

# **Sigurnost kod Oracle baze podataka**

*Seminarski rad*

Sistemi za upravljanje bazama podataka

Student: Uroš Pešić 1465

Profesor: Dr Aleksandar Stanimirović

Elektronski fakultet, Niš  
Maj 2024.

# Sadržaj

UVOD.....	3
1 – Sigurnost baza podataka.....	4
1.1 - Mere kontrole.....	5
1.2 – Kontrola pristupa.....	5
1.2.1 – Korisnički nalozi i revizija baze podataka.....	6
1.2.2 – Diskreciona kontrola pristupa.....	7
1.2.3 – Primeri upravljanja privilegijama.....	9
1.2.4 – Obavezna kontrola pristupa.....	12
1.2.5 – Poređenje diskrecione i obavezne kontrole pristupa.....	14
1.3 – Kontrola toka.....	14
1.5 – Enkripcija podataka.....	15
2 – Sigurnost kod Oracle baze podataka.....	16
2.1 – Kontrola pristupa zasnovana na ulogama.....	16
2.2 – Analiza privilegija u Oracle-u.....	18
2.3 – Objedinjena revizija (Unified Auditing).....	20
2.3 – Database Vault.....	23
2.4 – Oracle Label Security.....	29
2.5 – Sakrivanje poverljivih podataka.....	32
2.6 – DBSAT – Security Assessment Tool.....	34
ZAKLJUČAK.....	38
LITERATURA.....	39

# UVOD

Sigurnost predstavlja jedan od najbitnijih aspekata današnjih softverskih sistema. Do pre par decenija, pitanje sigurnosti je bilo ključno samo za neke specijalizovane sisteme, kao što su vojni sistemi, sistemi namenjeni vladama, finansijski sistemi, itd. Razvojem Interneta, postepeno su se menjali i načini poslovanja kompanija i softverski sistemi su postajali sve složeniji. Danas se svakodnevno generiše ogromna količina podataka. Veliki njihov deo čine osetljivi podaci, kao što su brojevi kreditnih kartica korisnika E-Commerce platformi, lozinke korisnika za pristup različitim servisima, adrese i drugi podaci lične prirode. Samim tim su i skladišta podataka postala česta meta hakerskih napada, kojima je potrebno doći. Sa druge strane, osetljive informacije korisnika neke platforme moraju biti zaštićene, kako od neautorizovanog pristupa, tako i od eventualne zloupotrebe od strane administratora ili drugih autorizovanih korisnika sistema. Slučajno curenje ili zloupotreba ovakvih podataka, može da donese brojne pravne probleme kompaniji koja stoji iza sistema.

Iz navedenih razloga, dobra sigurnost je neizostavni kriterijum koji moraju da ispunjavaju današnji sistemi baza podataka. U ovom seminarskom radu su u prvom poglavlju navedeni opšti pojmovi i problemi vezani za ovu temu. Takođe, definisane su generalne tehnike za rešavanje problema sigurnosti koje implementiraju današnji DBMS-ovi. U drugom poglavlju, dat je pregled tehnika koje Oracle DBMS koristi u cilju obezbeđivanja sistema i zaštite podataka.

# 1 – Sigurnost baza podataka

Sigurnost baza podataka predstavlja široku oblast koja se bavi problemima zaštite podataka od neautorizovanog pristupa, sprovođenja specifičnih sigurnosnih ograničenja, organizacije podataka i korisnika u više sigurnosnih klasa, itd. U nastavku su navedeni neki od primera sigurnosnih zahteva koji se javljaju u sistemima baza podataka:

- Prilikom registracije na neku softversku platformu, novi korisnici moraju biti jasno obavešteni o načinu na koji će se njihovi podaci čuvati i obrađivati, shodno zakonu o zaštiti podataka o ličnosti. Ukoliko dođe do kršenja ovih uslova, ili u slučaju da podaci postanu dostupni neovlašćenim organizacijama ili korisnicima, propisane su stroge zakonske kazne.
- Podaci koji su na osnovu politike neke institucije ili korporacije označeni kao poverljivi moraju biti zaštićeni. Na primer, medicinski izveštaji o pacijentima, plate zaposlenih, su neki od primera podataka koji ne smeju biti javno dostupni.
- U pojedinim organizacijama postoji potreba za postojanjem više sigurnosnih nivoa – na primer: strogo poverljivi, poverljivi i javni. U zavisnosti od položaja u hijerarhiji organizacije, zaposleni su takođe klasifikovani na isti način. Kod ovakvih sistema je potrebno obezbediti adekvatna sigurnosna rešenja, koja će da zaštite podatke svakog nivoa od nedozvoljenog pristupa sa nižeg sigurnosnog nivoa.

Glavni ciljevi sigurnosti baze podataka su obezbeđivanje integriteta (*eng. **Integrity***), dostupnosti (*eng. **Availability***) i poverljivosti (*eng. **Confidentiality***) podataka. Ugrožavanje sigurnosti, može dovesti do degradacije ili potpunog gubitka ovih osobina. One su definisane na sledeći način:

- **Integritet podataka** – Predstavlja zahtev da podaci budu zaštićeni od neovlašćene modifikacije. Modifikacija podataka podrazumeva dodavanje, ažuriranje i brisanje podataka. Narušavanje integriteta dovodi do kontaminacije podataka u bazi, koja dalje može dovesti do prevara ili donošenja pogrešnih poslovnih odluka.
- **Dostupnost podataka** – Predstavlja zahtev da podaci budu dostupni korisniku (ili programu) koji imaju pravo da njima pristupe. Dostupnost može biti narušena iz više razloga, jedan od primera su *DoS (Denial of Service)* napadi.
- **Poverljivost podataka** – Odnosi se na zaštitu podataka od neovlašćenog čitanja i njihovog javnog iznošenja. Posledice narušavanja poverljivosti variraju u zavisnosti od stepena tajnosti otkrivenih podataka, važećih državnih zakona, i sličnih faktora.

## 1.1 - Mere kontrole

Da bi se izbegli potencijalni problemi i obezbedili gore pomenuti ciljevi, u današnjim DBMS-ovima se najčešće implementiraju četiri mere kontrole:

- Kontrola pristupa (*eng. Access Control*)
- Kontrola toka podataka (*eng. Flow Control*)
- Kontrola zaključivanja (*eng. Inference Control*)
- Enkripcija podataka (*eng. Data encryption*)

U narednih par poglavlja dat je pregled nekih navedenih mera. Važno je napomenuti da su kontrola zaključivanja i kontrola toka usko povezane za neke specifične sisteme – kontrola zaključivanja je bitna mera u statističkim bazama podataka, pa će samim tim biti izostavljena iz ovog rada koji se fokusira na relacione baze podataka, dok je kontrola toka bitna u sistemima sa većim brojem sigurnosnih nivoa i biće kratko pomenuta u poglavlju 1.3.

## 1.2 – Kontrola pristupa

Kontrola pristupa se odnosi na mehanizme zaštite od neautorizovanog pristupa, kako sistemu u celini, tako i njegovim objektima (tabelama, atributima, itd). Kontrola pristupa celokupnom sistemu se vrši kreiranjem korisničkih naloga (korisnik u ovom kontekstu može biti osoba ili aplikacija koja koristi bazu podataka), dok su pojedini delovi sistema zaštićeni definisanjem privilegija – samo korisnici sa odgovarajućim privilegijama mogu da čitaju ili modifikuju pojedine podatke i tabele. Za ovaj aspekt sigurnosti sistema, prvenstveno je zadužen **administrator baze podataka (DBA – Database Administrator)**. Administrator baze podataka ima specijalni nalog za pristup DBMS-u, koji se često naziva **system**, ili **superuser** nalog, koji mu pruža funkcionalnosti koje običnim korisnicima nisu dozvoljene. Zaduženja administratora su:

1. **Kreiranje korisničkih naloga** – Administrator je zadužen da kreira naloge sa lozinkama za korisnike ili grupe korisnika koje pristupaju bazi.
2. **Dodeljivanje privilegija** – Administrator je zadužen da korisnicima dodeljuje privilegije pristupa objektima u sistemu.
3. **Oduzimanje privilegija** – Ukoliko je potrebno, administrator može da ukloni dodeljene privilegije.
4. **Dodela sigurnosnih nivoa** – U slučaju postojanja više sigurnosnih klasa, administrator je taj koji svakom nalogu pridružuje odgovarajući sigurnosni nivo.

Prvom navedenom stavkom kontroliše se pristup celokupnom sistemu. Stavke 2 i 3 spadaju u mehanizme **diskrecione kontrole pristupa**, dok se 4. stavka spada u tehnike **obavezne<sup>1</sup> kontrole pristupa**.

---

<sup>1</sup> *Eng. Mandatory*, ovaj termin u bukvalnom prevodu može biti zbunjujući. „Obavezne“ kontrole pristupa ne znači da moraju biti implementirane, naprotiv, postoje relacioni DBMS-ovi koji nemaju ovakve sigurnosne mehanizme.

**Diskreciona kontrola pristupa** predstavlja mehanizam kontrole pristupa korišćenjem **privilegija**, kojima se pojedinim korisnicima dozvoljavaju, odnosno zabranjuju operacije čitanja ili modifikacije objekata u sistemu, kao što su tabele, atributi, slogovi, i slično. Takođe, moguće je selektivno dozvoliti samo neke operacije (ne funkcionišu po principu sve ili ništa).

**Obavezna kontrola pristupa** se koristi za implementaciju više sigurnosnih nivoa. Ovim mehanizmom se sprečava da korisnici pristupe objektima koji su na višim nivoima sigurnosti od njihovog. Proširenje ovog mehanizma jeste i kontrola pristupa zasnovana na ulogama (*eng. Role-based*), kojom se definišu pravila i privilegije na osnovu uloga i ograničenja koje korisnici imaju u organizaciji koja koristi sistem.

Detaljni pregled diskrecione i obavezne kontrole pristupa dat je u poglavljima 1.2.2 i 1.2.3 redom.

### 1.2.1 – Korisnički nalozi i revizija baze podataka

Kao što je već pomenuto u prethodnom poglavlju, da bi pristupili bazi podataka korisnici moraju da zatraže korisnički nalog od administratora. Kada kreira nalog, administrator korisniku dostavlja broj naloga (username) i lozinku koja se koriste za prijavljivanje na sistem. DBMS vodi evidenciju o korisničkim nalogima u vidu specijalne enkriptovane tabele, koja ima kolonu za broj naloga i kolonu za lozinku. Prilikom prijavljivanja, DBMS proverava da li unete informacije odgovaraju nekoj vrsti tabele – u slučaju da ne postoji takva vrsta, korisniku je onemogućen pristup. Kreiranjem, tj. brisanjem korisničkog naloga, dodaje se, tj. briše se odgovarajuća stavka iz pomenute tabele. Na slici 1 je prikazan primer prijavljivanja administratora na Oracle bazu.

Connection Name	Connection Details
sysdba	sys@//10.0.2.15...

Name: sysdba

Database Type: Oracle

User Info: Proxy User

Authentication Type: Default

Username: sys

Password: .....

Role: SYSDBA

Save Password: ☐

Connection Type: Basic

Details: Advanced

Hostname: 10.0.2.15

Port: 1521

☐ SID

☒ Service name: xepdb1

Status : Success

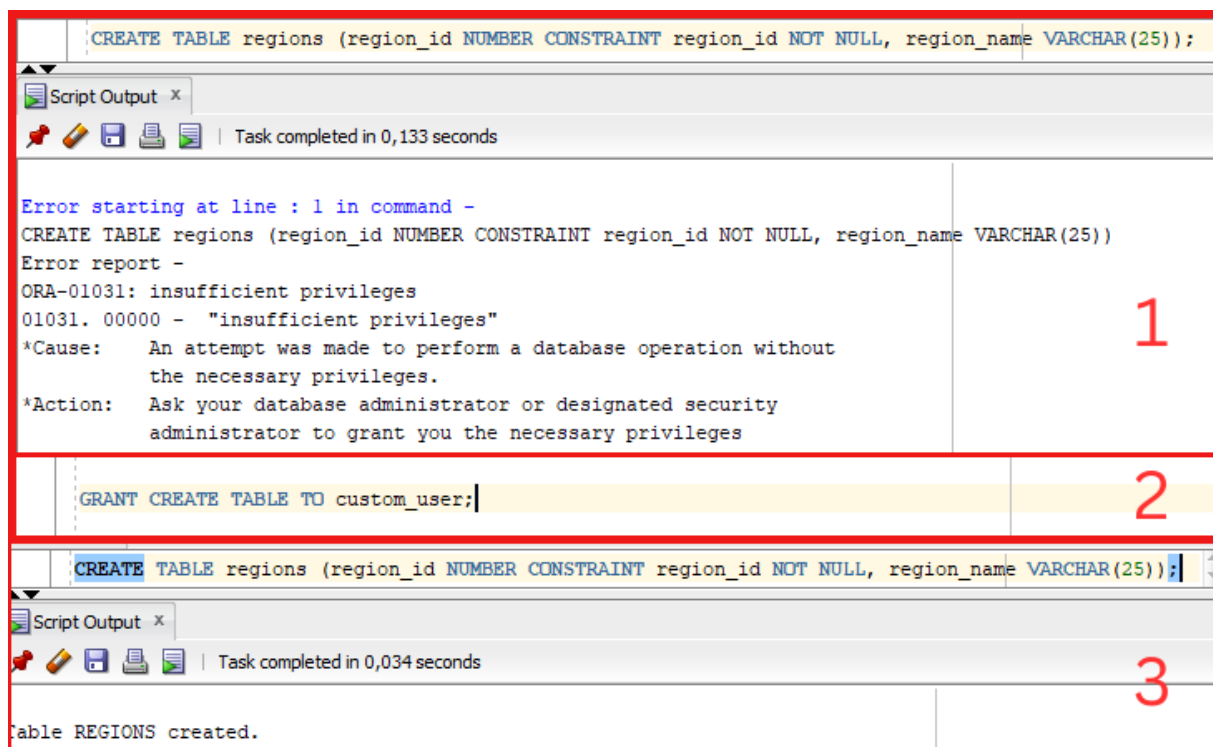
Slika 1: Prijavljivanje na **system** nalog Oracle baze korišćenjem Oracle SQL Developer-a

Takođe, DBMS vodi evidenciju o svim operacijama nad bazom koje korisnik izvršava u toku jedne **sesije**. Posebno je bitno pratiti operacije kojima se modifikuju podaci u sistemu. Svaka operacija koja se izvrši se upisuje u sistemski **log** fajl, zajedno sa brojem naloga i ID-em uređaja sa kog je izvršena. Ukoliko se posumnja da je došlo do nedozvoljenog manipulisanja podacima, pregledavanjem log fajla, tj. revizijom baze podataka (*eng. Database audit*) je moguće utvrditi koji korisnik je za to odgovoran i sa kog uređaja je to učinjeno. Mehanizam revizije baze je naročito bitan za sisteme u kojima se podaci često ažuriraju od strane velikog broja korisnika, kakvi su na primer bankarski sistemi.

## 1.2.2 – Diskreciona kontrola pristupa

Sigurni relacioni DBMS mora da pruža mogućnost selektivne dozvole pristupa tabelama i atributima u zavisnosti od tipa ulogovanog korisnika ili grupe kojoj korisnik pripada. Ovakva metoda kontrole pristupa se zove diskreciona kontrola pristupa i zasniva se na korišćenju privilegija, kao što je definisano u prethodnom poglavlju. Generalno, postoje dva vrste privilegija:

- **Definisane na nivou korisničkog naloga** – Ovako definisane privilegije su vezane za sam korisnički nalog i ne zavise od objekata u sistemu. Neki primeri ovakvih privilegija su: dozvola za kreiranje šema, tabela, ili pogleda, dodavanje ili brisanje atributa relacija, brisanje tabela i podataka, čitanje podataka korišćenjem SELECT naredbe, itd. Privilegije na nivou naloga nisu standardni deo SQL-a, već su različito implementirane u različitim DBMS-ovima. Na slici 2 je prikazan primer dodeljivanja privilegija za kreiranje tabela korisniku „*custom\_user*“ u Oracle DBMS-u.



Slika 2: Dodela privilegije za kreiranje tabela, korisniku "custom\_user"

- **Definisane na nivou relacije** – Ovako definisane privilegije se odnose na konkretne relacije (tabele) ili virtuelne relacije (pogleda). Njima se definiše koje operacije svaki postojeći korisnik može da izvrši nad relacijama u sistemu. Takođe je moguće definisati nad kojim podskupom atributa relacija je moguće primeniti dozvoljene operacije. Privilegije koje se definišu nad relacijama u sistemu mogu da se predstavje **matricom pristupa M** (eng. **Access matrix**), u kojoj vrste predstavljaju korisnike sistema (individualne korisnike ili programe), a kolone predstavljaju objekte sistema (tabele, pogleda, slogove). Element na poziciji  $M(i, j)$  označava koje operacije su dozvoljene korisniku  $i$ , nad objektom  $j$ . Primer matrice pristupa je dat na sledećoj slici.

Access Matrix	Table1	View1	Table2	Table3	View2
User_ID1	read/ write		write		
User_ID2		read		write	
User_ID3	read/ write	read		read	read

Slika 3: Primer matrice pristupa

Da bi se olakšalo upravljanje ovim tipom privilegija, svakoj relaciji je pridružen **vlasnički nalog** - najčešće je to nalog koji je korišćen za kreiranje relacije. Vlasnik ima sve privilegije nad svojim relacijama i može da dodeljuje i otklanja privilegije ostalim korisnicima. Naredbe za upravljanje ovim tipom privilegija su sastavni deo SQL-a i mogu biti:

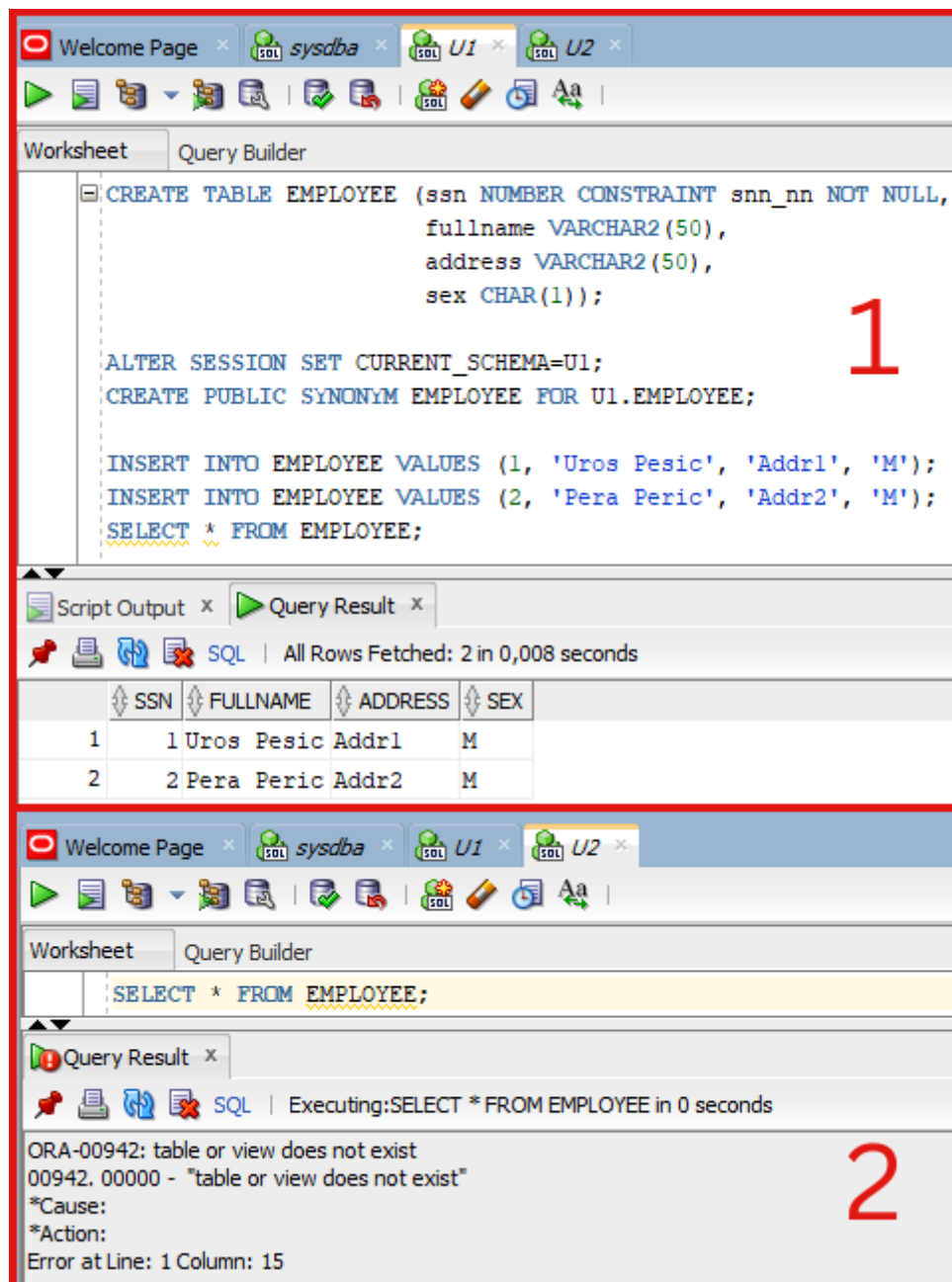
1. **SELECT (read) privilegije nad relacijom**
2. **Privilegije za modifikaciju relacije** – Dozvoljava upotrebu UPDATE, DELETE, INSERT operacija nad relacijom. Dodatno je moguće definisati skup atributa nad kojima je moguće izvršiti modifikaciju operacijama UPDATE i INSERT.
3. **Privilegije za referenciranje relacije** – Dozvoljava referenciranje relacije prilikom definisanja stranog ključa ili drugih ograničenja.

Često se u praksi ograničavanje korisnika na čitanje samo određenih atributa (ili određenih vrsta) relacije postiže korišćenjem pogleda. Prilikom kreiranja pogleda potrebno je da korisnik poseduje privilegije čitanja za sve relacije koje se javljaju u definiciji pogleda.



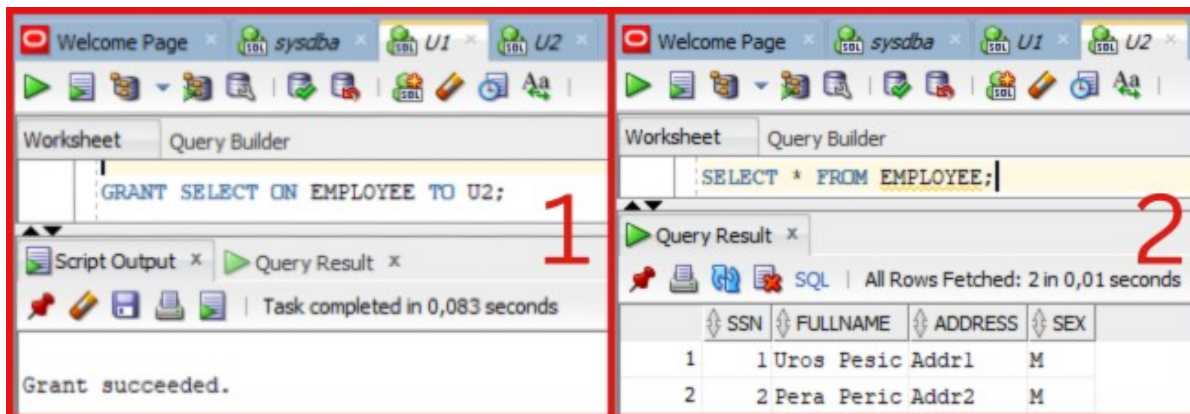
### 1.2.3 – Primeri upravljanja privilegijama

U ovom poglavlju će kroz par primera biti demonstrirano upravljanje privilegijama kod Oracle baze podataka. Za potrebe demonstracije kreirana su četiri korisnička naloga U1, U2, U3 i U4. Inicijalno je samo korisniku U1 naredbom **GRANT CREATE TABLE TO U1** dozvoljeno da kreira tabele. Zahvaljujući ovoj privilegiji, kreiramo tabelu *EMPLOYEE* sa naloga korisnika U1 (slika 4, sekcija 1).



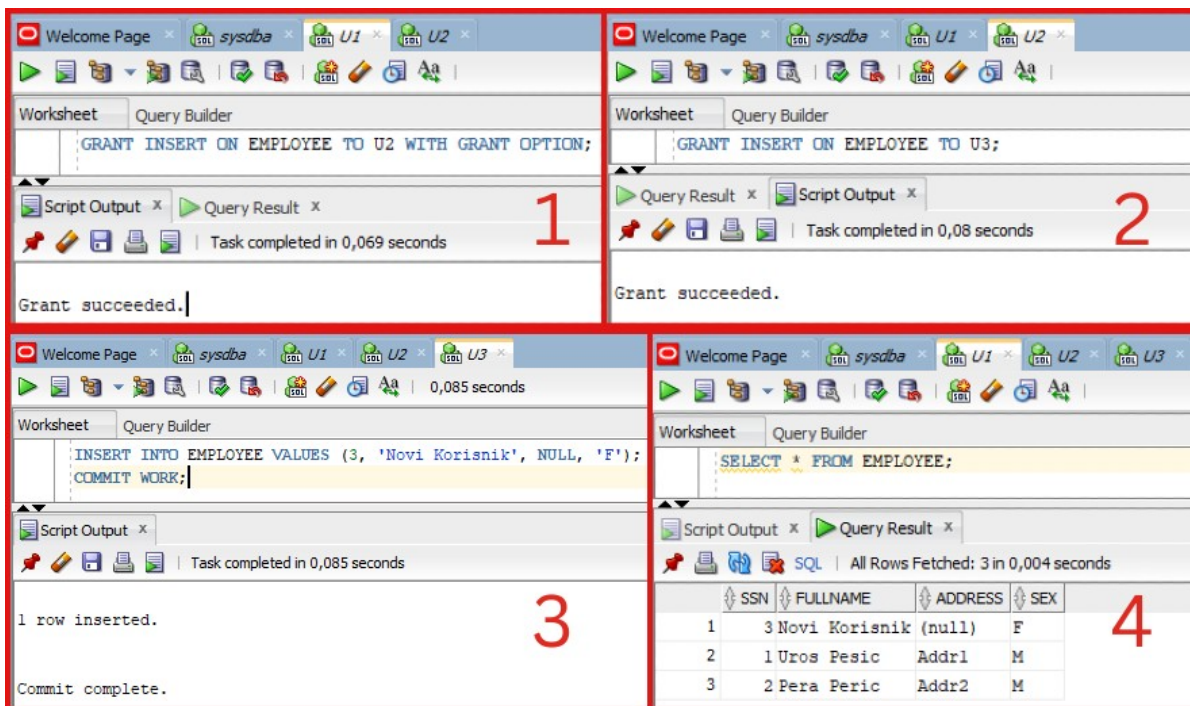
Slika 4: Kreiranje tabele i pokušaj čitanja

Možemo da primetimo da u drugom delu slike da iz ugla korisnika U2 tabela *EMPLOYEE* ne postoji, jer on nema nikakve privilegije za izvršavanje operacija nad njom<sup>2</sup>. Na sledećoj slici prikazan je rezultat iste operacije izvršene od strane korisnika U2, nakon što korisnik U1, koji je vlasnik tabele *EMPLOYEE* pa samim tim ima i sve privilegije nad njom, obezbedi mogućnost čitanja dodelom odgovarajuće privilegije.



Slika 5: Čitanje nakon dodele privilegije

Korisnik relacije takođe ima opciju da dodeli privilegiju korisniku, sa opcijom da je taj korisnik dalje propagira novim nalozima. To se postiže navođenjem **...WITH GRANT OPTION** opcije na kraju naredbe dodeljivanja privilegije. Ovakav primer je ilustrovan na slici 6.

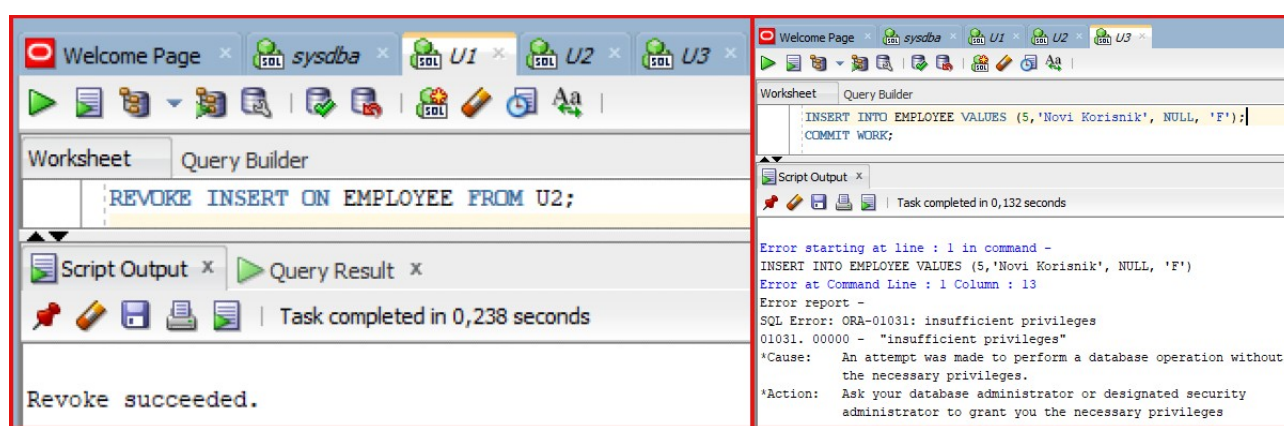


Slika 6: Primer propagacije privilegija

- 2 Ovakvom implementacijom obezbeđena je ispravna kontrola toka – ukoliko korisnik ne može da izvršava nijednu operaciju nad tabelom, on ne treba da bude svestan njenog postojanja. Sa druge strane, ukoliko korisnik ima definisane privilegije npr. za dodavanje podataka, ali ne i za čitanje, korisnik je svestan postojanja tabele i prilikom pokušaja čitanja će se ispisati adekvatna poruka (za razliku od poruke sa slike, na osnovu koje deluje da tabela ni ne postoji).

U primeru sa slike 6, korisnik U1 dodeljuje privilegiju za dodavanje podataka korisniku U2, uz mogućnost da U2 dalje prosleđuje ovu mogućnost. Zatim korisnik U2 dodeljuje ovu privilegiju korisniku U3 koji dodaje novu vrstu u tabelu *EMPLOYEE*. Kako korisnik U3 nema mogućnost čitanja iz ove tabele, na delu slike označenom brojem 4, koristimo nalog korisnika U1 radi provere operacije dodavanja.

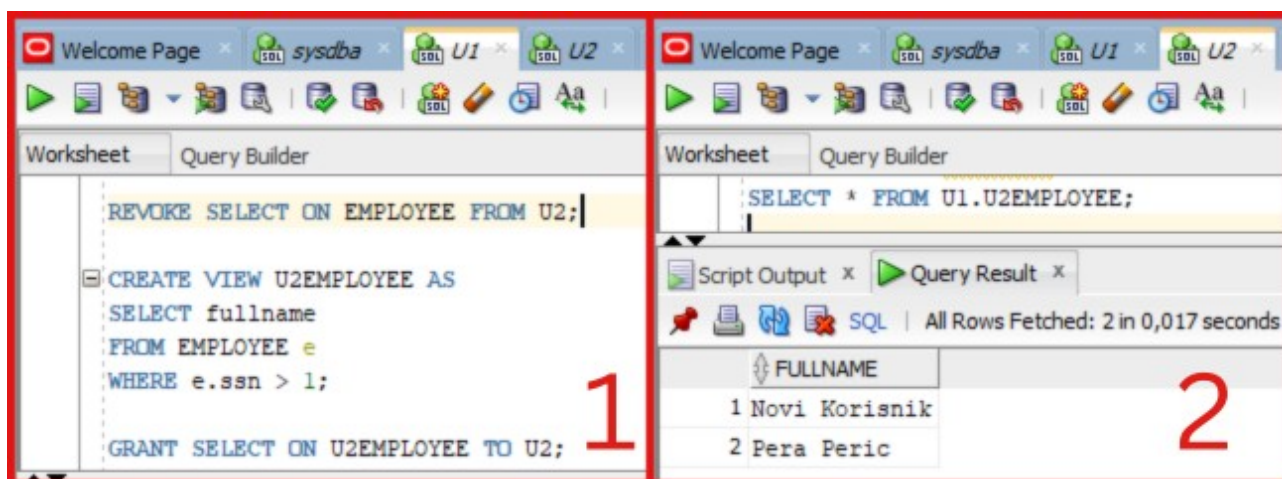
U slučajevima propagacije privilegija, potrebno je voditi evidenciju o „lancu“ dodeljivanja. Kada neki korisnik oduzme neku privilegiju koju je dodelio drugom korisniku sa GRANT opcijom, potrebno je sukcesivno oduzeti tu privilegiju svim ostalim korisnicima koji su tu istu privilegiju dobili neposredno preko korisnika od kojeg se privilegija oduzima. Jasnije objašnjenje možemo dobiti na osnovu primera sa slike 7. Nadovezujući se na prethodni primer, ukoliko korisnik U1 oduzme INSERT privilegiju korisnika U2, automatski će i korisnik U3 izgubiti mogućnost izvršenja INSERT operacije, jer je nju dobio od korisnika U2, koji gubi ovo pravo.



Slika 7: Lančano oduzimanje privilegija

U međuvremenu su razvijene različite tehnike za kontrolu propagacije privilegija. Generalno razlikujemo dve tehnike – ograničavanje **horizontalne propagacije** i ograničavanje **vertikalne propagacije**. Horizontalnu propagaciju je moguće ograničiti definisanjem celog broja  $k > 0$  tako da nalog kojem je dodeljena privilegija sa GRANT opcijom, može da je prosledi dalje na najviše  $k$  novih naloga. Kod vertikalnog ograničavanja se kontroliše dubina propagacije privilegija – definišemo faktor  $k > 0$  tako da nalog kojem je dodeljena privilegija sa GRANT opcijom može da prosledi ovu privilegiju novom nalogu čija je faktor najviše  $k-1$ .

Kao poslednji primer, navešćemo korišćenje pogleda kao mehanizam zaštite pojedinih atributa i vrsta od čitanja od strane neovlašćenog korisnika. Pretpostavimo da želimo da uklonimo privilegiju čitanja korisniku U2 nad tabelom *EMPLOYEE* i želimo da mu dozvolimo da ima uvid samo u imena zaposlenih čiji je *ssn* veći od 1. To možemo postići definisanjem odgovarajućeg pogleda, kao što je ilustrovano na slici 8.



Slika 8: Korišćenje pogleda za definisanje privilegija čitanja

### 1.2.4 – Obavezna kontrola pristupa

Diskreciona kontrola pristupa je glavni mehanizam koji se koristi za obezbeđivanje sigurnosti baze podataka. Međutim, ponekad ovakva zaštita po principu „sve ili ništa“ nije dovoljna, naročito u organizacijama sa složenijom sigurnosnom politikom. Obavezna kontrola pristupa obezbeđuje implementaciju više sigurnosnih nivoa za potrebe zaštite sistema u ovakvim organizacijama. Najčešće se mehanizmi obavezne kontrole koriste u kombinaciji sa diskrecionim mehanizmima. Kako su potrebe za ovakvim vidom višenivovske zaštite obično vezane za specifične sisteme, veliki broj relacionih DBMS-ova ne podržava ovakav vid kontrole, ili ga nudi u svojim specijalizovanim rešenjima, kao npr. Oracle.

Obično se definišu 4 sigurnosne klase: stroga tajna (*Top secret – TS*), tajna (*Secret – S*), poverljivo (*confidential – C*) i javno (*unclassified – U*), pri čemu je poredak klasa po nivou sigurnosti  $TS \geq S \geq C \geq U$ . Model za implementaciju više sigurnosnih nivoa koji se koristi u sistemima za upravljanje bazama podataka se zove *Bell-LaPadula* model, koji svakom subjektu (korisniku, nalogu, programu) i svakom objektu (relaciji, slogu, koloni, operaciji) dodeljuje odgovarajuću sigurnosnu klasu (jednu od 4 navedene). Ako sa  $class(S)$  označimo klasu subjekta  $S$ , a sa  $class(O)$  klasu objekta  $O$ , dva glavna sigurnosna principa ovog modela možemo definisati na sledeći način:

1. **Osnovno svojstvo sigurnosti** – Subjekt  $S$  ne može da ima dozvolu pristupa objektu  $O$  osim u slučaju kada je  $class(S) \geq class(O)$ .
2. **Zvezda svojstvo (star property)** – Subjekt  $S$  ne može da vrši upis u objekat  $O$ , osim u slučaju kada je  $class(S) \leq class(O)$ .

Dok je prvo svojstvo očigledno, drugo svojstvo je na prvi pogled zbunjujuće. Ovo svojstvo spada u domen kontrole toka informacija, o čemu će kratko biti reči u poglavlju 1.3, pa će i ovo svojstvo postati jasnije.



U relacionim bazama podataka, za potrebe implementacije obavezne kontrole pristupa, i pojedinačni atributi i celi slogovi se posmatraju kao objekti, tj. i čitava vrsta i svaki zasebni atribut vrste su klasifikovani u neku od pomenutih sigurnosnih klasa. Ako sa  $C_i$  označimo atribut klasifikacije za vrednost atributa  $i$ , a sa  $TC$  označimo atribut klasifikacije sloga, u ovako proširenom relacionom modelu, relaciju  $R$  koja ima  $n$  atributa možemo predstaviti kao  $R(A_1, C_1, A_2, C_2, \dots, A_n, C_n, TC)$ . Vrednost  $TC$  predstavlja predstavlja klasu za čitav slog i jednaka je  $\max(C_1, \dots, C_n)$ . Postojanje klasa za svaku pojedinačnu vrednost atributa u relaciji omogućava bolju kontrolu sigurnosti, ali i otežava implementaciju, o čemu će biti reči u nastavku.

U višestepenim relacijama se definiše pojam **prividnog ključa** (eng. *Apparent key*), koji predstavlja skup atributa koji bi u klasičnoj relaciji predstavljali primarni ključ. Glavni izazov u ovakvim sistemima predstavlja činjenica da različiti korisnici, u zavisnosti od sigurnosne klase kojoj pripadaju, mogu drugačije da vide podatke u sistemu. To se postiže na dva načina. U situacijama gde je to moguće, čuva se vrsta sa najvišom sigurnosnom klasom, na osnovu koje se procesom **filtriranja** generišu vrste koje je moguće pročitati na nižim sigurnosnim nivoima. Ovakva tehnika se primenjuje u slučajevima kada je potrebno pročitati vrste relacija. U drugom, komplikovanijem slučaju, potrebno je sačuvati više vrsta sa istim prividnim ključem na različitim nivoima klasifikacije. Ove vrste potiču od operacija upisa ili modifikacije od strane korisnika sa različitim sigurnosnih nivoa. Ovaj koncept se naziva **višestruko instanciranje** (eng. *Polyinstantiation*). Ova dva koncepta biće demonstrirana kratkim primerom.

(a) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

(c) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	NULL U	NULL U	U

(b) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	NULL C	C
Brown C	NULL C	Good C	C

(d) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Smith U	40000 C	Excellent C	C
Brown C	80000 S	Good C	S

Slika 9: Primer tabele sa više nivoa klasifikacije

Na slici 9(a) prikazan je primer tabele *EMPLOYEE* sa dve vrste. Pretpostavimo da korisnici žele da pročitaju ove vrste upitom `SELECT * FROM EMPLOYEE`. U zavisnosti od sigurnosne klase kojoj pripadaju, korisnici će dobiti različite rezultate upita. Korisnik sa klasom C (ili višom) će dobiti rezultat koji je prikazan na slici 9(b) – inicijalne vrednosti atributa sa klasom S (koja je iznad C u našem modelu) biće sakriveni i umesto njih će se kao rezultat zameniti NULL vrednostima. Korisnik sa sigurnosnom klasom U će moći da vidi samo prvu vrstu i to samo ime zaposlenog kao na slici 9(c). Ovakvo funkcionisanje princip filtriranja slogova. Za potrebe demonstracije višestrukog instanciranja, pretpostavimo da korisnik sa sigurnosnim nivoom C želi da ažurira vrednost atributa *Job\_performance* korisnika koji se zove „Smith“, sledećim upitom:

**UPDATE EMPLOYEE**

**SET** Job\_performance = „Excellent“

**WHERE** Name = „Smith“;

Bez obzira na to što korisnik inicijalno nije mogao da pročita vrednost ovog atributa, operacija ažuriranja mora biti dozvoljena – u suprotnom korisnik bi logički mogao da zaključi da ova vrednost atributa postoji i da je od njega sakrivena. Ovo je primer zaključivanja informacija kroz prikriveni kanal, o kojima će biti reči u poglavlju 1.3. U sigurnim sistemima ovo je potrebno sprečiti, tako da će se, nakon izvršenja gornjeg upita, u tabelu upisati nova vrsta sa istim vrednostima koje korisnik može da pročita i ažuriranom vrednošću za atribut *Job\_performance*, ali ovaj put sa nivoom klasifikacije C (jer je to nivo korisnika koji je inicirao operaciju), kao na slici 9(c). Slična logika se primenjuje i za implementaciju INSERT i DELETE operacija.

### 1.2.5 – Poređenje diskrecione i obavezne kontrole pristupa

Diskreciona kontrola pristupa je vrlo fleksibilna i jednostavnija je za implementaciju, pa je samim tim i često primenjen mehanizam u različitim sistemima. Nedostatak ovakvog vida kontrole je veća ranjivost prilikom napada – metodama diskrecione kontrole se ne upravlja tokom informacija, nakon što im korisnik uspešno pristupi. Sa druge strane, obavezna kontrola pristupa obezbeđuje visok stepen zaštite, upravo zbog toga što je tok informacija strogo regulisan, što ovaj vid kontrole čini pogodnim za primenu u sistemima u kojima je zaštita informacija od ključne važnosti – kao što su vojni sistemi, korporativni sistemi, itd. Nedostatak ovakve kontrole su vrlo rigorozna pravila klasifikacije i organizacije sigurnosnih nivoa, koja nije moguće primeniti u svakom okruženju. Zbog navedenih osobina, diskreciona kontrola pristupa su češće korišćene, dok se obavezna kontrola pristupa implementira u kombinaciji sa njima, gde je to potrebno.

## 1.3 – Kontrola toka

U poglavlju 1.2.4 pomenuto je star-svojstvo kojim se sprečava da subjekat S upisuje, tj. modifikuje objekat O koji je na nižem sigurnosnom nivou. Ovo svojstvo prvi pogled deluje neintuitivno, međutim, ovakvim mehanizmom je obezbeđen ispravan tok podataka. Pretpostavimo da ovo svojstvo ne važi i razmotrimo sledeći primer: neki korisnik kojem je pridružena sigurnosna klasa TS može kopirati podataka iz nekog TS objekta (tabele), modifikuje ga i zatim upiše u objekat sa klasom U. Na taj način podatak koji je inicijalno strogo poverljiv postaje dostupan svim autorizovanim korisnicima – tj. dolazi do „curenja“ podataka. Zbog toga je star-svojstvo neophodno da bi se obezbedio ispravan tok podataka. Kontrolom toka se upravo sprečavaju pomenute situacije, tj. sprečava se pojava **prikrivenih kanala** (eng. *Covert channels*) – kanali kroz koje podaci teku suprotno sigurnosnim ograničenjima.

## 1.5 – Enkripcija podataka

Kontrola pristupa i kontrola toka su neizostavni vidovi zaštite u sigurnim sistemima, međutim same po sebi ne obezbeđuju apsolutnu zaštitu od različitih napada. Kada se ove mere kontrole zaobiđu na bilo koji način, korisno je da podaci budu **enkriptovani**, kako bi se sprečilo njihovo čitanje od strane neautorizovanih korisnika. **Enkripcija** je proces konverzije podataka u **šifrovani tekst** (eng. *Cyphertext*), korišćenjem nekog algoritma za enkripciju. Postoje dve grupe algoritama za enkripciju:

- **Algoritmi koji koriste simetrični ključ** – Kod ovakvih algoritama se isti ključ koristi i za enkripciju i za dešifrovanje. Nijednim drugim ključem nije moguće dešifrovati šifrovani tekst. Primeri ovakvih algoritama su **Data Encryption Standard (DES)** i danas najčešće korišćeni **Advanced Encryption Standard (AES)**. Oba algoritma se primenjuju nad blokovima ulaznih podataka nad kojima vrše operacije zamene i permutacija na osnovu ključa. AES podržava veće blokove i duže ključeve, što ga čini težim za dešifrovanje i samim tim i bezbednijim. Problem sa korišćenjem simetričnih ključeva je potreba za njihovom distribucijom do svih korisnika koji treba da imaju mogućnost dešifrovanja. Ukoliko neautorizovani korisnici dođu do ključa, moći će i da vide zaštićene podatke. Ovaj problem se rešava korišćenjem druge grupe algoritama enkripcije.
- **Algoritmi koji koriste asimetrične ključeve** – Pojavom ove grupe algoritama prevaziđen je problem deljenja simetričnog ključa između korisnika. Svaki korisnik poseduje par ključeva koji su međusobno komplementarni – tekst šifrovan jednim od njih se dešifruje korišćenjem drugog i obrnuto. Jedan od ovih ključeva je javni i dostupan je svima, dok je drugi privatni i poznat je samo jednom korisniku. Zahvaljujući ovim osobinama ključeva, kada se podaci iz sistema šalju kroz mrežu ka nekom korisniku, oni se enkriptuju korišćenjem javnog ključa tog korisnika, koji zatim može lako da dešifruje poruku korišćenjem svog privatnog ključa. Ovi algoritmi imaju i drugu dobru osobinu, a to je mogućnost da pošiljalac enkriptuje podatke svojim privatnim ključem. Ovakve poruke je moguće lako dešifrovati korišćenjem javnog ključa pošiljaoca koji je dostupan, čime je moguće izvršiti validaciju, tj. uveriti se da poruka nije promenjena na svom putu ka primaocu. Primer ovakvog algoritma je **RSA**.

U bazama podataka se enkripcija vrši na više nivoa:

- **Transparentna enkripcija podataka (TDE)** – Svi podaci u bazi podataka koji „miruju“, tj. trenutno se ne čitaju, ažuriraju, ili šalju kroz mrežu, se čuvaju enkriptovani na disku. Pojam „transparentna“ se odnosi na činjenicu da korisnik nije svestan ove enkripcije – podaci se automatski dešifruju pre učitavanja u glavnu memoriju. Za ovaj vid enkripcije se koriste algoritmi koji koriste simetrični ključ, koji se još i naziva **ključ za enkripciju baze podataka**. Ukoliko hakeri dobiju pristup fizičkom skladištu, podaci koje pročitaju biće beskorisni, zbog čega je ovaj vid zaštite jako bitan.

- **Enkripcija individualnih kolona** – Kod ovog pristupa se svaka kolona enkriptuje zasebno, čime se postiže veća fleksibilnost prilikom izbora algoritama i ključeva. Takođe je moguće korišćenje različitih ključeva za različite kolone, čime se otežava dešifrovanje podataka brute-force metodama. Glavni nedostatak ovakve enkripcije je veliki gubitak u brzini dešifrovanja prilikom učitavanja podataka, jer je potrebno dešifrovati svaku kolonu zasebno.

Često u bazi postoje podaci koji ne smeju da se pamte u „sirovom“ obliku. Najbolji primer ovakvih podataka su korisničke lozinke, koje bi mogle da se zloupotrebe ukoliko bi bile dostupne u tekstualnom formatu. Sa druge strane, one su najčešće potrebne samo za proces autentifikacije korisnika. Zbog toga se lozinke u bazama podataka **heširaju** i tako enkriptovane čuvaju u sistemu. Algoritmima heširanja se ulazni tekst konvertuje u string fiksne dužine. Pritom se za istu ulazni podataka uvek dobija i ista izlazna vrednost. Primeri ovakvih algoritama su SHA, i MD5. Nakon što korisnik unese lozinku prilikom prijavljivanja, uneti string se takođe hešira i poredi sa stringom koji se čuva u bazi. Dodatno je moguće dodati nasumični tekst na lozinku pre heširanja, kako bi se otežalo njeno provaljivanje. Ova tehnika se naziva „soljenje“ (eng. ***Salt***ing).

## 2 – Sigurnost kod Oracle baze podataka

U poglavlju 1, je dat pregled nekih teorijskih osnova i glavnih tehnika kojima se obezbeđuje sigurnost sistema baza podataka. U poglavlju 1.2.3 su takođe navedeni neki primeri upravljanja privilegijama u Oracle bazi podataka. Zadatak ovog poglavlja je da na sličan način demonstrira još neke od pomenutih tehnika zaštite baze podataka na primeru Oracle DBMS-a. Osim toga biće definisani i specifični mehanizmi koje Oracle koristi za obezbeđivanje sigurnosti, uz neke dodatne, prateće alate koji su danas neizostavni deo ozbiljnjih informacionih sistema.

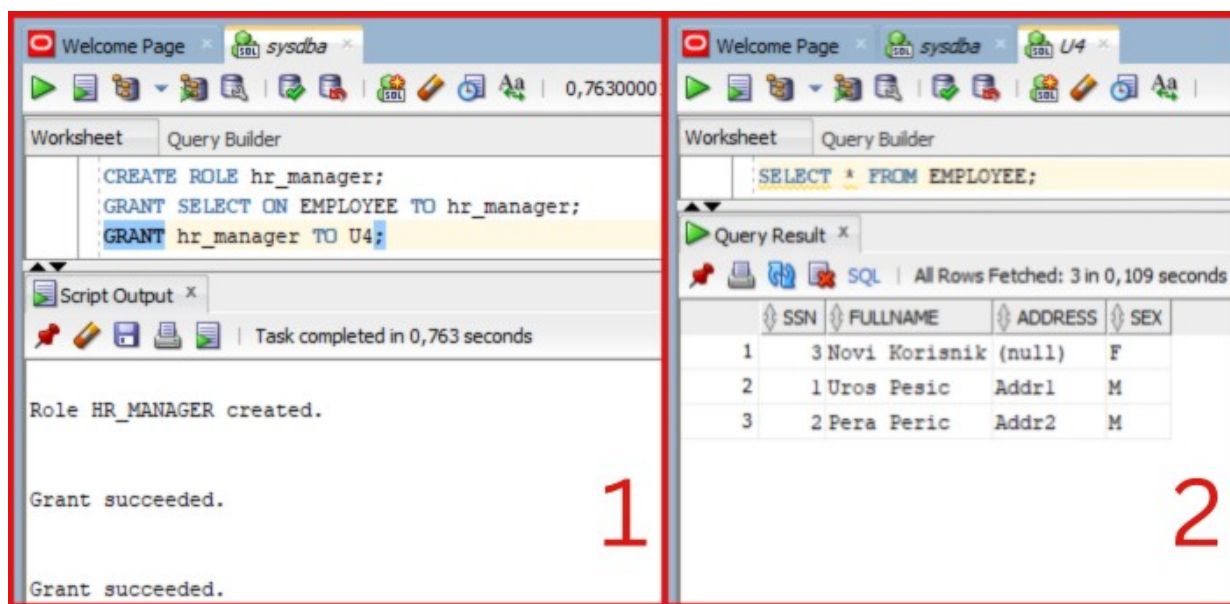
### 2.1 – Kontrola pristupa zasnovana na ulogama

Tema poglavlja 1.2.2 i 1.2.3 su bile tehnike diskrecione kontrole pristupa – kontrole pristupa koja se zasniva na dodeljivanju i uklanjanju privilegija korisnicima. Gotovo svi relacioni DBMS-ovi danas podržavaju grupisanje korisnika i njihovih privilegija u više grupa, tj. **uloga** (eng. ***Roles***). Iako ovaj mehanizam nije specifičan za Oracle, predstavlja njegov bitan aspekt, koji je potrebno razumeti da bi se moglo diskutovati o nekim drugim Oracle-ovim tehnikama o kojima će biti reči u nastavku.

Umesto da definišemo privilegije pojedinačno za svakog korisnika, mnogo je lakše definisati konkretne uloge koje korisnici imaju u sistemu – ove uloge uglavnom oslikavaju ulogu koju korisnik ima u samoj organizaciji. Uloge se kreiraju naredbom CREATE ROLE i brišu naredbom



DESTROY ROLE. Nakon kreiranja uloge, moguće joj je dodeliti privilegije na isti način kao i individualnim korisničkim nalogima. Ovakav primer prikazan je na slici 10.



Slika 10: Primer kreiranja uloge

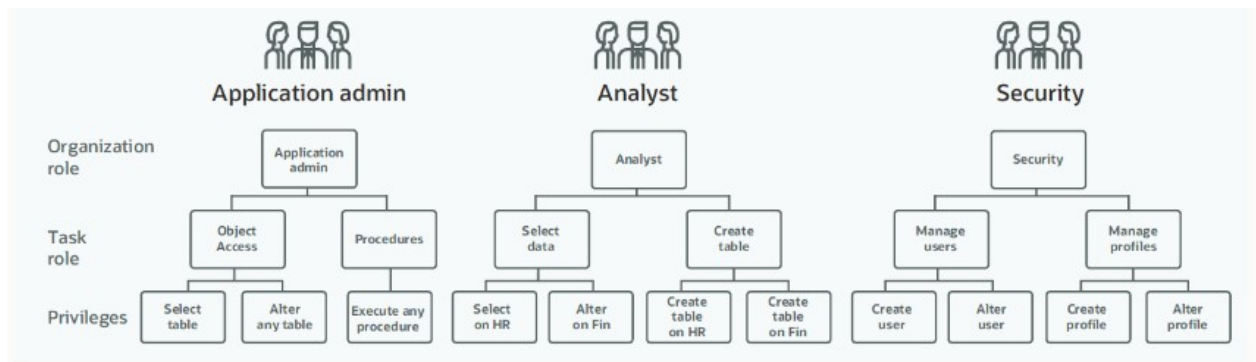
U ovom primeru je kreirana uloga „hr\_manager“ i dodeljena korisniku U4. Korisnicima koji imaju ovu ulogu je dodeljena privilegija čitanja tabele *EMPLOYEE*.

Uloge je moguće i hijerarhijski organizovati, tako da uloge u bazi u potpunosti oslikavaju i uloge koje korisnici u samoj organizaciji. Ukoliko za primer uzmemo neku softversku kompaniju, prirodno bismo definisali uloge *junior*, *medior*, *senior*. Sve privilegije *juniora* bi imao i *medior*, dok bi sve privilegije *juniora* i *mediora* imao i *senior*. Ovo je lako postići jedonstavnim dodeljivanjem jedne uloge drugoj korišćenjem GRANT naredbe, čime druga uloga preuzima sve privilegije prve uloge, tj. postaje njen nadskup. U našem primeru bi se za to izvršile sledeće naredbe:

**GRANT junior TO medior;**

**GRANT medior TO senior;**

Primer hijerarhijski uređenih uloga u nekoj organizaciji dat je na sledećoj slici:



Slika 11: Hijerarhijski uređene uloge

Grupisanjem korisnika u više uloga se olakšava upravljanje privilegijama. Međutim, potrebno je posvetiti pažnju pravilnom **razdvajanju zaduženja** (*eng. Separation of duties*). Većina korisnika baze podataka, počev od administratora koji upravlja bazom, pa do članova tima za poslovnu inteligenciju koji vrše analizu podataka, ima vrlo specifična zaduženja koja obavlja. Shodno tim zaduženjima je potrebno pažljivo kreirati odgovarajuće uloge i dodeliti im *samo one privilegije koje su njima potrebne*. Na ovaj način se postiže bolja bezbednost sistema, jer se sprečavaju zloupotrebe privilegija, kako od strane samih korisnika, tako i od strane hakera koji ostvare pristup nekom nalogu. Preporuka je da se i za administrativne zadatke kreiraju novi DBA nalozi, koji će imati samo podskup svih sistemskih mogućnosti koje su neophodne za njihovo ispunjavanje, upravo zbog toga što su DBA nalozi najčešća meta hakerskih napada. Ovom temom analize i bezbednim upravljanjem privilegija se bavimo u narednom poglavlju.

## 2.2 – Analiza privilegija u Oracle-u

Analiza privilegija predstavlja praćenje upotrebe privilegija od strane različitih korisnika i različitih uloga u definisanom vremenskom okviru, kako bi se detektovale neiskorišćene privilegije i tako ispoštovao model **minimalnih potrebnih privilegija**. Prema ovom modelu, korisnicima se dodeljuje tačno onaj skup privilegija koji je *neophodan* za izvršavanje njihovih zadataka. Na taj način se obezbeđuje ispravno razdvajanje zaduženja i samim tim bolja sigurnost sistema - kao što je opisano na kraju prethodnog poglavlja.

Čak i u sistemima u kojima su uloge i privilegije jasno definisane, često dolazi do njihovog bespotrebnog nagomilavanja i drugih previda. Kao primer, pretpostavimo da postoji korisnik u sistemu koji inicijalno treba da ima mogućnost čitanja podataka iz svih tabela u nekoj šemi. Međutim, umesto da mu administrator dodeli adekvatne privilegije za svaku od objekata (tabela) posebno, on mu naivno dodeli sistemsku privilegiju za selekciju iz bilo koje tabele (*SELECT ANY TABLE*). Iako smo na prvi pogled na ovaj način postigli isti efekat, ukoliko bismo kreirali bilo koju novu tabelu koja čuva osetljive podatke kojima pomenuti korisnik ne bi smeo da pristupi, greška postaje očigledna. Razmotrimo i drugi primer: u toku razvoja ili održavanja aplikacije, programeru su često potrebne neke dodatne privilegije koje ne bi smeo da poseduje kada je aplikacija puštena u rad. Često se dešava da prilikom ovakvih promena „konteksta“ dođe do propusta u upravljanju privilegijama. Postojanje mehanizma analize privilegija ovakve propuste je moguće brzo uočiti i adekvatno reagovati.

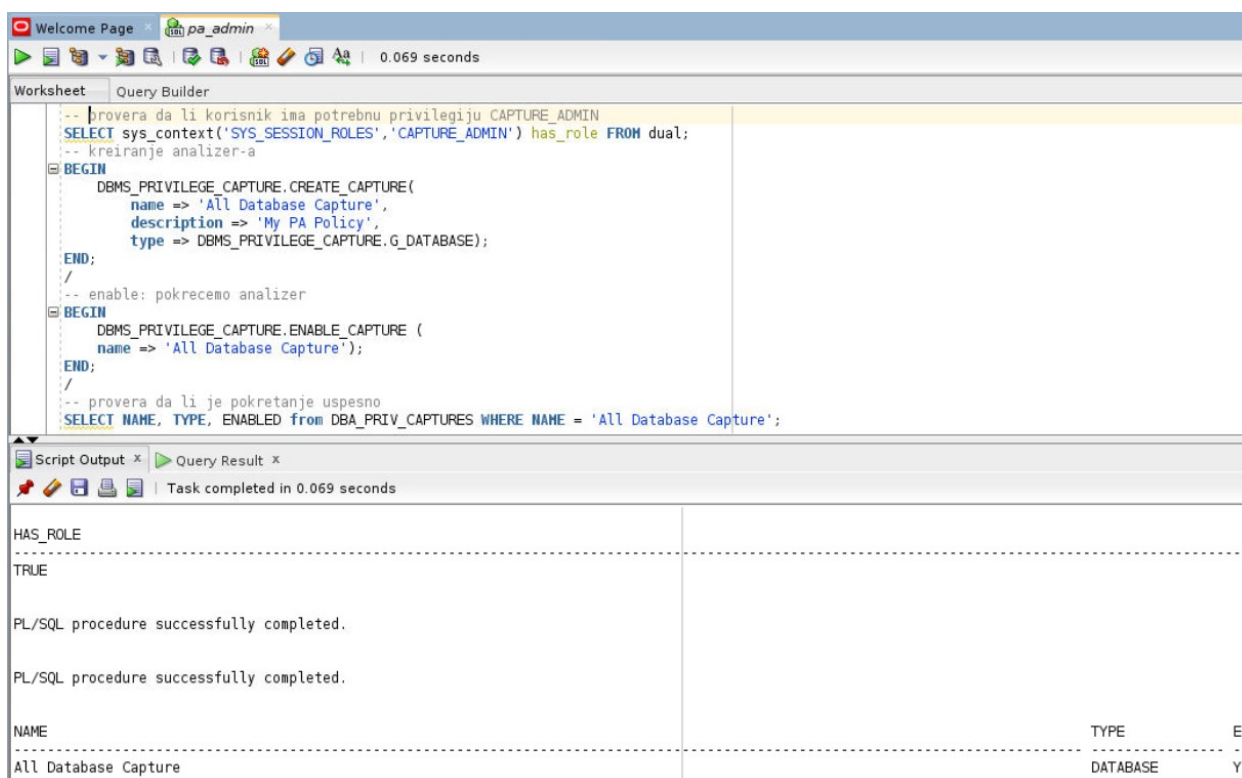
Postoje četiri različita tipa analize privilegija:

- **Analiza na osnovu konteksta (context-based)** – Korišćenjem SYS\_CONTEXT funkcije se definiše uslov i evidencija o privilegijama se beleži samo ukoliko je uslov ispunjen. Na ovaj način je moguće analizirati privilegije za specifičnog korisnika, za neku adresu hosta, itd.
- **Analiza zasnovana na ulogama** – Analizira se upotreba privilegija samo za konkretnu listu uloga, koja se navodi prilikom definisanja pravila za analizu.

- **Analiza na osnovu uloge i konteksta** – Predstavlja kombinaciju prva dva tipa analize.
- **Analiza na nivou baze (bezuslovna analiza)** – Analiza privilegija se vrši na nivou cele baze: za sve korisnike, tj. uloge u svim kontekstima. Jedini nalog za koji se ne vodi evidencija o privilegijama u ovom slučaju je SYS nalog.

Nalog sa koga se kreira i pokreće analiza privilegija mora da poseduje **CAPTURE\_ADMIN** privilegiju. Sve funkcije neophodne za definisanje pravila analize, tipa analize, pokretanje i ostale operacije su deo **DBMS\_PRIVILEGE\_CAPTURE** PL/SQL paketa u Oracle-u. Primer bezuslovne analize korišćenja privilegija je dat u nastavku.

Prvi korak u analizi je kreiranje pravila za analizu privilegija, pozivom procedure **CREATE\_CAPTURE**. Nakon toga je analizu moguće pokrenuti procedurom **ENABLE\_CAPTURE** u kojoj kao parametar treba staviti ime prethodno kreiranog pravila. Analiza se prekida procedurom **DISABLE\_CAPTURE**. Sve naredbe koje su izvršene u vremenskom prozoru između poziva pokretanja i prekida analize će biti analizirane, kako bi se prikupile informacije o korišćenim privilegijama. Nakon analize je moguće obrisati kreirano pravilo procedurom **DROP\_CAPTURE**. Na sledećoj slici je praktično demonstrirana upotreba ovih procedura.



Slika 12: Analiza privilegija korišćenjem DBMS\_PRIVILEGE\_CAPTURE paketa

U primeru sa slike je vrši bezuslovna analiza (što možemo zaključiti na osnovu parametra *type*). Za analizu na osnovu konteksta je ovaj parametar potrebno postaviti na G\_CONTEXT, za analizu na osnovu uloga na G\_ROLE, dok je za kombinovanu analizu potrebno postaviti vrednost G\_ROLE\_AND\_CONTEXT.

Pre završetka analize su izvršavani različiti SQL upiti od strane različitih korisnika, kako bi se simulirali realni uslovi u sistemu u produkciji. Pozivom procedure **DBMS\_PRIVILEGE\_CAPTURE.GENERATE\_RESULT** je izvršeno generisanje rezultata analize:

Grantee	Type	Used	Revoked	System Privileges		Object Privileges	
				Unused	Used	Unused	Used
CTXSYS	User			20		46	
PA_ADMIN	User			11	1	42	5
ORACLE_OCM	User			2		57	
WMSYS	User			30		15	
AUDSYS	User			7		25	4
DVSYS	User			4		28	
PU_PETE	User			11	1	5	5
System Privileges	Folder			2	1		
UNLIMITED TABLESPACE	System Privileg						
SELECT ANY TABLE	System Privileg						
CREATE SESSION	System Privileg	✓					

Slika 13: Rezultati analize korišćenja privilegija

Na slici 13 možemo da vidimo mali deo rezultata analize korišćenjem Oracle Enterprise Manager-a<sup>3</sup>. Sa slike možemo da zaključimo da je korisnik **PU\_PETE** koristio sistemsku privilegiju **CREATE\_SESSION**, tj. konektovao se na bazu, dok su druge dve sistemske privilegije koje ima ostale neiskorišćene (u ovom slučaju su to **SELECT ANY TABLE** i **UNLIMITED TABLESPACE**), na osnovu čega bismo mogli da razmotrimo ukidanje ove dve jako moćne privilegije ovom korisniku. U realnim uslovima bi se analiza vršila periodično u određenim vremenskim periodima, i na osnovu rezultata bi se donosili zaključci o tome koje je privilegije moguće ukloniti.

## 2.3 – Objedinjena revizija (Unified Auditing)

U prvom poglavlju je već definisan termin **revizije** baze podataka, kao vrlo bitan mehanizam za praćenje aktivnosti korisnika radi detektovanja sumnjivih obrazaca ponašanja i malicioznih delovanja. Ukoliko do ovakvih aktivnosti i dođe, revizijom je moguće utvrditi nalog sa kojeg su nedozvoljene operacije izvršene, a zahvaljujući tome je moguće daljom istragom detektovati kako i zašto je došlo do takvog sigurnosnog propusta. Na osnovu svega navedenog, lako je zaključiti da efikasni sistem revizije igra bitnu ulogu u sigurnosti čitavog sistema. Oracle je godinama, kroz različite verzije stalno unapređivao dostupne metode za reviziju baze podataka: stalno se proširivao skup objekata za koji je moguće pratiti operacije koje se nad njima izvršavaju,

3 Zbog bolje preglednosti, rezultati analize su prikazani korišćenjem pomoćnog alata - Oracle Enterprise Manager-a. Iste rezultate je moguće dobiti jednostavnim izvršavanjem upita nad pogledima **DBA\_USED\_PRIVS**, **DBA\_UNUSED\_PRIVS**, **DBA\_(UN)USED\_SYSPRIVS**, **DBA\_(UN)USED\_USERPRIVS**, itd.

menjao se način implementacije logovanja događaja radi poboljšanja performansi sistema (logovanje predstavlja dodatni overhead o kome je potrebno voditi računa), vremenom je implementiran i **fine-grained auditing** – tehnika zahvaljujući kojoj je postalo moguće pratiti operacije i nad individualnim kolonama od interesa, a ne na nivou čitavih tabela, itd. Konačno, Oracle je počev od verzije 12c definisao **objedinjenu reviziju**.

Objedinjena revizija, kako joj i ime govori, omogućava prikupljanje informacija o događajima u sistemu iz više izvora u jedinstvenu AUD\$UNIFIED tabelu, koja je deo AUDSYS šeme. Podaci za reviziju mogu biti:

- Događaji od interesa (definisani pravilima)
- Događaji od interesa za fine-grained reviziju
- Database Vault događaji
- Oracle Label Security događaji
- Događaji iz Oracle menadžera za oporavak (Recovery Manager)

Činjenica da se slogovi za reviziju iz svih ovih izvora čuvaju na jednom mestu značajno olakšava upravljanje revizijom, obezbeđuje bolju i jednostavniju kontrolu upotrebe memorijskog prostora zahvaljujući većoj selektivnosti pravila koja je moguće definisati, a samim tim i poboljšava performanse čitavog sistema. Takođe, tabela AUD\$UNIFIED u kojoj se čuvaju sve informacije o događajima je „zaključana“ iza specijalnog AUDSYS naloga, kome nije moguće pristupiti, čime su informacije dobro zaštićene od bilo kakvih promena. Njih je samo moguće pročitati i to kroz pogled *UNIFIED\_DICTIONARY\_VIEW*.

Kroz sledeća dva primera je demonstrirano kreiranje pravila za logovanje operacija nad tabelama i logovanje aktivnosti korisnika koja pripadaju nekoj ulozi. Korisnik koji kreira pravila za beleženje događaja za reviziju mora da ima privilegiju *AUDIT\_ADMIN*, dok korisnici koji vrše reviziju pregledavanjem logova moraju da imaju dodeljenu privilegiju *AUDIT\_VIEWER*. Na sledećoj slici prikazan je primer kreiranja pravila na osnovu koga se beleže informacije o izvršenim operacijama.

The screenshot shows the Oracle SQL Developer interface. The top pane is the 'Query Builder' with a SQL script. The script includes comments in Serbian and SQL commands to check if unified auditing is enabled, create an audit policy named 'AUDIT\_EMPLOYEESEARCH\_USAGE' for the 'EMPLOYEESEARCH\_PROD.DEMO\_HR\_EMPLOYEES' and 'EMPLOYEESEARCH\_PROD.DEMO\_HR\_USERS' tables, and enable the policy. The bottom pane shows the 'Script Output' and 'Query Result' tabs. The 'Query Result' tab displays a table with two columns: 'PARAMETER' and 'VALUE'. The first row shows 'Unified Auditing' with a value of 'TRUE'. Below the table, it states 'Audit POLICY created.'

```
-- proveravamo da li je unified auditing ukljucen
SELECT PARAMETER, VALUE FROM v$option WHERE PARAMETER='Unified Auditing';

-- kreira se policy koji prati sve operacije nad tabelama DEMO_HR_EMPLOYEES i DEMO_HR_USERS koje ne dolaze iz web aplikacije
-- ovaj uslov je definisan u when delu
CREATE AUDIT POLICY AUDIT_EMPLOYEESEARCH_USAGE
ACTIONS ALL ON EMPLOYEESEARCH_PROD.DEMO_HR_EMPLOYEES, ALL ON EMPLOYEESEARCH_PROD.DEMO_HR_USERS
WHEN 'SYS_CONTEXT(''USERENV'', 'SESSION_USER') = 'EMPLOYEESEARCH_PROD'
AND (SYS_CONTEXT(''USERENV'', 'OS_USER') != 'oracle'
OR SYS_CONTEXT(''USERENV'', 'MODULE') != 'JDBC Thin Client'
OR SYS_CONTEXT(''USERENV'', 'HOST') != 'dbsec-lab.dbsecvcn.oraclevcn.com')
EVALUATE PER STATEMENT;

-- enable policy
AUDIT POLICY audit_employeesearch_usage;
```

PARAMETER	VALUE
Unified Auditing	TRUE

Audit POLICY created.

Slika 14: Kreiranje pravila za beleženje informacija o operacijama



Sa slike možemo da uočimo da je kreirano pravilo naredbom **CREATE AUDIT POLICY**, kojim se nalaže beleženje informacija o svim operacijama koje se izvršavaju nad tabelama *DEMO\_HR\_EMPLOYEES* i *DEMO\_HR\_USERS* u šemi *EMPLOYEESEARCH\_PROD* i to samo o onim operacijama koje ne potiču sa web servera na adresi *dbsec-lab.dbsecvcn.oraclevcn.com*, ili nisu su pozvane od strane korisnika operativnog sistema „oracle“ – što možemo zaključiti na osnovu uslova koji je definisan u WHEN delu naredbe. Klauzulom **EVALUATE PER** se definiše kada će se vrednost pomenutog uslova procenjivati. Postoje sledeće mogućnosti:

- **PER STATEMENT** – Za svaku naredbu od interesa (naredbe od interesa su specificirane u **ACTIONS** delu) će se proveriti da li je vrednost uslova TRUE. Ukoliko jeste, naredba se beleži za reviziju. Ova varijanta je korišćena u našem primeru.
- **PER SESSION** – Vrednost uslova se računa samo jednom u toku sesije i to prilikom pojave prve naredbe od interesa. Ukoliko je vrednost uslova TRUE za tu prvu naredbu, sve ostale naredbe u toku sesije će biti zabeležene (naravno ukoliko su definisane u **ACTIONS** delu).
- **PER INSTANCE** – Slična logika kao kod prethodne varijante, sa tim što se uslov procenjuje jednom u toku životnog ciklusa instance (čak i u slučaju više različitih sesija uslov ostaje isti).

Nakon kreiranja opisanog uslova, potrebno je startovati pravilo naredbom **AUDIT POLICY**. Zatim je „ručno“ izvršeno par upita korišćenjem Oracle SQL Developera. Kako ovi upiti nisu upućeni iz web aplikacije, WHEN uslov je zadovoljen i informacije o operacijama su zabeležene, kao što možemo videti na slici 15.

The screenshot displays the Oracle SQL Developer interface. The top pane shows two SQL queries. The first query is a simple select statement for employee data. The second query is more complex, filtering the audit trail for specific policies. The bottom pane shows the results of the second query, which are three rows of audit events.

EVENT_TIMESTAMP	USERHOST	DBUSERNAME	ACTION_NAME	OBJECT_SCHEMA	OBJECT_NAME	SQL_TEXT
1 25-MAY-24 09.23.57.753051000 PM	dbsec-lab	EMPLOYEESEARCH_PROD	SELECT	EMPLOYEESEARCH_PROD	DEMO_HR_EMPLOYEES	select userid, firstname, lastname,
2 25-MAY-24 09.30.24.996486000 PM	dbsec-lab	EMPLOYEESEARCH_PROD	SELECT	EMPLOYEESEARCH_PROD	DEMO_HR_EMPLOYEES	select * from EMPLOYEESEARCH_PROD.DE
3 25-MAY-24 09.30.27.173532000 PM	dbsec-lab	EMPLOYEESEARCH_PROD	SELECT	EMPLOYEESEARCH_PROD	DEMO_HR_EMPLOYEES	select firstname, lastname from EMPL

Slika 15: Unified Auditing - Audit Trail - operacije

Takođe, na sličan način je moguće beležiti informacije o aktivnostima pojedinih korisničkih uloga, što je ilustrovano primerom na sledećoj slici:

**Worksheet 1: Query Builder** (Unshared SQL Worksheet (Ctrl+Shift+N))

```

create role mgr_role;
grant create tablespace to mgr_role;
grant mgr_role, create session to dba_nicole;
create user dba_junior identified by "Oracle123";
grant dba to dba_junior;

create audit policy aud_role_pol ROLES mgr_role;
create audit policy aud_dba_pol ROLES dba;
audit policy AUD_ROLE_POL;
audit policy AUD_DBA_POL;

select POLICY_NAME, AUDIT_OPTION, CONDITION_EVAL_OPT from AUDIT_UNIFIED_POLICIES where POLICY_NAME in ('AUD_ROLE_POL','AUD_DBA_POL');

```

**Query Result 1:** All Rows Fetched: 2 in 4.357 seconds

POLICY_NAME	AUDIT_OPTION	CONDITION_EVAL_OPT
1 AUD_ROLE_POL	MGR_ROLE	NONE
2 AUD_DBA_POL	DBA	NONE

**Worksheet 2: Query Builder**

```

select action_name, dbusername, action_name, OBJECT_SCHEMA, object_name, SQL_TEXT
from unified_audit_trail
where unified_audit_policies like '%AUD_DBA_POL%'
or unified_audit_policies like '%AUD_ROLE_POL%'
order by event_timestamp desc;

```

**Query Result 2:** All Rows Fetched: 5 in 0.021 seconds

ACTION_NAME	DBUSERNAME	ACTION_NAME_1	OBJECT_SCHEMA	OBJECT_NAME	SQL_TEXT
1 ALTER SYSTEM	DBA_JUNIOR	ALTER SYSTEM	(null)	(null)	alter system set job_queue_processes=100
2 ALTER SYSTEM	DBA_JUNIOR	ALTER SYSTEM	(null)	(null)	alter system set job_queue_processes=200
3 LOGON	DBA_JUNIOR	LOGON	(null)	(null)	(null)
4 DROP TABLESPACE	SYS	DROP TABLESPACE	(null)	TEST	drop tablespace test including contents and datafiles
5 CREATE TABLESPACE	SYS	CREATE TABLESPACE	(null)	TEST	create tablespace test datafile '/u01/oradata/cdb1/pdb1/test01.dbf'

Slika 16: Audit Trail - uloge

Prilikom prikupljanja podataka za reviziju, preporučuje se da se akcenat stavi na tabele i kolone koje sadrže osetljive podatke, kako ne bi došlo do bespotrebnog preopterećenja sistema. Takođe, iz istih razloga se preporučuje periodično arhiviranje i brisanje podataka za reviziju (trag revizije – eng. *audit trail*).

## 2.3 – Database Vault

Oracle Database Vault (sef baze podataka) je rešenje koje se isporučuje uz Enterprise verziju Oracle baze i predstavlja dodatni vid zaštite sistema od neautorizovanog pristupa, pomaže u sprečavanju zloupotrebe privilegija i može da zaštiti sistem od nenamernih ljudskih grešaka. Može se posmatrati kao još jedan zaštitni sloj sistema, u kojem je moguće definisati **zaštićene prostore** (eng. *Security realms*) i **komandna pravila** (eng. *Command rule*) na osnovu kojih je moguće sprečiti izvršenje neke operacije, čak i ukoliko je izvršava autorizovani korisnik. Komponente ovog podsistema koje služe za poboljšanje sigurnosti su sledeće:

- **Realms** – Predstavljaju „sigurnosne zone“ koje mogu da obuhvataju šeme, tabele, uloge i druge objekte sistema. Korišćenjem sigurnosnih zona je obezbeđena fleksibilnija kontrola pristupa zaštićenim objektima. Na primer, ukoliko je potrebno moguće je zaštititi šeme u sistemu koje sadrže tabele sa osetljivim podacima, i sprečiti pristup njima, čak i od strane autorizovanih korisnika (čak i od strane DBA naloga).

- **Komandna pravila** – Predstavljaju specijalni tip pravila, kojima je moguće definisati kada i pod kojim uslovima je moguće izvršiti bilo koju SQL naredbu. Na primer, moguće je sprečavati modifikaciju neke tabele u sistemu van vremenskog perioda 09h – 17h (van radnog vremena), jer bi to predstavljalo neki vid sumnjive modifikacije.
- **Skup pravila** – Skup individualnih pravila koja se koriste za kreiranje prostora zaštite i komandnih pravila. Ukoliko se skup sastoji iz više od jednog pravila, evaluaciju je moguće vršiti po principu *sva pravila su ispunjena*, ili *bar jedno pravilo je ispunjeno*
- **Faktori** – Različiti atributi, koji mogu biti IP adresa baze, username ulogovanog korisnika, čije vrednosti Database Vault koristi za evaluaciju pravila. Npr. na osnovu IP adrese sa kojeg je upućena naredba, moguće je proceniti da ona nije bezbedna za izvršenje.

Oracle obezbeđuje PL/SQL pakete sa definisanim API-em za rad sa svakom od navedenih komponenti. Za rad sa Database Vault-om, potrebno je definisati par postojećih korisnika, od kojih će jednom biti dodeljena privilegija *DV\_OWNER* – koja mu omogućava kreiranje i upravljanje komponentama Vault-a, dok će drugi imati privilegiju *DV\_ACCTMGR* koja mu omogućava upravljanje korisničkim nalogima. Preporučuje se da za ova dva tipa korisnika postoje i rezervni nalozi.

U narednom primeru demonstrirana je upotreba Database Vault-a za kreiranje realm-a za zaštitu svih objekata (tabela, pogleda, itd) u šemi *EMPLOYEESEARCH\_PROD*:

1. Najpre je potrebno omogućiti korišćenje Database Vault-a korišćenjem sledeće PL/SQL procedure:

```
BEGIN
DVSYS.CONFIGURE_DV(
    dvowner_uname => 'C##DVOWNER',
    dvacctmgr_uname => 'C##DVACCTMGR');
END;
```

Izvršavanjem naredbe „**SELECT \* FROM dba\_dv\_status**“ možemo se uveriti da je DV konfigurisan uspešno (*DV\_ENABLE STATUS* parametar).

NAME	STATUS
-----	-----
DV_APP_PROTECTION	NOT CONFIGURED
DV_CONFIGURE_STATUS	TRUE
DV_ENABLE_STATUS	TRUE

Slika 17: Provera rezultata *DVSYS.CONFIGURE\_DV* procedure



2. Zatim kreiramo realm kao na sledećoj slici:

The screenshot shows a SQL script in the editor and its execution results. The script creates a realm and then queries the database to verify its creation.

```
begin
  DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(
    realm_name => 'PROTECT_EMPLOYEESEARCH_PROD'
    ,object_owner => 'EMPLOYEESEARCH_PROD'
    ,object_name => '%'
    ,object_type => '%');
end;
/

select realm_name, owner, object_name, object_type
from dvsys.dba_dv_realms
where realm_name in (select name from dvsys.dv$realm where id# >= 5000);
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.039 seconds

REALM_NAME	OWNER	OBJECT_NAME	OBJECT_TYPE
1 PROTECT_EMPLOYEESEARCH_PROD	EMPLOYEESEARCH_PROD	%	%

NAME	DESCRIPTION	ENABLED
1 PROTECT_EMPLOYEESEARCH_PROD	A mandatory realm to protect the EMPLOYEESEARCH_PROD schema.	Y

Slika 18: Kreiranje Realm-a

Realm koji smo kreirali je odmah po kreiranju omogućen (*enabled* opcija) i konfigurisan je tako da svaki neautorizovani pokušaj pristupa evidentira za buduću reviziju (*audit\_options* opcija). Rezultatom *select* naredbe možemo da potvrdimo da je realm uspešno kreiran.

3. Nakon što je realm kreiran, potrebno je definisati koje objekte će on da obuhvata, tj. obezbeđuje. To radimo korišćenjem procedure **DVSYS.ADD\_OBJECT\_TO\_REALM**, kao na slici:

The screenshot shows the same SQL script as in Slika 18, but with the execution results showing that objects are being added to the realm.

```
begin
  DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(
    realm_name => 'PROTECT_EMPLOYEESEARCH_PROD'
    ,object_owner => 'EMPLOYEESEARCH_PROD'
    ,object_name => '%'
    ,object_type => '%');
end;
/

select realm_name, owner, object_name, object_type
from dvsys.dba_dv_realms
where realm_name in (select name from dvsys.dv$realm where id# >= 5000);
```

Script Output x Query Result x

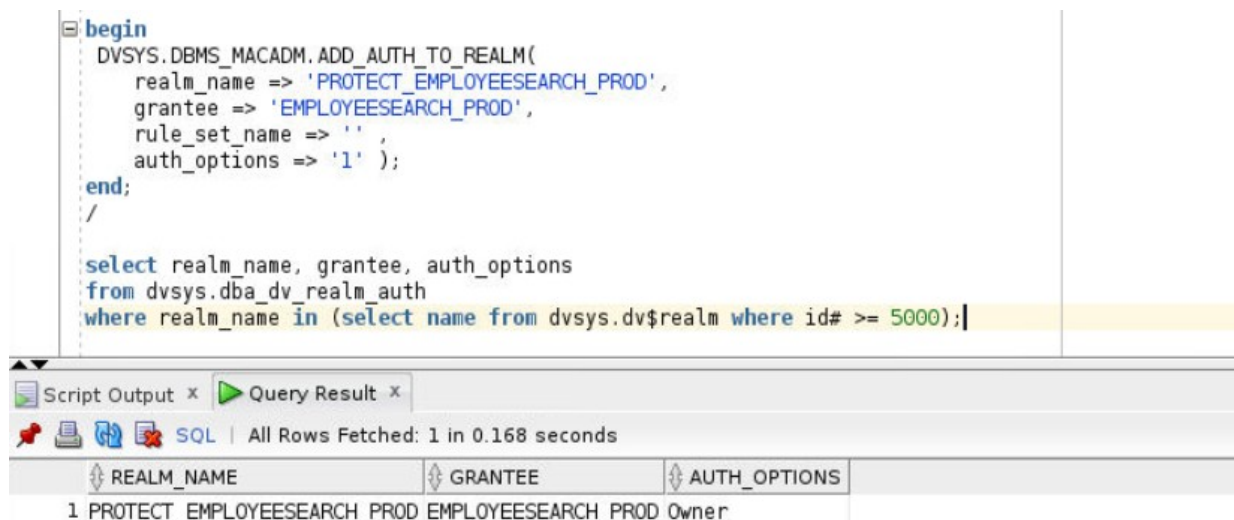
SQL | All Rows Fetched: 1 in 0.039 seconds

REALM_NAME	OWNER	OBJECT_NAME	OBJECT_TYPE
1 PROTECT_EMPLOYEESEARCH_PROD	EMPLOYEESEARCH_PROD	%	%

Slika 19: Dodeljivanje objekata prethodno kreiranom realm-u

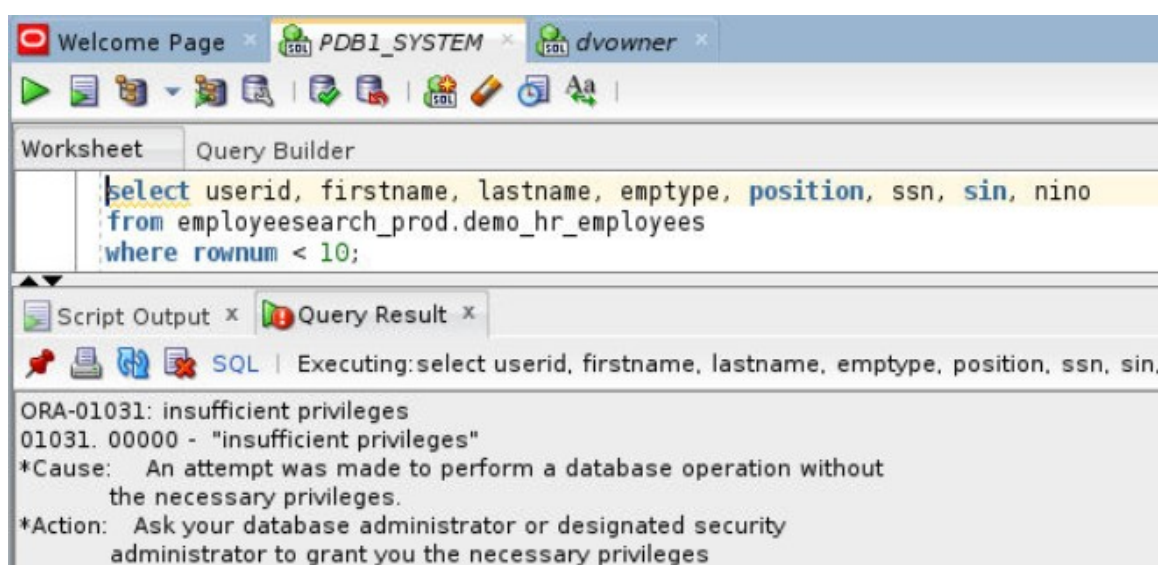
Svi objekti koji pripadaju šemi EMPLOYEESEARCH\_PROD su zaštićeni (što možemo da zaključimo na osnovu upotrebe % karaktera za parametre *object\_name* i *object\_type*).

1. Realm je kreiran dodeljeni su mu objekti koje obezbeđuje. Nakon toga potrebno je definisati kojim korisnicima (ili ulogama) je dozvoljeno da mu pristupe. Na taj način je moguće dodatno ograničiti prava pristupa za već autorizovane korisnike, kao što je već i pomenuto.



Slika 20: Autorizacija korisnika za pristup realm-u

- Na osnovu parametara možemo da zaključimo da je pristup šemi dozvoljen samo korisniku koji *EMPLOYEESEARCH\_PROD* (koji je i vlasnik šeme). Parametrom *auth\_options* se kontroliše da li je korisnik autorizovan kao vlasnik (vrednost 1 kao u našem primeru), ili kao učesnik (vrednost 0). Dodatno je moguće specificirati skup pravila koja moraju biti ispunjena kako bi se autorizovao pristup realmu – na taj način je moguće obezbediti još rigorozniju kontrolu.
2. Sada, ukoliko probamo da pročitamo podatke iz neke tabele zaštićene šeme, dobijamo poruku o nedovoljnim privilegijama, kao na slici 21. Interesantno je da na slici 20 ovu operaciju izvršava korisnik SYSDBA – čak ni on nema pristup zaštićenim objektima! Zbog toga DV predstavlja jedan od načina za obezbeđivanje pravilne podele zaduženja.



Slika 21: Pokušaj pristupa zaštićenoj tabeli

Drugi primer demonstrira korišćenje DV-a za kontrolu putanja sa kojih je moguće prijavljivanje na sistem. Definisanjem „pouzdanih konekcija“ se sistem dodatno obezbeđuje u slučaju da hakeri uspeju da se dokopaju naloga sa jakim privilegijama.

1. Prvo je potrebno kreirati adekvatno pravilo, kojim se definiše koji korisnik i sa koje adrese hosta može da pristupi sistemu. Takođe potrebno je kreirati i skup pravila koji će da sadrži prethodno kreirano pravilo, kao na sledećoj slici:

```

begin
  DVSYS.DBMS_MACADM.CREATE_RULE(
    rule_name => 'Application Connection',
    rule_expr => 'SYS_CONTEXT(''USERENV'', ''SESSION_USER'') = ''EMPLOYEESEARCH_PROD''
      AND SYS_CONTEXT(''USERENV'', ''OS_USER'') = ''oracle''
      AND SYS_CONTEXT(''USERENV'', ''MODULE'') = ''JDBC Thin Client''
      AND SYS_CONTEXT(''USERENV'', ''HOST'') = ''dbsec-lab''');
end;
/

begin
  DVSYS.DBMS_MACADM.CREATE_RULE_SET(
    rule_set_name => 'Trusted Application Path'
  ,description    => 'Protecting the App User'
  ,enabled        => DBMS_MACUTL.G_YES
  ,eval_options   => DBMS_MACUTL.G_RULESET_EVAL_ALL
  ,audit_options  => DBMS_MACUTL.G_RULESET_AUDIT_FAIL
  ,fail_options   => DBMS_MACUTL.G_RULESET_FAIL_SHOW
  ,fail_message   => 'You cannot use the app account this way.'
  ,fail_code      => -20000
  ,handler_options => null
  ,handler        => null
  ,is_static      => TRUE);
end;
/

```

Slika 22: Kreiranje pravila i skupa pravila

Parametrom *eval\_options* se kontroliše da li će se evaluacija vršiti po principu *sva pravila* ili *bar jedno*. Pošto u našem slučaju kreiramo samo jedno pravilo, ovaj parametar nije bitan.

2. Zatim dodajemo kreirano pravilo u skup pravila:

```

begin
  DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(
    rule_set_name => 'Trusted Application Path'
  ,rule_name      => 'Application Connection'
  ,rule_order     => 1
  ,enabled        => DBMS_MACUTL.G_YES);
end;
/

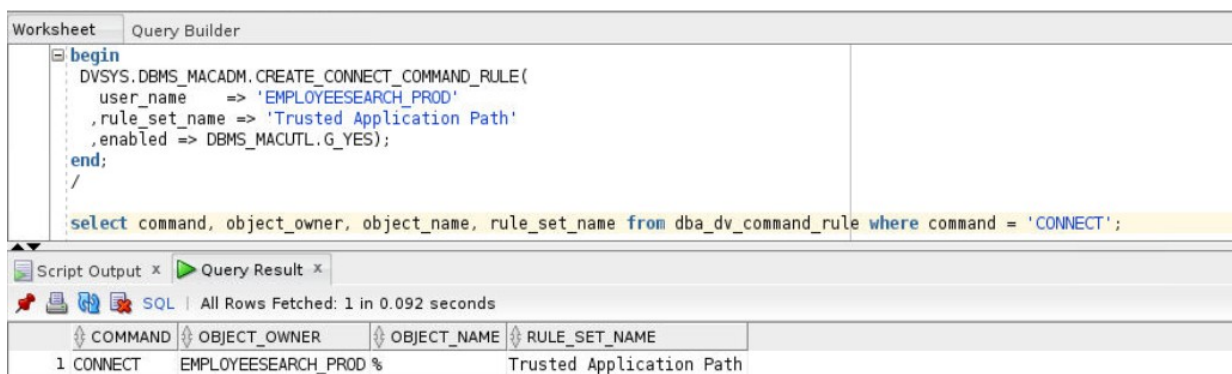
SELECT rule_set_name, enabled, eval_options_meaning, audit_options, fail_message, fail_code, is_static
FROM DBA_DV_RULE_SET
WHERE rule_set_name = 'Trusted Application Path';

```

RULE_SET_NAME	ENABLED	EVAL_OPTIONS_MEANING	AUDIT_OPTIONS	FAIL_MESSAGE	FAIL_CODE	IS_STATIC
1 Trusted Application Path	Y	All True	1	You cannot use the app account this way.	-20000	TRUE

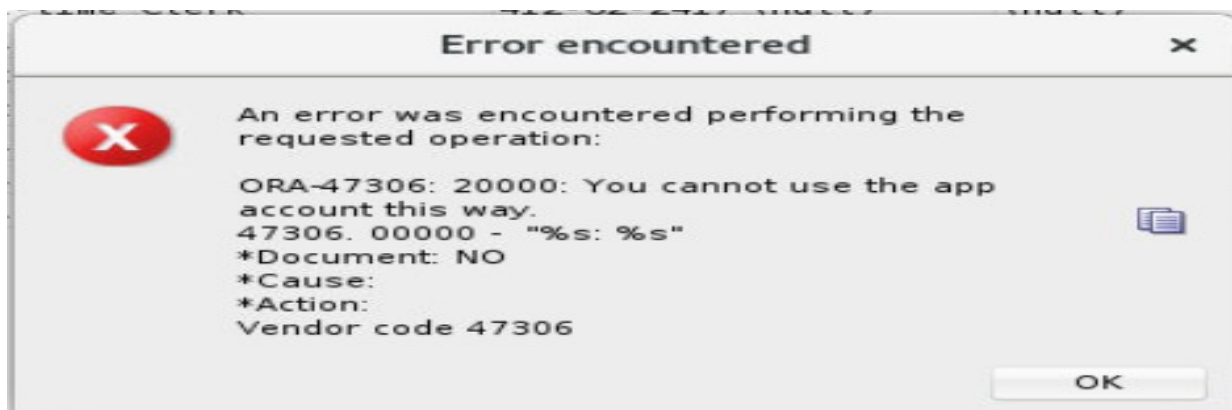
Slika 23: Dodavanje pravila u skup pravila

3. Kreiramo *CONNECT* komandno pravilo, kojim će svaka konekcija koja ne zadovoljava pravilo kreirano u koraku 1. biti odbijena, bez obzira na privilegije korisnika:



Slika 24: Kreiranje komandnog pravila

4. Ukoliko probamo da se prijavimo na sistem izvan aplikacije na adresi *dbsec-lab* (što je definisano pravilom) dobijamo poruku kao na slici 25:



Slika 25: Neuspela prijava van aplikacije

Database Vault, osim pomenutih, poseduje veliki skup drugih mogućnosti. Još jedna od interesantnijih, koju ćemo ovde pomenuti jeste korišćenje **simulacionog moda**. U ovom modu DV prati operacije shodno definisanom komandnom pravilu, međutim usled njegovog narušavanja se neće sprečiti pristup, već će se samo evidentirati. Simulacioni mod se koristi tako što se parametru *enabled* u proceduri *CREATE\_COMMAND\_RULE* dodeli vrednost *DVMS\_MACUTL.G\_SIMULATION*.

## 2.4 – Oracle Label Security

Oracle **labele** predstavljaju Oracle-ovu implementaciju obavezne kontrole pristupa. Dostupne su samo za enterprise verziju baze podataka. Glavna ideja OLS-a (Oracle Label Security) je dodeljivanje labela individualnim vrstama i korisnicima. Prilikom pristupa nekog korisnika vrstama labelirane tabele, autorizacija se obavlja upoređivanjem njegove labele sa vrednostima labele svake vrste. U slučaju čitanja, kao rezultat će biti prikazani filtrirane vrste tabele. Ovi koncepti su već objašnjeni u poglavlju 1.2.4, tako da će fokus ovog poglavlja biti na nekim specifičnostima koje Oracle uvodi u svojoj implementaciji obavezne kontrole pristupa. Takođe će ove pojedinosti biti demonstrirane primerom<sup>4</sup>.

Svaka labela u OLS-u se sastoji od sledećih komponenti:

- **Nivo (Level)** – Odnosi se na sigurnosnu klasu definisanu u poglavlju 1.2.4. Predstavlja se celim brojem, gde veći broj označava veću poverljivost podataka. Svaka labela mora da ima naveden nivo.
- **Odeljak (Compartment)** – Svaka labela može imati pridružen odeljak kojem pripada vrsta uz koju ona stoji. Na primer, podaci mogu imati tri sigurnosna nivoa (javni, tajni, strogo tajni), a da pritom pripadaju različitim timovima u kompaniji (IT, HR, tim za finansije). Na taj način je moguće ostvariti dvonivovsku podelu.
- **Grupa (Group)** – Grupom se definiše organizacija kojoj podaci pripadaju i koja sme da im pristupi. Na primer, ukoliko imamo jedinstveni server na kojem se čuvaju podaci organizacije, koji su prikupljeni sa različitih teritorija (EU, Severna Amerika, Azija) i moraju biti tako podeljeni, moguće je pridružiti im oznaku grupe, tako da podaci koji pripadaju jednom regionu ne mogu biti dostupni korisnicima iz drugih regiona. Grupe mogu biti hijerarhijski organizovane.

Svaka labela mora imati naveden nivo, dok su odeljak i grupa opcioni. Procesom pokretanja OLS-a, automatski se kreira specijalni korisnik *LBACSYS*, koji ima *LBAC\_DBA* ulogu, koja je potrebna za konfigurisanje labela u OLS-u. Kao što je već pomenuto, labele se dodeljuju korisnicima i podacima:

- **Labele podataka** – oslikavaju poverljivost podataka u sistemu, kao i dodatne kriterijume koje korisnik mora da ispuni da bi im pristupio (da bude deo istog odeljka ili grupe).
- **Labele korisnika** – za razliku od labela podataka, koje za nivo moraju imati tačno određenu vrednost, za korisnike možemo definisati opseg nivoa za koje ima mogućnost pristupa.

Konačno, kreirana labela ima sledeću formu:

*LEVEL [: COMPARTMENT<sub>1</sub> , ... , COMPARTMENT<sub>n</sub> : GROUP<sub>1</sub> , ... , GROUP<sub>m</sub>]*

---

<sup>4</sup> Ovaj primer se, kao i većina primera iz ovog poglavlja, zasniva na primerima Oracle LiveLabs radionica. Zbog specifičnosti problema i obima primera, u ovom radu će za svaki korak u konfiguraciji OLS-a biti data samo po jedna naredba za potrebe ilustracije.

U nastavku je demonstriran primer upotrebe OLS-a:

1. Inicijalno je potrebno omogućiti kreiranje OLS-a, sledećim komandama:

```
EXEC LBACSYS.CONFIGURE_OLS;  
EXEC LBACSYS.OLS_ENFORCEMENT_ENABLE_OLS;
```

2. Kreiramo OLS procedurom **CREATE\_POLICY** iz **SA\_SYSDBA** PL/SQL paketa:

```
BEGIN  
LBACSYS.SA_SYSDBA.CREATE_POLICY(  
    policy_name => 'OLS_DEMO_HR_APP',  
    column_name => 'OLSLABEL',  
    default_options => 'READ_CONTROL, WRITE_CONTROL, LABEL_DEFAULT,  
    HIDE');  
END;
```

Parametrima *column\_name* definiše se kako će se zvati kolona koja će sadržati labelu svake vrste, dok parametar *default\_options* definiše za koje operacije će se vršiti autorizacija labela, da li će labela biti vidljiva ili ne, i slično.

3. Nakon kreiranja policy-ja potrebno je definisati sigurnosne nivoe, procedurom **CREATE\_LEVEL** koja je deo **SA\_COMPONENTS** paketa:

```
BEGIN  
LBACSYS.SA_COMPONENTS.CREATE_LEVEL(  
    policy_name => 'OLS_DEMO_HR_APP',  
    level_num => 1000,  
    short_name => 'P',  
    long_name => 'Public');  
END;
```

Ovde je naveden jedan primer, u sistemu su kreirana 3 nivoa: *Public (P)*, *Confidential (C)* i *Highly Confidential (HC)*.

4. Podaci u sistemu su vezani za neki od konkretnih odeljaka u organizaciji. Postoje 4 odeljka: *HR*, *FINANCE (FIN)*, *INTELLECTUAL\_PROPERTY (IP)* i *INFORMATION\_Tech (IT)*. Za svaki od njih kreiramo OLS odeljak:

```
BEGIN  
LBACSYS.SA_COMPONENTS.CREATE_COMPARTMENT(  
    policy_name => 'OLS_DEMO_HR_APP',  
    comp_num => 400,  
    short_name => 'FIN',  
    long_name => 'FINANCE');  
END;
```



Parametar *comp\_num* ne utiče na oslikava nivo sigurnosti kao što je to slučaj sa parametrom *level\_num* kod nivoa, već samo utiče na redosled ispisivanja odeljaka u labeli.

- Podaci u našem simuliranom sistemu su vezani za određena geografska područja. To oslikavamo kreiranjem grupa, koje su hijerarhijski organizovane. Postoji grupa *GLOBAL*, koja obuhvata podgrupe *CANADA (CAN)*, *USA*, *LATIN\_AMERICA (LATAM)* i *EUROPE (EU)*, gde *EU* dodatno obuhvata grupu *GERMANY (GER)*. Primer kreiranja grupe:

**BEGIN**

```
LBACSYS.SA_COMPONENTS.CREATE_GROUP(  
  policy_name => 'OLS_DEMO_HR_APP',  
  group_num => 1100,  
  short_name => 'USA',  
  long_name => 'USA',  
  parent_name => 'GLOBAL');
```

**END;**

Parametar *parent\_name* označava grupu kojoj podgrupa pripada. Za grupu *GLOBAL*, koja je početna grupa koja sadrži sve ostale, ovaj parametar je *null*. Parametar *group\_num* je ekvivalentan parametru *comp\_num* kod definisanja odeljaka.

- Koristimo kreirane komponente da bismo definisali labelu koje će postojati u našem sistemu:

**BEGIN**

```
LBACSYS.SA_COMPONENTS.CREATE_GROUP(  
  policy_name => 'OLS_DEMO_HR_APP',  
  label_tag => 1100,  
  label_value => 'P:USA',  
  data_label => TRUE);
```

**END;**

- Podacima u sistemu su dodeljene labelu nasumično: neke od kreiranih labelu imaju definisan samo sigurnosni nivo, dok većina ima pridružen odeljak i grupu. Za potrebe demonstracije su kreirana dva korisnika *can\_candy* kojoj je dodeljena labela *P:CAN* i *eu\_evan* sa labelom *P:EU*.

- Nakon što su labelu dodeljene možemo uključiti policy:

**BEGIN**

```
LBACSYS.SA_POLICY_ADMIN.APPLY_TABLE_POLICY(  
  policy_name => 'OLS_DEMO_HR_APP',  
  schema_name => 'EMPLOYEESEARCH_PROD',  
  table_name => 'DEMO_HR_EMPLOYEES');
```

**END;**

Nakon što je policy uključen dodavanjem novih podataka je potrebno definisati i njihove labelu, bilo eksplicitno (npr. korišćenjem funkcije *to\_data\_label*) ili implicitno – labela dodatog podatka biće ista kao i labela korisnika koji je dodao podatak.

9. Na sledeće dve slike su prikazani rezultati pretraživanja labelirane tabele *DEMO\_HR\_EMPLOYEES*:

<pre>OLS_READ_LABEL ----- P::CAN  COUNT(*) ----- 162  ... How many cities can be seen by app user CAN_CANDY  CITY                COUNT_CITY ----- Toronto                162</pre>	<pre>OLS_READ_LABEL ----- P::EU,GER  COUNT(*) ----- 368  ... How many cities can be seen by app user EU_EVAN  CITY                COUNT_CITY ----- Berlin                125 London                122 Paris                121</pre>
--	---

Slika 26: Rezultati pribavljanja svih zaposlenih

Na levom delu slike možemo da vidimo da korisnik *can\_candy* može da vidi samo zaposlene iz Toronta – shodno njenoj labeli P:CAN. Sa desne strane korisnik *eu\_evan* može da vidi sve korisnike koji pripadaju grupama EU, i GER (jer je GER podgrupa grupe EU). U zavisnosti od labele, razlikuju se i podaci koje korisnici vide!

## 2.5 – Sakrivanje poverljivih podataka

Jedna od tehnika koje Oracle koristi za sakrivanje poverljivih podataka je **transparentno prikrivanje osetljivih podataka** (eng. *Transparent Sensitive Data Protection – TSDP*). Ova tehnika podrazumeva identifikaciju kolona osetljivih podataka i kreiranje pravila kojima se definiše na koji način će se sakriti informacije koje ovi podaci nose. U tu svrhu je moguće koristiti tehnike Oracle **radakcije podataka**. Postoje različiti načini sakrivanja podataka korišćenjem redakcije:

- **Potpuna redakcija** – Sav sadržaj kolona se sakriva. Način sakrivanja zavisi od tipa podataka kolone: kolone sa tipa NUMBER se menjaju nulom, dok se kolone tipa VARCHAR menjaju praznim stringom.
- **Parcijalna redakcija** – Prikriva se samo deo vrednosti. Na primer, broj kartice je moguće prekriti tako što će deo cifara biti zamenjen nekim specijalnim karakterom.
- **Redakcija korišćenjem regularnih izraza** – Koristi se za sakrivanje kolona koje imaju definisanu strukturu, ali mogu biti različite dužine, kao što je npr slučaj sa email-ovima.
- **Nasumična redakcija** – Podaci se prikrivaju tako što se menjaju nekom nasumično generisanom vrednošću.
- **Null redakcija** – Sve vrednosti u koloni koja je označena kao „osetljiva“ se menjaju vrednošću null.

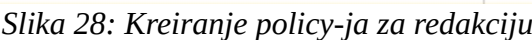
Glavna prednost transparentnog prikrivanja osetljivih podataka korišćenjem redakcije podataka je jednostavnost korišćenja. Prikrivanje vrednosti podataka se odvija u toku izvršenja i potpuno je



1. Prvenstveno je potrebno kreirati osetljivi tip podataka, korišćenjem ***DBMS\_TSDP\_MANAGE.ADD\_SENSITIVE\_TYPE*** PL/SQL procedure. Nakon toga se novokreirani tip dodeljuje nekoj konkretnoj koloni (ili kolonama) koja postoji u sistemu. Za to se koristi procedura ***DBMS\_TSDP\_MANAGE.ADD\_SENSITIVE\_COLUMN***. Ovi koraci su prikazani na sledećoj slici:



2. Kreira se policy kojim se definiše na koji način će se izvršiti parcijalna redakcija. Za kreiranje policy-ja se koristi ***DBMS\_TSDP\_PROTECT.ADD\_POLICY*** procedura:



Na slici možemo uočiti nekoliko parametara. Parametri koji su definisani za redakciju su redom: username korisnika od koga će podaci biti skriveni – u ovom primeru je to TSDP\_LABS, zatim tip redakcije – parcijalna redakcija i način sakrivanja vrednosti – kako se radi o broju kreditne kartice, ovako definisanom funkcijom će biti sakriveno prvih 8 cifara (što potvrđuju parametri \*, 1 i 8).

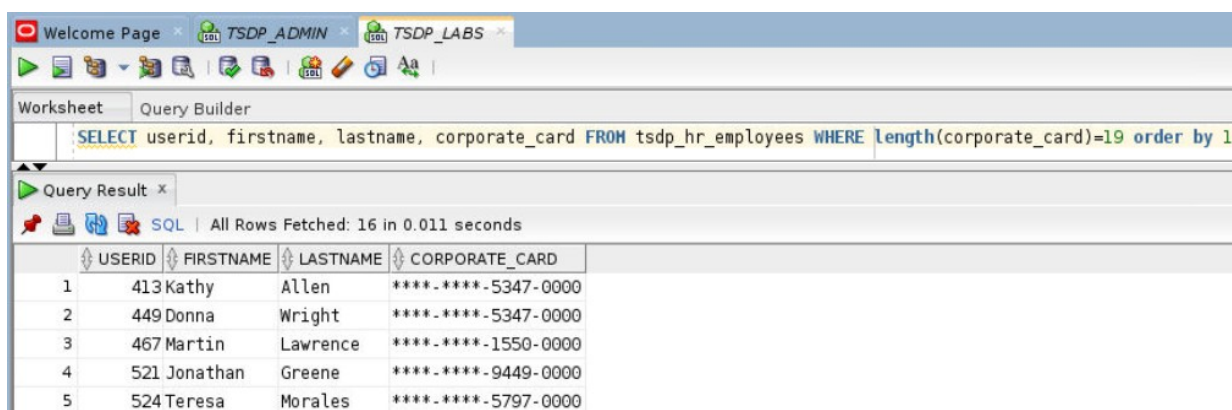
3. Nakon što kreiramo policy potrebno ga je povezati sa osetljivim tipom koji smo kreirali u prvom koraku, korišćenjem procedure **ASSOCIATE\_POLICY**, a zatim ga aktivirati korišćenjem procedure **ENABLE\_PROTECTION\_TYPE**. Obe procedure su deo **DBMS\_TSDP\_PROTECT** PL/SQL paketa.

```
-- primenjujemo kreirani policy na prethodno definisani "osetljivi" tip
BEGIN
  DBMS_TSDP_PROTECT.ASSOCIATE_POLICY(
    policy_name => 'redact_partial_cc',
    sensitive_type => 'credit_card_type',
    associate => true);
END;
/

-- aktiviramo policy
BEGIN
  DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE(
    sensitive_type => 'credit_card_type');
END;
```

Slika 29: Dodeljivanje i aktiviranje policy-ja

4. Na sledećoj slici možemo da vidimo rezultat selekcije kada je policy aktivan:



The screenshot shows the SQL Developer interface with a query executed in the TSDP\_LABS schema. The query selects user details and corporate card numbers, where the card numbers are redacted with asterisks. The results are displayed in a table with 5 rows.

	USERID	FIRSTNAME	LASTNAME	CORPORATE_CARD
1	413	Kathy	Allen	****.****-5347-0000
2	449	Donna	Wright	****.****-5347-0000
3	467	Martin	Lawrence	****.****-1550-0000
4	521	Jonathan	Greene	****.****-9449-0000
5	524	Teresa	Morales	****.****-5797-0000

Slika 30: Sakrivene vrednosti za kolonu CORPORATE\_CARD

## 2.6 – DBSAT – Security Assessment Tool

U prethodnim poglavljima su navedene neke od mnogobrojnih tehnika kojima se Oracle baza podataka služi, u cilju bolje zaštite sistema. U ovom poglavlju pažnja je posvećena dodatnom alatu, koji se danas vrlo često koristi kao prateći deo sistema koji umnogome olakšava posao održavanja sigurnosti na maksimalnom mogućem nivou, Oracle Database Security Assessment Tool – DBSAT.

DBSAT se koristi za brzu i efikasnu procenu trenutne sigurnosti sistema, može lako da otkrije koji su osetljivi podaci u sistemu koje je potrebno zaštititi i promoviše dobre sigurnosne prakse, koje bi trebalo primeniti. Sastoji se iz 3 komponente:

- **Collector** – Izvršava SQL upite i komande operativnog sistema, kako bi prikupio neophodne informacije o trenutnom stanju sistema. Rezultat kolekcije informacija je *json* fajl koji reporter kasnije koristi kako bi izvršio analizu.
- **Reporter** – Analizira prikupljene podatke i generiše izveštaj u Excel, HTML, json i tekstualnom formatu.
- **Discoverer** – Koristi se za analizu i otkrivanje osetljivih podataka u sistemu.

Korišćenjem komande *dbsat collect* se pokreće operacija prikupljanja podataka u sistemu. Fajl koji dobijemo kao rezultat ove operacije koristimo kao ulazni parametar komande *dbsat report* kojom se generišu rezultati analize, koji su predstavljeni na slici 31:

## Oracle Database Security Assessment

Highly Sensitive

### Assessment Date & Time

Date of Data Collection	Date of Report	Reporter Version
Thu May 23 2024 14:35:16 UTC+00:00	Thu May 23 2024 14:39:07 UTC+00:00	3.1 (Jan 2024) - b73a

### Database Identity

Name	Container (Type:ID)	Platform	Database Role	Log Mode	Created
CDB1	PDB1 (PDB:3)	Linux x86 64-bit	PRIMARY	NOARCHIVELOG	Wed Oct 30 2019 15:41:51 UTC+00:00

### Summary

Section	Pass	Evaluate	Advisory	Low Risk	Medium Risk	High Risk	Total Findings
<a href="#">Basic Information</a>	1	0	0	0	0	0	1
<a href="#">User Accounts</a>	6	10	1	5	2	0	24
<a href="#">Privileges and Roles</a>	4	25	1	0	0	0	30
<a href="#">Authorization Control</a>	0	3	2	0	0	0	5
<a href="#">Fine-Grained Access Control</a>	0	0	5	0	0	0	5
<a href="#">Auditing</a>	6	8	2	0	0	0	16
<a href="#">Encryption</a>	0	4	0	0	0	0	4
<a href="#">Database Configuration</a>	8	8	0	1	3	1	21
<a href="#">Network Configuration</a>	1	0	3	1	0	0	5
<a href="#">Operating System</a>	2	4	0	1	2	0	9
<b>Total</b>	<b>28</b>	<b>62</b>	<b>14</b>	<b>8</b>	<b>7</b>	<b>1</b>	<b>120</b>

Slika 31: DBSAT - Rezultat analize

Na slici 31 možemo da vidimo da je reporter izvukao 120 zaključaka, od kojih je 28 potvrđeno kao pozitivno, dok su ostali raspoređeni u 5 klasa: **proceniti**, **preporuka**, **nizak rizik**, **srednji rizik**, **visoki rizik**. Sva ova otkrića su razvrstana po domenima: nalazi vezani za korisničke naloge, privilegije, kontrolu pristupa, enkripciju, itd. Na sledećoj slici je prikazano kako izgleda kompletan nalaz koji je reporter pronašao:

## Sample Schemas

USER.SAMPLE		CIS	OBP	STIG
Sample schemas should be dropped				
<b>Status</b>	Low Risk			
<b>Summary</b>	Found 2 sample schemas.			
<b>Details</b>	Sample schemas: HR, SCOTT			
<b>Remarks</b>	Sample schemas are well-known accounts provided by Oracle to serve as simple examples for developers. They generally serve no purpose in a production database and should be removed because they unnecessarily increase the attack surface of the database.			
<b>References</b>	Oracle Best Practice CIS Benchmark: Recommendation 4.2 DISA STIG: V-220284			

Slika 32: DBSAT - Primer nalaza

Za svaki nalaz, reporter generiše status, kratak opis problema, detalje i razloge zbog kojih je evidentiran. Na slici je prikazan samo jedan primer koji zaista ne oslikava svu moć ovog alata.

Sledeća komponenta DBSAT-a je discoverer, koji može da otkrije koje su to potencijalno osetljive kolone u sistemu koje je potrebno zaštititi. Discoverer se podešava pomoću konfiguracionog fajla. Dodatno se specificira i .ini konfiguracioni fajl koji sadrži regex izraze za detekciju osetljivih tipova podataka na različitim jezicima, na osnovu kojih discoverer prepoznaje odgovarajuće kolone. Primer izlaza discoverera je dat na slikama 33 i 34.

## Summary

Sensitive Category	# Sensitive Tables	# Sensitive Columns	# Sensitive Rows
BIOGRAPHIC INFO - ADDRESS	9	36	6307209
BIOGRAPHIC INFO - EXTENDED PII	2	2	2000
FINANCIAL INFO - BANK DATA	2	2	599
FINANCIAL INFO - CARD DATA	7	7	3004
HEALTH INFO - PROVIDER DATA	1	1	149
IDENTIFICATION INFO - NATIONAL IDS	2	6	2000
IDENTIFICATION INFO - PERSONAL IDS	3	3	405
IDENTIFICATION INFO - PUBLIC IDS	9	26	2401125
IT INFO - USER DATA	13	15	12997
JOB INFO - COMPENSATION DATA	10	12	3149
JOB INFO - EMPLOYEE DATA	8	16	406
JOB INFO - ORG DATA	5	6	278
TOTAL	29*	132	8617413**

Slika 33: DBSAT - Discoverer analiza

## **Risk Level: High Risk**

### **Security for Environments with High Value Data: Detective plus Strong Preventive Controls**

Highly sensitive and regulated data should be protected from privileged users, and from users without a business need for the data. Activity of privileged accounts should be controlled to protect against insider threats, stolen credentials, and human error. Who can access the database and what can be executed should be controlled by establishing a trusted path and applying command rules. Sensitive data should be redacted on application read only screens. A Database Firewall ensures that only approved SQL statements or access by trusted users reaches the database - blocking unknown SQL injection attacks and the use of stolen login credentials.

Recommended controls include:

- Audit all sensitive operations including privileged user activities
- Audit access to application data that bypasses the application
- Encrypt data to prevent out-of-band access
- Mask sensitive data for test and development environments
- Restrict database administrators from accessing highly sensitive data
- Block the use of application login credentials from outside of the application
- Monitor database activity for anomalies
- Detect and prevent SQL Injection attacks
- Evaluate: Oracle Audit Vault and Database Firewall, Oracle Advanced Security, Oracle Data Masking and Subsetting, Oracle Database Vault

### **Tables Detected within Sensitive Category: BIOGRAPHIC INFO - ADDRESS**

<b>Risk Level</b>	High Risk
<b>Summary</b>	Found BIOGRAPHIC INFO - ADDRESS within 36 Column(s) in 9 Table(s)
<b>Location</b>	Tables: DMS_ADMIN.MASK DATA, EMPLOYEESEARCH DEV.DEMO HR EMPLOYEES, EMPLOYEESEARCH PROD.DEMO HR EMPLOYEES, HCM1.COUNTRIES, HCM1.LOCATIONS, HR.COUNTRIES, HR.LOCATIONS, LOOKUPS.LOOKUP_ADDRESSES, LOOKUPS.LOOKUP_PLACES

*Slika 34: DBSAT - Discoverer analiza*

DBSAT predstavlja jako moćan alat koji značajno olakšava posao security adminima. Značaj ovakvih alata je očigledna, naročito u jako kompleksnim sistemima sa velikim brojem korisnika. Osim ovog alata koriste se i neka druga rešenja, kao što su Oracle DataSafe, koji prevazilazi domen ovog seminarskog rada.



## ZAKLJUČAK

U ovom seminarskom radu prikazani su neki osnovni teorijski koncepti kojima se obezbeđuje sigurnost u sistemima za upravljanje bazama podataka i demonstrirana su neka od mnogih rešenja koja Oracle koristi u ove svrhe. Glavna ideja je bila naglašavanje značaja ove jako kompleksne oblasti, naročito danas kada organizacije postaju jako svesne vrednosti koju podaci imaju (o čemu, između ostalog svedoči i razvoj mašinskog učenja veštačke inteligencije danas). Toga su nažalost svesni i hakeri, pa su upravo baze podataka sve češća meta njihovih napada. Da bi jedan takav sistem odoleo ovim napadima, potrebno je konstantna evolucija i poboljšanje postojećih sigurnosnih rešenja. Međutim, potrebno je naglasiti i ljudski faktor u ovim sistemima – svaki sistem je siguran onoliko koliko i njegov najnesigurniji deo. Zbog toga je potrebno konstantno ulagati napor u edukaciju o dobrim sigurnosnim praksama i pravilno koristiti dostupne resurse sistema.

# LITERATURA

- [1] Oracle Security Primer - <https://download.oracle.com/database/oracle-database-security-primer.pdf>
- [2] Fundamentals of Database Systems 7th edition – Navathe, Elmasri
- [3] Oracle Security Guide - <https://docs.oracle.com/en/database/oracle/oracle-database/19/dbseg/>
- [4] Oracle LiveLabs: DB Security Basics - <https://apexapps.oracle.com/pls/apex/r/dbpm/livelabs/view-workshop?wid=698&clear=RR,180&session=1950409946582>
- [5] Oracle LiveLabs: DB Security Advanced - <https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?wid=726>
- [6] Oracle Database Security Guide - <https://docs.oracle.com/en/database/oracle/oracle-database/21/dbseg/database-security-guide.pdf>