibil pentru execuția backup-urilor);
- să anticipeze pe cât posibil creșterea BD și să prevină situațiile în care serverul poate fi afectat de lipsa de resurse: spațiul pe disc, memorie alocată, încărcarea procesorului sau influența altor aplicații asupra BD;
- să creeze și să urmărească în mod constant execuția job-urilor de mentenanță a BD, cum sunt jobul de actualizare a statisticilor, defragmentarea BD, gradul de încărcare a fișierelor BD.

Pot fi formulate următoarele măsuri preventive care trebuie urmate pentru a evita coruperea BD:

1. ***Urmărirea activității pe BD și gradul de încărcare al acesteia***, identificarea în timp real a blocajelor care pot apărea și eliminarea cauzelor.
2. ***Anticiparea creșterii BD***, alocarea din timp a spațiului necesar, astfel încât să nu apară blocaje la nivel de BD din acest motiv.
3. ***Actualizarea statisticilor*** este efectuată numai în cazul în care volumul de date crește foarte mult într-un timp foarte scurt sau acestea nu mai sunt actuale. Nu se justifică actualizarea zilnică a statisticilor pentru o BD cu o creștere moderată a volumul de date. În acest caz este o bună practică actualizarea săptămânală a acestora.
4. ***Crearea de indecși în funcție de specificul fiecărei BD.*** Teoretic este bine ca o BD să aibă câte un index pe fiecare *foreign key*. Aceasta nu este o regulă generală, introducerea de indecși trebuie făcută treptat și verificat impactul pe care îl au aceștia asupra BD.
5. ***Separarea datelor în funcție de importanță și modul în care sunt accesate.*** Dacă o BD folosește un tabel de dimensiuni foarte mari, ideal ar fi stocarea acestui tabel pe un *RAID* (a se vedea *Anexa 1*) separat de celelalte date. Teoretic, punerea indecșilor pe un mediu de stocare separat asigură un timp de răspuns mai bun al BD.

6.3. Tehnici de restabilre a bazei de date

Realizarea unei strategii de creare a copiilor de rezervă și/sau restabilire în caz de dezastru impune o analiză atentă a balanței costuri/beneficii pentru întreprinderea care necesită astfel de servicii. Restabilirea BD este procesul de revenire a BD într-o stare corectă după apariţia unei pene (căderi a BD).

Cauzele penelor sunt: căderile sistemului (hard sau soft), pene de mediu (distrugerea mediului de depozitare a datelor), erorile soft de aplicaţie, dezastre naturale, neglijenţa și sabotajul.

Indiferent de cauză, există două efecte principale:

1) pierderea memoriei principale, inclusiv a bufferelor BD;
2) pierderea copiei de pe disc a BD.

Sunt posibile două situaţii care depind de gradul de deteriorare a BD:

- dacă BD a fost deteriorată fizic (căderea discului pe care este stocată BD), este necesar să se restabilească ultima copie de rezervă şi să se aplice din nou operaţiile de reactualizare a tranzacţiilor efectuate, folosind jurnalul de tranzacții. Se recomandă ca jurnalul să fie stocat pe un disc separat, pentru a reduce riscul deteriorării lui simultan cu deteriorarea BD;

- dacă BD nu a fost deteriorată fizic, ci a devenit incoerentă din cauza unei căderi a sistemului în timpul execuţiei unei tranzacţii, nu este necesară utilizarea copiei de siguranţă, ci poate fi restabilită utilizând imaginile anterioare şi ulterioare conţinute în jurnalul tranzacțiilor.

Tehnici de restabilire a bazei de date:

1. ***Tehnica de restabilire cu ajutorul reactualizării amânate.*** Prin utilizarea acestui protocol, reactualizările nu sunt scrise în BD decât după ce tranzacţia a ajuns pe punctul de a fi efectuată. Dacă tranzacţia eşuează înainte de a ajunge în acest punct, ea nu modifică BD şi astfel nu mai sunt necesare anulări sau modificări. S-ar putea să fie nevoie doar să se reia reactualizările tranzacţiilor efectuate, deoarece este posibil ca efectul acestora să nu fi ajuns la BD. Pentru aceasta se foloseşte jurnalul de tranzacţii.

2. ***Tehnica de restabilire cu ajutorul reactualizării imediate***. Prin utilizarea acestui protocol, reactualizările sunt aplicate pe BD imediat ce au loc, fără a aştepta ca tranzacţia să ajungă pe punctul de a fi efectuată. Pe lângă reluarea reactualizărilor efectuate, acum ar putea să fie necesar să se anuleze efectele tranzacţiilor care nu au fost efectuate în momentul în care a survenit pana. Pentru aceasta se foloseşte jurnalul de tranzacţii. Este esenţial ca înregistrările din jurnal să fie scrise înainte de descrierea corespunzătoare în BD. Această operaţie e cunoscută sub denumirea de protocol de scriere în avans în jurnal.

3. ***Paginarea cu umbră.*** Este o alternativă a tehnicilor care folosesc jurnalul de tranzacţii. În cadrul acestei tehnici, în timpul unei tranzacţii se păstrează două tabele ale paginii: unul pentru pagina curentă și unul pentru pagina din umbră. La începutul tranzacţiei, cele două pagini sunt identice. Pagina din umbră nu se modifică niciodată pe parcursul tranzacţiei, în timp ce

pagina curentă reflectă toate reactualizările făcute pe BD de către tranzacţia respectivă. La încheierea tranzacţiei, tabelul paginii curente devine tabelul paginii din umbră.

Folosirea tehnicilor la restabilire a unei BD depinde de opțiunile folosite la crearea copiei de rezervă (completă, diferențială, jurnalizare și fișiere) și de starea BD.

Surse bibliografice:

1. A. Basta, M. Zgola. *Database Security*. Cengage Learning, 2011.
2. Microsoft SQL Docs, Restore a Database Backup Using SSMS 2021 https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-a-database-backup-using-ssms?view=sql-server-ver15 [Accesat: 28.12.2022]

VII. PROBLEMA INTERFERENȚEI ȘI CONCURENȚEI ÎN BAZE DE DATE

7.1. Tranzacții și proprietăți ACID

Un obiectiv major al SGBD este de a permite mai multor utilizatori să acceseze concurent datele partajate. Dar la sistemele în care o BD este accesată simultan de mai mulţi utilizatori apar situaţii de conflict condiționate de accesul concurent la datele care constituie resursă comună. Modul de rezolvare a conflictului depinde de natura cererilor de acces la date:

1. Dacă cererile de acces sunt de tip ***vizualizare (regăsire)***, atunci secvențialitatea accesului la mediul de memorare este suficientă şi nu mai este nevoie de precauţii suplimentare. Atât timp cât niciunul nu face modificări ale datelor, nu are importanţă ordinea în care se asigură accesul utilizatorului la date. Deci, în acest caz, rezolvarea situaţiilor conflictuale cauzate de operaţiile de citire simultană poate fi lăsată în seama sistemului de operare, care stabileşte ordinea de satisfacere a cererilor după criteriul asigurării concurenţei maxime de operare, minimizând timpul global de satisfacere a acestor cereri.

2. Dacă unele cereri sunt de tip ***actualizare (se modifică datele)***, este necesară aplicarea unor strategii adecvate de tratare a cererilor de acces. Exemplu tipic sunt sistemele de rezervare a locurilor în hotel sau a biletelor pentru avion, activitate în care mai mulţi agenţi fac permanent modificări. În acest caz există pericolul că două procese, care citesc şi modifică valoarea aceluiaşi obiect, ar putea interacţiona atunci când sunt executate simultan. Astfel, rezultă necesitatea impunerii unor restricţii asupra execuţiei concurente a proceselor care fac operaţii de citire şi modificare a datelor. O rezolvare imediată şi simplă ar fi blocarea BD pe durata rezolvării unei cereri. Dar asta echivalează cu blocarea concurenţei şi degradarea performanţelor sistemului, făcându-l uneori neutilizabil.

Definiția 7.1. Tranzacția reprezintă una sau mai multe instrucțiuni SQL care compun o singură unitate logică de lucru.

Definiția 7.2. O tranzacţie este o unitate logică de prelucrare indivizibilă (atomică) a datelor unei BD prin care se asigură consistenţa acesteia.

O tranzacție poate fi finalizată (încheiată) în două moduri:

✓ *tranzacția s-a finalizat cu succes* (atunci spunem că tranzacția a făcut 'commit') și în acest caz efectele tranzacției devin permanente în BD (BD este consistentă, coerentă);

✓ *tranzacția nu s-a finalizat cu succes*, atunci efectele tranzacției sunt eliminate din BD (în acest caz tranzacția face 'rollback', ceea ce presupune că este derulată înapoi) și tranzacția se încheie. Eliminarea efectelor unei tranzacții se numește *derularea înapoi a tranzacției*.

Pentru a delimita tranzacţiile, în majoritatea limbajelor de manipulare a datelor sunt disponibile instrucţiunile *BEGIN TRANSACTION, COMMIT* şi *ROLLBACK*. Dacă utilizatorul nu foloseşte aceste delimitări, de obicei întregul program este considerat ca reprezentând o singură tranzacţie, iar SGBD-ul execută automat o instrucţiune *COMMIT* atunci când programul este încheiat corect, sau una ROLLBACK în caz contrar. Deci, avem:

- *BEGIN TRANSACTION* – începe o nouă tranzacţie;
- *COMMIT* – salvează orice modificări şi încheie tranzacţia curentă;
- *ROLLBACK* – anulează orice modificări făcute în timpul tranzacţei curente şi încheie tranzacţia.

Dacă este utilizată instrucţiunea *BEGIN TRANSACTION*, atunci orice actualizare a datelor nu va fi executată instantaneu, ci numai după *COMMIT* sau *ROLLBACK* pentru a încheia tranzacţia.

Construcția tranzacțiilor trebuie să respecte proprietățile ***ACID***:

1. **Atomicitate** *este proprietatea 'totul sau nimic'. O tranzacție este o unitate indivizibilă care se execută în întregime sau deloc*. Dacă o tranzacţie este întreruptă dintr-o cauză oarecare, atunci SGBD-ul va asigura fie completarea şi validarea tranzacţiei, fie abandonarea tranzacţiei şi anularea tuturor efectelor acţiunilor efectuate de tranzacţie până în momentul întreruperii, după ce au fost eliminate cauzele care au întrerupt executarea tranzacţiei.

2. **Consistența** *presupune că o tranzacție trebuie să transforme BD dintr-o formă consistentă într-o altă formă, tot consistentă*. În stările intermediare prin care trece o BD în cursul unei tranzacţii, este posibil să existe unele inconsistenţe, dar starea finală în care ajunge BD după execuţia unei tranzacţii trebuie să fie consistentă. Starea unei BD este consistentă dacă respectă toate constrângerile de integritate implicite sau explicite. SGBD-ul nu verifică consistenţa BD după fiecare operaţie din tranzacţie, ci este sarcina proiectanţilor aplicaţiilor de BD ca operaţiile prevăzute în tranzacţii să producă o stare consistentă a BD. De exemplu, o coloană conține adevăr sau

fals (0 sau 1), dacă un utilizator încearcă să introducă o altă valoare, atunci regula de consistență nu permite acest lucru și se întoarce la starea inițială.

3. **Independența** *presupune că o tranzacţie se execută independent de oricare alta, adică efectele parțiale ale unei tranzacții incomplete nu trebuie să influențeze o altă tranzacție*. Dacă în același timp sunt executate mai multe tranzacţii concurente, acestea nu „văd" modificările parţiale efectuate de tranzacţia care a început modificările până în momentul validării tranzacţiei (până la *comit*). Proprietatea de independență a tranzacţiilor este importantă, deoarece elimină fenomenul de abandonare în cascadă a tranzacţiilor. Dacă rezultatele parţiale ale unei tranzacţii sunt vizibile altor tranzacţii înainte de validarea acesteia şi dacă se întâmplă ca această tranzacţie să fie abandonată şi anulată (*rollback*), atunci toate tranzacţiile care au accesat rezultatele parţiale ale acesteia vor trebui să fie anulate; aceste operaţii de anulare pot produce, la rândul lor, alte anulări ș.a.m.d. Acest fenomen de anulare în cascadă creează un efect de „domino". Independența este impusă prin metode de control al concurenţei, pe diferite niveluri de independență (achitarea facturii cu câteva poziții: apă, curățenie, ascensor ș.a. și achitarea unui produs cumpărat pe net).

4. **Durabilitate** *presupune că efectele unei tranzacții finalizate cu succes sunt definitiv înregistrate în BD*. Este proprietatea prin care, după validarea unei tranzacţii, modificările efectuate de aceasta în BD nu mai pot fi pierdute ca urmare a unor defectări ulterioare ale sistemului. Proprietatea de durabilitate este asigurată prin metode de restabilire (recovery) ale SGBD (dacă a fost *comit,* chiar dacă s-a stins lumina, această tranzacție deja nu mai poate fi anulată).

Pentru restabilirea BD în cazul când pot să apară inconsistențe în general, majoritatea BD sunt copiate periodic pe medii magnetice care se pastrează în locuri sigure. De asemenea, se ține o evidență a tuturor transformărilor efectuate în BD de când s-a efectuat ultima copie. Fișierul care conține aceste modificări se numește *jurnal*. Fiecare înregistrare din jurnal conține un identificator al programului care a făcut modificarea, valoarea veche și noua valoare introdusă pentru un element. Tot în jurnal se mai păstrează diferite momente importante din desfășurarea programelor (început, sfârșit, finalizarea unor operații etc.). Fişierul jurnal este memorat pe disc şi nu este afectat de diferite erori de execuţie, cu excepţia unei defectări catastrofice a discului. În plus, fişierul jurnal este salvat periodic pe un suport auxiliar ca o măsură de prevedere pentru astfel de defecte catastrofice. La finalizarea unei tranzacții, indiferent dacă ea s-a încheiat normal sau prin

eroare, BD trebuie să aibă același grad de integritate ca la momentul începerii execuției tranzacției respective.

7.2. Anomalii de interferență

Interacţiunea necontrolată a două sau mai multor tranzacţii poate duce la apariţia unor stări inconsistente ale BD şi la producerea unor rezultate eronate. Două tranzacţii Ti şi Tj sunt susceptibile (supuse, influențate) de interferenţă dacă rezultatul execuţiei lor concurente poate fi diferit de rezultatul execuţiei seriale. Între două tranzacţii poate să apară o interferenţă dacă acestea efectuează operaţii asupra unor date comune. Dacă aceste două tranzacţii sunt executate în mod concurent, atunci spunem că sunt conflictuale. Deci, două tranzacţii Ti şi Tj sunt conflictuale dacă sunt concurente şi susceptibile de interferenţă.

În funcţie de natura operaţiilor pe care le efectuează asupra datelor comune, între două tranzacţii pot să apară mai multe tipuri de interferenţe care provoacă anomalii de interferenţă:

a) ***anomalia de actualizare pierdută (lost updates)***, corespunde unui conflict de tip scriere-scriere şi constă în faptul că rezultatul actualizării efectuate de o tranzacţie se pierde ca urmare a reactualizării aceleiaşi date de către o altă tranzacţie, fără ca reactualizarea să fie influenţată de rezultatul primei actualizări;

b) ***anomalia de citire improprie (nonrepeatable read)***, corespunde unui conflict de tip scriere-citire şi apare atunci când o tranzacţie surprinde o stare temporar inconsistentă a BD. Această anomalie poate să apară atunci când una dintre tranzacţii este abandonată, iar altă tranzacţie concurentă a utilizat articolele modificate înainte de readucerea acestora la valoarea iniţială;

c) ***anomalia de citire irepetabilă (dity read)***, corespunde unui conflict de tip citire-scriere şi apare atunci când aceeaşi tranzacţie găseşte valori diferite la citiri repetate ale aceleiaşi date. Poate să apară dacă o tranzacţie T citeşte un articol de două ori, iar între cele două citiri o altă tranzacţie T_1 a modificat chiar acel articol. În această situaţie tranzacţia T primeşte două valori diferite ale aceluiaşi articol;

d) ***anomalia de citire fantomă (fantom read)***, apare atunci când o tranzacţie prelucrează un set de înregistrări ca rezultat al unei interogări; dacă în timpul acestei prelucrări o altă tranzacţie a inserat sau a şters o înregistrare care satisface condiţia interogării respective, atunci pot să apară sau să

dispară înregistrări din mulţimea de înregistrări rezultante iniţial, comportare asemănătoare cu aparţia sau dispariţia fantomelor.

Diferenţa dintre citirea fantomă şi citirea irepetabilă este aceea că citirea fantomă apare dacă tranzacţiile concurente modifică numărul de înregistrări utilizate de tranzacţia curentă (prin instrucţiuni INSERT sau DELETE), pe când citirea irepetabilă apare dacă tranzacţiile concurente modifică valorile din înregistrările existente şi prelucrate de tranzacţia curentă (prin instrucţiuni UPDATE). Astfel de anomalii, care pot să apară la execuţia concurentă necontrolată a două sau mai mai multor tranzacţii, evidenţiază necesitatea controlului concurenţei în BD. Consistenţa datelor memorate în BD trebuie să fie asigurată chiar în condiţiile în care o tranzacţie nu poate fi terminată cu succes, din cauza apariţiei unei erori de funcţionare.

SET TRANSACTION stabileşte proprietăţile tranzacţiilor şi admite câteva opţiuni de setare a modului de gestiune a tranzacţiilor.

Nivelul de izolare a tranzacţiilor (*isolation level*) cu valorile posibile:

- READ UNCOMMITTED;
- READ COMMITTED;
- REPETABLE READS;
- SERIALIZABLE;
- SNAPSHOT.

Nivelurile de izolare determină modul în care SGBD introduce diferite mecanisme de control al concurenţei (cel mai frecvent *locks* cu stări multiple). De exemplu, pe nivelul READ COMMITTED sunt prevăzute *locks* partajate pentru toate articolele citite, ceea ce împiedică apariţia citirilor improprii, dar aceste zăvoare sunt eliberate înainte de terminarea tranzacţiei şi, de aceea, pot rezulta citiri irepetabile şi citiri fantomă. Pe orice nivel de izolare, inclusiv pe cel mai slab (READ UNCOMMITTED), se folosesc mecanisme de control al concurenţei tranzacţiilor care previn pierderea actualizărilor. Astfel de anomalii sunt foarte grave, BD nu reflectă operaţiile care s-au efectuat asupra datelor şi nici nu există vreo posibilitate de refacere a acestor pierderi. De aceea nu este prevăzut niciun nivel de izolare care să permită pierderea actualizării datelor.

7.3. Blocări optimiste și pesimiste

Controlul execuţiei concurente a mai multor tranzacţii este necesar pentru a asigura proprietăţile de atomicitate, izolare şi durabilitate a tranzacţiilor şi, prin aceasta, consistenţa datelor memorate. Controlul

concurenţei poate fi realizat prin protocoale (set de reguli) impuse tranzacţiilor astfel încât, dacă acestea sunt respectate de fiecare tranzacţie, orice planificare în care astfel de tranzacţii participă este serializabilă şi, deci, corectă.

SGBD-urile folosesc mecanisme (algoritmi) ce asigură integritatea tranzacțiilor și mențin datele consistente când aceleași date sunt accesate de mai mulți utilizatori în același timp. Aceste mecanisme de control al concurenţei se împart în:

1) mecanisme de control prin *blocare*;
2) mecanisme de control prin *marcare*.

Protocoalele de control al concurenţei sunt implementate de SGBD astfel încât programatorii de aplicaţii nu operează în mod explicit cu blocări sau marcări, ci stabilesc opţiunile prin care SGBD adoptă anumite tehnici de control al concurenţei.

Există două tipuri mari de implementări ale controlului concurenței:

- metode cu control *optimist* asupra concurenţei: când o tranzacție efectuează modificări asupra datelor și dacă în același timp o altă tranzacție modifică aceleași date, atunci prima tranzacție face rollback;
- metode cu control *pesimist*, tranzacția care modifică datele blochează aceste date și alte tranzacții nu pot executa acțiuni asupra datelor până la finalizarea tranzacției începute.

Surse bibliografice:

1. K. Delaney. *SQL Server Concurrency: Locking, Blocking and Row Versioning*, RedGate. Simple Talk Publishing, 2012.
2. Microsoft SQL Docs, Transaction Isolation Levels, 2021 https://docs.microsoft.com/en-us/sql/t-sql/language-elements/transaction-isolation-levels?view=sql-server-ver15 [Accesat: 28.12.2022]

BIBILIOGRAFIE

a) Afyouni H. *Database Security and Auditing: Protecting Data Integrity and Accessibility*. Cengage Learning, 2006.
b) Basta A., Zgola M. *Database Security*. Cengage Learning, 2011.
c) Delaney K. *SQL Server Concurrency: Locking, Blocking and Row Versioning*. RedGate, Simple Talk Publishing, 2012.
d) Kyte T., Kuhn Darl. *Oracle Database Transactions and Locking Revealed*. Apress, 2014.
e) Natan R. *Implementing Database Security and Auditing*. Elsevier, 2005.
f) Thuraisingham B. *Database and Applications Security: Integrating Information Security and Data Management*. Auerbach Publications, 2005.
g) Watt A., Eng N. *Database Design*, 2014.
h) Microsoft SQL Docs, Permissions (Database Engine), 2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/permissions-database-engine?view=sql-server-ver15 [Accesat: 28.12.2021]
i) Microsoft SQL Docs, Create a server audit and database audit specification, 2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/auditing/ create-a-server-audit-and-database-audit-specification?view=sql-server-2017 [Accesat: 28.12.2021]
j) Microsoft SQL Docs, Transaction Isolation Levels, 2021 https://docs.microsoft.com/en-us/sql/t-sql/language-elements/transaction-isolation-levels?view=sql-server-ver15 [Accesat: 28.12.2021]
k) Microsoft SQL Docs, Restore a Database Backup Using SSMS 2021 https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-a-database-backup-using-ssms?view=sql-server-ver15 [Accesat: 28.12.2021]
l) Beginner SQL Tutorial, SQL Integrity Constraints https://beginner-sql-tutorial.com/sql-integrity-constraints.htm [Accesat: 28.12.2021]
m) Microsoft SQL Docs, SQL Server Audit (Database Engine) 2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/auditing/sql-server-audit-database-engine?view=sql-server-ver15 [Accesat: 28.12.2021].

n) Microsoft SQL Docs, Vulnerability assessment for SQL Server2021 https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-vulnerability-assessment?view=sql-server-2017 [Accesat: 28.12.2021]
o) CREATE TRIGGER (Transact-SQL) https://docs.microsoft.com/en-us/sql/t-sql/statements/ create-trigger-transact-sql?view=sql-server-ver15 [Accesat: 28.12.2021]

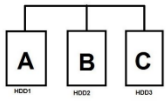
ANEXE

Anexa 1

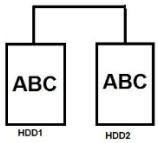
Tipuri de RAID

RAID (Redundant Array of Independent Disks sau Redundant Array of Inexpensive Disks) reprezintă o unire a mai multor HDD-uri (hard discuri) care are ca scop fie creșterea vitezelor scriere / citire, fie protecția datelor sau chiar combinația acestor două. Există mai multe tipuri de RAID. Cele mai folosite și cunoscute sunt RAID0, RAID1, RAID5 și RAID10.

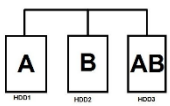
RAID0 este utilizat în scopul îmbunătățirii performanței și a vitezei de copiere și citire a datelor. Fie că avem o configurație RAID0 cu 3 HDD-uri și dorim să salvăm un fișier care este alcătuit din 3 părți (A, B și C). Salvarea se efectuează în felul următor: partea A a fișierului este salvată pe primul HDD, partea B – pe al doilea HDD, iar partea C – pe al treilea HDD. Scrierea/citirea pe mai multe HDD-uri se face simultan, mărindu-se performanța totală a sistemului. RAID0 este cea mai performantă și rapidă metodă de copiere, stocare și citire a datelor, însă defectarea unuia dintre cele 3 HDD-uri atrage după sine pierderea totală a informațiilor salvate.



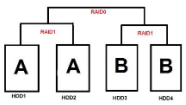
RAID1 se folosește când dorim o securitate sporită a datelor importante. Se folosesc 2 HDD-uri pentru o configurație de tip RAID1. Luăm același fișier cu 3 părți (A, B și C) ca și în exemplul de mai sus. Salvarea se face în felul următor: părțile A, B și C ale fișierului sunt salvate pe fiecare HDD în parte. Cu alte cuvinte, fișierul este duplicat, îl avem pe ambele HDD-uri în aceeași formă. Dacă unul dintre cele două HDD-uri se va defecta, atunci datele rămân în siguranță, deoarece fișierul există pe ambele HDD-uri. Dezavantajul este că spațiul de stocare este jumătate din suma capacității HDD-urilor.



RAID5 înglobează în același timp performanța ridicată și securitatea datelor în cazul deteriorării unui HDD. Ca să unim HDD-uri într-o configurație de tip RAID5, avem nevoie de minimum 3 HDD-uri. Fie că dorim să salvăm un fișier alcătuit din 2 părți (A și B). Partea A a fișierului este salvată pe primul HDD, partea B a fișierului este salvată pe al doilea HDD, iar pe al treilea HDD este salvat întregul fișier (A și B). Cu această configurație avem performanță și viteză ridicată, deoarece scrierea/citirea pe mai multe



HDD-uri se face simultan. Avem securitate, deoarece dacă se strică datele, oricare dintre HDD-uri rămâne accesibil.



RAID10 este o combinație între RAID0 și RAID1. Avem nevoie de minim 4 HDD-uri. RAID10 oferă citirea datelor după modelul RAID0, iar stocarea se face ca la RAID1. În acest fel beneficiem de viteza RAID0 și de siguranța RAID1. Dezavantajul este că putem folosi numai jumătate din capacitatea celor 4 HDD-uri.

Pentru îmbunătățirea vitezei sistemului cu o eventuală pierdere a datelor putem folosi RAID0. Pentru o securitate sporită a datelor fără creșterea vitezelor de scriere/citire, iar spațiul de stocare înjumătățit nu este un obstacol, se recomandă RAID1. Pentru utilizatorii avansați care doresc securitate sporită dar și performanțe crescute se recomandă RAID5, iar pentru cei mai exigenți cu un buget generos – RAID 10. Se pot crea combinații de mai multe RAID-uri între ele cu accentul pus fie pe siguranță, fie pe viteză.

Anexa 2

Instrucțiuni SQL

Tabelul 2.1. DDL (Data Definition Language): instrucțiuni de definire a datelor (permit descrierea structurii tabelelor)

Instrucțiune	Semnificație
CREATE	permite crearea tabelelor
ALTER	permite modificarea tabelelor
TRUNCATE	permite ștergerea conținutului unui tabel
DROP	permite ștergerea tabelelor
RENAME	permite modificarea denumirii unui tabel

Tabelul 2.2. DML (Data Manipulation Language): instrucțiuni de manipulare a datelor (permit modificarea conținutului tabelelor)

Instrucțiune	Semnificație
INSERT	permite adaugarea de noi înregistrari într-un tabel
UPDATE	permite actualizarea valorilor pentru înregistrarile dintr-un tabel
DELETE	permite stergerea înregistrarilor dintr-un tabel

Tabelul 2.3. DQL (Data Query Language): instrucțiuni de interogare a datelor

Instrucțiune	Semnificație
SELECT	permite regăsirea liniilor memorate în tabele

Tabelul 2.4. TCL (Transaction Control Language): instrucțiuni de procesare a tranzactiilor

Instrucțiune	Semnificație
SAVEPOINT	permite definirea unui punct de salvare, la care se poate reveni pentru a renunța la modificările făcute după acest punct asupra bazei de date
COMMIT	permite ca modificările făcute asupra bazei de date să devină permanente
ROLLBACK	permite renunțarea la anumite modificări făcute asupra bazei de date

Tabelul 2.5. DCL (Data Control Language): instrucțiuni pentru controlul datelor (permit definirea, modificarea și retragerea privilegiilor)

Instrucțiune	Semnificație
GRANT	permite acordarea de privilegii
REVOKE	permite retragerea privilegiilor

Anexa 3

Proceduri stocate

Tabelul 3.1. Proceduri stocate ce țin de securitatea bazelor de date

Procedura	Semnificație
sp_addlogin	crearea identificatorilor
sp_password	modificarea parolei
sp_helplogins	oferă un raport cu identificatorii creați pe server
sp_droplogins	permite ștergerea identificatorului
sp_grantlogin	permite acordarea permisiunilor de conectare
sp_revokelogin	permite ridicarea dreptului unui utilizator sau al unui grup de a se conecta la SQL Server
sp_denylogin	permite interzicerea dreptului unui utilizator sau al unui grup de a se conecta la SQL Server
sp_grantdbaccess	permite realizarea asocierii unui identificator cu un utilizator al bazei de date
sp_addalias	permite crearea unui alias
sp_revokedbaccess	ridică dreptul de acces la baza de date
sp_helpuser	oferă un raport cu utilizatorii creați în baza de date
sp_dropalias	permite renunțarea la alias
sp_changedbowner	permite modificarea proprietarului unei baze de date
sp_addrole	permite crearea unui rol
sp_droprole	permite înlăturarea unui rol dintr-o bază de date
sp_addrolemember	permite să fie adăugați utilizatori la rolurile fixe sau definite de utilizator
sp_droprolemember	permite eliminarea unui membru al unui rol

Tatiana PAȘA

SECURITATEA BAZELOR DE DATE

Note de curs

Redactare – *Ariadna Strungaru*
Asistență computerizată – *Maria Bondari*

Bun de tipar 5.05.2022. Formatul 70x100 $^{1}/_{12}$.
Coli de tipar 8,5. Coli editoriale 4,8.
Comanda 2. Tirajul 50 ex.

Centrul Editorial-Poligrafic al USM
str. Al. Mateevici, 60, Chişinău, MD 2009

100