Konceptimi dhe vënia në jetë e një shtrese sigurie në SMBD “Oracle”

Hakik PACI

Tiranë 2013
Udhëheqësi shkencor: Prof. Dr. Aleksandër Xhuvani

Juria

Prof. Dr. Fredi Premti, Kryetari, Juria e doktoratës
Prof. asc. Dr. Elinda Kajo, Anëtar, Juria e doktoratës
Prof. Dr. Kozeta Sevrani, Anëtar, Juria e doktoratës
Prof. Dr. Dritan Spahiu, Anëtar, Juria e doktoratës
Prof. Dr. Dhimiter Tole, Anëtar, Juria e doktoratës

Miratuar nga

Acad. Jorgaq KAÇANI, Rektori i UPT

_________________________________________ , Këshilli i Profesorëve, FTI
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABELA E PËRMBAJTJES</td>
<td></td>
</tr>
<tr>
<td>LISTA E FIGURAVE</td>
<td>VII</td>
</tr>
<tr>
<td>LISTA E TABELAVE</td>
<td>VIII</td>
</tr>
<tr>
<td>DEDIKIMI</td>
<td>IX</td>
</tr>
<tr>
<td>FALËNDERIMET</td>
<td>X</td>
</tr>
<tr>
<td>ABSTRAKTI</td>
<td>XI</td>
</tr>
<tr>
<td>KAPITULLI 1 - HYRJE</td>
<td>2</td>
</tr>
<tr>
<td>1.1 Motivimi</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Problemi i sigurisë në një SMDB “Oracle” për një DBA</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Pyetjet e kërkimit</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Metodologjia e kërkimit</td>
<td>4</td>
</tr>
<tr>
<td>1.4.1 Teori</td>
<td>4</td>
</tr>
<tr>
<td>1.4.2 Abstraksion</td>
<td>5</td>
</tr>
<tr>
<td>1.4.3 Projektimi</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Organizimi i tezës</td>
<td>5</td>
</tr>
<tr>
<td>KAPITULLI 2 - ORGANIZIMI I SIGURISË NË SMDB ORACLE</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Sistemi i Menaxhimit të Bazës së të Dhënave</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Siguria</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Modeli i sigurisë ne Oracle</td>
<td>9</td>
</tr>
<tr>
<td>2.3.1 Shtresat e sigurisë</td>
<td>9</td>
</tr>
<tr>
<td>2.3.2 Entitetet fizike</td>
<td>10</td>
</tr>
</tbody>
</table>
2.3.3 Entitetet logjike ................................................................. 10

2.4 Objektet e bazës së të dhënave ................................................................. 11

2.4.1 Tabela .................................................................................... 11

2.4.2 Trigger-i .............................................................................. 11

2.4.3 View ..................................................................................... 13

2.4.4 Programet e vendosura në bazën e të dhënave ( Stored programs ) ......... 14

2.4.5 Privilegji ............................................................................. 15

2.4.6 Roli ..................................................................................... 17

2.4.7 Fjalëkalimi ......................................................................... 17

KAPITULLI 3 - AUDITI I TË DHËNAVE DHE FSHEHJA E KODIT PL/SQL ....20

3.1 Auditi i të dhënave ......................................................................... 20

3.1.1 Ndërtimi i një programi për realizimin e proçesit të auditit në nivel rrjeshti. 21

3.1.2 Proçesi i auditimit për ngjarjen e shtimit. .............................................. 22

3.1.3 Proçesi i auditimit të ndryshimit dhe fshirjes ....................................... 24

3.1.4 Tabela audit_row ..................................................................... 25

3.1.5 Tabela AUDIT_COLUMN ............................................................ 26

3.1.6 Tabela AUDIT_DELETE ............................................................. 26

3.1.7 Trigger për auditimin e fshirjes .............................................................. 30

3.2 Kodimi i kodit në PL/SQL (Fshehja) ......................................................... 39

3.2.1 Proçesi wrap (fshehjes së kodit) .......................................................... 41

3.2.2 Kushtëzimet e proçesit të fshehjes (Wrapping) .................................. 44

3.2.3 Proçesi i kthimit të kodit (Unwrapping) ............................................. 44
KAPITULLI 4 - KONCEPTIMI I SHTRESËS DHE VETËMBROJTA..............46

4.1 Shtresa e Sigurisë e bazuar mbi trigger-a. .................................................................46
   4.1.1 Ndërtimi i trigger-ave në mënyrë automatike ..................................................49
4.2 Shtresa e sigurisë e bazuar mbi analizën e instruksioneve SQL ............................54
4.3 Shtresa Hibride ............................................................................................................56

KAPITULLI 5 - MENAXHIMI I SESIONEVE, ANALIZA E SQL (DDL DHE DML) DHE BAZA E TË DHËNAVE VIRTUALE ..............................................................59

5.1 Proçeset dhe sesionet ..................................................................................................59
5.2 Analiza e SQL ..............................................................................................................68
5.3 Ndalimi i leximit të kodit të programit të shtresës së sigurisë. ............................73
5.4 Baza e të dhënave virtuale .........................................................................................73
   5.4.1 Si punon VPD-ja .....................................................................................................74
   5.4.2 Përdorimi i VPD-së në shtresën e sigurisë .........................................................79
5.5 Enkriptimi i të dhënave në SMBD Oracle .................................................................79
   5.5.1 Principet e enkriptimit të të dhënave .................................................................80

KAPITULLI 6 - OBJEKTET PËRBËRËS TË SHTRESËS SË SIGURISË DHE FUNKSIONIMI I SAJ ......................................................................................................................82

6.1 Tabletat ......................................................................................................................82
6.2 Sekuencat ..................................................................................................................84
6.3 Objektet PL/SQL .......................................................................................................85
6.4 Mënyra e përdorimit të shtresës, bazuar në objektet e saj ......................................91
6.5 Mënyra e funksionimit të shtresës.........................................................................93
KAPITULLI 7 - AMBIENTI I TESTIMIT, MËNYRA E IMPLEMENTIMIT DHE Mbledhja e të dhënave.................................................................96

7.1 Karakteristikat e sistemit .................................................................................................................................96

7.2 Grumbullimi i të dhënave ...............................................................................................................................97

7.3 Riekzekutimi i instruksioneve SQL ...............................................................................................................99

7.4 Kohëvonesa që sjell shtresa e sigurisë ......................................................................................................100

KAPITULLI 8 - ANALIZA E TESTIMEVE, KONKLUZIONET E TESTIMIT DHE PËRMIRËSIMET E ALGORITMIT.................................................................101

8.1 Testimet e realizuara .....................................................................................................................................101

8.2 Analiza e testimeve .....................................................................................................................................103

8.3 Përmirësimet e algoritmit ............................................................................................................................104

KONKLUZIONE .................................................................................................................................................105
LISTA E FIGURAVE

Figura 2.1. Aksesimi i sistemit të menaxhimit të bazës së të dhënave (SMBD) ..................7

Figura 5.1. Proçeset e përdoruesve në SMBD Oracle ........................................................59
**LISTA E TABELAVE**

<table>
<thead>
<tr>
<th>Tabela</th>
<th>Deskriptiv</th>
<th>Përshtatje</th>
<th>Përdorimi</th>
<th>Njohur</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Fushat që shtohet për tabelat e auditimit</td>
<td></td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>2.</td>
<td>Lista e objekteve për realizimin e procesit të auditimit</td>
<td></td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>3.</td>
<td>View V$BGPROCESS</td>
<td></td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>4.</td>
<td>View V$SESSION</td>
<td></td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>5.</td>
<td>Lista e komandave bazuar mbi view V$SESSION</td>
<td></td>
<td></td>
<td>65</td>
</tr>
<tr>
<td>6.</td>
<td>View v$sql</td>
<td></td>
<td></td>
<td>69</td>
</tr>
<tr>
<td>7.</td>
<td>Karakteristikat e skemës se bazës së dhënave</td>
<td></td>
<td></td>
<td>97</td>
</tr>
<tr>
<td>8.</td>
<td>Tabela e kohëvonesë te shtuar nga shtresa e sigurisë</td>
<td></td>
<td></td>
<td>101</td>
</tr>
<tr>
<td>9.</td>
<td>Tabela e kohës së ekzekutimit të instruksioneve SQL në min</td>
<td></td>
<td></td>
<td>102</td>
</tr>
</tbody>
</table>
DEDIKIMI

Për tu plotësuar.
FALËNDERIMET

Për tu plotësuar.

Hakik PACI
ABSTRAKTI

KONCEPTIMI DHE VËNIA NË JETË E NJË SHTRESE SIGURIE NË SMBD

“ORACLE”

Udhëheqësi shkencor: Prof. Dr. Aleksandër Xhuvani

Në ditët e sotme çdo informacion po kalon në format elektronik, për shkak se është më e thjeshtë për ripërdorimin ose modifikimin e tyre. Ndonjëherë ne nuk kemi nevojë që të kemi një gjurmë të ndryshimeve që kemi bërë dhe ndonjëherë është e nevojshme që të dimë historinë e ndryshimeve. Këto të dhëna mund të ruhen në çdo bazë të dhënash, por jo të gjitha motorët e bazave të dhënave i plotësojnë kërkesat që mund të kërkojen mbi proçesimin e këtyre të dhënave. Në disa raste kërkesë primare është performanca, ndonjëherë siguria, ndonjëherë thjeshtësia (user-friendly) dhe ndonjëherë mund të jetë diçka tjetër. Një nga motorët më të mirë të bazave të dhënave, e cili mund të plotësojnë këto kërkesa, është “Oracle Database”.

Disa përdorues, të cilët kanë të drejta teknike (qasje) në shumë të dhënë, i përdorin këto të dhëna edhe pse ata nuk kanë drejtë ligjore për t’i parë ose modifikuar ato. Ata mund të përfitojnë nga këto të dhëna, duke i modifikuar ato.

Në SMBD “Oracle” si edhe në shumë të tjera, administratori i bazës së të dhënave mund t’u japë të drejta për ndonjë objekt përdorueseve te ndryshëm, kështu që disa përdorues kanë të drejtë që të shtojnë informacione të ndryshme (Insert), disa të tjera për të ndryshuar (Update, Delete) dhe të disa përdorues të tjerë vetëm për lexim (Select). Ndërsa një përdorues DBA (Database Administrator) ka të drejta të plotë teknikisht (qasje) në çdo objekt në këtë bazë të dhënash edhe pse mund të mos ketë të drejtë ligjore mbi të dhënët që ndodhen në këtë bazë të dhënash.

Qëllimi i këtij kërkimi do të jetë shtimi i një shtrese sigurie e cila do ti ndaloj përdoruesve të cilët nuk kanë të drejtë ligjore por kanë te drejtë teknike përdorimin e këtyre të dhënave.
PJESA I
Kapitulli 1 - HYRJE

Qëllimi i kësaj teze është ndërtimi i një shtrese sigurie në një SMDB “Oracle” e cila do të ofrojë mundësinë që një bazë të dhënash të jetë e sigurtë edhe kundrejt përdoruesve të cilët merren me administrimin e bazës së të dhënave. Në këtë tezë do të fokusohemi në menaxhimin e sesioneve dhe se si organizohet vetëmbrojta e objekteve të ndryshëm që janë pjesë e shtresës së sigurisë.

1.1 Motivimi

Përdorimi i informacionit në mënyrë elektronike në ditët e sotme ka një rëndësi shumë të madhe për arsyje se ofron një ripërëdorim dhe shpërndarje të tij në një kohë sa më të vogël. Për të arritur një performancë dhe një menaxhim sa më të mirë të informacionit përdoren jashtëzakonisht shumë bazat e të dhënave. Në shumicën e rasteve për të për zgjedhur se çfarë SMBD duhet të përdorim, duhet të bëhet një analizë e detajuar e parë në disa këndvështrime. Përgjithësisht kriteret e përcaktimit për për zgjedhje janë:

- Volumi i të dhënave që do të manexhojë SMDB.
- Performanca që ofron një SMDB.
- Kostot e implementimit.
- Sistemi i Backup-it dhe Rikuperimi i të dhënave në raste problematike.
- Shpërndarja gjeografike e makinave që do të përdoren instalimet e SMDB.
- Numri i përdoruesve.
- Siguria që ofron një SMBD, kjo pikë është e rëndësishme në rastet kur informacioni, i cili do të popullojë bazën e të dhënave, është i një rëndësie të veçantë. Në disa raste kjo pikë nuk është problematike sepse informacioni mund të jetë i hapur për të gjithë përdoruesit por në shumicën e rasteve duhet të jetë shumë i sigurtë madje duhet të mos aksesohet nga asnjë përdorues qoftë edhe për procese administrimi të bazës së të dhënave.
Disa përdorues, të cilat kanë të drejta teknie (qasje) në shumë të dhëna, i përdorin këto të dhëna edhe pse ata nuk kanë të drejtë ligjore për të parë ose modifikuar ato. Ata mund të përfitojnë nga këto të dhëna, duke i modifikuar ato.

Në SMBD “Oracle”, si edhe në shumë të tjera, administratori i bazës së të dhënave mund t’u japë të drejtë për ndonjë objekt përdorueseve të ndryshëm, kështu që disa përdorues kanë të drejtë që të shtojnë informacione të ndryshme (Insert), disa të tjerë për të ndryshuar (Update, Delete) dhe të disa përdorues të tjerë vetëm për lexim (Select). Ndërsa një përdorues DBA (Database Administrator) ka të drejta të plotë teknikisht (qasje) në çdo objekt në këtë bazë të dhënash edhe pse mund të mos të ketë të drejtë ligjore mbështetjet që ndodhen në këto objekte.

Qëllimi i kësaj teze është shtimi i një shtrese sigurie e cila do të ndalojë përdoruesit, të cilët nuk kanë të drejtën që të shtojnë informacione të ndryshme (Insert), disa të tjerë për të ndryshuar (Update, Delete) dhe të disa përdorues të tjerë vetëm për lexim (Select).

1.2 Problemi i sigurisë në një SMDB “Oracle” për një DBA

SMDB “Oracle” është një ndër bazat e të dhënave më cilësore dhe ofron një numër shumë të madh funksionalitetetësh krahasuar me bazat e të dhënave të tjera. Gjithashtu konsiderohet si SMDB më e sigurtë dhe më e përdorura në sistemet komerciale ku të dhënat janë shumë të rëndësishme në pikëpamjen e sigurisë si p.sh. në banka. Një problem i cili gjendet në një SMDB “Oracle” është të drejtat që mund të ketë përdoruesi i cili do të menaxhojë bazën e të dhënave. Ky përdorues i cili gëzonte rolin DBA ka të drejtë që t’u japë të drejtë përdoruesve të tjerë si dhe të drejtën për të ekzekutuar komanda SQL për menaxhinim e të dhënave të ndryshme në tabelat e ndryshme. Ky si problem nuk është një problem teknik por një problem konceptimi i organizimit të sigurisë. Për të mbrojtur të dhënat “Oracle” ofron mundësinë që të dhënë të kriptohen dhe të dekriptohen dhe ky proces nuk ndikon në performancë. Por edhe ky proces nuk e zgjidh problemin në fjalë. Në versionet e fundit “Enterprice” Oracle ka ofruar mundësinë që një përdorues, i cili gëzonte rolin DBA, të mos ketë të drejtë për të lexuar apo manipuluar të dhënat e një tabele të caktuar, pa i hequr atij asnjtë të drejtë për administrimin e bazës së të dhënave. Ky funksionalitet rrit jashtëzakonisht shumë sistemin e sigurisë në SMDB “Oracle”, por ky sistem mund të çaktivizohet nga një përdorues i cili ka të drejtë për të aksesuar SMDB nga makina ku është instaluar SMDB.

Për të ndërhyrë në motorin e kërkimit në një SMBD “Oracle” është e pamundur, sepse nuk është një sistem i hapur, por ofron mundësinë që të ekzekutohen skripte si në
JAVA, ashtu edhe në PL/SQL, nga ku ne mund të ndërtojmë një shtresë e cila do të monitorojë aktivitetin e përdoruesve.

1.3 Pyetjet e kërkimit

Në bazë të diskutimit në seksionin e mëparshëm, çështja kryesore e kërkimit te tezës është:

_Si mund të ndërtojmë një paketë, e cila do të mbrojë të dhënat nga një përdorues me të drejta pa ndërhyrë në motorin e SMBD dhe si mund të mbrojë veten nga sulmet e përdoruesve, të cilët kanë të drejta të aktivizojnë/çaktivizojnë, fshijnë apo të modifikojnë paketa të ndryshme?_

Për t’ju përgjigjuria kësaj pyetje duhet ta shikojmë nga pikëpamjet e mëposhtme:

- **Gjendja aktuale.** Cila është gjendja aktuale e sigurisë që ofrohet nga “Oracle” dhe cilat janë mundësitë e kontrollit të sesioneve?
- **Kërkesat minimale.** Cilat janë kërkesat minimale të cilat duhet të gëzojë një SMBD “Oracle”, në mënyrë që të na lejojë të implementojmë shtresën e sigurisë?
- **Vetëmbrojta.** Cilat janë mundësitë që një objekt i caktuar të kontrollojë DDL- të ekzekutohen kundrejt atij?
- **Performanca.** Si do të ndikojë në performance aktivizimi i moduleve për mbrojtjen e të dhënave?

1.4 Metodologjia e kërkimit

Kërkimet në shkencat kompjuterike në përgjithshëmi ndahen në tri paradigma, që janë:

1.4.1 Teori

Teoria është e rrënjosur në matematikë dhe synon zhvillimin e teorive të vlefshme dhe ka këto hapa:

- Karakterizimi i objekteve të studimit.
- Hipotezat e mundshme që lidhen me të.
- Vërtetimi.
Interpretimi i rezultateve.

1.4.2 Abstraksion
Abstraksioni bazohet në metodën eksperimentale shkencore. Në ketë metodë një fenomen shikohet dhe vlerësohet duke u mbështetur në rezultatet dhe analizat e eksperimenteve shkencore. Hapat janë si më poshtë:

- Ndërtimi i hipotezës.
- Ndërtimi i një modeli dhe parashikimi.
- Ndërtimi i një eksperimenti dhe mbledhja e të dhënave.
- Analiza e rezultateve

1.4.3 Projektimi
Projektimi është një metodologji e cila përdoret shpesh në inxhinieri dhe ka si qëllim që t’i japë zgjidhje një problemi të caktuar. Hapat janë si më poshtë:

- Kërkesat bazë.
- Specifikimet.
- Projektimi dhe implementimi i sistemit.
- Testimi i sistemit

Hulumtimi i paraqitur në këtë tezë përshtatet natyrisht në paradigmën “Design”, dhe ka si qëllim që të zgjidhë problemin e sigurisë të lartpërmendur në një SMBD “Oracle”.

1.5 Organizimi i tezës
Kjo tezë është e organizuar në tri pjesë, ku secila pjesë ka një fokus të caktuar. Fokus i pjesës së parë është një përmbledhje e qëllimit të kërkimit si dhe një informim mbi punime të ngjashme me këtë fushë. Pjesa e dytë prezanton zgjidhjen tonë për ndërtimin e shtresës së sigurisë dhe mënyrën se si implenxohet kjo zgjidhje. Në pjesën e tretë do të trajtohen rezultatet eksperimentale mbi një SMBD reale, nga ku do të pasqyrohen edhe konkluzionet e punimit.

Pjesa I - Hyrje
Kjo pjesë përmban hyrje si dhe kërkimin mbi literaturën dhe punimet që kanë lidhje me këtë fushë.

**Kapitulli 1** – Ky kapitull përmban këtë hyrje ku përcaktohet motivimi dhe metodologjia e përdorur për këtë punim.

**Kapitulli 2** – Ky kapitull përmban informacion rrëth organizimin e sigurisë në SMBD “Oracle”.

**Kapitulli 3** – Ky kapitull do të paraqesë studime rrëth auditimin në SMDB “Oracle” dhe përdorimin e PL/SQL wrap.

**Pjesa II** – Shtresa e sigurisë

**Kapitulli 4** - Ky kapitull do të përmbajë konceptimin e shtresës për mbrojtjen e të dhënave dhe logjika se si do të vetëmbrohet paketa.

**Kapitulli 5** - Ky kapitull do të përmbajë menaxhinim e sesioneve, analizën e SQL-së (DDL dhe DML), koncepti i VPD-së dhe enkriptimin e të dhënave.

**Kapitulli 6** – Objektet për të rësë e të paketës si dhe funksionimi i tyre.

**Pjesa III** – Vënia në jetë e shtresës dhe testimi

**Kapitulli 7** – Ambienti i testimit, mënyra e implementimit dhe mbledhja e të dhënave.

**Kapitulli 8** – Analiza e testimeve, konkluzionet e testimit dhe përmirësimet e algoritmit.

**Kapitulli 9** – Konkluzionet e përgjithshme të tezës.
Kapitulli 2 - ORGANIZIMI I SIGURISË NË SMDB ORACLE

Në këtë kapitull do të njihemi me organizimin e sigurisë në një SMDB “Oracle” duke analizuar disa elemente të sigurisë si rolet, përdoruesit, të drejat mbi të dhënat, auditimi, sinonimet etj.

2.1 Sistemi i Menaxhimit të Bazës së të Dhënave

Një bazë të dhënash është një bashkësi të dhënash të mirëorganizuara, që ndryshe këto të dhëna quhen rekorde. Për të aksesuar një bazë të dhënash duhet që të gjithë kërkesat të kalojnë nëpërmjet sistemit të menaxhimit të bazës së të dhënave (SMBD), e cila ilustrohet si në figurën e mëposhtme:

![Diagram](https://via.placeholder.com/150)

Figura 2.1. Aksesimi i sistemit të menaxhimit të bazës së të dhënave (SMBD)

Aksesimi i të dhënave në një SMDB Oracle bëhet nëpërmjet programeve të veçanta, të cilët presin kërkesa nga programe klient dhe quhet “tns listener”, i cili interpreton komandat që i dërgohen dhe i drejton këto komanda drejt bazës së të dhënave.
Çdo transaksion i cili do të ekzekutohet ka një fillim dhe një përfundim, ku si përfundim mund të jetë *Commit* apo *Abort*. Transaksionet të cilat vetëm lexojnë të dhëna, nuk kanë nevojë për një fund.

Në përgjithësi, një transaksion ka katër veçori që ndryshë njihen si veçoritë ACID (Haerder and Reuter, 1983) që janë si më poshtë:

- **Atomiciteti.** Transaksioni duhet të ekzekutohet me sukses ose duhet të duket që nuk është ekzekutuar i plotë, me qëllim që SMBD të anulojë të gjithë veprimet e kryera nga transaksioni i cili nuk është ekzekutuar i plotë.
- **Konsistencë.** Një transaksion pas ekzekutimit apo anulimit të tij duhet patjetër që bazën e të dhënave ta kalojë nga një gjendje konsistente në një gjendje tjetër konsistente, që do të thotë nuk duhet të thyejë asnjë nga kufizimet (*constraints*) e bazës së të dhënave.
- **Izolimi.** Asnjë transaksion nuk duhet të ndërhyjë apo të ketë efekte nga një transaksion tjetër.
- **Qëndrueshmërria.** Një transaksion i cili përfundon me sukses (*commit*) duhet të qëndrojë të jetë gjendje pavarësisht nga arsjet e ndryshme, qoftë edhe kur mund të bllukohet SMBD apo kur nuk përgjigjet makina.

### 2.2 Siguria

Siguria në pikëpamjen kompjuterike ka disa kuptime dhe shumica e përdoruesve të sistemeve kompjuterike i ndajnë si më poshtë:

- **Fshehtësia dhe konfidencialiteti:** Në ketë këndvështrim të dhënat nuk duhet të aksesohen nga persona të paautorizuar për përdorimin e këtyre të dhënave. Kjo është edhe pikëpamja e kësaj teze.
- **Saktësinë, integriteti dhe Autenticiteti (vërtetësia):** Saktësia dhe integriteti do të thotë se të dhënët nuk mund të jetën, me qëllim që të korruptuar ose të modifikuar. Autenticiteti (vërtetësia) siguron një mënyrë për të verifikuar origjinën e të dhënave.
- **Disponueshmërria dhe rikuperimi:** Në ketë vështrim siguria merr kuptime që një sistem vazhdon të punojë dhe do të ofroi të dhënëve në rastet kur ka probleme.
Të tre këto këndvështrime të sigurisë nё pamje të parё ngjasojnё shumё, por, nё tё vёrtetё, kur shikohen për zgjidhjet teknike, kёto janё shumё të ndryshme dhe sisteme të ndryshme pёrdorin metoda të ndryshme për tё zgjidhur kёto çёshtje. P.sh.: pёrdorimi i njё ID dhe fjalёkalimi do tё na mundёsonte qё të kemi informacion se nga ku kanё ardhur tё dhёnat nё sistem. Pёrdorimi njё algoritmi kriptimi do tё na siguronte qё të dhёnat tё mos jenё tё ndryshueshme nga pёrdoruesit e paautorizuar, ndёrsa njё sistem backup-i do tё siguronte pikёpamjen e tretё tё sigurisё tё lartpёrmendur.

Nё varёsi se çfarё rёndёsie kanё tё dhёnat, zgjidhet edhe se si duhet tё jetё e organizuar siguria e njё bazё tё dhёnash.

Pёr tё ofruar njё siguri sa mё tё lartё duhet qё tё kemi edhe njё siguri tё ofruar nga hardware-i gjithashtu edhe mundёsinё qё organizojmё kёtё siguri edhe nё nivel software. Nё disa raste tё dhёnat janё tё njё rёndёsi shumё tё lartё sa qё edhe ndёrtuesi e sistemit tё sigurisё duhet tё mos kenё akses nё kёto tё dhёna.

2.3 Modeli i sigurisё ne Oracle

Modeli i sistemit tё sigurisё ështё njё model i bazuar nё logjikën e shtresёzimit, e cila mbrojta file-t dhe objektet e bazёs sё tё dhёnave si brenda ashtu edhe jashtё saj. Nё kёtё pjesё do tё njihemi mё nё detaje se si ështё e organizuar siguria nё SMBD “Oracle”.

2.3.1 Shtresat e sigurisё

Shtresat e sigurisё, të cilat mund tё implementohen, duhet tё përmbushin kёrkesat e mёposhtme:

- Mbrojta e file-ve tё SMBD “Oracle” nё nivel tё sistemit Operativ.
- Mbrojtja e kodit i cili ndёrvepron me bazёn e tё dhёnave.
- Kontrolli i lidhjeve me bazёn e tё dhёnave.
- Kontrolli i aksesimit tё tabelave duke pёrdorur rolet, grant-et, trigger-at dhe procedurat.
- Mundёsia për tё pёrdorur kriptim tё tё dhёnave, firma digitale, etj., tё cilёt rrisin nivelin e sigurisё.
2.3.2 **Entitetet fizike**

Me qëllim që të ofrohet një siguri sa më e lartë duhet që entitetet fizike të një SMBD Oracle të organizohen në mënyrë të tillë që të minimizohen sulmet dhe të mbrohen këto entitetet. File-et e mëposhtme janë shumë të rëndësishme në pikëpamjen e sigurisë:

- File-t fizike të cilët formojnë “tablespace” e bazës së të dhënave, ku përfshihen file-t “redo log”, kontrolli, arshiva (archive log), datafiles.
- File-t për parametrit inicilaizues – ky file përmban të dhëna për inicilaizimin e bazës së të dhënave.
- File-t e konfigurimit (config.ora) – ky file përmban informacion për sa të përkë konfigurimit të bazës së të dhënave.
- File-t e konfigurimit të rrjetit (LISTENER.ORA, SQLNET.ORA, TNSNAMES.ORA). Këto file përbajnë informacion për mënyrën se si lidhen klientët me SMBD.

2.3.3 **Entitetet logjike**

Përveç entitetet fizike SMBD Oracle është i organizuar edhe në entitete logjike, të cilat mbrohen dhe mirëmbahen vetë nga SMBD.

Më poshtë janë dhënë listat e këtyre entitetve:

- **Përdoruesi**: Përdoruesi krijohen nga një përdorues me të drejtë për të krijuar përdorues. Një përdorues mund të ketë objekte të tij ose mund të përdor objektet e një përdoruesi tjetër.
- **Skema**: Përbëhet nga bashkësia e objekteve që i përkin janë përdoruesi.
- **Privilegji**: Të drejtat që gëzon një përdorues mbi të dhënat ose mbi një objekt. Privilegjet ndahen në dy nivele, në nivel sistemi dhe në nivel objekt.
- **Roli**: Një grup privilegjesh të cilat mund të jepen një përdoruesi të caktuari. Rolet mund të të organizohen në mënyrë hierarkike, që do të thotë që një roli mund të përmby të rënë role të tjera.
- **Tabela**: Objekte ku mund të vendosen të dhënët.
- **View**: Objekte të cilat bazohen mbi tabelat të ndryshme për të paraqitur të dhënët.
• **Trigger**: Objekte të cilat janë të programueshme dhe gjenerojnë ngjarje në rastet kur kemi ndonjë veprim Insert, Update ose Delete.

• **Pjesë programi ose programe të plota**: Programe të ndërtuar në PL/SQL të cilat janë të vendosura brenda bazës së të dhënave dhe janë të kompiluara.

• **Sinonimi**: Emërtimi të cilat përdoren nga një përdorues për t’ju referuar një objekti.

### 2.4 Objektet e bazës së të dhënave

#### 2.4.1 Tabela

Tabela është objekti ku mbahen të dhënat në një bazë të dhënash. Një tabelë mund të shihohet si një file virtual, e cila krijohet dhe mirëmbahet brenda “datafile”-ve të bazës së të dhënave.

Kur ndërtohet një tabelë nga një përdorues duhet të specifikohen disa parametra të detryueshëm si emri, kolonat, lloji i kolonave dhe gjetësia e tyre. Gjithashtu mund të specifikohen edhe parametra të tjerë si tablespace-i, në të cilin do të bëjë pjesë, por ky parametër nuk është i detryueshëm dhe, në rastin kur nuk specifikohet, ai vendoset në tablespace ku është vendosur përdoruesi.

Pronësia e një tabele në rastet kur nuk specifikohet, i përkët një përdoruesi, i cili e ndërton këtë objekt. Por një përdorues, me të drejta për të krijuar tabela si DBA, mundet që kur të ndërtojë një tabelë, t’i përcaktojë edhe përdoruesin i cili do të ketë në pronësi këtë tabelë, duke i vendosur emrin e përdoruesit para emrit të tabelës si në shembullin në vijim,

`CREATE TABLE perdorues2.tabele ...`

#### 2.4.2 Trigger-i

Një trigger është një program, i cili gjendet brenda bazës së të dhënave, por ekzektutimi i këtij programi ndodh vetëm në rastet kur ndodhin transaksione mbi tabelën, të cilës i përket ky program. Ky program ndryshon nga llojet e programeve të tjera, të cilët ndodhen në bazën e të dhënave sepse ekzektutohet vetëm në rastet kur ndodh një ngjarje mbi këtë tabelë dhe nuk mund të ekzektutohet nga përdoruesit sic mund të ndodh me programet e tjetra, të cilat mund t’i përkasin të njëjtit përdorues. Duke qenë se këta janë programe që vetekzektutohen mund të përdoren për kontrolle të ndryshme, qoftë për
integritetin e të dhënave ashtu edhe për sigurinë e tyre. P.sh: në rast se një tabelë përmban dy lloje të dhënash dhe për secilin lloj të dhënash duhet që kur të krijohet një rekord në këtë tabelë, ku numri i fushave të detyrueshme të jetë i ndryshëm, ky rast nuk mund të zgjidhet duke përdorur kufizimet në ndërtimin e tabelës, por duke përdorur trigger-at, ku do të vendoset logjika se si dallohen këto rekorde dhe cilat janë fushat e detyrueshme.

Një trigger është program, i cili i përkë një objekti të llojit tabelë të caktuar, që do të thotë që në rastet kur fshihet tabela, atëherë edhe trigger-i fshihet automatikisht.

Për të ndërtuar një trigger duhet patjetër të përcaktohen ngjarja/ngjarjet si dhe tabela që i përket ky trigger. Ngjarjet për të cilat mund të ndërtojnë trigger-a për një tabelë të caktuar janë INSERT, UPDATE ose DELETE. Përveç ngjarjes, një trigger ka nevojë që t’i përcaktohet edhe momenti kur duhet të ekzekutohet. Këto momente janë para dhe pas ngjarjes dhe mund të përdoren sipas rasteve që nevojiten. P.sh në rast se trigger-i duhet të kontrollojë për saktësinë e të dhënave, atëherë duhet që programi i cili do të bëjë këtë kontroll të ekzekutohet përpara, ndërsa në rastet si auditimi i të dhënave, për të mbajtur gjurmët e ndryshimeve, duhet të ekzekutohet pas përfundimit të ngjarjeve. Më poshtë jeptë DDL-ja për ndërtimin e një trigger-i, cili ekzekutohet pas ngjarjeve mbi një tabelë dhe shton në një tabelë tjetër ngjarjen që ndodhi.

CREATE OR REPLACE TRIGGER Emri_trigger
AFTER INSERT OR UPDATE OR DELETE
ON Tabela_e_audituar
FOR EACH ROW

-- Deklarimi i variablave

DECLARE
    tyact   VARCHAR2(1); -- I per shtim, U- Ndryshim, D- Fshirje
BEGIN

    -- Logjika e programit

    IF INSERTING THEN
        tyact := 'I';
    ELSIF updating THEN
        tyact := 'U';
    END IF;
ELSIF deleting THEN
    tyact := 'D';
END IF;

INSERT INTO tb_audit
(   perdoruesi, veprimi, tabela, id, data_veprimit )
VALUES
(   user, tyact, ' Tabela_e_audituar ', :old.id, Sysdate );
END;

Përveç veprimeve INSERT, UPDATE ose DELETE, mbi një tabelë mund të ekzekutohen edhe SELECT, por nuk është e mundur që të ndërtohet një trigger për një ngjarje e cila lexon informacionin.

2.4.3 View

Një objekt view në një SMBD Oracle është një përcaktim i mënyrës se si mund të lexohen disa të dhëna nga tabela të ndryshme, si të ishte një tabelë vetë. Pra një view nuk përmban asnjë të dhënë përveç logjikës se si do të merren disa të dhëna dhe do të paraqitet, ku si burim mund të jetë një ose më shumë tabela, view apo pjesë te ndryshme kodi, të cilat kthejnë informacione të ndryshme në bazë të ekzekutimeve të tyre. View është shumë e përdorshme në ndërtimin e aplikacioneve dhe në shumë raste përdoren edhe për qëllime sigurie sepse, duke përdorur view, ne mund të mos paraqesim të gjithë informacionin, i cili ndodhet në një tabelë të caktuar.

Edhe pse view ofron një mbrojtje, për informacionin duhet të tregohet kujdes me përdorimin e tyre për vetë faktin që për çdo ekzekutim të një SELECT, që është pjesë e
një view-e të dhënat që gjenerohen, vendosen në tabelë tempore. Në rastet kur numri i rekordeve është i madh, atëherë këto të dhënave do të kërkojnë kohë për vendosjen në këto tabela.

2.4.4 Programet e vendosura në bazën e të dhënave (Stored programs)

Programet, të cilat janë ndërtuar në PL/SQL, mund të vendosen në një bazë të dhënash “Oracle” në gjendje të kompiluar. Shpesh këto programe quhen proçedura ose funksione, ku bashkësia e tyre, të cilat janë të organizuara, formojnë paketat. Diferenca e vetme ndërmjet funksioneve dhe proçedurave është që një funksion është i detyruar që të kthejë një vlerë, ndërsa një proçedure nuk kthen asnjë vlerë pas ekzekutimit, por mënyra se si programohet në të dyja është e njëjtë. Një proçedurë mund që të ekzekutohet nga një përdorues, nga një aplikacion apo edhe nga ndonjë program tjetër, i cili është vendosur brenda bazës së të dhënave.

Programet e vendosura në bazën e të dhënave kanë përparësi krahasuar me ekzekutimet dinamike të SQL-ve të ndryshme. Më poshtë paraqiten përparësitë kryesore:

- **Minimizimi i trafikut:** Përdorimi i programeve, që vendosen në një bazë të dhënash, minimizojnë trafikun e rrijet për arsy se të dhënat përputohen brenda bazës së të dhënave, si rrjedhim jo vetëm minimizon trafikun por rrit edhe performancën e sistemit në tërësi.

- **Mbrojta e logjikës së biznesit:** Programet, të cilat vendosen brenda bazës së të dhënave dhe përdoren jashtëzakonisht për të ndërtuar logjikën e punës së aplikacionit për të cilin po përdoret edhe baza e të dhënave. Duke qenë se mbajnë një pjesë shumë të rëndësishme të logjikës, duhet që të jenë edhe të mbrojtura në mënyrë të tillë që të mos kuptohet logjika se si përputohen informacioni.

- **Mbrojta nga sulmet:** Në shumë raste thyerja e sistemeve bëhet nëpërmjet sulmeve “SQL Injection” të cilat përdoren për të marrë informacione nga tabelë të tjera. Përdorimi i programeve të vendosura brenda bazës së të dhënave.

Oracle për të mbrojtur këto programe, të cilat janë të kompiluara, ofron edhe mundësinë që kod i tyre të mos jetë i lexueshëm për përdoruesit edhe pse mund të gëzojnë rolin DBA. Për të mbrojtur këto programe përdoren “PL/SQL Wrapper”, e cila kodon kodin burim të një proçedurë, funksioni ose paketë. Duke përdorur këtë teknikë
është e mundur që informacioni, se si është logjika e procesimit të të dhëna
ve, të mos jetë e ekspozuar, por kod i koduar i programeve, që vendosen brenda bazës së të dhëna
ve në Oracle, mund të shikohet duke përdorur view ALL_SOURCE. Edhe pse Oracle ka ofruar
një mënyrë kodimi të kodit burim, do të shikojmë në kapitullin pasardhës se është e
mundur që të dekodohet informacioni. Kjo do të jetë edhe një pikë e cila do të përfshihet
edhe në shtresën e sigurisë që do të ndërtojmë.

2.4.5 Privilegji

Privilegji është e drejtë që i jetet një përdoruesi të caktuar për të aksesuar bazën e
të dhëna. Në momentin që një administrator i bazës së të dhëna ndërton një
përdorues ky i fundit nuk mund të aksesojë bazën e të dhëna deri në momentin që i
jepet privilegji i sistemit CREATE SESSION. Privilegjet ndahen në dy kategorini, në
privilegje sistemi dhe privilegje objekti. Privilegjet e sistemit lejojnë një përdorues që të
lidhet me bazën e të dhëna dhe të krijojë, modifikojë ose fshijë objekte të ndryshme,
ndërsa privilegjet e objekteve lejojnë një përdorues të aksesojë të dhëna, që i përkasin një
objekti, ose të ekzekutojë këtë objekt, nëse është objekt i ekzekutueshëm.

Janë shumë lloje të ndryshme privilegje sistemi, të cilat gjenden në një bazë të
dhënash Oracle dhe numri i tyre është mbi 100 të tilla dhe ndryshon në varësi të versionit.
Lista e tyre mund të gjendet në objektin e bazës së të dhëna me emrin
system_privilege_map. Disa nga privilegjet janë si më poshtë:

CREATE SESSION
CREATE TABLE
ALTER SESSION
CREATE ANY VIEW

Privilegjet mbi objekte janë ato që kanë të bëjnë me të dhënë të përmban një
objekt i caktuar. Numri i tyre ndryshon në varësi të versionit dhe në versionin 11 R2 janë
26 të tilla. Më poshtë janë paraqitur 16 më të përdorshmet, të cilat janë edhe në versione
falas të Oracle.

- ALTER: Ky privilegj përdoret për të ndryshuar strukturën e një objekti, që
  mund të jetë tabelë ose sekuençë. P.sh një tabelë mund të ndryshohet për të
  shtuar një kolonë të caktuar.

- AUDIT: Përdoret për të audituar një objekt të caktuar, p.sh kur janë shtuar të
dhënët në një tabelë të caktuar.
**COMMENT:** Përdoret për të vendosur shënime në një tabelë për kolona të ndryshme.

**DELETE:** Privilegj për fshirjen e të dhënave nga tabela të ndryshme.

**GRANT:** Privilegj i cili i lejon një përdoruesi të drejta mbi një objekt.

**INDEX:** Lejon përdoruesin të krijojë indeksime të ndryshme vetëm mbi tabela.

**INSERT:** Përdoret për t’i dhënë të drejtën për të shtuar rekorde në një tabelë të caktuar.

**LOCK:** Ky privilegj përdoret në rastet kur duhet që një objekt të bëhet locked (Bllokohet aksesimi).

**RENAME:** Privilegj për të bërë riemërimin e një tabele.

**SELECT:** Privilegj për të lexuar të dhënat nga një tabelë apo view.

**UPDATE:** Të dhënët në tabela të ndryshme shpesh duhet të ndryshohen të dhënë dhe ky proçes u lejohet vetëm përdoruesve të cilët kanë të drejtën mbi privilegjin UPDATE. Ky privilegj nuk mund të zëvendësohet me DELETE/INSERT për shkak se tabelat mund të kenë të dhënë një varësi.

**REFERENCES:** Ky privilegji lejon përcaktimin e një kufizimi dhe përçaktimin se nga ku vjen një çelës i jashtëm në një tabelë të caktuar.

**EXECUTE:** Disa objekte, si procedurat apo funksionet, nuk mund t’u jepen privilegje si Select apo Insert etj., për shkak se janë objekte të cilat përmbajnë programe me gjendje të kompiluara dhe të ndërtuara në PL/SQL. Këtyre objekteve u jepet e drejta për t’u ekzekutuar me anë të privilegjit EXECUTE.

**CREATE:** Me anë të këtij privilegji i jepet e drejta një përdoruesi të krijojë një objekt p.sh tabelë.

**READ:** Lejon përdoruesin të lexojë nga një direktori në Oracle.

**WRITE:** Lejon përdoruesin të ketë të drejta për shkrim në një direktori.

Privilegjet të cilat janë më të përdorshme janë SELECT, INSERT, UPDATE, dhe DELETE, të cilat merren me të dhënët që ndodhën në objektet e caktuara.
2.4.6 Roli

Roli është bashkësia e disa privilegjeve me qëllim që të kemi një organizim sa më të mirë të sistemit të sigurisë. Një rol mund të gëzojë një ose më shumë përdorues njëkohësisht. Duke qenë se roli është një bashkësi privilegijesh, atëherë çdo përdorues, të cilit i caktohet një rol, gëzon të gjitha privilegjet që janë pjesë e këtij roli. Veçori tjetër e rolit është se nuk i delegohen të drejatat për privilegje drejtëpërsërdërejtë përdoruesit, por privilegjet i ngarkohen përdoruesit sapo ai lidhet me bazën e të dhënave. Kjo do të thotë që në rastet kur një rol modifikohet, ku mund t’i shtohet apo hiqet një privilegji, atëherë përdoruesi sapo të logohet në sistem, atij i reflektohen ndryshimet e bëra.

Arsyeja pse përdoren rolet është se në SMBD Oracle privilegjet jepen për një objekt dhe vetëm për një përdorues njëkohësisht. Duke qenë se një aplikacion mund të përdoret nga shumë përdorues të ndryshëm, të cilët duhet të kenë të njëjtat të drejta mbi objekte të ndryshme, atëherë do të ishte shumë e vështirë dhe do të harxhonte shumë kohë për të menaxhuar përdoruesit. Me anë të një roli mund të thjeshtohet si proces.

Për të kontrolluar privilegjet që mund të ketë një përdorues, mund të kontrollojmë view-et e Oracle për të cilat duhen të drejta për aksesimin e tyre.

Në disa raste i njëjtë përdorues mund të ketë disa detyra të ndryshme si shkak i organizimit të pozicionit të punës. Për këtë problem Oracle e ka organizuar skemën e sigurisë në mënyrë të tillë që një përdorues mund të gëzojë më shumë se një rol, edhe në rastet kur dy ose më shumë role të ndryshme gëzojnë të njëjtat privilegje mbi objekte të njëjta. Duke qenë se në SMBD Oracle nuk ka privilegje të cila do të kundërshtonin njërit-tjetrin atëherë në sistem nuk do të ketë asnjë ndikim në rastet kur një përdorues gëzon disa role.

Një pjesë e roleve ndërtohen në momentin që ndërtohet baza e të dhënave dhe këto role janë role të cilat përmbyjnë privilegje mbi objekte, por privilegje sistemi.

2.4.7 Fjalëkalimi

Fjalëkalimet janë pikë shumë e rëndësishme e sigurisë së të dhënave. Në Oracle fjalëkalimet mbahen të koduar dhe nuk mundet të kthehen dhe të gjendet si fjalëkalim. Në shumicën e rasteve, për të sulmuar një bazë të dhënash, përdoren metodat e kombinimeve të karaktereve dhe testimi i tyre për të aksesuar bazën e të dhënave. Për të eliminuar këto problem Oracle përdor profillet e përdoruesve, në të cilat mund të bëhet specifikimi për sa i përkët gjetësisë, kompleksitetit, numrit të tentativave të pasukseshme për hyrje ne sistem, jetëgjatësia e fjalëkalimit, etj.
Por a është e mundur që t’i ndryshojmë fjalëkalimin një përdoruesi me një të ri dhe pas disa veprimeve t’i kthejmë të njëjtin fjalëkalim?

Administratori i bazës së të dhënave ka të drejtë të menaxhojë përdoruesit, të krijojë përdorues të ri, t’i bëjë rivendosje fjalëkalimit, etj. Për të bërë rivendosje të fjalëkalimit të një përdoruesi është e mundur me anë të DDL-së, si më poshtë:

\[\text{alter user perdorusi identified by fjalëkalimi;}\]

Siç shikohet në DDL-në e mësipërme, kur vendoset fjalëkalimi duhet që ai të vendoset në mënyrë të paakorduar. Por, përveç kësaj metode, për ndryshimin e fjalëkalimit ekziston edhe një mënyrë tjetër në Oracle, të cilën ne mund të vendosim vlerën e koduar të fjalëkalimit, si më poshtë:

\[\text{alter user perdorusi identified by values ‘vlera e koduar’;}\]

Përdorimi i kësaj metode i lejon administratorit të përdoruesve të ndryshojë fjalëkalimin dhe të vendosë një vlerë të caktuar të koduar. Kjo do të thotë se një administrator, edhe pse nuk e di fjalëkalimin e një përdoruesi, mund t’ia ndryshojë dhe më pas t’ia rikthejë fjalëkalimin në gjendjen që ishte, duke e lexuar këtë të fundit nga disa view të caktuara të Oracle si në vijim.

\[\text{select t.username} \\
\text{t.password,} \\
\text{t.expiry_date} \\
\text{from dba_users t}\]

Ekzekutimi i SQL-së së mësipërme do të kthejë fjalëkalimin e përdoruesve në mënyrë të koduar dhe këto mund të ripërdoren për ta rikthyer fjalëkalimin në paraardhësin.

Për të rritur sigurinë në Oracle është e mundur që të ndërtohen procedura, të cilat përdorin një logikë tjetër përça i përkët fjalëkalimeve. Që të mos jenë të lexueshme këto procedura duhet të përdoret PL/SQL wrap. Në këto procedura mund të bëhen edhe vonesa të ndryshme, më qëllim që, në rastet kur tentohet thyerja e fjalëkalimin me mënyrat e kombinimeve të karaktereve, të ketë vonesa në çdo testim si dhe të kontrollohet se nga çfarë adrese IP vijnë sulmet, në mënyrë që të bllokohen tentativat për sulm. Në këto procedura është e mundur që të përdoren edhe teknika të tjera, të cilat
mund të lidhen me faktorë të tjerë, të cilat mund të analizohen edhe duke përdorur funksionet standarde të inteligjencës artificiale, të cilat gjenden në një bazë të dhënash Oracle.
Kapitulli 3 - AUDITI I TË DHËNAVE DHE FSHEHJA E

KODIT PL/SQL

3.1 Auditi i të dhënave

Monitorimi i aktiviteteve që ndodhin në një bazë të dhënash dhe regjistrimi i tyre quhet auditi i bazës së të dhënave. Proçesi i audimit bëhet për arsyet të ndryshme, ku më kryesorja është ajo e sigurisë, por mund të bëhet edhe për të bërë një analizë të perfomancës, apo edhe për të mbajtur gjurmët e ndryshimeve të ndryshme që mund t’u bëhen disa të dhënave.

SMBD Oracle ofron proçes audimit të aktivitetit që ndodh në bazën e të dhënave dhe mund të auditojë proçese të ndryshme si leximi, modifikimi apo fshirja e informacionit që ndodhet në një ose shumë tabel. Gjithashtu, është e mundur që të monitorohen edhe ndryshimet strukturale që mund të ndodhin mbi objekte të caktuara si krijimi, fshirja apo ndryshimi një objekti si dhe të monitorohen edhe ekzekutimet e objekteve të ekzekutueshme, siç janë procesdurat apo funksionet.

Proçesi standard i audimit në një SMBD Oracle nuk suporton auditimin në nivel rrjesht, që do të thotë është nuk mund të mbahen ndryshimet që bëhen në një rrjeshti të caktuari të tabelës, por mund që të mbahet informacioni vetëm për llojin e veprimit që ka ndodhur. Për të realizuar auditimin në nivel rrjeshti duhet të përdoren programe (procedura/funksione), të cilat ekzekutohen në rastet kur ndodh një veprim në tabelën e cila por auditohet. Mënya se si ndërtohet një sistem auditimi në nivel rrjeshti do ta shihojmë tek “Ndërtimi i një aplikacioni për audititmin e të dhënave” që trajtohet në këtë kapitull.

Proçesi i audimit të rrjet nivelin e sigurisë por na lejon të mund të identifikohen veprimet që janë ndërmarrë dhe nga kush janë ndërmbërrë këto veprime, nga ku mund të kuptohen edhe ndryshimet e kryera. Auditimi në nivel rrjeshti është efektiv atëherë kur aplikacioni nuk ofron vetë një sistem audit, që të mbajë gjurmët e ndryshimeve, dhe kur informacioni është i një rëndësie të veçantë.

Përcaktimi se cili nga llojet e audimit do të përdoret, përcaktohet duke marrë në konsideratë edhe perfomancën, e cila ndikohet shumë nga proçesi i audimit. P.sh në rastin kur auditohet në nivel rrjeshti dhe numri i të dhënave është relativisht i madh dhe
numri i aktiviteteve është gjithashtu i madh, atëherë duhet të merret shumë në konsideratë edhe kapacitetit.

Në rastet kur proçesi i auditit përdoret për të kontrolluar nëse dikush po kryen veprime të palejueshme, duhet që të aktivizohet që në fillim. Ndërsa ne raste kur auditi përdoret për të evidentuar problemet e performancës, atëherë nuk është e nevojshme që të aktivizohet që në fillim, por vetëm në momentin kur kërkohet të bëhet analiza e performancës. Në rastin e parë, auditi shihohet si proçes, i cili do të na japë informacioni mbështet dhe veprimet që ndodhin në të si dhe nga cilët përdorues, ndërsa në rastin e dytë grumbullohet informacioni sasior mbi objektin që përdoret më tepër, periudhën kur ka më shumë kërkesa ndaj sistemit, etj.

Vend i rëndësishëm i proçesit të auditit është edhe vendi se ku duhet të vendoset informacioni, i cili vjen nga proçesi i auditimit. Në rastet kur është e mundur që informacioni të vendoset jashtë bazës së të dhënave, ka një siguri më të lartë nga ku mund të kontrollohet informacioni. Por edhe pse mund të kemi mundësi që informacionin e auditit ta vendosim në një file jashtë bazës së të dhënave, në disa raste nuk rekomandohet të shkruhet në file, sidomos në rastet kur proçesi i auditimit përdoret për analizë performante.

Përveç rasteve kur administratori ka mundësi që të zgjedhë se ku do të vendoset informacioni, i cili gjenerohet nga proçesi i auditit ka raste kur vendi, ku do të vendoset informacioni, është i përcaktuar. P.sh në rastet kur do të auditohet proçesi i startimit të bazës së të dhënave, i fikjes së saj, i ndryshimit të datafile-ve, etj., informacioni nuk mund të vendoset brenda bazës së të dhënave.

3.1.1 Ndërtimi i një programi për realizimin e proçesit të auditit në nivel rrjeshti

Shpesh është e nevojshme që të kontrollohet se cili përdorues, kur dhe nga ku, ka shtuar një rekord, ka fshirë apo ndryshuar. Siç e thamë edhe më sipër, Oracle ofron mundësinë e auditimit të veprimeve që mund të ndodhin në nivel tabele, por në disa raste është e nevojshme të kontrollohet edhe rezultati i veprimeve të kryera dhe jo vetëm lloji i veprimeve. Për të realizuar këtë proçes auditimi duhet që ndërtohen programe, të cilat do të vendosen brenda bazës së të dhënave, dhe do të ekzekutohen në momentin që ndodh një veprim, e thënë ndryshe do të përdoren trigger-at. Duhet të merret në konsideratë edhe performanca e të gjithë sistemit pas aktivizimit të paketave të auditimit, me qëllim që të optimizohen proçeset e ndryshme apo të kontrollohen hapësirat fizike.
Ndërtimi i këtij programi në Oracle kërkon që, përveç objekteve të ndërtuara në PL/SQL, të ndërtohen edhe objekte të tjera ndihmëse si tabela, sekuenca, etj. Më poshtë është lista e objekteve që nevojiten për ndërtimin e këtij aplikacioni:

- Tri tabela ku do të vendosen të dhënat.
- Dy sekuenca.
- Një paketë PL/SQL, e cila do të përmbajë funksionet apo procedurat e nevojshme për ndërtimin e programit.
- Tre trigger-a për çdo tabelë, të cilët do t’i nënshtrohen proçesit të auditimit.
- Tri kolona për çdo tabelë.

Aplikacioni do të ndërtohet në mënyrë të tillë që zhvilluesit e skemës, në të cilën do të auditohen veprimet e kryera, të mos jenë në varësi të sistemit të auditit. Edhe në rastet kur ka ndryshim të strukturës së tabelave, të mos ketë efekt mbi aplikacionin, për të cilin po ndërtohet auditi.

Duke qenë se programi i cili do të monitorojë proçeset dhe do të registrojë ndryshimet, nuk do të ndikohet nga skema, e cila përmban objektet e audituara, atëherë sistemi i auditimit mund të vendoset në jetë edhe pa dijeninë e zhvilluesve.

Sistemi i auditimit nuk mundet të anashkalohet sepse ekzekutimet e trigger-ave menaxhohen nga Oracle dhe është e vendosur në një skemë një momenti kur ka ndodh ngjarja. Kjo do të ofrojë mundësinë që të ndajmë edhe proçesin e auditimit për çdo tabele dhe çdo ngjarje. P.sh në rastet kur auditi do të përdoret vetëm për të matur performancën, në rastet e shtimit apo ndryshimit të të dhënave, atëherë auditimi i kësaj tabele për proçesin e fshirjes mund të çaktivizohet.

### 3.1.2 Proçesi i auditimit për ngjarjen e shtimit.

Për registruar të gjitha shtimet që bëhen në tabela që janë në proçes auditimi ka dy mënyra për të mbajtur gjurmët, që janë si në vijim:

- **Gjenerimi i tabelave të reja.** Gjenerohen tabela identike me ato që janë ndërtuar nga zhvilluesit të aplikacionit, për të cilin do të auditohen të dhënat dhe shtrohen tri kolona, ku do të mbahet një ID, përdoruesi dhe momenti kur ka ndodh ngjarja.
Shtimi i kolonave në tabelat ekzistuese. Një zgjidhje tjetër do të ishte edhe shtimi i tri kolonave në tabelat ekzistuese, por pa prishur strukturën e çelësave të tabelave dhe pa ndryshuar kufizimet e këtyre tabelave.

Në rastin tonë, duke qenë se ky aplikacion është vetëm për qëllime studimi, kemi zgjidhur mënyrën e dytë e cila mund të kuptohet nga zhvilluesit, praninë e programeve të auditimit, por ka përparësi krahasuar me rastin e parë sepse nuk ndikon shumë në performancë dhe nuk harxhon shumë hapësirë.

Në tabelën e mëposhtme jepen fushat që do të shtohen për çdo tabelë për auditimin e shtimit të rrjeshtave.

**Tabela 1. Fushat që shtohet për tabelat e auditimit**

<table>
<thead>
<tr>
<th>Kolona</th>
<th>Lloji i fushës</th>
<th>Vlera Fillestare</th>
<th>Përshkrimi</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>rid</strong></td>
<td>NUMBER(10)</td>
<td></td>
<td>Id unike për çdo rresht.</td>
</tr>
<tr>
<td><strong>reg_user</strong></td>
<td>VARCHAR2(12)</td>
<td>RTRIM(user)</td>
<td>Përdoruesi që ndryshoi të dhënët.</td>
</tr>
<tr>
<td><strong>reg_date</strong></td>
<td>DATE</td>
<td>SYSDATE</td>
<td>Data dhe ora kur kandodh procesi</td>
</tr>
</tbody>
</table>

Fusha rid (row id) do të plotësohet nga sekuenca, ndërsa dy fushat e tjera janë fusha, të cilat në momentin që ngelen bosh, marrin vlerë, ku e para merr si vlerë emrin e përdoruesit, ndërsa reg_date do të marr si vlerë orën dhe datën e bazës së të dhënave.

Sekuenca do të përdoret për të gjeneruar një vlerë në çdo thirrje të saj e cila do të plotësojë fushën rid gjatë procesit të shtimit të një rrjeshti. Kjo fushë nuk ndikon në çelësit e tabelës, por mund të përdore si çelës në rastet kur është e nevojshme. Sekuenca është përdorur për të zëvendësuar llojin “Autonumber”, që mund të gjenet në bazat e të dhënave të tjera. Ndërtimi i kësaj sekuenca realizohet duke ekzekutuar DML-në e mëposhtme:

```sql
CREATE SEQUENCE seq_rid
START WITH 1        /* numri me te cilin do të filloje sekuenca */
INCREMENT BY 1      /* vlera e inkrementimit te saj */
ORDER               /* gjenerim sipas radhes */
NOCYCLE             /* nuk lejon që te perseriten vlera */
CACHE 100           /* numri i elementeve ne memorje, perdoret per te optimizime ne varesi te numri te shtimeve që
```

Fusha rid (row id) do të plotësohet nga sekuenca, ndërsa dy fushat e tjera janë fusha, të cilat në momentin që ngelen bosh, marrin vlerë, ku e para merr si vlerë emrin e përdoruesit, ndërsa reg_date do të marr si vlerë orën dhe datën e bazës së të dhënave.
GRANT select ON seq_rid TO public;
-- i jepet e drejta per te gjithe

Programi i cili do të ekzekutohet në mënyrë automatike duke përdorur trigger-at do të vendoset në ngjarjen para shtimit për tabelën në fjalën për çdo rresht. Trigger-i do të ketë strukturën e mëposhtme:

```sql
CREATE OR REPLACE TRIGGER emri_trigerit_bi0
BEFORE INSERT
ON tabela_audituar
FOR EACH ROW
WHEN (new.rid IS NULL)
BEGIN
/*
Ky trigger do të ekzekutohet perpara se të shtohet një rresht në tabelën në fjale dhe vetëm në rastet kur rid (fusha e shtuar nga ne) nuk do të ketë vlerë. Vlera e tij do të merret nga sekuencia e ndertuar më sipër më anë të Select-it si mëposhtë. */

SELECT TRUNC(seq_rid.nextval)
INTO :new.rid
FROM dual;
END;
```

3.1.3 Proçesi i auditimit të ndryshimit dhe fshirjes

Për të audituar ndryshimet dhe fshirjet që ndodhin në një tabele duhet që të shtojmë edhe objekte të tjera, në të cilat do të pasqyrohen edhe ndryshimet apo rrjeshtat, të cilët janë fshirë.

Në tabelën e mëposhtme jepen objektet, të cilat duhet të shtohen për të realizuar proçesin e auditimit.
Tabela 2. **Lista e objekteve për realizimin e proçesit të auditimit.**

<table>
<thead>
<tr>
<th>Emri i Objektit</th>
<th>Lloji i Objektit</th>
<th>Përshkrimi</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>audit_row</strong></td>
<td>Tabele</td>
<td>Rekordet për transaksionet UPDATE/DELETE</td>
</tr>
<tr>
<td><strong>audit_column</strong></td>
<td>Tabele</td>
<td>Rekordet për vlerat e vjetra dhe te reja për çdo ndryshim</td>
</tr>
<tr>
<td><strong>audit_delete</strong></td>
<td>Tabele</td>
<td>Të dhënat e rrejtimit të fshirjes</td>
</tr>
<tr>
<td><strong>seq_rid</strong></td>
<td>Sekuence</td>
<td>Sekuence për nr unik</td>
</tr>
<tr>
<td><strong>seq_audit</strong></td>
<td>Sekuence</td>
<td>Sekuence për nr unik</td>
</tr>
<tr>
<td><strong>audit_trail</strong></td>
<td>PL/SQL pakete</td>
<td>Përmbyet procedurat e auditimit</td>
</tr>
<tr>
<td><strong>record_transaction</strong></td>
<td>proçedurë</td>
<td>Plotëson tabelën audit_row</td>
</tr>
<tr>
<td><strong>column_update</strong></td>
<td>proçedurë</td>
<td>Plotëson tabelën audit_column</td>
</tr>
<tr>
<td><code>&lt;tablename&gt;_bi0</code></td>
<td>trigger</td>
<td>Trigger për shtimet</td>
</tr>
<tr>
<td><code>&lt;tablename&gt;_au0</code></td>
<td>trigger</td>
<td>Trigger për ndryshimet</td>
</tr>
<tr>
<td><code>&lt;tablename&gt;_ad0</code></td>
<td>trigger</td>
<td>Trigger për fshirje</td>
</tr>
</tbody>
</table>

### 3.1.4 Tabela audit_row

Kjo tabelë do të përmbyet informacion për tabelën mbi të cilën ka ndodhur ngjarja, e cila ka ekzekutuar trigger-in, llojin e ngjarjes, datën kur ka ndodhur ngjarja si dhe përdoruesi i cili ka ekzekutuar transaksion për tabelën. Duke përdorur DLL e mëposhtme mund që të gjenë rohet kjo tabelë:

```sql
CREATE TABLE audit_row
(
    raid        NUMBER(10)   NOT NULL, /* Row audit ID */
    table_name  VARCHAR2(30) NOT NULL, /* Tabela */
    rid         NUMBER(10)   NOT NULL, /* RID unike nga tabela */
    action      VARCHAR2(1)  NOT NULL, /* Veprim (I, U, D) */
    timestamp   DATE DEFAULT SYSDATE NOT NULL, /* Çasti i ngjarjes*/
    user_name   VARCHAR2(12) DEFAULT RTRIM(USER) NOT NULL -- Përdoruesi
);```

3.1.5 Tabela AUDIT_COLUMN

Kjo tabelë do të mbajë informacion për ndryshimet e ndodhura në një tabelë të caktuar nga një për dorues në një çast të caktuar. Kjo tabelë do të marrë çelësa nga tabela e parë vetëm në rastet kur kemi ngjarje ndryshimi.

DDL-ja e mëposhtme përdoret për ndërtimin e saj:

```sql
CREATE TABLE audit_column
(
    raid       NUMBER(10)   NOT NULL,
    /* Row audit ID, çelës nga tabela audit_row */
    caid       NUMBER(10)   NOT NULL,
    /* Kolona audit ID, numër unik i cili gjenerohet nga sekuenca për tjetër e këtij moduli dhe do të përdoret për arsyeje se në një transaksion mund që te kemi më shumë se një kolone e cila ndryshohet, që do të thotë që për një ndryshim duhet të mbajmë gjurme për disa kolona */
    column_name VARCHAR2(30) NOT NULL  , /* Emri i kolonës */
    old_value   VARCHAR2(2000),       /* Vlera e vjetër */
    new_value   VARCHAR2(2000)          /* Vlera e re */
); 
```

3.1.6 Tabela AUDIT_DELETE

Në këtë tabelë do të vendoset i gijthë informacioni i proçesit të fshirjes, ku këto të dhëna do të ndahen me një karakter të vëçantë në mënyrë që proçesi i auditimit të mos ndërlikohet më shumë se sa është objektivi i kësaj teze. Zgjidhja më e mirë do të ishte që të gjeneroheshin tabela të cilët të përfshinin të gjitha fushat e tabelave që po auditohen, si dhe kolona shtesë për menaxhimin e veprimeve. Gjithësia, edhe në rastin tonë nuk ka ndonjë ndryshim të madh dhe vetëm fushat binare nuk mund që të mbahen vlerat që ato kishin para fshirjes. Për të ndërtuar këtë tabelë do të përdoret DDL-ja e mëposhtme:

```sql
CREATE TABLE audit_delete
(
    raid       NUMBER(10)     NOT NULL,     /* Row audit ID */
    row_data   VARCHAR2(4000) NOT NULL
)     /* te dhenat */
```
Për të audituar procesin e ndryshimit të të dhëna duhet që të gjenerohet një trigger, si ai i mësipërmi, për çdo tabelë. Duke qenë se numri i kolonave ndryshon për çdo tabelë dhe përdoruesi mund të ndryshojë një ose disa nga kolonat mbi të cilat ai ka të drejtë, atëherë lind nevoja që të gjenerohet një skript, i cili kontrollon se cilë nga vlerat e kolonat ka pësuar ndryshime dhe këto ndryshime duhet të vendosen në tabelën e auditimit të ndryshimeve. Më poshtë po paraqitet një trigger për procesin e auditimit e ndryshimit të të dhëna për tabelën “employee”, e cila është në skemën HR, që gjenden në skemat shembull të SMBD “Oracle”.

CREATE OR REPLACE TRIGGER emp_au0
AFTER UPDATE OF
    job, mgr, empno, ename, hiredate, sal, comm, deptno, rid
ON employee
FOR EACH ROW

/*
ky trigger do të ekzekutohet pas çdo ndryshimi ne të dhënat e tabelës employee dhe ka ndryshuar një ose me shumë se një fushe nga fushat që jepen me poshtë :

    job, mgr, empno, ename, hiredate, sal, comm, deptno, rid

duke përdorur këtë filtrër ne mund që te monitorojmë vetëm fushat te cilat ka vlere për procesin e auditimit dhe ne këtë mënyre minimizojmë hapësirën që nevojitet për auditimi si dhe minimizohen efektet negative ne performacën e sistemit.
*/

DECLARE
    raid NUMBER(10);
-- variabël i cili do të përdoret për leximin e vlerës nga sekuenca
BEGIN

    -- lexohet sekuenca (Row Audit ID).
SELECT seq_audit.nextval INTO raid FROM dual;

-- Popullimi me të dhëna i tabelës AUDIT_ROW e cila përmban
-- informacion se cilë nga rreshta kanë pësuar ndryshime nga
-- veprimet e një përdoruesi.

audit_trail.record_transaction
  (raid, 'EMP', :old.rid, 'U');

-- për të populluar me të dhëna tabelën AUDIT_COLUMN e cila
-- mban informacione cila fushe ka ndryshuar për një ndryshim
-- i cili është pasqyruar ne tabelën AUDIT_ROW me anë të
-- procedurës se me sipërme.
-- Kemi përdorur funksioni NVL i cili përdoret për funshat
-- bosh me qëllim krahasimin e vlerave. Ky funksion është
-- një funksion i cili gjendet ne çdo version te SMBD Oracle.
-- për të kuptuar se cila kolone ka pësuar ndryshime atëherë
-- do të kontrollohen vlerat e vjetra me ato të reja për
-- secilën nga ato

-- kolona empno
IF NVL(:old.empno,0) != NVL(:new.empno,0) THEN

  audit_trail.column_update
    (raid, 'EMPNO', :old.empno, :new.empno);

END IF;

-- kolona ename
IF NVL(:old.ename,' ') != NVL(:new.ename,' ') THEN

  audit_trail.column_update
    (raid, 'ENAME', :old.ename, :new.ename);


END IF;
-- kolona job
IF NVL(:old.job,' ') != NVL(:new.job,' ') THEN

    audit_trail.column_update
        (raid, 'JOB', :old.job, :new.job);

END IF;

-- kolona mgr
IF NVL(:old.mgr,0) != NVL(:new.mgr,0) THEN

    audit_trail.column_update
        (raid, 'MGR', :old.mgr, :new.mgr);

END IF;

-- kolona hiredate
IF NVL(TO_CHAR(:old.hiredate,'dd-mon-yyyy'),'31-dec-9595')
    != NVL(TO_CHAR(:new.hiredate,'dd-mon-yyyy'),'31-dec-9595') THEN

    audit_trail.column_update (raid, 'HIREDATE',
        TO_CHAR(:old.hiredate,'DD-MON-YYYY'),
        TO_CHAR(:new.hiredate,'DD-MON-YYYY');
END IF;

-- kolona sal
IF NVL(:old.sal,0) != NVL(:new.sal,0) THEN

    audit_trail.column_update
        (raid, 'SAL', :old.sal, :new.sal);

END IF;

-- kolona comm
IF NVL(:old.comm,0) != NVL(:new.comm,0) THEN
    audit_trail.column_update
    (raid, 'COMM', :old.comm, :new.comm);
END IF;

-- kolona rid
IF NVL(:old.rid,0) != NVL(:new.rid,0) THEN
    audit_trail.column_update
    (raid, 'RID', :old.rid, :new.rid);
END IF;
END;

3.1.7 Trigger për auditimin e fshirjes

Edhe ky trigger do të ekzekutohet pas proçesit të fshirjes dhe duhet të mbajë rrjeshtin e fshirë të vendosur në një tabelë. Siç është shpjeguar më sipër, ky trigger bashkon të dhënët në një fushë të vetme dhe këtë fushë e vendos tek tabela e cila mban detajet e një proçesi fshirjeje. Edhe ky skript do të ndërtohet për çdo tabelë dhe për secilën kolonë, vlera e së cilës është e rëndësishme për proçesin e fshirjes. Tabelat, të cilat do të popullohen me vlera për këtë proçes auditmi, do të jenë AUDIT_ROW dhe AUDIT_DELETE. Më poshtë jepet trigger-i i një tabele të skemës HR.
CREATE OR REPLACE TRIGGER emp_ad0
    AFTER DELETE
    ON emp
    FOR EACH ROW

DECLARE
    raid NUMBER(10);
    deleted_data VARCHAR2(4000);

BEGIN

    -- lexim nga sekuencia (Row Audit ID).

    SELECT seq_audit.nextval
INTO raid
FROM dual;

-- Popullimi me te dhëna i tabelës AUDIT_ROW e cila përmban
-- informacion se cili nga rreshta është fshir nga
-- veprimet e një përdoruesi.

audit_trail.record_transaction(raid, 'EMP', :old.rid, 'D');

-- konvertimi i te dhënave
deleted_data :=
    TO_CHAR(:old.empno) || '|' ||
    :old.ename || '|' || :old.job || '|' ||
    TO_CHAR(:old.mgr) || '|' ||
    TO_CHAR(:old.hiredate,'DD-MON-YYYY') || '|' ||
    TO_CHAR(:old.sal) || '|' ||
    TO_CHAR(:old.comm) || '|' ||
    TO_CHAR(:old.deptno) || '|' ||
    TO_CHAR(:old.rid) || '|' ||
    :old.creator || '|' ||
    TO_CHAR(:old.timestamp,'DD-MON-YYYY') || '|';

-- shtimi i të dhënave ne tabelën audit_delete
    INSERT INTO audit_delete VALUES (raid, deleted_data);
END;

Siç shihet, në të dy trigger-at e ndërtuar më sipër për të shtuar të dhënët në tabelat përkatëse, janë përdorur procedura të cilat ndodhen në paketën audit_trail e cila përbëhet nga këto procedura:

    record_transaction
    column_update

Për këto procedura nevojiten edhe disa parametra, të cilat janë si më poshtë:

raid    = Audit ID, merret nga sekuënc
tabname = Emri i tabelës ne te cilen ka ndodhur ngjarja.
erid    = ID e rreshtit ekzistues.
act     = Ngjarja.
colname = Emri i kolonës që është përfshi ne ngjarje.
oldval = vlera para ndryshimit, vlera vjetër
newval = Vlera pas ndryshimit, vlera e re

Paketa për këtë proces do të jetë si më poshtë:

CREATE OR REPLACE PACKAGE audit_trail IS
PROCEDURE record_transaction
(
 raids IN NUMBER,
 tabname IN VARCHAR2,
 erid IN NUMBER,
 act IN VARCHAR2
);

PROCEDURE column_update
(
 raids IN NUMBER,
 colname IN VARCHAR2,
 oldval IN VARCHAR2,
 newval IN VARCHAR2
);

END audit_trail;

CREATE OR REPLACE PACKAGE BODY audit_trail IS
PROCEDURE record_transaction
(
 raids IN NUMBER,
 tabname IN VARCHAR2,
 erid IN NUMBER,
 act IN VARCHAR2
 ) IS

BEGIN


-- populohet me të dhena tabela audit_row e cila do të mbajë informacion mbi ngjarjet e ndodhura.
-- në parametra nuk janë perfshirë të dhënat e fushave
-- timestamp dhe user_name për qëllime performance dhe vlera
-- e tyre merret nga sysdate dhe user

INSERT INTO audit_row
  (raid, table_name, rid, action, timestamp, user_name)
VALUES
  (raid, tabname, erid, act, sysdate, user);

-- në raste kurgjenerohet ndonje gabim, si psh është fshirë
-- ndonjë nga kolonat e tabelës audit_row.

IF sql%NOTFOUND THEN
  RAISE_APPLICATION_ERROR
    (-20000, Gabim ne procesin e auditimit.');</END_IF;
END;

PROCEDURE column_update
  (raid IN NUMBER,
   colname IN VARCHAR2,
   oldval IN VARCHAR2,
   newval IN VARCHAR2
  ) IS

BEGIN

-- me këtë procedurë do të populohet me të dhena tabela
-- audit_column ku mbahen të dhënët e ndryshimeve.

INSERT INTO audit_column
  (raid, caid, column_name, old_value, new_value)
VALUES (raid, seq_audit.nextval, colname, oldval, newval);

-- gabime

IF sql%NOTFOUND THEN
    RAISE_APPLICATION_ERROR (-20000, Gabim ne proçesin e auditimit.');
END IF;
END;
END audit_trail;

Ndërtimi i skripteve për të kompletuar procesin e auditimit është një proces, i cili, nëse do të ndërtohej sic u ndërtua më sipër, do të ishte jashtëzakonisht shumë i vështirë dhe do të duhet kohë për ta realizuar në rastet kur numri i tabelave është relativisht i madh.

Për të zgjidhur këtë problem mund të shfrytëzohen view e Oracle, të cilat kanë kthejnë informacion për strukturën e objekteve të ndryshme që në rastin tonë është për tabelat që do të auditohen.

View user_tables përmban informacion për tabelën në fjalë dhe me anë të një SQL-je të thjeshtë jemi në gjendje që të marrim si rezultat skriptin për procesin e auditimit. Më poshtë jepet një shembull për sa i përket procesit të shtimit dhe si mund të ndërtohet një trigger në mënyrë dinamike.

SELECT RPAD ('create or replace trigger ', 80, ' ') ||
    RPAD(' before insert', 80, ' ') ||
    RPAD(' on ', 80, ' ') ||
    RPAD(' for each row', 80, ' ') ||
    RPAD('when (new.rid is null or new.rid = 0)', 80, ' ') ||
    RPAD(' ', 80, ' ') ||
    RPAD('begin', 80, ' ') ||
    RPAD(' select trunc(', 80, ' ') ||
    RPAD(' into :new.rid', 80, ' ') ||
    RPAD(' from dual;', 80, ' ') ||
Kjo SQL ka nevojë që, kur të ekzekutohet, t’i jepet vetëm emri i tabelës për të cilën ne kërkojmë që të na gjenerohet skripti për ndërtimin e trigger-ët, për proçesin e auditimit të shtimit të rrjeshtave.

Në të njëjtën logjikë, mund të ndërtojmë edhe për proçeset e ndryshimeve dhe proçeset e fshirjes dhe janë si më poshtë:

```
-- triggeri i ndryshimeve

SELECT RPAD('create or replace trigger ' ||
    table_name || '_AU0', 80, ' ') ||
    RPAD('  after update of', 80, ' ') col1
FROM user_tables
WHERE table_name = UPPER('&tabname');

SELECT '  ' || LOWER(utc.column_name) ||',' col1
FROM user_tab_columns utc
WHERE utc.table_name = UPPER('&tabname')
    AND tc.data_type in ('CHAR', 'VARCHAR',
        'VARCHAR2', 'DATE', 'NUMBER')
    AND utc.column_name not in ('RID', 'CREATOR', 'TIMESTAMP')
ORDER BY utc.column_id;

SELECT '  ' || LOWER(utc.column_name) col1
FROM user_tab_columns utc
WHERE utc.table_name = UPPER('&tabname')
    AND utc.column_name = 'RID';

SELECT RPAD('  on ' || LOWER(table_name),80,' ') ||
    RPAD(' for each row',80,' ') ||
    RPAD(' ',80,' ') ||
    RPAD('declare',80,' ') ||
```
RPAD(' raid number(10);',80,' ') ||
RPAD(' ',80,' ') ||
RPAD('begin',80,' ') ||
RPAD(' select ' ||
'seq_audit.nextval into raid from dual;',80,' ') ||
RPAD(' ',80,' ') ||
RPAD(' audit_trail.record_transaction',80,' ') ||
RPAD(' (raid, ' || CHR(39) || table_name || CHR(39) ||
', :old.rid, ' || CHR(39) ||'U'||CHR(39)||');',80,' ') ||
RPAD(' ',80,' ') col1
FROM user_tables
WHERE table_name = upper('&&tabname');

SELECT RPAD(' if nvl(:old.' || LOWER(a.column_name) || '||',80,' ') ||
DECODE(a.data_type, 'NUMBER','0','DATE', CHR(39) || '31-dec-9595' || CHR(39),
CHR(39) || ' ' || CHR(39) ) || ' ) != ',80,' ') col4,
RPAD(' NVL(:new.'||LOWER(a.column_name)||', '' ') ||
DECODE(a.data_type,'NUMBER','0','DATE', CHR(39)||'31-dec-9595'||
CHR(39), CHR(39) || ' ' || CHR(39) ) ||') then',80,' ') col5,
RPAD(' audit_trail.column_update',80,' ') col6,
RPAD(' (raid,'||chr(39)||a.column_name||CHR(39)||',',80,' ')col7,
RPAD(' :old.' || LOWER(a.column_name) || ', :new.' || lower(a.column_name) || ');',80,' ') col8,
' end if;' col3
FROM user_tab_columns a
WHERE a.table_name = upper('&&tabname')
AND a.data_type IN ('CHAR', 'VARCHAR', 'VARCHAR2', 'DATE', 'NUMBER')
AND a.column_name NOT IN ('CREATOR', 'TIMESTAMP')
ORDER BY a.column_id;

-- triggeri i fshirjes

SELECT RPAD('create or replace trigger ' ||
    table_name || '_AD0',80,' ') ||
RPAD(' after delete',80,' ') ||
RPAD(' on ' || lower(table_name),80,' ') ||
RPAD(' for each row',80,' ') ||
RPAD(' ,80,' ') ||
RPAD(' declare',80,' ') ||
RPAD(' raid number(10);',80,' ') ||
RPAD(' deleted_data varchar2(2000);',80,' ') ||
RPAD(' ',80,' ') ||
RPAD('begin',80,' ') ||
RPAD(' select seq_audit.nextval into raid from dual;',80,' ') ||
RPAD(' ',80,' ') ||
RPAD(' audit_trail.record_transaction',80,' ') ||
RPAD(' (raid, ' || chr(39) || table_name || chr(39) ||
' , :old.rid, ' || chr(39) || 'D' || chr(39) || ');',80,' ') ||
RPAD(' ',80,' ') ||
RPAD(' deleted_data :=',80,' ') col1
FROM user_tables
WHERE table_name = upper('&&tabname');

SELECT column_id co10,
RPAD(' ' ||
DECODE(a.data_type, 'NUMBER','to_char(',
'DATE','to_char(' null) || ':old.' ||
LOWER(a.column_name) ||
DECODE(a.data_type, 'NUMBER','),
'DATE',');', null) || ' ' ||
CHR(124) || CHR(124) || ' ' ||
CHR(39) || CHR(124) || CHR(39) || ' ' ||
CHR(124) || CHR(124), 80, ' ') col1
FROM user_tab_columns a
WHERE a.table_name = UPPER('&&tabname')
AND a.data_type IN ('CHAR', 'VARCHAR', 'VARCHAR2', 'DATE', 'NUMBER')
AND column_id <
(SELECT MAX(a2.column_id)
FROM user_tab_columns a2
WHERE  a2.table_name = UPPER('&&tabname')
    AND  a2.data_type IN ('CHAR', 'VARCHAR',
                       'VARCHAR2', 'DATE', 'NUMBER'))
UNION
SELECT column_id col0,
       RPAD('  ' ||
       DECODE(a.data_type, 'NUMBER','to_char(',
           'DATE','to_char(', null) || ':old.' ||
           LOWER(a.column_name) ||
           DECODE(a.data_type, 'NUMBER',')',
           'DATE',')', null) || ';', 80, ' ') col1
FROM user_tab_columns a
WHERE a.table_name = UPPER('&&tabname')
    AND a.data_type IN ('CHAR', 'VARCHAR', 'VARCHAR2', 'DATE', 'NUMBER')
    AND column_id =
       (SELECT MAX(a2.column_id)
           FROM user_tab_columns a2
           WHERE a2.table_name = UPPER('&&tabname')
                   AND a2.data_type IN ('CHAR', 'VARCHAR',
                                       'VARCHAR2', 'DATE', 'NUMBER'))
ORDER BY 1;

SELECT null
FROM dual;

SELECT RPAD('insert into ' || 'audit_delete',80,' ') ||
       RPAD('(raid, row_data) values (raid, deleted_data);',80,' ') col1
FROM dual;

SELECT 0 col0, 'end;' col1
FROM dual
UNION
SELECT 1 col0, '/' col1
FROM dual
ORDER BY 1;
3.2 Kodimi i kodit në PL/SQL (Fshehja)

SMDB Oracle ofron mundësinë e përdorimit të blloqeve të kodit të shkruar brenda bazës së të dhënave në një gjuhë të zgjeruar, bazuar mbi SQL-në, që quhet Procedural Language/SQL (PL/SQL), e cila është shumë më e fuqishme se gjuha SQL. Të gjitha programet apo modulet e ndryshme që mund të ndërtohen duke përdorur PL/SQL-në, kanë një strukturë të organizuar në blloqe kodit të vendosura në mënyrë sekuenciale, siç paraqitet më poshtë:

```sql
DECLARE
    /*
    Seksoni i deklarimit: variablave, llojet, etj.
    */
BEGIN
    /*
        Seksoni i ekzekutimit
    */
    /
    /*
        ky seksion eshte i nevojshem per cdo program ne PL/SQL.
    */
EXCEPTION
    /*
        Seksoni i perjashtimeve:
        Përdoret per pjesën e gabimeve.
    */
END;

Vetëm pjesa e ekzekutueshme është e detyrueshme, ndërsa pjesa tjeter është opsionale. Në ndërtimin e programeve në PL/SQL lejohen vetëm ato pjesë të SQL-së, të cilat përdoren për manipulimin e të dhënave si SELECT, INSERT, UPDATE, DELETE si dhe kontrolli i transaksioneve. Përdorimi i DML si CREATE, DROP apo ALTER nuk janë të lejueshme që të përdoren drejtpërsëdrujti në një bllok PL/SQL. Përveç kodit SQL,
mund të përfshihen edhe pjesë të tjera kodi, si ciklet, vendim marrjet, thirrjet e funksioneve apo proceduralave.

Një procedurë, funksion, trigger apo një paketë, janë pjesë kodi të ndërtuara në PL/SQL, të cilat aksesojnë të dhënët e objektet e bazës së të dhëna dhe janë të vendosura në bazën e të dhëna si objekte më vete.

Në përgjithësi përdoresi për valid-imin e të dhëna apo për interpretim të ndryshme të pjesës logjike të aplikacionit që do i përdoresi.

Përdoresimi nga objektet PL/SQL (procedura, funksione, trigger-at dhe paketat) që janë të ndërtuara në bazën e të dhëna në ofron këto përparësi:

- **Mbingarkesa:** Duke qenë se këto objekte janë të vendosura direkt në bazën e të dhënë, përdoresi i tyre eliminon një pjesë të Overhead-it në rastet kur kemi të bëjjmë me aplikime. Këto të fundit dërgojnë instruksonë të ndryshme SQL dhe marrin përgjigje, të cilat në varësi të vlerave të tyre mund të ridërgohen në bazën e të dhënëve.

- **Trafiku:** Në shumicën e rasteve përdoresimi i instruksonëve SQL, rezultatet e të cilave ndikojnë në instruksonët pasardhëse, gjenerojnë trafik të dhënash, sepse në momentin që të dhënat e instruksonëve paraardhëse kanë si rezultat të dhënë nga objekte të ndryshme në bazën e të dhënë dhe nga këto të dhënë do të përzhgjidhen vetëm një pjesë e tyre, atëherë kemi një trafik. Ndërkohe, përpunimi i të dhëna nga objektet PL/SQL, që ndodhen në bazën e të dhënëve, nuk do të gjenerojë trafik.

- **Enkapsulimi e logjikes se biznesit:** Përdoresimi i objekteve PL/SQL në bazën e të dhënë të lejojnë programuesit të përfshihën një pjesë të logjikës së biznesit, e cila mund të vendosej edhe në aplikimet klient por nuk do të siguronte të njëjtin integritet të dhënash si në përdoresimin e objekteve PL/SQL.

- **Mbrojta nga sulmet “SQL Injection”:** Në rast kur mund të përdoresi instruksonet SQL në mënyrë të drejtpeshrëndëshme, do të krijonte mundësinë e sulmeve “SQL Injection”. Ndërkohe, përdoresimi i objekteve PL/SQL e eliminon këtë problem serioz për sigurinë e bazës së të dhënave sepse nuk merr si parametër vetëm të dhëna dhe jo komanda SQL.
3.2.1 Procesi wrap (fshehjes së kodit)

Procesi i fshehjes së kodit PL/SQL na ndihmon që të mbrojmë kodin burim nga përdorues të ndryshëm, të cilët mund t’u interesojë ndryshimi i tij apo edhe të kuptojnë logjikën e punës së një programi. Kodi, i cili ka kaluar në procesin e fshehjes, kalon në një kod, i cili është i koduar dhe nuk mund të kuptohet apo të ndryshohet, por ekzistojnë metoda të cilat mund të bëjnë dekodimin e tij. Këto metoda do t’i shikojmë më poshtë, e cila është një problem, i cili është zgjidhur në pjesën e vetëmbrojtjes së kodit.

SMBD Oracle ofron dy mënyra se si mund të fsihet kodit i ndërtuar në PL/SQL duke përdorur një program të tretë “wrap utility” ose paketën DBMS_DDL, e cila përmban procedura të ndryshme për fshehjen e kodit.

Programi “Wrap utility”: Ky program merr si hyrje (input) dosje, të cilat përmbajnë kodin në PL/SQL dhe gjeneron një dosje ku gjendet kodin në mënyrë të koduar, i cili në pas mund të përdoret në bazën e të dhënave për të ndërtuar objektet PL/SQL në mënyrë të koduar.

Për të ekzekutuar këtë program duhet të përdoret comand promt-i, duke përdorur sintaksën e mëposhtme:

```
wrap iname=input_file [ oname=output_file ]
```

Paketa DBMS_DDL: Kjo paketë përmban procedura, të cilat bëjnë të mundur fshehjen e kodit për një objekt PL/SQL, i cili ndodhet në bazën e të dhënave. Por ndryshë nga mënya tjetër, përdorimi i kësaj pakete ka një mjëngjesi, sepse kodon vetëm një objekt PL/SQL në një ekzekutim.

Paketa DBMS_DLL përmban proçedurën CREATE_WRAPPED, e cila merr si parametër një pjesë kodit në PL/SQL dhe ndërton objektin në mënyrë të koduar. Paketa DBMS_DLL është nën përdoruesin sys.

Në shembullin e mëposhtëm tregohet se si mund të përdoret CREATE_WRAPPED në mënyrë që të ndërtohet një paketë e koduar, ku të përfsihet si pjesa specifikuese e paketës ashtu edhe trupi i saj.

DECLARE

```
package_text VARCHAR2(32767);
```

```
FUNCTION generate_spec (pkgname VARCHAR2) RETURN VARCHAR2 AS
```

BEGIN

RETURN 'CREATE PACKAGE ' || pkgname || ' AS

PROCEDURE raise_salary (emp_id NUMBER, amount NUMBER);

PROCEDURE fire_employee (emp_id NUMBER);

END ' || pkgname || ';';

END generate_spec;

FUNCTION generate_body (pkgname VARCHAR2)

RETURN VARCHAR2 AS

BEGIN

RETURN 'CREATE PACKAGE BODY ' || pkgname || ' AS

PROCEDURE raise_salary (emp_id NUMBER, amount NUMBER) IS
BEGIN

UPDATE employees
SET salary = salary + amount
    WHERE employee_id = emp_id;

END raise_salary;

PROCEDURE fire_employee (emp_id NUMBER) IS
BEGIN

DELETE FROM employees WHERE employee_id = emp_id;

END fire_employee;

END ' || pkgname || ';';
END generate_body;

BEGIN

-- gjenerimi i pjesës specifikë te pakettes
package_text := generate_spec('emp_actions')

-- kodimi i pjesës specifikë
DBMS_DDL.CREATE_WRAPPED(package_text);

-- gjenerimi i trupit te pakettes
package_text := generate_body('emp_actions');

-- kodimi i trupit te pakettes
DBMS_DDL.CREATE_WRAPPED(package_text);

END;
/

-- ekzekutimi i një objekti PL/SQL te koduar
CALL emp_actions.raise_salary(120, 100);

Nëse do të kontrollojmë view-të *_SOURCE, të cilat janë objekte të SMDB-së (data dictionary views) ku janë të përcaktuara të gjithëview, që janë pjesë e kësaj baze të dhënash, do të shikojmë që kodit është i koduar ose i fshehur (versionet e fundit te Oracle DB 11g R2). Për të kontrolluar këto view mund të përdorim instruksionin e mëposhtëm SQL.

    SELECT text
    FROM USER_SOURCE
    WHERE name = 'EMP_ACTIONS';

Rezultati do të jete si ne vijim:

TEXT
-----------------------------------
3.2.2 Kushtëzimet e proçesit të fshehjes (Wrapping)

Proçesi i fshehjes nuk është një metodë e sigurt për të fshehur fjalëkalimet ose emrat e tabelave.

Fshehja e kodit është e pamundur për trigger-at. Për të zgjidhur këtë problem duhet të ndërtohet një proçedurë e koduar, e cila thirret nga trigger-i.

Problemi që është ashtu i përdorur e fshehjes në perendim të fshehjes së kodit, është edhe fakti që nuk detektohen gabimet sintaksore apo semantike.

Kodi burim i fshehur nuk mund të kalojë në versione më të ulëta të SMDB-së por vetëm në versione më të larta.

3.2.3 Proçesi i kthimit të kodit (Unwrapping)

Proçesi i kodimit të objekteve PL/SQL mund të kthehet pas dhe një objet i koduar mund të dekodohet dhe të arrihet të shëndet kodburim. P. Finnigan në një artikull të tij mbështet sigurinë e Oracle paraqitu si një mbështetësi nga mund të dekodohet një objet i fshehur në PL/SQL, duke analizuari kodin DIANA. Mekanizmi i fshehjes së kodit ka ndryshuar nga versioni në version, në 10g dhe 11i është më e vështirë, por është i ngjashëm.

Për të dekoduar kodin e objekteve te fshehura ka shumë programe falas si dhe faqe online, të cilat realizojnë këtë proçes (http://www.codecrete.net/UnwrapIt).

Duke qenë se ekziston mundësia që të lexohet logjika e programeve, që ndërtohen në PL/SQL, duhet të gjendet një mënyrë që të mbrojmë shtresën e sigurisë, që do të ndërtohet.
Pjesa II – Shtresa e sigurisë

Në këtë pjesë do të shpjegohet mënyra se si është konceptuar shtresa e sigurisë për mbrojtjen e të dhënave. Dihet se një përdorues me të drejta DBA ka mundësinë që të ndryshojë objektet e bazës së të dhënave si dhe të ndryshojë edhe statusin e tyre, pra ka mundësinë që një objekt ta bëjë aktiv ose pasiv. Kjo e drejtë i lejon atij që të kalojë në gjendje pasive edhe objektet e shtresës së sigurisë, që po ndërtohen. Për të eliminuar këtë problem është e nevojshme që të ndërtohet një mënyrë vetëmbrojtjeje, e cila nuk do të lejojë asnjë përdorues të kalojë ne status pasiv objektet e shtresës së sigurisë.
Kapitulli 4 - KONCEPTIMI I SHTRESËS DHE VETËMBROJTA

Në shumë raste, administratori i bazës së të dhënave duhet të ketë mundësinë që të menaxhojë objekte të ndryshme, si tabela, view, indekse, etj. Një pjesë e funksioneve janë për qëllim optimizimi dhe përmirësimi të performancës së një proçesi pune të caktuar. Për këtë arsy dhe duhet që të këtyre përdoruesve të mos u hiqen të gjitha të drejtat mbi objekte të caktuara, por vetëm të drejtat që kanë të bëjnë me ndryshimin e të dhënave, që praktikisht janë të dhënat e tabelave apo sekuencat (të cilat në përgjithësi nuk përbëjnë informacion të rëndësishëm).

Për të realizuar një mbrojtje të tillë mund të përdoren dy mënyra, ku e para ndërton trigger-a të ndryshëm për secilën tabelë, ku vetë trigger-i kontrollon nëse përdoruesi, i cili po kren një veprim Insert, Update apo Delete mbi disa të dhëna e tabelës, ka të drejta të cilat janë të ndryshuar. Për të drejtë, trigger-i nuk duhet të pengojë procesin dhe, në rast të kundërt, nuk duhet të lejojë veprimin. Ndërsa mënryra e dytë konsiston në mundësinë e ndërtimit të një programi, i cili monitoron çdo ekzekutim, i cili ndodh në bazën e të dhënave (çdo instruksion SQL). Në rastet kur ndonjë nga këto instruksione përbëjnë Insert, Update apo Delete e orientuar drejt një objekti të caktuar, mbi të cilin nuk ka të drejta të caktuara në shtresën e sigurisë, atëherë ky instruksion vritet dhe nuk lejohet ekzekutimi i tij. Kjo mënryrë bazohet mbi menaxhimin e sesioneve të përdoruesve.

4.1 Shtresa e Sigurisë e bazuar mbi trigger-a.

Kjo mënryrë bazohet mbi logjikën që të ndalohet veprimi përpara se të afektohet tabela e të dhënave, e cila është e mbrojtur (e përçaktuar në shtresën e sigurisë) nga përdorues të paautorizuar për ndryshim e këtyre të dhënave. Duke qenë se kontrolli duhet të bëhet përpara, atëherë është e nevojshme përborimi i trigger-ave, të cilët vetë-ekzekutohen përpara se të ndodh ngjarja, që janë trigger-a me pjesën before të specifikuar në pjesën ndërtimit të tij.

Më poshtë paraqitet sintaksa e një trigger-i, i cili vetekzekutohet para se të ndodh ngjarja e shtimit dhe një shembull se si mund të ndryshohen vlerat para se të ndodh ngjarja.
CREATE or REPLACE TRIGGER emri_i_trigger
BEFORE INSERT
    ON emri_i_tabeles
    [ FOR EACH ROW ]

DECLARE
    -- deklarimi i variablave
BEGIN
    -- kodi i trigger-it

EXCEPTION
    WHEN ...
    -- gabimi

END;

CREATE OR REPLACE TRIGGER orders_before_insert
BEFORE INSERT
    ON orders
    FOR EACH ROW

DECLARE
    v_username varchar2(10);

BEGIN
    -- Përcaktimi I përdoruesit I cili po bën veprimet
    SELECT user INTO v_username
    FROM dual;

    -- ndryshimi I vlere se një fushe te caktuar
    :new.create_date := sysdate;

    -- përcaktimi I vlere se created_by me përdoruesin
Bazuar në logjikën e trigger-it të mësipërm, mund të ndërtohet edhe për rastet e Update dhe Delete, ose në rastet kur do të kemi vetëm kontrollin e sigurisë të përfshirë në trigger, atëherë është mundur që të përfshihen të tria rastet në të njëjtën trigger të vetëm.

Ndërtimi i shtresës së sigurisë bazuar logjikën e trigger-ave ka një përparësi shumë të madhe sepse nuk analizohet asnjë pjesë tjetër e instruksioneve SQL dhe ekzekutimi i trigger-ave menaxhohet nga vetë Oracle dhe lejon mundësinë e ekzekutimeve të disa trigger-ave në mënyrë paralele, të cilët u përkasin objekteve të ndryshme dhe vijnë nga përdorues të ndryshëm. Kjo nuk do të afekojë shumë performancën.

Gjatë ndërtimit të trigger-ave duhet të merren në konsideratë edhe të drejtat e përdoruesit aktual mbi objektit e shtresës së sigurisë, më qëllim që të kontrollohet nëse një përdorues ka të drejta mbi një tabelë. Mbi këtë tabelë duhet të jepet akses vetëm për leximin e të dhënave.

Çdo trigger, i cili është pjesë e shtresës së sigurisë, i cili është i ndërtuar mbi një objekt ekzistues, duhet të ketë pjesën standarde, e cila me anë të një instruksioni Select kontrollon nëse përdoruesi, që ka ekzekutuar instruksionin, i cili gjeneroi trigger-in, ka të drejta apo jo. Në rast se nuk ka të drejta, atëherë instruksioni nuk lejon të përfundojë me sukses duke përdorur raise_application_error.

Në shembullin e mëposhtëm tregohet si një përdorues, i cili është me emrin test, nuk ka të drejti të bëjë shtim apo ndryshim në një tabelë me emrin employee, edhe pse mbi të ka të drejta për Insert apo Update.

```sql
create or replace trigger emp_biu
    before insert or update on employee
    for each row
DECLARE
v_username varchar2(10);
Begin

    SELECT user INTO v_username
    FROM dual;
END;
```
if v_username = 'test' then
    raise_application_error (-00001,
        'Ju nuk keni te drejta per kete veprim');
end if;
end;

Në rastin kur ky trigger do të ekzekutohet nga ndonjë instruksion SQL, i cili përmban Insert apo Update, i thirrur nga përdoruesi me emrin test, atëherë do të ndërpritet ky instruksion. Në vijim paraqitet ekzekutimi i një instruksioni, i cili përmban Update nga përdoruesi me emër test.

SQL> update employee set salary = 20000;
update employee set salary = 20000
* ERROR at line 1:
ORA-00001: Ju nuk keni te drejta per kete veprim.

Siç është theksuar më sipër, objekti trigger nuk mund të fshihet (të kodohet kodi burim) dhe për të realizuar kodimin e tij duhet të ndërtohet një procedurë, e cila është e koduar dhe më pas të thirret brenda trigger-it. Edhe pse kjo metode është shumë e mirë në performance, nuk na lejon që të vetëmbrohet nga përdoruesit DBA sepse ata mund të kalojnë me status pasiv trigger-in, i cili thërret procedurën e koduar, që verifikon sigurinë. Për të zgjidhur këtë problem duhet të ndërtohen disa skripte, të cilat do të monitorojnë instruksionet DDL, me qëllim që të bllokohen ata që do të prekin strukturën apo statusin e trigger-ave.

4.1.1 Ndërtimi i trigger-ave në mënyrë automatike

Ndërtimi i shtresës së sigurisë bazuar mbi trigger-at e tabelave, të cilat ekzekutohen automatikisht në momentet kur ndodh një transaksion mbi një tabelë të caktuar, kërkon që të ndërtohen objekte shtesë (objekte të ndërtuar me PL/SQL). Vendosja e këtyre objekteve në skemën ku janë ndërtuar tabelat, të cilat do të mbrohen nga shtresë e sigurisë, do të lejonte përdoruesit, të cilët kanë të drejta që të shikonin këto objekte dhe të kuptonin që kemi një mbrojtje të menaxhuar nga trigger-at. Për këtë arsye, të gjithë trigger-at duhet të vendosen nën një përdoruesi tjetër, i cili ka të drejtë të ndërtojë trigger-a të një përdoruesi tjetër brenda së njëtës bazë të dhënash. Disa baza të dhënash nuk e lejojnë që të ndërtohen trigger-a për një skëmën tjetër, ndërsa Oracle e ofron këtë mundësi.
Duke qenë se në ndërtimin e aplikacioneve ndryshimet apo përmirësimet e ndryshme shpesh ndryshojnë edhe strukturën e të dhëna, e cila çon edhe në ndryshimin e objekteve te bazës së të dhëna, atëherë duhet që shtresa e sigurisë të mos jetë e varur nga struktura e tabelave. Kjo do të theqë që, trigger-at e ndërtuar duhet të jenë konceptuar në mënyrë të tillë që të mos kemi efekte në rastet kur ndryshon struktura e një objekti të caktuar (shtimi, fshirja apo ndryshimi i fushave të një tabele të caktuara). Për të zgjidhur këtë problem mjafton që ndërtimi i trigger-ut të jetë i konceptuar në mënyrë të mënyrë të pavarur nga fushat e tabelave, të cilat do të jenë pjesë e sistemit të sigurisë. Kjo do të theqë që, trigger-i ose duhet të kontrollojë në mënyrë dinamike strukturën e tabelës, ose të mos jetë i interesuar për strukturën, por vetëm për emrin e tabelës. Duke qenë se shtresa e sigurisë ka si qëllim mbrojtjen e të dhëna në nivel rrjeshti dhe jo në nivel qelize, atëherë mjafton që dihet vetëm emri i tabelës dhe emri i përdoruesit, i cili ka ekzekutuar instruksionin SQL, e cila ka ekzekutuar trigger-in.

Me qëllim që të minimizojmë punën, e cila do të duhet për të ndërtuar një trigger, i cili është pjesë e shtresës së sigurisë për një skemë, e cila do të ketë shumë tabela, të cilat do të jenë pjesë e mbrojtjes, atëherë duhet të ndërtojmë strukturën e tabelës, ose të mos jetë e njëjtë për të gjetë objektet. Kjo zgjidhet duke ndërtuar një funksion, i cili do të marrë si parametrë emrin e përdoruesit, i cili ka ekzekutuar instruksionin SQL, llojin e instruksionit (Insert/Update/Delete), si dhe emrin e objektit mbi të cilin ndodh ngjarja. Funksioni do të kontrollojë nëse statusi i sigurisë është aktiv dhe nëse përdoruesi ka të drejtën për ekzekutimit e instruksionit. Në rastet kur ka të drejtë, atëherë si rezultat do të jetë 1 dhe në të kundërt 0.

Përdorimi i këtij funksioni na mundëson që të kemi të njëjtën strukturë të trigger-ave për të gjitha tabelat dhe ndryshon vetëm parametrat me të cilat thërritet funksioni i sigurisë.

Më poshtë paraqitet një trigger, i cili thërret funksionin s_kontrollo_objekt dhe në rast se ky funksion kthen vlerën 0, atëherë do të ndërpritet ekzekutimi i tij.

```sql
CREATE OR REPLACE TRIGGER X_test
before INSERT
OR UPDATE
OR DELETE ON aplikimi.objekti
FOR EACH ROW
```
DECLARE

sys_time DATE;
terminal varchar2(15);
perdoruesi varchar2(30);
veprimi varchar2(30);

BEGIN

/*********************************************************
* Kod i gjeneruar automatikisht
* Trigger auditimi mbi aplikimi.objekti "
***********************************************************/

BEGIN

/* Variablat sys_date dhe terminal mund te përdoren për log-e */
sys_time := SYSDATE;
terminal := SUBSTR(USERENV('TERMINAL'), 1, 15);

perdoruesi:= SUBSTR(upper(user), 1,30);

IF INSERTING THEN
    -- rasti kur kemi ngjarje te shtimit
    veprimi := 'INSERT';
ELSIF DELETING THEN
    veprimi := 'DELETE';
ELSE /* UPDATING */
    veprimi := 'UPDATE';
END IF;

if s_kontrollo_objekt(perdoruesi, veprimi, 'aplikimi.objekti')=0 then
    raise_application_error (-00001,
        'Ju nuk keni te drejta per kete veprim');
end if;
exception
   when others then
      null;
end;
END;

Siç shihohet, edhe në kodin e trigger-it nuk kemi varësi nga struktura e tabelës për të cilën është ndërtuar trigger-i. Kjo na jep mundësinë që të ndërtojmë një procéduri, e cila do të na mundësojë ndërtimin e trigger-ave në mënyrë dinamike dhe përdoruesve të shtresës së sigurisë nuk u lind nevoja që të shkruajnë kode për ndërtimin e trigger-ave.

Proçedura, e cila do të ndërtojë trigger-at, do të marrë si parametër emrin e trigger-it, i cili nuk duhet të ekzistojë, si dhe emrin e tabelës (përshërë edhe emrin përdoruesit që i përkët kjo tabelë). Kodi i kësaj procédurë do të jetë si më poshtë:

```sql
create or replace PROCEDURE nderto_trigger( p_table_name in varchar2,
                                           p_trigger_name in varchar2,
                                           p_owner_name in varchar2)
IS
   --
   v_full_table_name varchar2(100) := p_owner_name || '.' || p_table_name;
   v_trigger_sql varchar2(32000);
   --
   v_count number;
BEGIN

   'CREATE OR REPLACE TRIGGER ' || p_trigger_name || chr(10) || 'BEFORE INSERT OR UPDATE OR DELETE ON ' || v_full_table_name || chr(10) || 'FOR EACH ROW' || chr(10) || 'DECLARE' || chr(10) || 'perdoruesi varchar2(30); '
```
BEGIN
  PERDORUESI:= SUBSTR(UPPER(USER), 1,30);
  IF INSERTING THEN
    -- rasti kur kemi ngjarje te shtimit
    VEMRIMI := ''INSERT'';
  ELSIF DELETING THEN
    VEMRIMI := ''DELETE'';
  ELSE /* UPDATING */
    VEMRIMI := ''UPDATE'';
  END IF;
  IF S_KONTROLLO_OBJEKT(PERDORUESI, VEMRIMI, 'APLIKIMI.OBJEKTI')=0 THEN
    RAISE_APPLICATION_ERROR (-00001,''JU NUK KENI TE DREJTA PER KETE VEMRIM'');
  END IF;
  EXCEPTION
    When others THEN
      Null;
  END;
END;
EXECUTE IMMEDIATE v_trigger_sql;
END;
4.2 Shtresa e sigurisë e bazuar mbi analizën e instruksioneve SQL

Një zgjidhje _tjetër për ndërtimin e shtresës së sigurisë është edhe ajo e monitorimit dhe analizës së çdo instruksoni SQL, i cili ekzekutohet mbi bazën e të dhënave. Kjo mënyrë duhet të vendoset para ekzekutimit, duke përdorur view-e (v$sqltext), të cilat Oracle na i ofron për të lexuar SQL, që ekzekutohen nga përdorues të ndryshëm. Duke përdorur këto view na jetep mundësia që të shihojmë çdo instrukson, i cili vjen nga përdorues të caktuar dhe më pas këto instruksonë mund të analizohen përpara se të kalojnë në ekzekutim. Ndryshe nga mënyra e trigger-ave, ku nuk kishim nevojë që ne të bënim analiza të ndryshme, për të kuptuar nëse një instrukson kishte të bënte me sigurinë e të dhënave, këtu duhet që të ndërtohen një program (paketë ose procedural në PL/SQL ose java duke qenë se Oracle na lejon që të ekzekutojmë kode java brenda bazës së të dhënave), i cili do të na kthej informacionit nëse një SQL e dhënë është DDL apo DML dhe mbi cilit objekte do të ekzekutohet. Pra, ky program duhet të kontrollojë një mënyrë inteligjente nëse kemi instrukson SQL, të cilat nuk leohen apo jo. Arsyeja pse duhet të jetë inteligjente ka të bëjë me formën e instruksonëve SQL, të cilat janë dinamike. P.sh. një Instrukson SQL mund të përmbytë Insert në tabelën A me vlera që vijnë nga tabelat B dhe C, që do të thotë që mbi tabelat B dhe C kemi Select. Duke qenë se SQL mund të përmbytjen nën SQL (sub query), atëherë ky program duhet të jetë i ndërtuar me logikën rekursive dhe të arrijë të ndajë çdo pjesë të instruksonit, kryeson në pjesë më të vogla, të cilat do t’i kalojnë analizës dhe rezultati i secilës pjesë do të kombinohen për të marrë një rezultat final.

Kjo mënyrë na ofron mundësinë që të kontrollojmë edhe instruksonët DDL, të cilat mund të jenë për qëllime optimizimi apo ndryshimi të strukturës së objekteve, por mund që të jetë edhe për qëllim të modifikjojë programin, i cili monitoron të gjithë sesionet dhe instruksonet SQL, që ekzekutohen. Kjo do t’i lejonte një përdoruesi me të drejta DBA, që të ndryshojë shtresën e sigurisë apo ta kalojë me status jo aktiv. Duke qenë se programi i ndërtuar do të monitorojë të gjithë instruksonet SQL, si ata DML për përpunimin e informacionit, ashtu edhe ata DDL për përcaktimin e objekteve të bazës së të dhënave, ashtu edhe TCL për kontrollin e bazës së të dhënave.

Në momentin që programi, i cili po monitoron sesionet e përdoruesve do të detektojë një DDL apo një TCL, e cila do të ndëpresë mirëfunksionimin e shtresës së sigurisë, atëherë ky instrukson do të ndërpritet. Në këtë mënyrë është e mundur që të arrihet një vetëmbrojte e shtresës së sigurisë nga përdorues me të drejta DBA.
Edhe pse kjo mënyrë është shumë efikase dhe ofron mundësinë e vetëmbrojtës, duhet të optimizohet me qëllim që të mos ndikojë në performancën e bazës së të dhënave duke analizuar çdo SQL. Duke qenë se instruksionet ndahen në dy lloje, që janë instruksione të brendshme (internal - background) dhe instruksione të jashtme, atëherë instruksionet SQL që janë internal, të cilat ekzekutohen nga vetë motori i bazës së të dhënave për qëllimin e mirëmbajtjes së bazës së të dhënave, mund të përjashtohen nga monitorimi dhe kjo do të sillte një rritje të performancës. Gjithashtu programi, i cili përdoret për detektimin e pjesëve të palejueshme të instruksioneve SQL, duhet të jetë shumë kompleks sepse instruksionet SQL nuk janë të njëjta, edhe pse kanë strukturë të mirëcaktuar.

Përjashtime të tjera nga monitorimi mund të bëhen edhe për përdoruesit, të cilët shkëmbbejnë të dhëna në mënyrë automatike ndërmjet bazave të të dhënave të ndryshme, si në rastet e replikimit. Kjo për arsyen se, nëse kemi disa baza te dhënash, të cilat kanë të implementuar shtresën e sigurisë, atëherë një instruksion, i cili ka ndryshuar të dhëna, është kontrolluar nga shtresa e sigurisë, atëherë nuk është e nevojshme që ky ndryshim të kontrollohen përshër. Por për këto raste duhet që të mos lejohet, që një përdorues DBA të ndryshojë fjalëkalimin e përdoruesit, i cili menaxhon procesin e shkëmbimit të të dhënave, me qëllim që të eliminojë mundësinë që të përdoren kredencialet e këtij përdoruesi.

Nisur nga logjika e përjashtimeve të përdoruesve nga monitorimi, programi duhet të kontrollojë vetëm sesionet, të cilat konsiderohen sesione me rrezik sulmi (ndryshimi i të dhënave edhe pse nuk kanë të drejta juridike por kanë të drejta teknike).

Programi, i cili do të monitorojë sesionet, nuk është e mundur që të punojë në paralel për arsy se mund të kemi instruksione, të cilat mund të ndikojnë në të drejtat e instruksioneve pasardhëse, dhe nëse do të ekzekutoheshin në paralel, do të sillnin një funksionim jo të mirë. Në shembullin e mëposhtëm jepet një rast kur mund të hasen probleme.

Përdoruesit : A, B dhe Admin
Tabelat : TB1, TB2, TB3
Te drejtat : A ka Insert ne TB1, TB3 dhe Update ne TB2
          B ka Insert ne TB2 dhe Delete ne TB3
Lista e Transaksioneve :
      A – Insert ne TB1
      B– Insert ne TB2
Në rastin kur do të kishim një ekzekutim serial, atëherë do të shikonim që rrjesht i shtuar nga B në transaksionin e dytë nuk do të ndryshojë. Ndërsa ekzekutimi në paralel (edhe në rastin kur do të ekzekutohen dy nga dy) do të shikonim që transaksioni i Admin-it do të ekzekutohet në paralel me transaksionin e përdoruesit A, i cili është një Update në tabelën TB2. Duke qenë se në momentin që përdoruesi A po ekzekuton transaksionin e dytë të tijin, transaksioni i Admin-it nuk ka përfunduar akoma, atëherë përdoruesi A ka të drejta që të bëjë ndryshimin e rrjeshtit të shtuar nga përdoruesi B në tabelën B2.

**A ndikon kjo në performancën e bazës së të dhënave?**

Për të dhënë një përgjigje kësaj pyetje duhet të shikonet sa është raporti i sesioneve që monitorohen, se sa është kohe vonesa mesatare e analizës së instruksioneve SQL. Por problemi i ekzekutimeve në mënyrë seriale ka një ndikim shumë të madh në performancë dhe për këtë arsye përdorimi i kësaj metode nuk do të ishte zgjidhja optimale, përjashto rastet kur numri i transaksioneve është shumë i ulët edhe në rastet e pikut.

### 4.3 Shtresa Hibride

Për të rritur sa më shumë performancën e sistemit një zgjidhje shumë efikase do të ishte një kombinim i të dyja mënyrave të përshkruara më sipër. Kur instruksione SQL, të cilat përmbyjnë Insert, Update apo Delete, do të menaxhohen me anë të trigger-ave, ndërsa DDL, të cilat mund të ndikojnë te shtresa e sigurisë, do të kontrollohen nga pjesa e menaxhimit të sesioneve të përdoruesve. Nisur nga analiza statistikore e bërë në bazën e të dhënave, e cila është përdorur për të testuar shtresën e sigurisë, është vënë re që numri i transaksioneve DDL është më i madh vetëm në momentin, në të cilin do të vendoset në jetë baza e të dhënave, skema apo përdoruesit. Në rastet kur kemi një proces pune normale më shumë se 99% e transaksioneve janë DML. Kjo do të thotë që, me heqjen e transaksioneve DML nga procesi i monitorimit, do të kemi një performancë shumë më të mirë se sa në rastet e mënyrës së dytë. Gjithashtu, trigger-at mund të ekzekutohen në mënyrë të pavarur nga njëri-tjetri.

Një optimizimi i mëtejshëm është edhe përjashtimi i sa më shumë përdoruesve nga procesi i monitorimit të sesioneve. Kjo është e mundur për arsye se shumica e përdoruesve do të kenë të drejta DML dhe, duke qenë se kjo pjesë e transaksioneve do të
menaxhohet nga trigger-at, ata herë këto përdorues mund të përjashtohen nga lista e përdoruesve, të cilëve do t’u menaxhohen sesionet.

Sa më i vogël të jetë numri i përdoruesve, të cilët do të monitorohën, aq më e lartë është performanca. Ata herë duhet të ndërtohet një program, i cili do të ekzekutohet në momentin e implementimit të shtresës së sigurisë dhe sa herë që kemi DDL, ku i jepen të drejta përdoruesve për ekzekutime DDL. Ky program do të ndërtojë një pasqyrë të të drejtave aktuale për DDL-në dhe në këtë mënyrë është e mundur që të nxirret lista e përdoruesve, të cilët kanë të drejta për të bërë ndryshime mbi objekte të ndryshme, të cilat janë të mbrojtura nga shtresë e sigurisë apo edhe vetë mbi shtresën.

Programi i monitorimit të sesioneve do të kontrollojë vetëm për instruksionet e mëposhtme:

- ALTER ... (të gjithë instruksionet që fillojnë me ALTER)
- ANALYZE
- ASSOCIATE STATISTICS
- AUDIT
- COMMENT
- CREATE ... (të gjithë instruksionet që fillojnë me CREATE)
- DISASSOCIATE STATISTICS
- DROP ... (të gjithë instruksionet që fillojnë me DROP)
- FLASHBACK ... (të gjithë instruksionet që fillojnë me FLASHBACK)
- GRANT
- NOAUDIT
- PURGE
- RENAME
- REVOKE
- TRUNCATE

Një pjesë e këtyre instruksioneve nuk janë instruksione, të cilat mund të ndikojnë në sigurinë e bazës se të dhënave siç është p.sh.: comment apo associate statistics. Duke
eliminuar sa më shumë instruksione, të cilat nuk kanë efekte tek siguria, aq më shumë do të rritet performanca.

Një mënysë tjetër për rritjen e performancës do të jetë edhe analiza e objekteve PL/SQL, të cilat mund të ndryshojnë objektet e tjera, pra analiza e procedurave, funksioneve apo paketave. Për të bërë këtë analizë duhet të ndërtohen programe shumë inteligjente dhe të arrijnë të kuptojnë se çfarë realizon një objekt, i cili po analizohet për arsyje se vetë programet e ndërtuara në PL/SQL mund të jenë të afta të fshehin apo të ekzekutojnë kod, i cili gjenerohet në momentin e ekzekutimit të programit. Duke qenë se një analizë e tillë kërkon shumë kohë dhe është jashtë objektivave të këtij punimi, algoritmi hibrid nuk do të merret me analizën e objekteve ekzistuese, me qëllim që ata, që nuk përmbën rrezik, të mos përfshihen në procesin e monitorimit.
Kapitulli 5 - MENAXHIMI I SESIONEVE, ANALIZA E SQL

(DDL DHE DML) DHE BAZA E TË DHËNAVE VIRTUALE

5.1 Proçeset dhe sesionet

Çdo përdorues, i cili lidhet me bazën e të dhënave dhe ekzekuton instruksione SQL, e realizon nëpërmjet proçeseve, të cilat janë mekanizma.

Proçeset ndahen në proçese, të cilat merren me pjesën e aplikacioneve, që lidhen me bazën e të dhënave si dhe me hapa të ndryshme brenda bazës së të dhënës Oracle.

Struktura e secilit proçes varet nga konfigurimet e bazës së të dhënave, sistemi operativ në të cilën është i vendosur baza e të dhënave si dhe nga parametrat e tjerë të bazës së të dhënave të zgjedhura. Më poshtë paraqitet një instancë e bazës së të dhënave, ku dallohen të dy llojet e proçeseve.

![Diagrama 5.1. Proçeset e përdoruesve në SMBD Oracle](image)

Figura 5.1. Proçeset e përdoruesve në SMBD Oracle

Në momentin që një përdorues ekzekuton një aplikacion, i cili kërkon që të aksesojë instancën e bazës së të dhënave SMDB Oracle, krijon një proçes, i cili do të ekzekutojë programin. Në momentin e krijimit të proçesit Oracle përdor dy koncepte për të realizuar komunikimin, që janë:
- **Lidhja**: Lidhja është rruga se si aplikacioni komunikon me instancën e bazës së të dhënave, e cila realizohet me anë të mekanizmave të ndryshëm (mekanizmi i proceseve të brendshme apo të rrjetit).

- **Sesioni**: Sesioni është një lidhje e veçantë e një përdoruesi me instancën e bazës së të dhënave, duke përdorur proceset e përdoruesit. P.sh. në momentin që një përdorues kërkon të lidhet me bazën e të dhënave, duke përdorur një emër dhe password të vlefshëm, atëherë instanca e bazës së të dhënave formohen një sesion. Sesioni zgjat nga momenti që përdoruesi lidhet me bazën e të dhënave deri në momentin që shkëputet përdoruesi apo mbylljet aplikimi.

Duke përdorur të njëjtin përdorues dhe fjalëkalim të vlefshëm është e mundur që të ndërtohen sesione të shumta, të cilat janë të pavarura nga njëri-tjetri. Pra, një përdorues mund ta aksesojë instancën e bazës së të dhënave njëkohësisht nga makina të ndryshme apo nga aplikacione të ndryshme.

Proceset ekzekutohen në sfond të instancës së bazës së të dhënave. Këto procese merren me optimizimin e performancës si dhe me mirëmbajtjen e bazës së të dhënave dhe janë të pavarura nga përdoruesit. Një instancë e bazës së të dhënave mund të ekzekutojë një numër të madh procesesh në sfond njëkohësisht. Më poshtë janë paraqitur proceset më të përdorura nga një instancë Oracle sipas emërtimeve të tyre në anglisht:

- **ARCn (Archiver Processes)**: Ky proces bëhet aktiv vetëm në rastin kur baza e të dhënave është e konfiguruar në mënyrën ARCHIVELOG.

- **CKPT (Checkpoint Processes)**: Ky proces menaxhon datafile-et e Oracle në momente të caktuara.

- **DBWn (Database Writer Processes)**: Ky proces realizon shkrimin e memories cache në datafile-et e bazës së të dhënave në momentin që instanca e shikon të arsyeshme që duhet të regjistrohet cache-ja në bazën e të dhënave (p.sh. në rastet kur përfundon një transaksion i caktuara).

- **Job Queue Processes**: Ky proces merret me menaxhimin e detyrave, të cilat i ka përcaktuar një përdorues për t’u ekzekutuar sipas një orari të caktuara.
- **LGWR (Log Writer Proçess)**: Ky proces bën të mundur shkrimin nё redo log të gjithë transaksionet, të cilat kanë përfunduar dhe kanë marrë një numër transaksioni unik. Redo log-et mё pas, nёse është aktivizuar procesi i ARCH, kalon nё arshivim.

- **PMON (Proçess Monitor Proçess)**: Ky proces ka si dytër qё të pastrojë memorien cache dhe burimet e tjera, të cilat i përkisnin një procesi, i cili ka dështuar mё qёllim ripërdorimi.

- **RECO (Recoverer Proçess)**: Ky proces është aktiv nё rastet kur kemi njё bazë të dhënash të shpërndarë dhe menaxhon dështimet qё ndodhin nё kёto lloj bazashe të dhënash.

Gjithashtu, ekzistojnё edhe proçese të tjera, të cilat ekzekutohen nё sfond, qё mund t'i kontrollojmё nё nga view `V$BGPROÇESS`, e cila ka strukturё si nё tabelën e mёposhtme:

**Tabela 3. View V$BGPROÇESS**

<table>
<thead>
<tr>
<th>Kolona</th>
<th>Lloji</th>
<th>Përshkrimi</th>
</tr>
</thead>
<tbody>
<tr>
<td>PADDR</td>
<td>RAW(4</td>
<td>8)</td>
</tr>
<tr>
<td>PSERIAL#</td>
<td>NUMBER</td>
<td>Numri serial</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR2(5)</td>
<td>Emri i proçesit</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2(64)</td>
<td>Përshkrimi</td>
</tr>
<tr>
<td>ERROR</td>
<td>NUMBER</td>
<td>Gabimi</td>
</tr>
</tbody>
</table>

Duke qenё se shtresa e sigurisё nuk do të merret me proçese, të cilat ekzekutohen nё sfond, atёherё ështё e nevojshme qё të analizohet informacioni, të cilin mund ta marrim nga view `V$SESSION`, qё ka strukturё si nё tabelën e mёposhtme:

**Tabela 4. View V$SESSION**

<table>
<thead>
<tr>
<th>Kolona</th>
<th>Lloji</th>
<th>Përshkrimi</th>
</tr>
</thead>
<tbody>
<tr>
<td>SADDR</td>
<td>RAW(4</td>
<td>8)</td>
</tr>
<tr>
<td>SID</td>
<td>NUMBER</td>
<td>ID e sesionit</td>
</tr>
<tr>
<td>SERIAL#</td>
<td>NUMBER</td>
<td>Numri serial i sesionit.</td>
</tr>
<tr>
<td>AUDSID</td>
<td>NUMBER</td>
<td>ID e sesionit te Auditimit</td>
</tr>
<tr>
<td>PADDR</td>
<td>RAW(4</td>
<td>8)</td>
</tr>
<tr>
<td>USER#</td>
<td>NUMBER</td>
<td>Identifikuesi i pёrdoruesit ne</td>
</tr>
<tr>
<td>Column</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USERNAME</td>
<td>VARCHAR2(30)</td>
<td>Emri i përdoruesit</td>
</tr>
<tr>
<td>COMMAND</td>
<td>NUMBER</td>
<td>Komanda ne ekzekutim. Detaje te kësaj kolone paraqitet ne tabelën ne vijim.</td>
</tr>
<tr>
<td>OWNERID</td>
<td>NUMBER</td>
<td>ID e përdoruesit që zotëron sesionin.</td>
</tr>
<tr>
<td>TADDR</td>
<td>VARCHAR2(8)</td>
<td>Adresa e transaksionit</td>
</tr>
<tr>
<td>LOCKWAIT</td>
<td>VARCHAR2(8)</td>
<td>Adresa e sesionit bllokus (lock - waiting for); NULL nëse nuk ka</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR2(8)</td>
<td>Statusi i sesionit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACTIVE – Sesioni është duke ekzekutuar SQL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INACTIVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KILLED – Sesioni është i ndërprerë</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CACHED – sesioni është ne cache përkoheisht nga Oracle*XA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNIPEDE - Sesion jo aktiv, ne pritje te klientit</td>
</tr>
<tr>
<td>SERVER</td>
<td>VARCHAR2(9)</td>
<td>Lloji i shërbimit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEDICATED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHARED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSEUDO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>SCHEMA#</td>
<td>NUMBER</td>
<td>ID e Skemës</td>
</tr>
<tr>
<td>SCHEMANAME</td>
<td>VARCHAR2(30)</td>
<td>skema</td>
</tr>
<tr>
<td>OSUSER</td>
<td>VARCHAR2(30)</td>
<td>Emri i përdoruesit te sistemit operativ klient</td>
</tr>
<tr>
<td>PROCESS</td>
<td>VARCHAR2(24)</td>
<td>Proçes ID e sistemit operativ klient</td>
</tr>
<tr>
<td>MACHINE</td>
<td>VARCHAR2(64)</td>
<td>Emri i makinës</td>
</tr>
<tr>
<td>TERMINAL</td>
<td>VARCHAR2(30)</td>
<td>Terminali</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>VARCHAR2(48)</td>
<td>Aplikimi</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR2(10)</td>
<td>Lloji i sesionit</td>
</tr>
<tr>
<td>SQL_ADDRESS</td>
<td>RAW(4</td>
<td>8)</td>
</tr>
<tr>
<td>SQL_HASH_VALUE</td>
<td>NUMBER</td>
<td>SQL_HASH_VALUE – Adresa e HASH që është ne ekzekutim</td>
</tr>
<tr>
<td>SQL_ID</td>
<td>VARCHAR2(13)</td>
<td>ID e instruksionit SQL që është ne ekzekutim</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SQL_CHILD_NUMBER</td>
<td>NUMBER</td>
<td>Numri i SQL nën varësi</td>
</tr>
<tr>
<td>SQL_EXEC_START</td>
<td>DATE</td>
<td>Koha kur ka filluar ekzekutimi i SQL aktive; NULL nëse SQL_ID është NULL</td>
</tr>
<tr>
<td>SQL_EXEC_ID</td>
<td>NUMBER</td>
<td>ID e ekzekutimit te SQL; (V$SQL_MONITOR)</td>
</tr>
<tr>
<td>PREV_SQL_ADDR</td>
<td>RAW(4</td>
<td>8)</td>
</tr>
<tr>
<td>PREV_HASH_VALUE</td>
<td>NUMBER</td>
<td>Adresa HASH e instruksionit te fundit SQL</td>
</tr>
<tr>
<td>PREV_SQL_ADDR</td>
<td>VARCHAR2(13)</td>
<td>ID e instruksionit SQL te fundit</td>
</tr>
<tr>
<td>PREV_SQL_ID</td>
<td>VARCHAR2(13)</td>
<td>Numri i SQL nën varësi te ekzekutuar</td>
</tr>
<tr>
<td>PREV_EXEC_START</td>
<td>DATE</td>
<td>Koha kur ka filluar ekzekutimi i fundit</td>
</tr>
<tr>
<td>PREV_EXEC_ID</td>
<td>NUMBER</td>
<td>ID e objekti PL/SQL që është ne stak; NULL nëse nuk ka</td>
</tr>
<tr>
<td>PLSQL_ENTRY_OBJECT_ID</td>
<td>NUMBER</td>
<td>ID e programit PL/SQL që është ne stak ne fillim. NULL nëse nuk ka</td>
</tr>
<tr>
<td>PLSQL_ENTRY_SUBPROGRAM_ID</td>
<td>NUMBER</td>
<td>ID e Objekti PL/SQL që është ne ekzekutim; NULL nëse po ekzekutohet SQL</td>
</tr>
<tr>
<td>PLSQL_OBJECT_ID</td>
<td>NUMBER</td>
<td>ID e programit që është ne ekzekutim; NULL nëse po ekzekutohet SQL</td>
</tr>
<tr>
<td>MODULE</td>
<td>VARCHAR2(48)</td>
<td>Emri i modulit</td>
</tr>
<tr>
<td>MODULE_HASH</td>
<td>NUMBER</td>
<td>Vlera Hash e modulit</td>
</tr>
<tr>
<td>ACTION</td>
<td>VARCHAR2(32)</td>
<td>Veprimi</td>
</tr>
<tr>
<td>ACTION_HASH</td>
<td>NUMBER</td>
<td>Vlera Hash e Veprimi</td>
</tr>
<tr>
<td>CLIENT_INFO</td>
<td>VARCHAR2(64)</td>
<td>Informacion i klientit</td>
</tr>
<tr>
<td>FIXED_TABLE_SEQUENCE</td>
<td>NUMBER</td>
<td>Ky numër përmbyan një vlere e cila ritet çdo here përfundoj një veprim ne bazën e te dhënave</td>
</tr>
<tr>
<td>ROW_WAIT_OBJ#</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>ROW_WAIT_FILE#</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>ROW_WAIT_BLOCK#</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>ROW_WAIT_ROW#</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>LOGON_TIME</td>
<td>DATE</td>
<td>Koha e hapës se sesioni</td>
</tr>
<tr>
<td>LAST_CALL_ET</td>
<td>NUMBER</td>
<td>Koha nga momenti kur sesioni</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PDML_ENABLED</td>
<td>VARCHAR2(3)</td>
<td>Statusi i ekzekutimeve paralele te DML</td>
</tr>
<tr>
<td>FAILOVER_TYPE</td>
<td>VARCHAR2(13)</td>
<td>Statusi i ekzekutimeve paralele te DDL</td>
</tr>
<tr>
<td>FAILED_OVER</td>
<td>VARCHAR2(3)</td>
<td>Kërkimet paralele</td>
</tr>
<tr>
<td>RESOURCE_CONSUMER_Group</td>
<td>VARCHAR2(32)</td>
<td>ID e klientit</td>
</tr>
<tr>
<td>PDML_STATUS</td>
<td>VARCHAR2(8)</td>
<td>Statusi i sesionit blokues:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NO HOLDER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GLOBAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT IN WAIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>BLOCKING_INSTANCE</td>
<td>NUMBER</td>
<td>ID e instancës se sesionit blokues</td>
</tr>
<tr>
<td>BLOCKING_SESSION</td>
<td>NUMBER</td>
<td>ID e sesionit blokues</td>
</tr>
<tr>
<td>SEQ#</td>
<td>NUMBER</td>
<td>Numër unik</td>
</tr>
<tr>
<td>EVENT#</td>
<td>NUMBER</td>
<td>Numri i ngjarjes</td>
</tr>
<tr>
<td>EVENT</td>
<td>VARCHAR2(64)</td>
<td>Ngjarja</td>
</tr>
<tr>
<td>P1TEXT</td>
<td>VARCHAR2(64)</td>
<td>Përshkrimi i ngjarjes se pare ne pritje</td>
</tr>
<tr>
<td>P1</td>
<td>NUMBER</td>
<td>Parametri i ngjarjes se ne pritje</td>
</tr>
<tr>
<td>P1RAW</td>
<td>RAW(8)</td>
<td>Parametri i ngjarjes se ne pritje (hexadecimal)</td>
</tr>
<tr>
<td>P2TEXT</td>
<td>VARCHAR2(64)</td>
<td>Ngjarja se dyte ne pritje</td>
</tr>
<tr>
<td>P2</td>
<td>NUMBER</td>
<td>Përshkrimi i ngjarjes se dyte ne pritje</td>
</tr>
<tr>
<td>P2RAW</td>
<td>RAW(8)</td>
<td>Parametri i ngjarjes se ne pritje (hexadecimal)</td>
</tr>
<tr>
<td>P3TEXT</td>
<td>VARCHAR2(64)</td>
<td>Ngjarja se trete ne pritje</td>
</tr>
<tr>
<td>P3</td>
<td>NUMBER</td>
<td>Parametri i ngjarjes se trete ne pritje</td>
</tr>
</tbody>
</table>
Tabela e mëposhtme paraqet komandat të cilat mund të kontrollohen nga view e mësipërme. Edhe kjo tabelë është shumë e rëndësishme për të kuptuar se si është e ndërtuar SQL-ja, e cila do t’i nënshtrohet analizës nisur nga sesionet.

**Tabela 5. Lista e komandave bazuar mbi view V$SESSION**

<table>
<thead>
<tr>
<th>Nr</th>
<th>Komanda</th>
<th>Nr</th>
<th>Komanda</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CREATE TABLE</td>
<td>2</td>
<td>INSERT</td>
</tr>
<tr>
<td>3</td>
<td>SELECT</td>
<td>4</td>
<td>CREATE CLUSTER</td>
</tr>
<tr>
<td>5</td>
<td>ALTER CLUSTER</td>
<td>6</td>
<td>UPDATE</td>
</tr>
<tr>
<td>7</td>
<td>DELETE</td>
<td>8</td>
<td>DROP CLUSTER</td>
</tr>
<tr>
<td>9</td>
<td>CREATE INDEX</td>
<td>10</td>
<td>DROP INDEX</td>
</tr>
<tr>
<td>11</td>
<td>ALTER INDEX</td>
<td>12</td>
<td>DROP TABLE</td>
</tr>
<tr>
<td>13</td>
<td>CREATE SEQUENCE</td>
<td>14</td>
<td>ALTER SEQUENCE</td>
</tr>
<tr>
<td>15</td>
<td>ALTER TABLE</td>
<td>16</td>
<td>DROP SEQUENCE</td>
</tr>
<tr>
<td>17</td>
<td>GRANT OBJECT</td>
<td>18</td>
<td>REVOKE OBJECT</td>
</tr>
<tr>
<td>19</td>
<td>CREATE SYNONYM</td>
<td>20</td>
<td>DROP SYNONYM</td>
</tr>
<tr>
<td>21</td>
<td>CREATE VIEW</td>
<td>22</td>
<td>DROP VIEW</td>
</tr>
<tr>
<td>Number</td>
<td>Command</td>
<td>Number</td>
<td>Command</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>--------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>23</td>
<td>VALIDATE INDEX</td>
<td>24</td>
<td>CREATE PROCEDURE</td>
</tr>
<tr>
<td>25</td>
<td>ALTER PROCEDURE</td>
<td>26</td>
<td>LOCK</td>
</tr>
<tr>
<td>27</td>
<td>NO-OP</td>
<td>28</td>
<td>RENAME</td>
</tr>
<tr>
<td>29</td>
<td>COMMENT</td>
<td>30</td>
<td>AUDIT OBJECT</td>
</tr>
<tr>
<td>31</td>
<td>NOAUDIT OBJECT</td>
<td>32</td>
<td>CREATE DATABASE LINK</td>
</tr>
<tr>
<td>33</td>
<td>DROP DATABASE LINK</td>
<td>34</td>
<td>CREATE DATABASE</td>
</tr>
<tr>
<td>35</td>
<td>ALTER DATABASE</td>
<td>36</td>
<td>CREATE ROLLBACK SEG</td>
</tr>
<tr>
<td>37</td>
<td>ALTER ROLLBACK SEG</td>
<td>38</td>
<td>DROP ROLLBACK SEG</td>
</tr>
<tr>
<td>39</td>
<td>CREATE TABLESPACE</td>
<td>40</td>
<td>ALTER TABLESPACE</td>
</tr>
<tr>
<td>41</td>
<td>DROP TABLESPACE</td>
<td>42</td>
<td>ALTER SESSION</td>
</tr>
<tr>
<td>43</td>
<td>ALTER USER</td>
<td>44</td>
<td>COMMIT</td>
</tr>
<tr>
<td>45</td>
<td>ROLLBACK</td>
<td>46</td>
<td>SAVEPOINT</td>
</tr>
<tr>
<td>47</td>
<td>PL/SQL EXECUTE</td>
<td>48</td>
<td>SET TRANSACTION</td>
</tr>
<tr>
<td>49</td>
<td>ALTER SYSTEM</td>
<td>50</td>
<td>EXPLAIN</td>
</tr>
<tr>
<td>51</td>
<td>CREATE USER</td>
<td>52</td>
<td>CREATE ROLE</td>
</tr>
<tr>
<td>53</td>
<td>DROP USER</td>
<td>54</td>
<td>DROP ROLE</td>
</tr>
<tr>
<td>55</td>
<td>SET ROLE</td>
<td>56</td>
<td>CREATE SCHEMA</td>
</tr>
<tr>
<td>57</td>
<td>CREATE CONTROL FILE</td>
<td>59</td>
<td>CREATE TRIGGER</td>
</tr>
<tr>
<td>60</td>
<td>ALTER TRIGGER</td>
<td>61</td>
<td>DROP TRIGGER</td>
</tr>
<tr>
<td>62</td>
<td>ANALYZE TABLE</td>
<td>63</td>
<td>ANALYZE INDEX</td>
</tr>
<tr>
<td>64</td>
<td>ANALYZE CLUSTER</td>
<td>65</td>
<td>CREATE PROFILE</td>
</tr>
<tr>
<td>66</td>
<td>DROP PROFILE</td>
<td>67</td>
<td>ALTER PROFILE</td>
</tr>
<tr>
<td>68</td>
<td>DROP PROCEDURE</td>
<td>70</td>
<td>ALTER RESOURCE COST</td>
</tr>
<tr>
<td>71</td>
<td>CREATE MATERIALIZED VIEW LOG</td>
<td>72</td>
<td>ALTER MATERIALIZED VIEW LOG</td>
</tr>
<tr>
<td>73</td>
<td>DROP MATERIALIZED VIEW LOG</td>
<td>74</td>
<td>CREATE MATERIALIZED VIEW</td>
</tr>
<tr>
<td>75</td>
<td>ALTER MATERIALIZED VIEW</td>
<td>76</td>
<td>DROP MATERIALIZED VIEW</td>
</tr>
<tr>
<td>77</td>
<td>CREATE TYPE</td>
<td>78</td>
<td>DROP TYPE</td>
</tr>
<tr>
<td>79</td>
<td>ALTER ROLE</td>
<td>80</td>
<td>ALTER TYPE</td>
</tr>
<tr>
<td>81</td>
<td>CREATE TYPE BODY</td>
<td>82</td>
<td>ALTER TYPE BODY</td>
</tr>
<tr>
<td>83</td>
<td>DROP TYPE BODY</td>
<td>84</td>
<td>DROP LIBRARY</td>
</tr>
<tr>
<td>85</td>
<td>TRUNCATE TABLE</td>
<td>86</td>
<td>TRUNCATE CLUSTER</td>
</tr>
<tr>
<td>91</td>
<td>CREATE FUNCTION</td>
<td>92</td>
<td>ALTER FUNCTION</td>
</tr>
<tr>
<td>93</td>
<td>DROP FUNCTION</td>
<td>94</td>
<td>CREATE PACKAGE</td>
</tr>
<tr>
<td>95</td>
<td>ALTER PACKAGE</td>
<td>96</td>
<td>DROP PACKAGE</td>
</tr>
<tr>
<td>97</td>
<td>CREATE PACKAGE BODY</td>
<td>98</td>
<td>ALTER PACKAGE BODY</td>
</tr>
<tr>
<td></td>
<td>DROP PACKAGE BODY</td>
<td>100</td>
<td>LOGON</td>
</tr>
<tr>
<td>---</td>
<td>------------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>101</td>
<td>LOGOFF</td>
<td>102</td>
<td>LOGOFF BY CLEANUP</td>
</tr>
<tr>
<td>103</td>
<td>SESSION REC</td>
<td>104</td>
<td>SYSTEM AUDIT</td>
</tr>
<tr>
<td>105</td>
<td>SYSTEM NOAUDIT</td>
<td>106</td>
<td>AUDIT DEFAULT</td>
</tr>
<tr>
<td>107</td>
<td>NOAUDIT DEFAULT</td>
<td>108</td>
<td>SYSTEM GRANT</td>
</tr>
<tr>
<td>109</td>
<td>SYSTEM REVOKE</td>
<td>110</td>
<td>CREATE PUBLIC SYNONYM</td>
</tr>
<tr>
<td>111</td>
<td>DROP PUBLIC SYNONYM</td>
<td>112</td>
<td>CREATE PUBLIC DATABASE LINK</td>
</tr>
<tr>
<td>113</td>
<td>DROP PUBLIC DATABASE LINK</td>
<td>114</td>
<td>GRANT ROLE</td>
</tr>
<tr>
<td>115</td>
<td>REVOKE ROLE</td>
<td>116</td>
<td>EXECUTE PROCEDURÉ</td>
</tr>
<tr>
<td>117</td>
<td>USER COMMENT</td>
<td>118</td>
<td>ENABLE TRIGGER</td>
</tr>
<tr>
<td>119</td>
<td>DISABLE TRIGGER</td>
<td>120</td>
<td>ENABLE ALL TRIGGERS</td>
</tr>
<tr>
<td>121</td>
<td>DISABLE ALL TRIGGERS</td>
<td>122</td>
<td>NETWORK ERROR</td>
</tr>
<tr>
<td>123</td>
<td>EXECUTE TYPE</td>
<td>157</td>
<td>CREATE DIRECTORY</td>
</tr>
<tr>
<td>158</td>
<td>DROP DIRECTORY</td>
<td>159</td>
<td>CREATE LIBRARY</td>
</tr>
<tr>
<td>160</td>
<td>CREATE JAVA</td>
<td>161</td>
<td>ALTER JAVA</td>
</tr>
<tr>
<td>162</td>
<td>DROP JAVA</td>
<td>163</td>
<td>CREATE OPERATOR</td>
</tr>
<tr>
<td>164</td>
<td>CREATE INDEXTYPE</td>
<td>165</td>
<td>DROP INDEXTYPE</td>
</tr>
<tr>
<td>167</td>
<td>DROP OPERATOR</td>
<td>168</td>
<td>ASSOCIATE STATISTICS</td>
</tr>
<tr>
<td>169</td>
<td>DISASSOCIATE STATISTICS</td>
<td>170</td>
<td>CALL METHOD</td>
</tr>
<tr>
<td>171</td>
<td>CREATE SUMMARY</td>
<td>172</td>
<td>ALTER SUMMARY</td>
</tr>
<tr>
<td>173</td>
<td>DROP SUMMARY</td>
<td>174</td>
<td>CREATE DIMENSION</td>
</tr>
<tr>
<td>175</td>
<td>ALTER DIMENSION</td>
<td>176</td>
<td>DROP DIMENSION</td>
</tr>
<tr>
<td>177</td>
<td>CREATE CONTEXT</td>
<td>178</td>
<td>DROP CONTEXT</td>
</tr>
<tr>
<td>179</td>
<td>ALTER OUTLINE</td>
<td>180</td>
<td>CREATE OUTLINE</td>
</tr>
<tr>
<td>181</td>
<td>DROP OUTLINE</td>
<td>182</td>
<td>UPDATE INDEXES</td>
</tr>
<tr>
<td>183</td>
<td>ALTER OPERATOR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nisur nga këto të dhëna, të cilat paraqiten në dy tabelat e mësipërme, është e mundur që të menaxhohen sesionet në mënyrë të tillë, që të kontrollohen vetëm instruksionet SQL, të cilat janë të llojit DDL dhe janë përdorur për të ndikuar tek shtresa e sigurisë. Pra, nuk përshihen të gjitha instruksionet DDL dhe përjashtohen një pjesë e mirë e tyre dhe si rezultat do të kemi një performancë akoma më të mirë. P.sh. **Create Indexes** apo **user comment**, etj., nuk ndikojnë te elementet e sigurisë për faktin se këto lloj instruksionesh nuk modifikojnë të dhënët apo ndryshojnë statusin dhe logjikën e programeve PL/SQL, të cilët menaxhojnë sesionet.
Për rastet kur kemi një instrukson DDL, i cili është i drejtuar drejt objekteve të shtresës së sigurisë me qëllim modifikimin e tyre apo statusin e tyre, atëherë duhet që të analizohet kjo SQL (në rastet kur është e nevojshme), ose mund të ndërpritet ky sesion nëse një kërkesë e tillë vjen nga një përdorues, i cili nuk ka të drejta. Zgjidhja, e cila do të sëllte një performancë sa më të mirë do ishte analizimi jo në shumë detaje i instruksonit SQL, por të kontrollohej nëse është e drejtuar në drejtim të njërit nga objektet, të cilat janë pjesë e shtresës së sigurisë dhe nëse ndikon në statusin e saj apo në logjikën e saj. Kjo do të thotë që nuk duhet të kemi ekzekutime të llojeve Drop, Replace apo Alter. Objektet, të cilat do të mbrohen nga shtresa e sigurisë, do të jenë që të gjitha të përcaktuara në një tabelë, e cila është e ndërtuar në skemën, në të cilën do të vendeaset shtresë e sigurisë. Gjithashtu, duhet që shtresa e sigurisë të mos lejojë që të shtohen, ndryshohen apo të fshihen rekorde në tabelat e konfigurimeve. Edhe këtu mund të përdore logikat e shtjelluara më sipër, si ajo e menaxhimit të sesioneve (në këto raste kemi instrukson e DML), ashtu edhe nëpërmjet trigger-ave të ndërtuar për këto tabela. Me qëllim që të ndikojnë sa më pak në performancën e motorit të bazës së të dhënave, është më e përshtatshme përdorimi i trigger-ave.

5.2 Analiza e SQL

Për të analizuar rastet kur duhet të kontrollojmë instruksonet DDL, nëse këto ndikojnë tek shtresa e sigurisë, atëherë na vijnë në ndihmë disa view, të cilat paraqesin detaje për një instrukson të caktuar. View e mëposhtme përdoresë për të marrë informacionin e nevojshëm :

- v$sql: Përmban informacion të pagrupuar për çdo SQL, pra mban një rrjesht për çdo ekzekutim SQL.
- v$sqlarea: Përmban informacion të grupuar për çdo SQL, pra është v$sql pa dublime.
- v$sqltext: Përmban instruksonet SQL, por në rastet kur një instrukson është më i gjatë se 64 karaktere, atëherë ndahen në disa rrjeshta.

Për të bërë të mundur lidhjen ndërmjet sesioneve dhe view-eve të lartpërmdurë, mund të përdorim vlerat hash të tyre si në vijim:

```sql
select s.sid,
    s.serial#,
    a.sql_text
```
from v$session s
join v$sqlarea a on a.hash_value = s.sql_hash_value;

Më poshtë paraqitet struktura e view v$sql, e cila është edhe më e rëndësishmja në këtë proces.

**Tabela 6. View v$sql**

<table>
<thead>
<tr>
<th>Kolona</th>
<th>Lloji</th>
<th>Përqyshimi</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_TEXT</td>
<td>VARCHAR2(1000)</td>
<td>Permban njësime karakteret e para te instruksionit SQL.</td>
</tr>
<tr>
<td>SQL_FULLTEXT</td>
<td>CLOB</td>
<td>I gjithë instruksioni SQL.</td>
</tr>
<tr>
<td>SQL_ID</td>
<td>VARCHAR2(13)</td>
<td>ID e SQL</td>
</tr>
<tr>
<td>SHARABLE_MEM</td>
<td>NUMBER</td>
<td>Memorja ne byte e perëndorur</td>
</tr>
<tr>
<td>PERSISTENT_MEM</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>RUNTIME_MEM</td>
<td>NUMBER</td>
<td>Madhesia e memories e nevojshme per ekzekutim.</td>
</tr>
<tr>
<td>SORTS</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>LOADED_VERSIONS</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>OPEN_VERSIONS</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>USERS_OPENING</td>
<td>NUMBER</td>
<td>Numri i perëndoruesve që ekzekuton instruksioned SQL</td>
</tr>
<tr>
<td>FETCHES</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>EXECUTIONS</td>
<td>NUMBER</td>
<td>Numri i ekzekutimeve</td>
</tr>
<tr>
<td>PX_SERVERS_EXECUTIONS</td>
<td>NUMBER</td>
<td>Numri i ekzekutimeve ne paralel</td>
</tr>
<tr>
<td>END_OF_FETCH_COUNT</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>USERS_EXECUTING</td>
<td>NUMBER</td>
<td>Numri i perëndoruesve që ekzekuton kete SQL.</td>
</tr>
<tr>
<td>LOADS</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>FIRST_LOAD_TIME</td>
<td>VARCHAR2(19)</td>
<td>Koha kur eshte ngarkuar</td>
</tr>
<tr>
<td>INVALIDATIONS</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>PARSE_CALLS</td>
<td>NUMBER</td>
<td>Numri i thirjeve</td>
</tr>
<tr>
<td>DISK_READS</td>
<td>NUMBER</td>
<td>Numri i leximeve ne disk</td>
</tr>
<tr>
<td>DIRECT_WRITES</td>
<td>NUMBER</td>
<td>Numri i shkrimve direkte</td>
</tr>
<tr>
<td>BUFFER_GETS</td>
<td>NUMBER</td>
<td>Numri i leximit te buffer-it</td>
</tr>
<tr>
<td>APPLICATION_WAIT_TIME</td>
<td>NUMBER</td>
<td>Kohevonesa e aplikimit (ne microsekonda)</td>
</tr>
<tr>
<td>Column Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CONCURRENCY_WAIT_TIME</td>
<td>NUMBER</td>
<td>Koha e konkurences (Concurrency wait time) (ne microsekonda)</td>
</tr>
<tr>
<td>CLUSTER_WAIT_TIME</td>
<td>NUMBER</td>
<td>Koha e pritjes Cluster (ne microsekonda)</td>
</tr>
<tr>
<td>USER_IO_WAIT_TIME</td>
<td>NUMBER</td>
<td>Koha e pritjes I/O te perdonuesit (ne microsekonda)</td>
</tr>
<tr>
<td>PLSQL_EXEC_TIME</td>
<td>NUMBER</td>
<td>Koha e ekzekutimit PL/SQL (ne microsekonda)</td>
</tr>
<tr>
<td>JAVA_EXEC_TIME</td>
<td>NUMBER</td>
<td>Koha e ekzekutimit Java (ne microsekonda)</td>
</tr>
<tr>
<td>ROWS_PROCESSED</td>
<td>NUMBER</td>
<td>Numri i reshtave</td>
</tr>
<tr>
<td>COMMAND_TYPE</td>
<td>NUMBER</td>
<td>Lloji i komandes ne Oracle</td>
</tr>
<tr>
<td>OPTIMIZER_MODE</td>
<td>VARCHAR2(10)</td>
<td>Menyra se si eshte ekzekutuar SQL</td>
</tr>
<tr>
<td>OPTIMIZER_COST</td>
<td>NUMBER</td>
<td>Kostoja e optimizimit</td>
</tr>
<tr>
<td>OPTIMIZER_ENV</td>
<td>RAW(691)</td>
<td>Ambienti i optimizimit</td>
</tr>
<tr>
<td>OPTIMIZER_ENV_HASH_VALUE</td>
<td>NUMBER</td>
<td>Vlera Hash e ambientit te optimizimit</td>
</tr>
<tr>
<td>PARSING_USER_ID</td>
<td>NUMBER</td>
<td>ID e perdoruesit</td>
</tr>
<tr>
<td>PARSING_SCHEMA_ID</td>
<td>NUMBER</td>
<td>ID e skemes</td>
</tr>
<tr>
<td>PARSING_SCHEMA_NAME</td>
<td>VARCHAR2(30)</td>
<td>Emri i skemes</td>
</tr>
<tr>
<td>KEPT_VERSIONS</td>
<td>NUMBER</td>
<td>versioni</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>RAW(4</td>
<td>8)</td>
</tr>
<tr>
<td>TYPE_CHK_HEAP</td>
<td>RAW(4)</td>
<td>Lloji i kontrollit te heap-it</td>
</tr>
<tr>
<td>HASH_VALUE</td>
<td>NUMBER</td>
<td>Vlera Hash</td>
</tr>
<tr>
<td>OLD_HASH_VALUE</td>
<td>NUMBER</td>
<td>Vlera vjeter Hash</td>
</tr>
<tr>
<td>PLAN_HASH_VALUE</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>CHILD_NUMBER</td>
<td>NUMBER</td>
<td>Numri femijeve</td>
</tr>
<tr>
<td>SERVICE</td>
<td>VARCHAR2(64)</td>
<td>Emri i sherbimit</td>
</tr>
<tr>
<td>SERVICE_HASH</td>
<td>NUMBER</td>
<td>Vlera hash e sherbimit</td>
</tr>
<tr>
<td>MODULE</td>
<td>VARCHAR2(64)</td>
<td>Emri i modulit</td>
</tr>
<tr>
<td>MODULE_HASH</td>
<td>NUMBER</td>
<td>Vlera Hash e modulit</td>
</tr>
<tr>
<td>ACTION</td>
<td>VARCHAR2(64)</td>
<td>Veprimi</td>
</tr>
<tr>
<td>ACTION_HASH</td>
<td>NUMBER</td>
<td>Vlera Hash e veprimi</td>
</tr>
<tr>
<td>Serializable_aborts</td>
<td>NUMBER</td>
<td>Numri i deshtimeve</td>
</tr>
<tr>
<td>OUTLINE_CATEGORY</td>
<td>VARCHAR2(64)</td>
<td></td>
</tr>
<tr>
<td>CPU_TIME</td>
<td>NUMBER</td>
<td>Koha e CPU (ne microsekonda)</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>ELAPSED_TIME</td>
<td>NUMBER</td>
<td>Koha e perllogaritur</td>
</tr>
<tr>
<td>OUTLINE_SID</td>
<td>NUMBER</td>
<td>ID e sesionit</td>
</tr>
<tr>
<td>CHILD_ADDRESS</td>
<td>RAW(4</td>
<td>8)</td>
</tr>
<tr>
<td>SQLTYPE</td>
<td>NUMBER</td>
<td>Versioni i SQL te perdorur</td>
</tr>
<tr>
<td>REMOTE</td>
<td>VARCHAR2(1)</td>
<td>Ne largesi (remote)</td>
</tr>
<tr>
<td>OBJECT_STATUS</td>
<td>VARCHAR2(19)</td>
<td>Statusi:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALID_AUTH_ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALID_COMPILE_ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALID_UNAUTH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INVALID_UNAUTH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INVALID</td>
</tr>
<tr>
<td>LITERAL_HASH_VALUE</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>LAST_LOAD_TIME</td>
<td>VARCHAR2(19)</td>
<td>Koha e fundit e ngarkimit</td>
</tr>
<tr>
<td>IS_OBSOLETE</td>
<td>VARCHAR2(1)</td>
<td></td>
</tr>
<tr>
<td>CHILD_LATCH</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>SQL_PROFILE</td>
<td>VARCHAR2(64)</td>
<td>Profili SQL</td>
</tr>
<tr>
<td>PROGRAM_ID</td>
<td>NUMBER</td>
<td>ID e programit</td>
</tr>
<tr>
<td>PROGRAM_LINE#</td>
<td>NUMBER</td>
<td>Numri i linjes se programit</td>
</tr>
<tr>
<td>EXACT_MATCHING_SIGNA TURE</td>
<td>NUMBER</td>
<td>Normalizimi i SQL</td>
</tr>
<tr>
<td>FORCE_MATCHING_SIGNA TURE</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>LAST_ACTIVE_TIME</td>
<td>DATE</td>
<td>Koha e fundit aktive e planit te ekzekutimit</td>
</tr>
</tbody>
</table>

Duke përdorur view v$sql, e cila është paraqitur më sipër si dhe sesionet nga kur do të bëhet filtrimi i përdoruesve, si dhe llojit të komandave, të cilat do të lejohen, atëherë duhet që në fushën SQL_FULLTEXT të mos kemi asnjë nga llojet e instruksioneve të ndaluara, të cilat të jenë të orientuara në drejtim të objekteve, të cilat janë pjesë e shtresës së sigurisë. Pra, SQL_FULLTEXT duhet të jetë pjesë e kushtit where si më poshtë:

```sql

(SQL_FULLTEXT not like '%alter %' or
 SQL_FULLTEXT not like '%drop%' or
 SQL_FULLTEXT not like '%Replace%'

```
Deri në ketë moment është diskutuar se si është e mundur që të kapet një instruksion SQL, i cili është i llojit DDL dhe është i drejtuar drejt disa objekteve të caktuara dhe është në një grup të caktuara instruksionesh. Për të parandaluar ekzekutimin e këtyre instruksioneve duhet që sesioni, i cili ka një instruksion të palejuar, të ndërpritet dhe kjo përdoret duke përdorur instruksionin e mëposhtëm:

```
alter system kill session

Ky instruksion ka nevojë të specifikohet sid-i dhe numri serial i sesionit, të cilat në mund që t’i marrin nga view v$session.

Kodi në vijim është një shembull se si mund të ndërpriten disa sesione, të cilët janë nga një makinë me emrin ‘makina’ dhe janë loguar në instalçën e bazës së të dhënave si përdoruesi ‘SYS’

BEGIN

FOR c IN (

SELECT s.sid, s.serial#
FROM v$session s
WHERE (s.MACHINE = 'Makina')
AND s.USERNAME <> 'SYS'
AND s.STATUS <> 'KILLED'
)
LOOP

EXECUTE IMMEDIATE 'alter system kill session ''' ||
c.sid || ',' || c.serial# || '''';

END LOOP;

END;
```

Mënyra se si janë paraqitur gjërat në konceptimin e shtresës si dhe mënyra se si do të filtrohen instruksionet SQL të llojit DDL, nuk është gjithmonë e saktë sepse në rastet kur një instruksion përmbyt edhe komente, ku një pjesë e komentin është emri i një objekti, i cili është pjesë e shtresës së sigurisë, atëherë programi do ta konsiderojë si një...
sulm në drejtim të shtresës dhe për qëllim ai do e ndërpresë këtë session. Për të eliminuar këtë problem, duhet të ndërtohet një funksion i cili do të analizojë çdo pjesë të instrukzionit dhe të bazohet në hierarkinë e ekzekutimit të secilit lloj instruksoni, siç shpjegohet nga Oracle. Kjo do të ndikojë në performancën e analizës së instrukzionit SQL dhe rrjedhimisht edhe në performancën e bazës së të dhënhave.

Numri i këtyre gabime mund të minimizohet duke përdorur emërtime të ndryshme për objektet e shtresës, të cilat nuk janë të kuptimta ose që përmbyjnë karaktere të veçanta në emërtimin e tyre. Kjo nuk do të eliminohet plotësisht, por ul numrin e rasteve.

Një zgjidhje më e pranuashi do ishte mundësia e përdorimit të dy funksioneve, si ai i thjeshtë, ashtu edhe ai i ndërlikuari dhe të vendoset vetëm ai që është i nevojshëm pas analizave të ndryshme të performancës me të dyja rastet.

5.3 Ndalimi i leximit të kodit të programit të shtresës së sigurisë.

Në kapitullin e tretë është paraqitur mënyra se si mund të fshihihet kod i një objekti i ndërtaur në PL/SQL. Kjo mënyrë nuk do të është më e pranuashe kur ia është i kërkuar. Një ndërte me të madhe dhe të ishte mundur që të kthenim kodin në mënyrë të dekoduar, në rastet kur kishim kodin burim të koduar. Për të zgjidhur këtë problem duhet të sigurë së shtresës së sigurisë edhe mundësinë e kontrollit të instruksoni SQL, të cilat janë të llojet Select të drejtuar në drejtim të view-e * _SOURCE. Kjo mund të realizohet duke përdorur logikën e mësipërme ose, nëse është e mundur, të modifikojet instruksoni SQL, në mënyrë të tillë që të përijashtohen objektet e shtresës së sigurisë.

5.4 Baza e të dhënave virtuale

SMBD Oracle ofron gjithashtu edhe një komponent shumë të rëndësishëm në fushën e sigurisë së të dhënave. Ky komponent njihet me disa emërtime, si VPD (Virtual Private Databases), “Fine-grained access control” apo edhe “Row Level Security”. Pavarësisht nga emërtime, ky koncept, i cili mund të implementohet në SMBD Oracle, ofron një mbrojtje shumë të sigurt të të dhënave dhe menaxhohet nga DBA.

Në rastet kur implementohet VPD-ja, ajo bën të mundur që çdo instruksoni SQL, i cili është i orientuar në drejtim të objekteve, të cilat janë të varur nga të drejtat e përdoruesve dhe përmban informacion specifik, për sa i përkut rrjeshtave të tabelave apo view-e, të cilat
do të aksesohen të dhënat. Duke qenë se siguria kalon në nivel rrjeshti, kjo ka marrë emërtimin “Row Level Security”.


5.4.1 Si punon VPD-ja

Në momentin kur një përdorues, i cili ka të drejta të ekzekutojë instruksione SQL (Select) mbi një objekt të caktuar, SMDB Oracle fillimisht thërret funksionin e sigurisë, i cili kthen një kusht “Where” ku do të limitohet numri i rrjeshtave të kërkimit. Më pas VPD-ja transformon instruksionet SQL, duke u bashkëngjitur kushtin në kohë reale. Pas ekzekutimit, kemi një rezultat, ku janë të dhënët të filtruara sipas kushtit të vendosur nga VPD-ja.

Metodologjia VPD është përdorur gjerësisht për sistemet Oracle në Web, ku siguria duhet të mbahet në përputhje me numrin e përdoruesve, por në të njëjtën kohë të sigurojë një metodë, ku qasje të të dhënëve mund të kontrollohen sa më shumë me anë të proçedurave. Përdorimi i VPD-së për siguri në SMDB Oracle kërkon përdorimin e funksioneve PL/SQL për të përcaktuar logjikën e sigurisë.

Përdorimi i VPD-së na ofron:

- Siguri e shumëfishtë.
- Përdorimi i një baze të dhënash për shumë përdorues, në mënryrë që të mos ndikojnë tek të dhënët e njëri-tjetrit.
- Përdoruesit nuk mund që të anashkalojnë sigurinë, për faktin se filtrimi i të dhënave bëhet në server dhe jo në nivel aplikimi.

Në vijim kemi një shembull se si mund të përdoret VPD-ja në SMDB Oracle. Në këtë shembull, supozohet se një kompani përbëhet nga departamente të ndryshme (ku secili ka një të dhënë në tabelën e departamenteve) dhe çdo punonjës i takon pikërisht një departamentin. Një departament mund të ketë të dhëna sekrete, që mbahen në tabelën

\[ \text{Row Level Security} \]
department_secrets. Kodi për ndërtimin e objekteve të kësaj skeme do të jetë si më poshtë:

```sql
create table department
(
    dep_id int primary key,
    name    varchar2(30)
);

create table employee
(
    dep_id references department,
    name    varchar2(30)
);

create table department_secrets
(
    dep_id references department,
    secret varchar2(30)
);
```

Kodi i mëposhtëm përdoret për popullimin e të dhënave të tabelave të mësipërme.

```sql
insert into department values (1, 'Research and Development');
insert into department values (2, 'Sales');
insert into department values (3, 'Human Resources');

insert into employee values (2, 'Peter');
insert into employee values (3, 'Julia');
insert into employee values (3, 'Sandy');
insert into employee values (1, 'Frank');
insert into employee values (2, 'Eric');
insert into employee values (1, 'Joel');

insert into department_secrets values (1, 'R+D Secret #1');
insert into department_secrets values (1, 'R+D Secret #2');
insert into department_secrets values (2, 'Sales Secret #1');
```
insert into department_secrets values (2, 'Sales Secret #2');
insert into department_secrets values (3, 'HR Secret #1');
insert into department_secrets values (3, 'HR Secret #2');

For each employee, it must be possible to view all the secrets of their department, but no secret of another department.

To realize this requirement, we need to create a package, trigger and a group.

Më poshtë paraqitet paketa e kësaj pjesë të sigurisë.

create or replace package pck_vpd
as
  p_dep_id department.dep_id%type;

procedure set_dep_id(v_dep_id department.dep_id%type);

function predicate (obj_schema varchar2, obj_name varchar2) return varchar2;
end pck_vpd;
/

create or replace package body pck_vpd as

procedure set_dep_id(v_dep_id department.dep_id%type) is
begin
  p_dep_id := v_dep_id;
end set_dep_id;

function predicate (obj_schema varchar2, obj_name varchar2) return varchar2 is
begin
  return 'dep_id = ' || p_dep_id;
end predicate;

end pck_vpd;
Më pas, për çaktohet një trigger, i cili do të ekzekutohet sapo një përdorues lidhet me bazën e dhënave. Trigger-i do të gjejë se kujt departamenti i përket ky përdorues dhe do të thërrasë procedurën set_dep_id që është pjesë e paketës së mësipërme.

create or replace trigger trg_vpd
    after logon on database

declare
    v_dep_id department.dep_id%type;
begin

    select dep_id into v_dep_id
    from employee
    where upper(name) = user;

    pck_vpd.set_dep_id(v_dep_id);
end;
/

Pas ndërtimit të trigger-it duhet të vendoset rregullat e sigurisë, duke përdorur paketën dbms_rls, e cila ka procedurën add_policy që do të thirret si në vijim:

begin
    dbms_rls.add_policy
    (  
        user, -- përdoruesi
        'department_secrets', -- tabela
        'choosable policy name', -- emri i rregullit user,
        'pck_vpd.predicate', -- funksioni i kushti logjik
        'select,update,delete');
        -- rastet kur duhet te ekzektuohet ngjarja
    end;

Për VPD-në e ndërtuar më sipër po bëhen testet e mëposhtme, me qëllim që të kuptohet se si do të funksionojë.
Ndërtimi i përdoruesve:
create user frank identified by frank default tablespace users temporary tablespace temp;
create user peter identified by peter default tablespace users temporary tablespace temp;
create user julia identified by julia default tablespace users temporary tablespace temp;

Privilegjet:
grant all on department_secrets to frank;
grant all on department_secrets to peter;
grant all on department_secrets to julia;
grant create session to frank;
grant create session to peter;
grant create session to julia;

Sinonimet:
create public synonym department_secrets for department_secrets;

Frank (i cili është pjesë e R+D) ekzekuton SQL e mëposhtme dhe do të marrë rezultatin si në vijim:
connect frank/frank;
select * from department_secrets;
   DEP_ID SECRET
------- ------------------------
     1 R+D Secret #1
     1 R+D Secret #2

Peter (i cili është pjesë e Sales) ekzekuton SQL e mëposhtme dhe do të marrë rezultatin, si në vijim:
connect peter/peter;

select * from department_secrets;
   DEP_ID SECRET
2 Sales Secret #1
2 Sales Secret #2

Sic shikohet, edhe nga rezultati i ekzekutimit kemi një ndarje të të dhënavë për të dy përdoruesit, bazuar në departamentin e tyre.

5.4.2 Përdorimi i VPD-së në shtresën e sigurisë

A është e mundur që të përdoret koncepti i VPD-së si pjesë e skemës sonë të sigurisë?

Shtresa e sigurisë e konceptuar nga në modelin hibrid dhe përdor të njëjtën logjikë pune, të bazuar mbi trigger-at për instruksionet DML, por i vetmi ndryshim është që ajo bllokon të gjitha të dhënat dhe jo vetëm disa rrjeshta, sic bën të mundur VPD-ja. Dëri tani kemi diskutuar vetëm për instruksionet INSERT, UPDATE dhe DELETE, ndërsa për SELECT nuk është ndërtuar asnjë pjesë sepse nuk është e mundur që të përdoren trigger-at në ngjarjet SELECT. Për të realizuar këtë pjesë, mund të përdorim konceptin e VPD-së dhe si kusht logjik do të kemi një kusht, i cili vjen nga një funksion i ngjashëm me funksionin e kontrollit të sigurisë për tri rastet e tjera.

Nëse vendosim një kusht logjikë gjithmonë të vërtetë, si (1=1), atëherë ne nuk ndërpresim apo modifikojmë në asnjë mënyrë instruksionin SQL SELECT të përdoruesve, të cilët kanë të drejta, si dhe ndikimi i këtij lloj kushtit në performancë është pothuajse zero, sepse është githmonë i vërtetë. Në rastet kur duhet të kthehet rezultati negativ, pra që të mos i jepet e drejtë përdoruesit që të lexojnë të dhënat, edhe pse është DBA, por nuk është pjesë e sigurisë, atëherë do të kthehet kushti logjik (1=0), i cili nuk është asnjëherë i vërtetë dhe nuk ndikon as ky kusht në performancë.

Ky problem mund të bëhej edhe duke ndryshuar instruksionet SQL në momentin e monitorimit, por për qëllime performance do të përdoret VPD-ja.

5.5 Enkriptimi i të dhënavë në SMBD Oracle

Me rritjen e numrit të aplikimeve, të cilat janë online shtohen edhe sfidat për rritjen e sigurisë së të dhënavë, ku shumica e tyre mund të grupohen në mekanizmat tradicionalë të sigurisë si më poshtë:

- Përdorimi i fjalëkalimeve me një politikë të fortë autentifikimi.
• Të drejtat e përdoruesve për të cilat ai mund të shikojë apo të ndryshojë.
• Auditimi i të dhënave për çdo përdorues.
• Enkriptimi i të dhënave për të mbrojtur të dhënat sensitive gjatë transmetimeve në rrjet.

Enkriptimi është një komponent i rëndësishëm i disa prej këtyre zgjidhjeve. Shumë prej këtyre përdorimeve të enkripimeve janë relativisht transparente për përdoruesit ose aplikimet. Për shembull, shumë “Web Browser” suportojnë SSL-në dhe përdoruesit në përgjithësi nuk kanë nevojë të bëjnë ndonjë gjë të veçantë për të mundësuar enkriptimin SSL.

Oracle suporton enkriptim në nivel rrjeti ndërmjet bazës së të dhënave dhe klientëve, pavarësisht se cili version është në përdorim (duke filluar nga versioni 7).

Enkriptimi i të dhënave nuk është një zgjidhje e plotë të sigurisë së të dhënave, por për disa raste të veçanta është shumë i nevojshëm për përdorim i tij, si p.sh. në rastet kur kemi një bazë të dhënash, e cila përmban informacion të rëndësishëm si numrat e kartave të kreditit, fjalëkalimet, etj. Në këtë formë mund të sigurohet që edhe të dhënët në bazën e të dhënave janë të enkriptuara, ashtu edhe në transmetimet në rrjet janë të enkriptuara.

Edhe në rastet kur mund që të thyhet një bazë të dhënash për arsye të moskonfigurimit të saktë apo përdorimit të fjalëkalimeve shumë të thjeshtë apo për arsye të tjera, përdorimi i enkriptimit do të na siguronte që të dhënat të mos jenë të kuptueshme.

5.5.1 Principet e enkriptimit të të dhënave

Siç ka shumë arsye për ta përdorur enkriptimin e të dhënave, ashtu ka edhe arsye për mos e përdorur atë. Në disa raste, enkriptimi mund të përketësojë sigurinë, në vend që të përmirësojë atë. Në vijim jepen disa principe mbi përdorimin e enkriptimit.

• Enkriptimi nuk zgjidh problemën e aksesimit të të dhënave. Në shumicën e rasteve është e nevojshme që të bëhen limitime në të dhënat që aksesohen nga përdorues të ndryshëm, si p.sh., një sistem për menaxhimin e burimeve njerëzore, do t’i lejojë çdo përdoruesi që të ketë akses tek të dhënat e tij, ndërkohë që menaxhimi duhet të shikojë të dhëna për të gjithë punonjësit. Në këto raste nuk mund të përdoret enkriptimi i të dhënave, por duhet të përdorim mekanizmin e kontrollit të aksesimit të të dhënave.
Enkriptimi nuk ofron mbrojtje nga DBA-ja. Edhe pse Oracle ofron një mekanizëm kontrolli të aksesimit të të dhënave shumë të sigurt, është e pamundur që të eliminohen të gjitha problemet, p.sh. DBA edhe pse nuk duhet të shikojë apo të modifikojë të dhënat në pikëpamjen ligjore, ai teknikisht e ka të mundur aksesimin e tyre. Zgjidhja e këtyre rasteve realizohet duke përdorur VPD-në, por në mënyrë të tillë që DBA-ja të mos ketë mundësi ta ndryshojë. VPD-ja ka një konfirmim të një përgjegjësi ligjor mbi të dhënat.

Enkriptimi i të gjitha të dhënave nuk rrit sigurinë. Një nga gabimet më të shpeshqta që ndodh me enkriptimin e të dhënave është pikërisht edhe enkriptimi i të gjitha të dhënave. Kjo i jep mundësinë që një person, i cili kërkon që të gjejë çelësin e enkriptimit të ketë më shumë mundësi, duke ditur që një pjesë e të dhënave janë të njohtura për të, p.sh. një çelës primar, i cili rritet me një për çdo rast shtimi të një rekordi të ri. Gjithashtu, kjo do të çojë në ulje të performancës shumë sepse për çdo lexim, shkrim apo ndryshim duhet që të dhënat të dekriptohen fillimisht dhe më pas të kalojnë për t’u përpuñuar.

Gjithësia, përdorimi i enkriptimit në shumë raste është mëse e nevojshme. P.sh. në rastet kur duhet që të sigurohemi që disa të dhëna nuk duhet që të lexohen nga përdorues të veçantë, por njëkohësisht duhet edhe që këto të dhënë të tëbëhen backup në mënyrë të vazhdueshme. Në këto raste, nëse nuk do të përdorim enkriptimin, një DBA mund të marrë një eksportim të të dhënave dhe ta vendosë në një bazë të dhënash tjetër (makine tjetër por i njëjtë version i SMDB Oracle), e cila nuk përmban shtresën tonë të sigurisë dhe automatikisht ai mund të lexojë të dhënë. Kështu që duhet që të dhënat te enkriptohen në momentin e eksportimit të tyre (në momentin që janë offline).

Për të realizuar enkriptimin e të dhënave apo dekriptimin e tyre, Oracle ofron një paketë PL/SQL, e cila bën të mundur përdorimin e algoritmeve të ndryshëm për këtë qëllim. Paketa DBMS_CRYPTO ofron mundësinë e përdorimit të këtyre algoritmeve:

- Data Encryption Standard (DES), Triple DES (3DES, 2-key)
- Advanced Encryption Standard (AES)
- SHA-1 Cryptographic Hash
- SHA-1 Message Authentication Code (MAC)
Kapitulli 6 - **OBJEKTET PËRBËRËS TË SHTRESËS SË SIGURISË DHE FUNKSIONIMI I SAJ**

Me qëllim që të jetë sa më funksionale shtresa e sigurisë është shumë rëndësishme mënyra se si do të organizohet skema e sigurisë, me qëllim që të kemi sa më pak vonesa në rastet kur kërkohet të përpunohet informacioni. Duke qenë se mbi 99.9% e rasteve të aksesimit të të dhënave të shtresës së sigurisë do të bëhet nga instruksionet Select, atëherë është e nevojshme që skema të jetë e mirëindeksuar. Skema gjithashtu duhet të ofrojë mundësinë dhe të mbahen gjurmë mbështetet, të cilat shtresa e sigurisë i ka cilësuar si të rrezikshme.

Objektet e skemës do të jenë të organizuara në tabela, sekuencu dhe objekte të ndërtuara në PL/SQL.

**6.1 Tabletat**

Për ndërtimin e shtresës së sigurisë janë të nevojshme tabelat e mëposhte të cilat do të përdoren për konfigurime si dhe për të mbajtur gjurmë të ndryshme.

- **Tabela përdoresi.** Kjo tabelë do të mbajë të gjithë përdoresit të cilët do të jenë pjesë e shtresës së sigurisë, ku secili përdores do të ketë një id unike për secilin përdores, i cili shtohet në shtresën e sigurisë, emërtimin e tij, statusin nëse është aktiv apo jo, datën e regjistrimit si dhe datën e ndryshimit të statusit. Me qëllim që të minimizohet sa më shumë koha për aksesimin e të dhënave, atëherë id do të jetë një fushë numerike, e cila do të rritet pas cdo rekordi që do të vendoset në ketë tabelë. Për të realizuar këtë proces duhet të ndërtohet një trigger, i cili do të vetekzekutohet në momentin para shtimit të një rekordi të ri. Gjithashtu, me anë të trigger-ave do të kontrollohet nëse një DBA kërkon që të ndryshojë të dhënët në këtë tabelë apo ndryshimi po bëhet nëpërëmjet proçedurës për regjistrimin përdoresit nga personi i autorizuar.

- **Tabela Objekti.** Në këtë tabelë do të mbahen informacione mbi ato tabela të cilat bëjnë pjesë në shtresën e sigurisë. Kjo tabelë do të jetë e përbërë nga objekti_id (numerike që rritet për cdo objekt dhe cëlës primar), përdoresi në të cilin bën pjesë kjo tabelë, emërtimin e tabelës, statusi, data e regjistrimit si dhe data e ndryshimit të fundit. Edhe për këtë tabelë është e nevojshme të
ndërtohen trigger-at e nevojshme, të cilët do të mbrojnë të dhënët e saj nga DML të palejueshme si dhe gjenërimi të id, duke qenë se SMDB Oracle nuk ofron mundësinë kolonave autoinkrementale.

- **Tabela roli.** Për të ndërtuar një skemë sa më funksionale dhe të jetë sa më e thjeshtë në përdorim, është e nevojshme që të ndërtohen edhe rolet, të cilat nuk janë gjë tjetër veçse një grupim privilegjesh mbështetën objekte të caktuara. Për të ndërtuar rolet është e nevojshme që të përdorët tabela roli, e cila do të ketë në strukturën fushën roli_id numerike dhe që rriritet për çdo rekord të ri, emërtimin e roilit, statusin e roilit, datën e regjistrimit dhe datën e ndryshimit. Në të njëjtën mënyrë, si edhe tabelat e mësipërme, duhet që të sigurohet mbrojtja e të dhënave në këtë tabeli si dhe gjenërimi i roli_id në momentin e shtimit.

- **Tabela roli_përdorues.** Për çdo përdorues, i cili është pjesë e shtresës së sigurisë, duhet të mbahen rolet që ky përdorues gëzon në mënyrë të tillë, që të kontrollohet e drejta mbi objekte të caktuara të kësaj shtrese. Kjo tabelë do të jetë e përmbërc nga fusha roli_id, e cila vjen nga tabela roli, përdoruesi_id që vjen nga tabela përdoruesi, statusi, data e regjistrimit, data e ndryshimit. Në të njëjtën mënyrë si edhe tabelat e mësipërme duhet që të sigurohet mbrojtja e të dhënave në këtë tabeli.

- **Tabela roli_objekt.** Çdo rol i ndërtuar duhet që të ketë edhe të drejta të caktuara mbi objektet, të cilat do të merren në mbrojtje nga shtresa e sigurisë. Për të mbajtur këto të dhëna duhet të përdorët tabela roli_objekt, e cila përmban fushat roli_id që vjen nga tabela roli, objekti_id që vjen nga tabela objekti, statusi, data e regjistrimit, data e ndryshimit, si dhe fushat, të cilat përcaktojnë privilegjet Select, Insert, Delete dhe Update. Privilegjet mund të mbaheshin edhe në një tabelë jetë të mënjë që të kontrollojen për një objekt që i përkët një roli të caktuari, të mos kontrollohen 4 rekorde, janë vendosur në një të vetëm. Kjo mënyrë organizimi minimizon efektet e ndikimit në uljen e performancës nga shtresa e sigurisë. Në të njëjtën mënyrë, si edhe tabelat e mësipërme, duhet që të sigurohet mbrojtja e të dhënave në këtë tabeli.

- **Tabela konfig.** Shtresa e sigurisë ka nevojë për konfigurime të ndryshme, të cilat mund të jenë dinamike, si p.sh. statusi nëse skema është aktive, fjalëkalimi i shtresës, informacion se si mund të ndryshohet fjalëkalimi në rastet kur humbet ai, etj. Këto të dhëna do të mbahen në tabelën konfig, e cila
do të përbëhet nga konfig_id numerike (në rastet kur kemi një kuptim të përçaktuar në kod për çdo id ose mund të jetë edhe varchar2 në rastet kur do të përdoren emërtimi si p.sh. statusi), fusha vlera e cila mbështet për një id të caktuar, statusi, data e regjistrimit, përdoruesi i regjistrimit dhe data e ndryshimit të saj. Edhe kjo tabelë ka nevojë për mbrojtjen e të dhënave, e cila
do të bëhet në të njëjtën mënyrë si në rastet e mësipërme duke përdorur trigger-at.

- Tabela ddl_log. Për të analizuar çdo instrukpcion DDL, e cila orientohet drejt objekteve, të cilat janë nën mbrojtjen e shtresës së sigurisë apo përdoruesit, që mund të ndryshohen për qëllime përdorimi të paautorizuar të tyre, si p.sh ndryshimi i fjalëkalimit, duhet që të mbahen në një tabelë të caktuar. Kjo tabelë përbëhet nga ddl_log_id fushë numerike, e cila rritet për çdo rekord të ri në të, fusha DDL_ekzekutuar e cila përmban të gjithë instrukpcioni DDL, koha e ekzekutimit (data, ora), përdoruesi i cili ka ekzekutuar instrukpcionin, nrreziku i instrukcionit, kompjuteri nga ku vjen ky instrukcion, statusi i tij (analizuar ose jo). Edhe kjo tabelë ka nevojë për mbrojtjen e të dhënave, e cila do të bëhet në të njëjtën mënyrë si në rastet e mësipërme duke përdorur trigger-at, të cilët gjithashtu do të gjenerojnë ndryshimi nga asnjë përdorues, por vetëm do të shtohen rekorde nga procedurat e shtresës së sigurisë.

- Tabela sesione_vrare. Kjo tabelë do të mbajë të gjithë sesionet në momentin që ndodh një ndërprerje e sesionit nga shtresa e sigurisë me qëllim mbrojtjen e vetë shtresës. Përbëhet nga fusha sesion_vrare_id numerike dhe që rritet, data dhe ora, kompjuteri nga ku është ekzekutuar instrukpcioni i rrezikshëm, instrukpcioni, përdoruesi. Edhe kjo tabelë ka nevojë për mbrojtjen e të dhënave, e cila do të bëhet në të njëjtën mënyrë si në rastet e mësipërme duke përdorur trigger-at.

### 6.2 Sekuencat

Duke qenë se në SMDB Oracle nuk ka fusha të llojit “Autonumber” dhe nevojitet që si çelës të përdoret një fushë numerike e cila rritet sa here qe shtohet një e dhënë atëherë do te përdorim sekuencat si në vijim:
- **Sekuenca seq_përdoruesi_id.** Kjo sekuençë do të përdoret nga trigger-i para ngjarjes së shtimit të një rekordi në tabelën përdoruesi.

- **Sekuenca seq_roli_id.** Kjo sekuençë do të përdoret nga trigger-i para ngjarjes së shtimit të një rekordi në tabelën roli.

- **Sekuenca seq_objekti_id.** Kjo sekuençë do të përdoret nga trigger-i para ngjarjes së shtimit të një rekordi në tabelën objekti.

- **Sekuenca seq ddl_log_id.** Kjo sekuençë do të përdoret nga trigger-i para ngjarjes së shtimit të një rekordi në tabelën ddl_log.

- **Sekuenca seq_sesion_vrare_id.** Kjo sekuençë do të përdoret nga trigger-i para ngjarjes së shtimit të një rekordi në tabelën sesion_vrare.

### 6.3 Objektet PL/SQL

Pjesa kryesore e logjikës së shtresës së sigurisë është përfshirë në procedurat dhe funksionet, të cilat janë pjesë e pakteqës së sigurisë. Paketa, e cila do të jetë e koduar dhe e mbrojtur në mënyrë që të mos ndryshohet mënyra se si punon kjo shtresë, është e përbërë nga funksionet dhe procedurat e mëposhtme:

- **Funksioni s_kontrollo_objekt.** Ky funksion ka qëllim që të kontrollojë nëse një përdorues i caktuar ka të drejtë të ekzekutojë një DML të drejtuar drejt një objekti të caktuar, në rast se ka të drejtë atëherë do të kthejë vlerën 1 dhe në rast të kundër të 0. Thirrja e këtij funksioni do të behet nga trigger-at në mënyrën e mëposhtme:

    ```sql
    CREATE OR REPLACE TRIGGER X_test
    before INSERT OR UPDATE OR DELETE ON aplikimi.objekti
    FOR EACH ROW
    DECLARE
        sys_time      DATE;
        terminal      varchar2(15);
        perdoruesi   varchar2(30);
        veprimi       varchar2(30);
    BEGIN
        /* *******************************************************
        * Kod i gjeneruar automatikisht
        */
    ```

* Trigger auditimi mbi aplikimi.objekti

BEGIN
    sys_time := SYSDATE;
    terminal := SUBSTR(USERENV('TERMINAL'), 1, 15);
    perdoruesi:= SUBSTR(upper(user), 1,30);

    IF INSERTING THEN
        veprimi   :=   'INSERT';
    ELSIF DELETING THEN
        veprimi   :=   'DELETE';
    ELSE
        veprimi   :=   'UPDATE';
    END IF;

    if s_kontrollo_objekt(perdoruesi, veprimi, 'aplikimi.objekti')=0
    then
        raise_application_error(-00001,'Ju nuk keni te drejta per kete veprim');
    end if;
except
    when others then
        null;
end;
END;

- **Funksioni s_kontrollo_objekt_kushti.** Ky funksion ka si qëllim që të kontrollojë nëse një përdorues i caktuar ka të drejtë të ekzekutojë një Select të drejtuar drejt një tabele të caktuar, në rast se ka të drejtë atëherë do të kthejë vlerën (1=1) dhe në rast të kundërt (0=1). Ky funksion do të përdoret nga VPD-ja në rastet kur është e nevojshme, me qëllim ndalimin e leximit të vlerave të një tabele të caktuar dhe nuk ndikon në performancë, duke qenë se ky si kusht nuk harxhon shumë cikle makine për vërtetësinë e tij. Thirrja e tij bëhet me të njëjtat parametra si funksioni s_kontrollo_objekt.


- **Procedura aktivizë_sigurine.** Kjo procedurë bën të mundur ndryshimin e statusit të shtresës së sigurisë, e cila merr dy parametra që janë fjalëkalimi dhe statusi që do të jetë aktiv apo pasiv. Kjo procedurë fillimisht kontrollon nëse fjalëkalimi është i vlefnëshm dhe më pas ndryshon statusin e shtresës së sigurisë. Kjo procedurë do të përdoret nga një përdorues, i cili ka të drejta për një vendimmarrje të tillë.

- **Procedura ndrysho_fjalëkalim.** Kjo procedurë bën të mundur ndryshimin e fjalëkalimit të shtresës së sigurisë, i cili mbahet në tabelat e konfigurimit në mënyrë të enkriptuara. Kjo procedurë thirret me anë të tri parametra, që janë fjalëkalimi i vjetër, fjalëkalimi i ri dhe konfirmimi. Gjithashtu në rast rezultati të suksesshëm kthen vlerën 1 në rast rezultati të pasuksesshëm vlerën 0 si dhe nëse kemi ndonjë gabim teknik kthen id e gabimit. Procedura duke përdorur të dhënët e konfigurimit mund të kërkojë që një fjalëkalim i ri të plotësojë disa kritere të veçanta, si p.sh gjatësia, përdorimi i një karakteri jo shkronje, etj., të cilat bëjnë të mundur që fjalëkalimi të jetë më i sigurt. Një veçori tjetër e këtij funksioni është se në të mund të vendosim edhe kohë vonesa të caktuar apo numër përdorimesh të limituara që do të eliminonin thyerjen e fjalëkalimit me metodat brute-force.

- **Procedura humbur_fjalëkalimi.** Kjo procedurë bën të mundur ndryshimin e fjalëkalimit aktual, në rastet kur është humbur ai. Përdorimi i tij bëhet duke i dhënë si parametër të dhëna, të cilat janë regjistruar më parë në tabelën e konfigurimit me anë të këtij funksioni. Këtu mund të përcaktohet që fjalëkalimi i ri mund të ndryshohet vetëm nga një kompjuter i caktuar, në një orar të caktuar, etj., me qëllim rritjen sa më shumë të sigurisë.

- **Procedura shto_objekt.** Kjo procedurë realizon shtimin e një objekti të caktuar në shtresën e sigurisë ku si parametrët merr përdoruesin, i cili ka në zotërim objektin si dhe emrin e tabelës që do të shtohet. Mund të përdoren dy versione të kësaj procedurë ku e para lejon përdorimin e një trigger-i për të përcaktuar id e objektit dhe e dyta e përshin brenda trupit të saj edhe pjesën e gjinerimit të id nga sekuencia. Versioni i dytë është më i sigurt, duke qenë se trigger-at nuk mund të kodohen ashtu si edhe procedurat. Procedura përpara se të shtojë këtë objekt do të kontrollojë, nëse tabela që po shtohet, është pjesë active e shtresës së sigurisë me qëllim eliminimin e dublikimeve të rekordeve. Me qëllim që procedura të mos ekzekutohet nga çdo përdorues,
duhet që të kontrollohet edhe fjalëkalimi për të realizuar këtë veprim. Në momentin që shtohet një objekt i ri (tabelë e re) në shtresën e sigurisë atëherë automatikisht do të thirret proçedura, e cila gjeneron trigger-in për këtë tabelë. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Proçedura hiq_objekt.** Kjo proçedurë bën të mundur që një tabelë, e cila është pjesë e shtresës së sigurisë të hiqet nga objektet e shtresës së sigurisë. Në rastin e kësaj proçedurë bëhet vetëm ndryshimi i statusit në tabelën objekt dhe në tabelën rol_i_objekt dhe jo fshirjen e tyre. Kjo bëhet me qëllim që të mos kemi humbje të historikut të veprimeve mbi rolet apo mbi objektet, që kanë qenë pjesë e shtresës së sigurisë. Me qëllim që proçedura të mos ekzekutohet nga çdo përdorues, duhet që të kontrollohet edhe fjalëkalimi për të realizuar këtë veprim. Në momentin që shtohet një objekt i ri (tabelë e re) në shtresën e sigurisë, atëherë automatikisht do të thirret procedura e cila do të fshijë trigger-in për këtë tabelë. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Proçedura ndërto_rol.** Kjo proçedurë ka si qëllim që të shtojë role të reja, të cilat nuk ekzistojnë në listën e roleve aktiv në shtresën e sigurisë. Fillimisht, proçedura kontrollohet nëse ekziston një rol i tillë aktiv dhe nëse jo atëherë shton këtë rekord në tabelën duke marrë nga sekuenca id e këtij roli (id mund që të vendoset edhe me trigger). Me qëllim që proçedura të mos ekzekutohet nga çdo përdorues, duhet që të kontrollohet edhe fjalëkalimi për të realizuar këtë veprim. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Proçedura fshi_rol.** Kjo proçedure ka për qëllim të kalojë me status joaktiv një rol aktiv, i cili mund të këtë edhe të drejta për objekte të ndryshme të caktuara, në cilët bëjnë pjesë në shtresën e sigurisë me status aktiv. Proçedura kontrollojnëse statusi i rolit të dhënë si parametër është aktiv dhe nëse po, atëherë bën ndryshim menjëherë, pasi është bërë kontrolli nëse kjo proçedurë është ekzekutoar nga persona të autorizuar, duke përdorur fjalëkalimin e shtresës së sigurisë. Duke qenë se kontrolli i të drejtave bëhet para ekzekutimit të çdo instruksioni DML atëherë nuk është e nevojshme që një përdorues, i cili gëzon një rol, të ndërpresë lidhjen e tij në momentin që
fshihet ky rol. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Procedura shto_rol_përdorues.** Kjo procedurë përdoret në rastet kur duhet të shtojmë një rol të caktuar, i cili ekziston dhe është aktiv për një përdorues, i cili gjithashtu duhet të jetë në tabelën e përdoruesve dhe me status aktiv. Gjatë ekzekutimit të kësaj procedurë, përcalëse se të shtohen të dhënat e reja në tabelën roli_përdorues, bëhen verifikimet e nevojshme në mënyrë të tillë që të mos kemi dublikime të të dhënat dhe të kemi një si rolin ashtu edhe përdoruesin të vlefshëm. Kjo procedurë merr tri parametra që janë emri i rolit, emri i përdoruesit, si dhe fjalëkalimi i shtresës së sigurisë. Në rastet kur në tabelë ekziston një rekord, i cili përputhet me parametrat me të cilën është thirrur kjo procedurë, kontrollohen statusi nëse është aktiv si rekord dhe nëse jo, atëherë ndryshohet statusi nga joaktiv në aktiv. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Procedura hiq_rol_përdorues.** Kjo procedurë përdoret në rastet kur duhet të hiqim një rol të caktuar, i cili ekziston dhe është aktiv për një përdorues, i cili gjithashtu duhet të jetë në tabelën e përdoruesve dhe me status aktiv. Kjo procedurë merr tri parametra, që janë emri i rolit, emri i përdoruesit si dhe fjalëkalimi i shtresës së sigurisë. Në rastet kur në tabelë ekziston një rekord, i cili përputhet me parametrat me të cilën është thirrur kjo procedurë, kontrollohen statusi nëse është aktiv si rekord dhe nëse jo, atëherë ndryshohet statusi nga joaktiv në aktiv. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Procedura eksport_sesione_vrare.** Me anë të kësaj procedurë është e mundur që të eksportohen të dhënat nga tabela sesione_vrare në një format CSV me qëllim që të analizohen të dhënat më në detaje. Procedura ka nevojë për disa parametra, me qëllim që të kemi rezultatet e kërkuara. Dy nga këto parametra përcaktojnë datën e fillimit dhe datën e përfundimit të rezultateve që kërkojen, një parametër të cilët përcakton adresën ku do të vendoset dosja CSV, e cila do të përmbajë informacionit. Duke qenë se baza e të dhënave nuk lejon që të shkruhet në çdo vend të hapësirës, që menaxhohet nga sistemi operativ, atëherë është e nevojshme që të specifikohet një adresë, ku baza e të dhënave ka të drejtë për shkrim. Me qëllim që procedura të mos ekzekutohet nga çdo përdorues duhet që të kontrollohet edhe fjalëkalimi për të realizuar
këtë veprim. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Procedura eksport_ddl_log.** Me anë të kësaj procedurë, është e mundur që të eksportohen të dhënat nga tabela ddl_log në një format CSV, me qëllim që të analizohen të dhënat më në detaje për të kuptuar se çfarë instruksionesh DDL janë ekzekutuar gjatë një periudhë kohore te kaktuar. Procedura ka nevojë për katër parametra me qëllim që të kemi rezultatet e kërkuara. Dy nga këto parametra përcaktojnë datën e fillimit dhe datën e përfundimit të rezultateve që kërkojen, një parametër tjetër përcakton adresën ku do të vendoset dosja CSV, e cila do të përmbanë informacionin. Duke qenë se baza e të dhënave nuk lejon që të shkruhet në çdo vend të hapësirës që menaxhohet nga sistemi operativ, atëherë është e nevojshme që të specifikohet një adresë, ku baza e të dhënave ka të drejtë për shkrim. Me qëllim që procedura të mos ekzekutohet nga çdo përdorues, duhet që të kontrollohet edhe fjalëkalimi për të realizuar këtë veprim. Në rast ekzekutimi me sukses kthehet vlera 1 dhe në të kundërt vlera 0.

- **Procedura shto_vpd.** Kjo procedurë përdoret për rastet kur duhet që një tabelë e caktuar t’i nënshtrohet logjikës VPD, me qëllim që të blokohen të drejtat për leximin e të dhënave, e cila përdor funksionin që kthen kushtin për aksesimin e të dhënave, në rastet kur duhet të lejohet kthen (1=1) dhe në rastin e kundërt kthehet (1=0). Kjo procedurë thirret vetëm në rastet kur duhet të blikohen instrukSIONE Select.

- **Procedura hiq_vpd.** Kjo procedurë ekzekutohet në rastet kur një tabelë, e cila është në shtresën e sigurisë dhe një rol, i cili kishte privilegjin Select mbi këtë tabelë, e humbet këtë privilegji. Kjo bën të mundur që kjo tabelë të dalë nga skema e VPD-së.

- **Procedura monitoro_siguria.** Kjo procedurë është baza e vetëmbrojtjes së shtresës së sigurisë, e cila do të ekzekutohet automatikisht me fillimin e punës së SMBD Oracle. Qëllimi i kësaj procedurë është që të monitorojë të gjithë instruksionet DDL, të cilat do të drejtohen drejt objekteve, të cilat janë në shtresën e sigurisë si dhe instruksonë të tjera DDL, të cilat mund të sjellin probleme sigurie për shtresën. Për të optimizuar sa më shumë performancën, një pjesë e instruksonëve janë të koduara brenda trupit të procedurës si p.sh: `drop` e orientuar drejt procedurave të sigurisë apo ndonjë tjetër që vret...
sesionin e sigurisë. Në rastet kur procedura gjen një rast, i cili përbën rrezik, atëherë kjo e dhënë shtohet në tabelën ddl_log dhe kur ky sesion vritet, atëherë vendoset në tabelën sesionе_vrare. Proçedura bëhet joaktive në rastin kur është kërkuar nga administratori i shtresës së sigurisë që të ndërpresë mbrojtjen me anë të proçedurës aktivizo_sigurine. Kjo proçedurë nuk ka nevojë për fjalëkalim sepse duhet të ekzekutohet automatikisht pa qenë e nevojshme që të thirret nga ndonjë përdorues.

- **Proçedura fshi_shtresën.** Kjo proçedurë bëhen të mundur që të fshihet shkresa e sigurisë nga baza e të dhënave. Kjo proçedurë i vendoset përdoruesit sys dhe thirret vetëm në rastet kur në sistem jemi futur si SYSDBA.

Fillimisht shtresa e sigurisë do të jetë e plotësuar me rolin bazë, i cili merr në mbrojtje vetë tabela e shtresës së sigurisë së qëllim që të kemi edhe këtu gjenem automatik të trigger-ave.

### 6.4 Mënyra e përdorimit të shtresës, bazuar në objektet e saj

Shtresa e sigurisë, e cila bazohet mbi objektet e larçermendura do të përdoret nga një përdorues, i cili do të menaxhojë shtresën dhe nuk është e nevojshme këtë njohuri mbi menaxhimin e bazës së të dhënave apo të këtë njohuri gjuhën SQL. Për të realizuar këtë objektiv duhet të ndërtohet një ndërfaqë grafike e cila duhet të realizojë kërkesat e mëposhtme:

- **Lidhja me bazën e të dhënave nëpërmjet përdoruesit sys, i cili është edhe përdoruesi me të drejta më të shuma me qëllim instalimin e shtresës së sigurisë (ndërtrimi i skemës së sigurisë për vendosjen e objekteve).**
- **Moduli për ndryshimin fjalëkalimit të shtresës së sigurisë si dhe vendosja e mënyrave për rikuperimin e fjalëkalimit në rastet kur humbet ai.**
- **Moduli për importimin e përdoruesve të cilët do të jenë pjesë e shtresës së sigurisë.** Lista e përdoruesve që janë në bazën e të dhëna mund të merret me anë të instruksionit SQL si në vijim:

```sql
select t.username, t.user_id
from dba_users t
```
• Moduli për përcaktimin e objekteve (tabelave) të cilat do të jenë nën mbrojtjen e shtresës së sigurisë. Kjo listë do të ngarkohet duke përdorur SQL në vijim:

```sql
SELECT owner,
       table_name
FROM all_tables
```

Ose në raste kur njohim përdoruesin,

```sql
SELECT table_name
FROM user_tables
```

• Moduli për ndërtimin e rolit, i cili do të mundësojë ndërtimin e një roli të caktuar dhe përcaktimin e të drejtave mbi objekte të ndryshme, të cilat janë pjesë e shtresës së sigurisë.

• Moduli për eksportimin e ddl_log dhe sesione_vrare.

Të gjitha këto module mund të ndërtohen nga gjuhë të ndryshme programimi, mjafton që të kemi një lidhje me bazën e të dhënave, me qëllim që të ekzekutojmë instruksione të ndryshme SQL (vetëm Select) dhe të thirren procedurat e ndërtuara dhe të shpjeguara më sipër, duke plotësuar parametrat e nevojshme. Në vijim kemi një shembull se si mund të thirret një procedurë nga VB.Net, e cila është në një bazë të dhënash Oracle.

```vbnet
Dim x As Exception
Dim Ds As New DataSet()
Dim Oraclecon As New OracleConnection("Server=DB;Uid=uid;Pwd=pwd")

Oraclecon.Open()

Dim myCMD As New OracleCommand()
myCMD.Connection = Oraclecon
myCMD.CommandText = "paketa.procedura_1"
myCMD.CommandType = CommandType.StoredProcedurë
myCMD.Parameters.Add(New OracleParameter("p1",
                                         OracleType.Number)).Value = 1
```
myCMD.Parameters.Add(New OracleParameter("p2", OracleType.Cursor)).Direction = ParameterDirection.Output

Dim MyDA As New OracleDataAdapter(myCMD)

Try
    MyDA.Fill(Ds)
    Catch x
        MessageBox.Show(x.Message.ToString)
    End Try

DataGrid1.DataSource = Ds.Tables(0)
Oraclecon.Close()

Duke qenë se përdorimi nga ana jonë për testime ka nevojë që të kontrollohen informacione të tjera, të cilat do të japin detaje të shumta për funksionimin e shtresës së sigurisë, atëherë do të jetë më e vlefshme thirrja e proçedurave në mënyrë manuale duke përdorur ndërfaqe si SQL-plus apo SQL-Developer.

6.5 Mënya e funksionimit të shtresës

Në momentin që instalohet shtresa e sigurisë, ajo kalon me statusin aktiv dhe automatikisht mbrobëj objektet e saj në mënyrë që të mos këtë akses për të ndryshuar apo për të lexuar të dhënët, që ndodhen në këto tabel. Në këtë moment është e nevojshme që të bëhen konfigurimet e nevojshme, si përcaktimi i objekteve, përcaktimi i roleve, përcaktimi i të drejtave të çdo përdorues duke u dhënë rolë të caktuara.

Pas konfigurimit të proçedurës së të drejtave, shtresa e sigurisë automatikisht ndërton trigger-at, të cilët më pas do t’i nënshtrohen edhe mbrojtjes me qëllim që një përdorues, i cili është me të drejta DBA, të mos ndryshojë atë apo statusin e tij nga aktiv në status pasiv gjë, që do të çojë në lejimin e modifikimit të të dhënave në tabelën ku bën pjesë ky trigger.

Analiza e instruksioneve, të cilët bëjnë në mundur kontrollin dhe identifikimin e një sulmi të caktuar, bëhet me anë të monitorimit të sesioneve dhe filtrimit te instruksioneve DDL, të cilat janë të orientuara në drejtim të objekteve të shtresës së sigurisë ku bën pjesë edhe trigger-i. I gjithë ky proçes menaxhohet me anë të proçedurës monitoro_siguria, e cila ekzekutohet apo starton baza e të dhënave. Kjo proçedurë është vazhdimisht në punë dhe nuk lejon në asnjë mënyrë ndërprerjen e monitorimin apo
vrasjen e kësaj proçedure nga një përdorues tjetër, duke i analizuar instruksionet që dërgohen. Proçedura nuk është aktive vetëm në rastet kur administratori i shtresës së sigurisë ka kërkuar që të ndërresë mbrojtjen e shtresës së sigurisë me anë të proçedurës aktivizo_sigurine, e cila duhet të thirret sipas karakteristikave të saj, të cilat janë përcaktuar më parë.

Duke e parë nga mënyrat e përdorimit kësaj shtrese, arrihet në konkluzionin që kjo shtresë ka dy lloje përdorimesh:

- Përdorimi për menaxhimin e statusit të saj.
- Përdorimi për konfigurimin e saj.

Për ta thjeshtuar sa më shumë këtë proçes, mund që ndërfaqja grafike të konceptohet mbi këtë logjikë, por me një kusht tjetër që proçedurat e konfigurimit të thirren vetëm në rastet kur shtresa e sigurisë nuk është aktive. Kjo do të thotë që, edhe nëse dikush mund të menaxhojë këtë proçes, atij i duhet që administratori i shtresës së sigurisë, i cili ka të drejta juridike për menaxhimin e sigurisë, t’i japë të drejta për të ndërmarrë veprime.
Pjesa III – Vënia në jetë e shtresës, testimi dhe konkluzione
Kapitulli 7 - AMBIENTI I TESTIMIT, MËNYRA E IMPLEMENTIMIT DHE MBLEDHJA E TË DHËNAVE

Shtresa e sigurisë është e nevojshme që të vendoset në një ambient real testimi, me qëllim që të analizohen rezultatet dhe të shihet mundësia se si mund të përminësohet performeranca dhe të rritet sa më shumë nivel i sigurisë. Për të testuar funksionimin e kësaj shtrese sigurie është përdorur një bazë të dhënash reale e një kompanie, e cila ka shumë përdorues të shpërndarë në disa pika dhe shumëllojshmëri përdoruesish, që do të kërkonin konfigurime të ndryshme përza i përket roleve.

7.1 Karakteristikat e sistemit

Baza e të dhënave Oracle 10g R2 është instaluar në një makinë me këto karakteristika:

- **Prodhuesi**: IBM
- **Modeli**: System x3650 M2
- **Procesori**: Intel Xeon CPU – E5504
- **Memoria**: 8 GB
- **Lloji i sistemit**: 64 bit
- **Sistemi Operativ**: Windows 2008 R2 Enterprise, SP1

Baza e të dhënave : Oracle 10g R2 Enterprise

Kjo makinë mban bazën e të dhënave, në të cilën mbahen të dhënët, që do të përdoren nga sistemi me të cilin punon kompania. Për të minimizuar efektet e programeve, të cilat do të analizojnë sesionet për çdo instruksion, i cili vjen nga lista e përdoruesve do të përdoret një makinë e dytë, e cila fillimisht do të jetë identike përza i përkët skemës së të dhënave dhe më pas në këtë makinë do të ekzekutohen të gjitha instruksionet SQL, që janë ekzekutuar në makinën e parë nga përdoruesit. Por këtu do të përdoren disa forma të ndryshme të algoritimit të shtresës së sigurisë, siç është shpjeguar më parë dhe do të shihet ndikimi i saj në performancë.

Skema e bazë së të dhënave, e cila do t’i nënshtrohet testimeve, ka këto karakteristika:
Bazuar në këto informacione do të ndërtohen disa skripte, të cilat do të bëjnë të mundur që të mbledhin instruksionet DML dhe t’i vendosin në tabelë të një baze të dhënash të tretë, e cila do të aksesohet duke ndërtuar një DB_link ndërmjet bazës së të dhënave kryesore dhe asaj ku do të vendosen instruksionet. Kjo bëhet me qëllim që të mos kemi ndikim në performancën e bazës së të dhënave kryesore, e cila do të merret si referencë në krahazim me algoritmet që do të përdorim për shtresën e sigurisë.

Pra, skema e testimit do të jetë e përbërë nga tre makina në të cilat do të jetë e instaluar SMDB Oracle 10g, ku njëra është baza e të dhënave mbi të cilën punon kompania, makina identike e saj do të jetë makina ku do të provohen algoritmet dhe ndikimi i saj në performancë dhe makina e tretë shërben vetëm për të mbajtur instruksionet SQL që janë ekzektuar. Ndërmjet makinës së parë dhe të tretë do të kemi një DB_link, e cila do të realizojë lidhjen e këtyre dy bazave të të dhënave me qëllim minimizimin e efekteve të skripteve, të cilat do të grumbullojnë instruksionet SQL që vijnë nga përdoruesit.

### 7.2 Grumbullimi i të dhënave

Për të grumbulluar të dhënë, të cilat vijnë nga përdorues të ndryshëm, me qëllim matjen e efekteve që do të këtë shtresa në performancën e sistemit, janë ndërtuar skripte, të cilat realizojnë mbledhje e instruksioneve SQL, të orientuara në drejtim të skemës kryesore. Këto instruksione merren nga view V$SQLAREA dhe V$SESSION, siç shpjegohet tek menaxhimi i sesioneve. Jo të gjitha të dhënë, të cilat vijnë nga këto dy view janë të nevojshme që të mbahen, por vetëm ato që do të përdoren për të testuar të gjithë shtresën e sigurisë, që janë:
Përdoruesi

Instruksioni SQL

Lloji i instruksionit

Data_ora e ekzekutimit

Kohë vonesa (rastet e konkurrencës)

Koha e ekzekutimit

Problemi, i cili lind me grumbullimin e këtyre të dhënave, është në rastet kur kemi fusha të llojëve BLOB, të cilat mund të kenë madhësi të mëdha, si dhe nuk mbahen në instruksionin SQL, i cili po ekzekutohet, por ka një referencë në një pozicion të memories, ku ndodhet objekti blob.

Për të eliminuar këtë problem është nevojshme, që të përfishtohet nga testimi të gjithë tabelat, të cilat përmbyjnë fusha të tilla që në skemën, ku po bëhen testimet, janë vetëm dy tabela, të cilat nuk bëjnë pjesë tek tabelat me numër të madh transaksionesh.

Të dhënat që merren nga view-et e mësipërme do të dërgohen, duke përdorur DB_link, në një makinë të dytë dhe në këtë rast nuk kemi vonesa apo efekte në performancën e bazës së të dhënave kryesore.

Para se të fillojë ky proces për grumbullimin e të dhënave, është e nevojshme që të bëhet një kopje identike e makinës kryesore (klonim i bazës së të dhënave) dhe të vendoset në serverin ku do të bëhet matja në ndikimin e performancës të shtresës së sigurisë.

Instruksionet e grumbulluara do të sistemohen në bazën e të dhënave, e cila ndodhet në makinën e tretë në mënryrë të tillë që të mbahet një sekucë përmbyt nje ekzekutimit, me qëllim që kur të riezkukutohen, të kemi të njëjtën sekuencë ekzekutimi.

Të dhënët janë grumbulluar për një periudhë tremujore dhe më pas këto instruksione janë ekzekutuar në bazën e të dhënave të testimit, duke përdorur një program, i cili lexon instruksionet nga tabela dhe i dërgon për ekzekutim. Theksojmë se ekzekutimi ndërmjet instruksioneve SQL nuk mund të bëhet në mënryrë identike të shpërndarë në kohë, sikur kanë ndodhur në realitet me qëllim përhshpetimë të marrjes së rezultateve. Duke qenë se testimet bëhen për të kuptuar se cila nga mënryrat e menaxhimit të sigurisë ndikon më pak në performancë dhe janë bërë mbi të njëjtën makinë, nëse
përdoret e njëjta mënyrë ekzekutimi të instruksioneve, atëherë nuk kemi efekte në diferençën e performancës.

7.3 Riekzekutimi i instruksioneve SQL

Për të matur ndikimin e secilës nga mënyrat e shtresës së sigurisë (ajo e bazuar ne trigger-a, monitorimi i sesioneve dhe mënyra hibrë) duhet që të njëjtat instruksione SQL të grumbulluara të ekzekutohen sipas të tria mënyrave. Për të realizuar këtë proces ka dy mënyra, që janë si në vijim:

- **Mënyra identike**: Kjo mënyrë i ekzekuton gjithë instruksionet SQL sipas sekuencës, të cilat janë vendosur në tabelë dhe me kohën vonesë ndërmjet njëra-tjetrës sëçështë edhe në realitet, duke qenë se ne e njohim kohën e ekzekutimit të secilit instruksion.

- **Mënyra e përshtepjtuar**: Kjo mënyrë i ekzekuton instruksionet SQL sipas sekuencës, por duke mos pritur të njëjtën kohën ndërmjet instruksioneve por duke e minimizuar atë në një vonesë shumë të vogël në rastet kur kemi diferenca të konsideruese me ndërmjet kohëve të ekzekutimit të instruksioneve.

Mënyra e parë do të kërkojë një kohën identike me atë që janë mbledhur të dhënët, ndërsa mënyra e dytë është shumë herë më e shpejtë sepse mesatarisht jo më shumë se 60 minuta kohën harxhojnë instruksionet ditore. Kjo do të thotë që ne mund të ekzekutojmë të njëjtën grup instruksionesh brenda një kohë më të vogël se në realitet. Kjo do të na ofrojë mundësi që të bëjmë disa teste për të kontrolluar ndikimin në performancë.

Për ekzekutimin e instruksioneve është ndërtuar një program, i cili merr nga baza e të dhënave, në të cilën ndodhen instruksionet, dhe i ekzekuton ato në mënyrën sekuenciale në bazë të sekuencës që është vendosur në momentin e grumbullimit. Në rastet kur koha e ekzekutimit të instruksionit paraardhës me atë që është në radhë për ekzekutim, është më e vogël se 5 sekonda, atëherë ekzekutimi i do të vazhdojë në mënyrën të tillë që të kemi të njëjtën vonesë si në realitet. Ndërsa për instruksionet, të cilat janë me kohën vonesë më të madhe se 5 sekonda, atëherë këto instruksione do të ekzekutohen 5 sekonda më vone, pasi ka përforfunduar ekzekutimi i instruksionit të fundit. Kjo do të minimizojë jashtë zakonisht shumë kohën e testimeve. P.sh nga ekzekutimi i
instruksionit të fundit për ditën e parë deri në instruksionin e parë të ditës së dytë kemi një kohë vonesë rreth 16 orë.

Programi nuk duhet të sjellë kohë vonesa të tjera sepse instruksionet do të ekzekutohen në mënyrë jo të sinkronizuar, që do të thotë një instruksion nuk duhet të presë një instruksion tjetër për t’u ekzekutuar.

Testim ideal, për të bërë krahasimme me sistemin real, do të ishte që të monitorojmë edhe kohën kur kemi lidhje ndërmjet përdoruesve të ndryshëm, me qëllim që të ndërtojmë edhe në testim të njëjtën logjikë që të kemi të njëjtën menaxhim të SGA-së apo PGA-së.

7.4 Kohëvonesa që sjell shtresa e sigurisë

Duke qenë se shtresa e sigurisë për çdo instruksion SQL do të kontrollojë nëse përdoruesi ka të drejtë ekzekutimi të saj mbi objektet, ku është drejtuar ky instruksion dhe çdo kontroll, sjell një kohëvonesë në ekzekutim. Për këtë qëllim do të matet koha e ekzekutimit të trigger-it (duke e konvertuar në procedurë), si dhe kohë që i duhet funksionit, i cili kthen rezultatin nëse ky përdorues ka të drejta për aksesimin e të dhënave.

Matja e kësaj kohe mund të ndryshojë nga numri i instruksioneve, që janë në ekzekutim si dhe numri i përdoruesve të lidhur me bazën e të dhënave. Për të realizuar një matje sa më të saktë dhe për të ditur se sa është efekti i shtresës së sigurisë, është ndërtuar një procedurë, e cila e ekzekuton funksionin me parametra të rastësishëm, por të vlefshëm, dhe mat kohën se sa duhet për ekzekutimin e tij. Kjo procedurë e bën për një numër shumë të madh herësh (në testimet e bëra është marr numri real i instruksioneve të ekzekutuara gjatë periudhës së mbledhjes së të dhënave). Në shumë pak raste kemi një anormalitet të kohës, që ose është shumë e vogël ose jashtëzakonisht shumë e madhe. Duke përjashtuar këto raste, bazuar në matjet e kryera rezulton që kemi një kohëvonesë mesatare të barabartë me 16 milisekonda që i shtohet çdo transaksioni.

Bazuar në numrin mesatar të instruksioneve që kemi, shtresa jonë në total do të shtojë vetëm 240 sekonda vonesë gjatë të gjithë 15000 transaksioneve ditore, por e shpërndarë në për çdo ekzekutim që nuk do të këtë efekt të përdoruesi.
Kapitulli 8 - ANALIZA E TESTIMEVE, KONKLUZIONET E TESTIMIT DHE PËRMIRËSIMET E ALGORITMIT

8.1 Testimet e realizuara

Gjatë studimit kërkimor për ndërtimin e kësaj shtrese sigurie janë marrë në konsideratë tri mënyra se si mund të ndërtotoj kjo shtresë, të cilat janë trajtuar në kapitujt e mëparshëm. Për të parë efektin e secilës nga këto kërkesa në performancën e bazës së të dhënave, janë bërë këto teste:

- Shtresa e bazuar në trigger-a dhe me një shpërndarje të instruksioneve me mënyrën e përshtypuar të instruksioneve SQL (Testi 1).
- Shtresa e bazuar në menaxhimin e sesioneve dhe me një shpërndarje të instruksioneve me mënyrën e përshtypuar të instruksioneve SQL (Testi 2).
- Shtresa hibride dhe me një shpërndarje të instruksioneve me mënyrën e përshtypuar të instruksioneve SQL (Testi 3).

Gjatë procedurës së testimeve, për secilin nga tre testet e mësipërme janë ekzekutuar të njëjtat instruksione SQL të grumbulluara dhe me të njëjtën mënyrë për 10 herë. Në tabelën e mëposhtme kemi për secilin testim rezultatet në total të kohëvoneshës në sekonda, që ka shtuar shtresa e sigurisë, këto matje janë bërë duke modifikuar funksionin e kontrollit të të drejtave, i cili shton në një tabelë kohën e ekzekutimit të funksionit. Testimet janë bërë për 900,000 transaksione të grumbulluara.

Tabela 8. Tabela e kohëvoneshës të shtuar nga shtresa e sigurisë

<table>
<thead>
<tr>
<th>Nr. provës</th>
<th>Testi 1</th>
<th>Testi 2</th>
<th>Testi 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1430</td>
<td>1428</td>
<td>1420</td>
</tr>
<tr>
<td>2.</td>
<td>1445</td>
<td>1420</td>
<td>1443</td>
</tr>
<tr>
<td>3.</td>
<td>1480</td>
<td>1430</td>
<td>1448</td>
</tr>
<tr>
<td>4.</td>
<td>1401</td>
<td>1488</td>
<td>1497</td>
</tr>
<tr>
<td>5.</td>
<td>1454</td>
<td>1456</td>
<td>1449</td>
</tr>
<tr>
<td>6.</td>
<td>1433</td>
<td>1441</td>
<td>1441</td>
</tr>
<tr>
<td>7.</td>
<td>1442</td>
<td>1451</td>
<td>1422</td>
</tr>
</tbody>
</table>
Siç shihohet nga kjo tabelë, nuk kemi diferenca të mëdha në kohën që i duhet funksionit për të kontrolluar të drejtat për secilin rast.

Përveç testit, për të matur kohëvonesën që shtohet për shkak se duhet të kontrollohen të drejtat e përdoruesve, kemi edhe kohëvonesë, e cila vjen si shkak i monitorimit të sesioneve, që duhet të bëhet, e cila mund të matet në total të kohës së ekzekutimit për secilin test. Për të realizuar matjen e këtyre kohëve programi, i cili bën të mundur shpëndarjen e instruksioneve SQL, regjistron kohën që i duhet një instruksioni në një tabelë të caktuar. Duke qenë se në mënyrën e menaxhimit të sesioneve ekzekutimet i kemi në serial, atëherë vihet re që kemi një ndikimi shumë të madh në kohën në total. Në tabelën në vijim kemi kohën e harxhuar nga secili test në të 10 provat që janë bërë. Kohët e paraqitura në këtë tabelë janë në minuta.

Tabela 9. **Tabela e kohës së ekzekutimit të instruksioneve SQL në min**

<table>
<thead>
<tr>
<th>Nr. provës</th>
<th>Testi 1</th>
<th>Testi 2</th>
<th>Testi 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>2030</td>
<td>4389</td>
<td>2032</td>
</tr>
<tr>
<td>2.</td>
<td>2033</td>
<td>4435</td>
<td>2035</td>
</tr>
<tr>
<td>3.</td>
<td>2028</td>
<td>4398</td>
<td>2033</td>
</tr>
<tr>
<td>4.</td>
<td>2040</td>
<td>4407</td>
<td>2039</td>
</tr>
<tr>
<td>5.</td>
<td>2025</td>
<td>4409</td>
<td>2030</td>
</tr>
<tr>
<td>6.</td>
<td>2031</td>
<td>4418</td>
<td>2035</td>
</tr>
<tr>
<td>7.</td>
<td>2031</td>
<td>4415</td>
<td>2033</td>
</tr>
<tr>
<td>8.</td>
<td>2035</td>
<td>4408</td>
<td>2034</td>
</tr>
<tr>
<td>9.</td>
<td>2037</td>
<td>4421</td>
<td>2040</td>
</tr>
<tr>
<td>10.</td>
<td>2032</td>
<td>4404</td>
<td>2033</td>
</tr>
</tbody>
</table>

Një e dhënë e rëndësishtë është edhe raporti ndërmjet llojeve të instruksioneve, që janë rreth 99.91% e tyre janë instruksione DML dhe vetëm 0.09% e tyre janë instruksione DDL.
8.2 Analiza e testimeve

Siç shikohet në tabelën 8, vonesa që shtohen nga koha për të kontrolluar të drejtat e çdo përdoruesi në bazë të skemës të konfiguruar janë aq të vogla (mesatarisht 0.016 sekonda për instruksion), sa që nuk ndikojnë në proçesin e punës nga përdoruesit të cilët kanë akses në të dhëna.

Ajo, që bie ne sy, është koha që duhet për t’u ekzekutuar e gjithë bashkësia e transaksioneve dhe vihet re që ndërmjet testit të parë dhe atij të dytë kemi shumë pak ndryshime dhe kjo vjen për faktin që gjatë testit të tretë kemi instruksione DDL, të cilat monitorohen dhe kemi një ekzekutim serial të tyre. Në shumicën e rasteve këto janë instruksione janë për rindërtimin e indekseve, që janë përdorur për të rritur performancën gjatë proçesit të menaxhimit të bazës së të dhënave nga administratori.

Një vonesë tjetër, e cila nuk është e matshme, është edhe ajo që sjell sistemi operativ në menaxhimin e proçeseve që janë në ekzekutim. Kjo pikë shpjegon edhe rastet kur testi i tretë është pak më i shpejtë se ai i pari. Kjo nuk duhet të ndodh në kushte të njëta, duke qenë se në shtresën hibride kemi analizë të instruksioneve DDL, duke menaxhuar sesionet.

Siç shikohet edhe nga tabela 9, kemi një kohë shumë të madhe në ekzekutimin e instruksioneve SQL në testin e dytë, i cili bazohet mbi menaxhimin e sesioneve dhe analizën e çdo instruksioni SQL. Kjo ndikon në mënyrën e ekzekutimit të proçeseve sepse nuk i ekzekuton ato në paralel, por në serial, dhe ky është edhe faktori kryesor i ndikimit në performancës.

Gjatë një analize më të detajuar të testimeve të testit të dytë vihet re që kemi shpesh edhe “time out” për shkak se baza e të dhënave vonon për ekzekutimin e një instruksioni të caktuara, ndërkohe që vetë programi, i cili ekzekuton instruksionet e grumbulluara SQL i dërgon ato në mënyrë të pasinkronizuar në varësi të kohës që ato janë ekzekutuar, duke përdorur mënyrën e përshpejtimit që është shpjeguar më parë.

Bazuar në testimet e mësipërme, mënyra e ndërtimit të shtresës së sigurisë e bazuar mbi analizën e çdo instruksioni SQL, pavarësisht nëse ai është DDL apo DML, nuk është efikase sepse ndikon jashtëzakonisht shumë në performancë dhe sjell një keqfunkcionim SMBD-së në rastet kur kemi instruksione SQL komplekse, të cilat mund të kërkojnë kohë relativisht shumë të madhe në ekzekutim, siç ndodh p.sh në rastet e ndërtimit të raporteve.
8.3 Përmirësimet e algoritmit

Duke parë testimet e realizuara vëlë re se mënyra e dytë nuk mund që të përdoret për ndërtimin e shtresës së sigurisë dhe ajo, që ka një ndikim sa më të vogël në performancë, është mënyra e parë, por ajo nuk na ofron një vetëmbrojtje. Duke qenë se edhe kjo mënyrë nuk është e plotë, atëherë ajo që duhet përmirësohet është mënyra hibrise.

Vëlë re që mënyra hibrise analizon nëpërmjet sesioneve vetëm instruksionet SQL, të cilat janë të llojit DDL, që do të thotë që numri i tyre është shumë i vogël. Në disa raste këto instruksione nuk përbëjnë rrezik sulmi ndaj të dhënave. Kjo do të thotë këto lloj instruksionesh mund të lejohen dhe të mos jenë pjesë e analizës, sic janë përmandur edhe tek konceptimi i shtresës hibrise. Kjo mënyrë do të sjellë shumë përmirësime në performancë, duke qenë se shumica e instruksioneve DDL, të cilat konsumojnë kohë nuk ndikojnë në sigurinë e të dhënave, sic janë ndërtimi i indekseve, menaxhimi i hapësirave të lira (menaxhimi i tablespace-ve), etj.
KONKLUZIONE

Gjatë studimit të gjithë procesit të ndërtimit të shtresës së sigurisë është arritur në disa lloj konkluzionesh, të cilat kanë të bëjnë me performancën, rritjen e sigurisë apo edhe menaxhimin e sesioneve.

Konkluzioni kryesor, që del nga ky studim, është që në SMBD Oracle, i cili edhe pse nuk është Open Source, na ofron mundësinë e menaxhimit të kërkesave të ndryshme për të arritur rezultate që kërkojen duke përdorur paketat, që ofron Oracle, si p.sh në rastin tonë VPD apo enkriptimi i të dhënave.

Oracle ka ofruar mundësinë që nëpërmjet VPD-së, ne të arrimë që të bllokojmë një përdorues të caktuar në akcesimin e të dhënave, si për lexim ashtu edhe për shkrim, por problemi i cili është zgjidhur si rezultat i këtij kërkimi, është që nuk i lejohet një përdoruesi me të drejtë si DBA të këtë mundësinë që të lejojnë të dhëna, duke modifikuar VPD apo duke bërë statusin joaktiv të objekteve të ndryshëm.

Ndikimi i shtresës së sigurisë në performancë është minimal, për arsye se kohë për kontrollin e të drejtave është shumë e vogël krahasuar me kohën që konsumojnë shumica e instruksioneve SQL dhe të vetëm instruksione, të cilat do të ndikojnë në performancë, janë ato DDL që prekin objektet, të cilat janë nënbrojtje. Kjo do të thotë që shumë më pak se 0.09% e instruksioneve DDL do t’i nënshtrohen procesit të analizës së sigurisë.

Me qëllim përmirësimin dhe rritjen e performancës (minimizimit të efekteve që ka në performancë kjo shtresë sigurie), përdorimi i log-eve dhe rastet kur kemi ndërprerje të sesioneve do të na gjeneronin shumë informacion edhe mbi përdoruesit apo burimet e sulmeve.

Logjika e kësaj shtrese sigurie mund të implementohet edhe në SMBD të tjera, të cilat ofrojnë koncepte ose lejojnë ndërtimin e tyre, si ato që ofron Oracle që janë menaxhimi dhe analiza e sesioneve, përdorimi i VPD, fshihja e kodit etj.
REFERENCA


[14] Oracle® Database Reference 10g Release 2 (10.2), Part Number B14237-04


[17] Bolfing F.: Database Replication with Oracle 11g and MS SQL Server 2008


[25] “Differences between Read-Only, Updatable and Writeable Materialized Views”, Metalink article, 162711.1

[26] Bolfing F. “Database Replication with Oracle 11g and MS SQL Server 2008”

[27] “Oracle Master Replication, Performance and Scalability”, Metalink article 76448.1

