

Sicurezza in Oracle

Autore: Barletta Massimiliano

Mestre, 15 luglio 2004

Indice

1	Introduzione	1
1.1	Obiettivi	1
1.2	Sicurezza	2
1.3	Privacy	2
1.4	Livelli di protezione	3
1.4.1	Autenticazione	3
1.4.2	Autorizzazione	3
1.4.3	Auditing	3
1.5	Principi di sicurezza	4
1.5.1	Segretezza e Confidenzialità	4
1.5.2	Integrità ed Autenticità	4
1.5.3	Accessibilità	5
1.6	Organizzazione della tesi	5
2	Componenti di un sistema Oracle	6
2.1	Sicurezza in un sistema Oracle	6
2.2	Tablespace	6
2.3	Oggetti	7
2.3.1	Tabelle	7
2.3.2	Trigger	8
2.3.3	Viste	8
2.3.4	Programmi memorizzati	9
2.3.5	Utenti e Ruoli	10
2.3.6	Profili	11
2.3.7	Sinonimi	12

2.3.8	Privilegi	14
2.4	Meccanismi di autenticazione	15
2.4.1	Password	16
2.5	Data Dictionary	18
2.6	Oracle e rete	22
2.6.1	Configurare il listener net8: accettare o rifiutare richieste da specifici indirizzi IP	25
3	Auditing	26
3.1	Auditing	26
3.2	Perché auditing	27
3.3	Abilitare l'auditing	28
3.4	Risultato dell'auditing	30
3.5	Opzioni di auditing	32
3.5.1	Statement	32
3.5.2	Privilegi	33
3.5.3	Oggetti	33
3.6	Privilegi per l'auditing	33
3.7	Auditing e Triggers	34
3.8	Esempi di utilizzo	34
3.8.1	Tentativi di autenticazione falliti	34
3.8.2	Tentativi di accesso al database con utenti che non esistono	35
3.8.3	Tentativi di accesso ad ore non usuali	36
4	Backup	37
4.1	Frequenza di backup	37
4.2	Archivelog	38
4.3	Forme di backup	39
4.3.1	Export e Import	39
4.3.2	Backup a freddo	41
4.3.3	Backup a caldo	42
5	Row Level Security	44
5.1	Cosa significa	44

5.2	Esempio	45
5.2.1	Assunzioni	46
5.2.2	Context security	47
5.2.3	Dynamic Access Predicates	48
5.2.4	DBMS_RLS package	49
5.2.5	Test	50
6	Security Check-List	52
6.1	Bloccare gli utenti di default	52
6.2	Cambiare le password degli utenti di default	53
6.3	Controllo delle password	53
6.4	Proteggere il data dictionary	53
6.5	Privilegi e ruoli	53
6.6	Restringere l'accesso alla rete	54
6.7	Auditing	54
6.8	Backup	55
6.9	Applicare patch di sicurezza	55
7	Conclusioni	56
A	DBMS_RLS package	58
B	DBMS_SESSION package	62
	Bibliografia	63

Elenco delle figure

2.1	Sinonimi pubblici vs Sinonimi privati	13
2.2	Connessione client/server	23
2.3	Listener - richiesta di connessione	24
4.1	Esportare un database	40

Elenco delle tabelle

2.1	Viste di particolare importanza	19
2.2	Dynamic Performances Tables	22
3.1	Viste di auditing	32
A.1	DBMS_RLS programs	58
B.1	set_context parameters	62

Capitolo 1

Introduzione

1.1 Obiettivi

I dati e le informazioni sono indubbiamente il bene più prezioso per ciascuna azienda pertanto la loro salvaguardia è di vitale importanza. La maggior parte dei sistemi informativi odierni utilizza una base dati o database¹ per la registrazione ed elaborazione di queste informazioni.

I moderni database costituiscono pertanto l'elemento portante di ogni sistema di business ed altre attività di natura differente. Molti dei dati in essi conservati sono di natura confidenziale (numeri di carte di credito, dati finanziari, ecc) e/o critici per il corretto funzionamento dell'azienda stessa. Per questi motivi è necessario provvedere ad analizzare potenziali rischi e predisporre opportune misure di sicurezza.

L'obiettivo principale di questa tesi è quello di porre l'accento sulle possibili "vulnerabilità" del database Oracle e le misure di sicurezza adottate per evitare che un malintenzionato prenda il pieno controllo del sistema. Ho scelto Oracle² come caso studio in quanto a livello aziendale appare essere la base dati maggiormente utilizzata.

Naturalmente a causa della vastità dell'argomento considero questa tesi un punto di partenza per poter successivamente approfondire con maggior dettaglio le tematiche trattate. Dopo aver letto questa tesi mi auguro

¹Insieme di dati strutturati in record e campi

²Produttore di DBMS relazionali, <http://www.oracle.com>

che il lettore possa avere le idee più chiare e possa così comprendere quali siano i rischi di una non corretta amministrazione di un sistema Oracle e quali possano essere i danni per una azienda che gestisce molti dati sensibili e/o strategici.

1.2 Sicurezza

Si sente spesso parlare di “sicurezza” ma come altre parole della lingua italiana questa può assumere un diverso significato a seconda del contesto in cui viene utilizzata. Possiamo infatti parlare di sicurezza personale, sicurezza di edifici, sicurezza di un conto in banca. Nel nostro caso specifico posso associare alla parola “sicurezza” il significato di tutte quelle procedure sia tecniche sia personali che dovrebbero permettere di assicurare che persone non autorizzate non accedano ad informazioni private e che persone autorizzate non mettano a repentaglio il sistema e i dati. Naturalmente è opportuno ricordarsi che la creazione di un ambiente sicuro non si esprime in qualcosa di statico, ma è un’operazione continua nel tempo spesso in funzione di nuove tecnologie disponibili.

1.3 Privacy

L’esigenza della sicurezza dei sistemi informatici e la protezione di dati ed informazioni necessarie ed indispensabili risulta essere sempre più rilevante. Nonostante questo non possiamo risolvere la sicurezza in termini di oggetti quali un firewall né in termini di decreti. L’art. 15 della legge 31.12.1996, n. 675 emanata a tutela della riservatezza dei dati personali ribadisce le necessità di responsabilizzare soprattutto chi è preposto alla loro protezione. Al comma 1 infatti, dispone che *“i dati personali oggetto di trattamento devono essere custoditi e controllati, anche in relazione alle conoscenze acquisite in base al progresso tecnico, alla natura dei dati e alle specifiche caratteristiche del trattamento, in modo da ridurre al minimo, mediante l’adozione di idonee e preventive misure di sicurezza, i rischi di distruzione o perdita, anche accidentale, dei dati stessi, di accesso non autorizzato o di trattamento non consentito o non conforme*

alle finalità della raccolta".

Le norme possono essere un utile mezzo di conoscenza per poter prendere decisioni in via preventiva ma certamente non sono il mezzo definitivo per raggiungere la sicurezza assoluta.

1.4 Livelli di protezione

La sicurezza dei database assume almeno questi tre livelli di protezione: autenticazione, autorizzazione, auditing.

1.4.1 Autenticazione

L'autenticazione è quel processo che identifica correttamente gli utenti e gli permette di accedere al sistema. Normalmente un normale processo di identificazione potrebbe consistere nella verifica di un nome utente e l'autenticazione nella verifica della validità della password associata. L'autenticazione è un elemento necessario per valutare in quale circostanza qualcuno fa qualcosa.

Esistono comunque altri sistemi di autenticazione quali smart-card, certificati X.509³ o elementi biometrici (impronta digitale).

1.4.2 Autorizzazione

Una volta riconosciuto dal sistema, l'utente ha accesso ai propri privilegi ed autorizzazioni. Vengono decise pertanto quali siano le risorse a cui l'utente può aver accesso e quali siano le modalità operative.

1.4.3 Auditing

Questa fase è finalizzata, mediante alcune funzionalità idonee, ad identificare e riconoscere potenziali abusi oltre ad assicurare l'integrità delle informazioni. L'auditing può essere un ottimo strumento per il controllo del database ma non dovrebbe limitarsi a rilevare le sole informazioni

³un certificato a chiave pubblica secondo il formato X.509

sull'accesso ai dati degli utenti ma dovrebbe anche tracciare la modalità d'accesso per valutare le possibili intrusioni dell'utente.

Le funzionalità di auditing hanno un impatto negativo sulle performance dell'intero sistema nonostante ciò è opportuno e consigliabile ricorrere alle stesse cercando un giusto compromesso tra le prestazioni e le quantità di dati raccolti mediante il monitoraggio.

1.5 Principi di sicurezza

L'idea maggiormente diffusa di sicurezza riguarda la segretezza delle informazioni contenute in un database. Nonostante si tratti di un aspetto molto importante non è comunque l'unico. Non è infatti possibile prescindere dalla considerazione di altri aspetti quali integrità, autenticità ed accessibilità.

1.5.1 Segretezza e Confidenzialità

I dati devono poter essere consultati e/o modificati soltanto da parte di chi sia debitamente autorizzato (privacy).

I controlli degli accessi devono garantire che i dati non possano essere rivelati accidentalmente o senza autorizzazione. La crittazione è un processo piuttosto standard normalmente adottato per garantire un elevato grado di confidenzialità.

È fondamentale che la crittazione sia applicata sia nel momento in cui i dati entrano sia nel momento in cui i dati escono dal database. Le intrusioni in rete sono spesso la minaccia esterna più facile e ricorrente, per cui la crittazione del traffico di rete non può essere considerata soltanto un'opzione ma deve essere o diventare una necessità.

1.5.2 Integrità ed Autenticità

I dati non devono poter essere manipolati dolosamente od accidentalmente ed inoltre la loro provenienza deve essere verificabile. L'integrità dei dati può essere fornita, ad esempio, attraverso un data-checksum, una verifica tecnologica del fatto che il dato sia ancora quello che era

stato originariamente imputato o salvato. L'integrità può essere implementata sia per i dati in movimento in una rete, sia per quelli conservati nel database utilizzando algoritmi sperimentati quali MD5⁴ o SHA-1⁵. L'integrità può inoltre essere fornita anche attraverso un severo controllo degli accessi e la possibilità di mantenere un ambiente "least-privilege". Con least-privilege s'intende che ciascun utente ha solo i privilegi per svolgere il proprio lavoro.

1.5.3 Accessibilità

I dati devono essere sempre disponibili eventualmente anche attraverso il loro immediato ripristino. Pertanto, anche in funzione delle politiche di sicurezza, è opportuno predisporre alcuni meccanismi di ridondanza, anche su più livelli (hardware e/o software).

1.6 Organizzazione della tesi

Ho pensato di organizzare la tesi in sei capitoli.

Dopo un capitolo introduttivo al tema della sicurezza, ho cercato innanzitutto di descrivere le componenti fondamentali di un sistema Oracle (oggetti, tabelle, viste, triggers, ecc) con lo scopo di favorire una lettura più agevole dei capitoli successivi.

Successivamente ho affrontato le tematiche riguardanti l'auditing (cap. 3), il backup del database (cap.4) ed una caratteristica di sicurezza aggiuntiva tipica di oracle, row level security (cap.5). Come conclusione ho inserito una possibile check-list di sicurezza che può essere consultata come prima fonte post-installazione.

⁴Message Digest - 5

⁵Secure HAsH Algorithm -1

Capitolo 2

Componenti di un sistema Oracle

2.1 Sicurezza in un sistema Oracle

In questo capitolo saranno introdotte alcune componenti essenziali, tra cui file ed oggetti del database, ai fini della sicurezza in Oracle. Sarà inoltre introdotto il concetto di data dictionary da un punto di vista della sicurezza e saranno definiti alcuni concetti indispensabili per implementare la sicurezza del proprio sistema: viste, privilegi, ruoli e account utenti, profili, sinonimi, password.

Come detto in precedenza questi concetti saranno indispensabili per comprendere le tematiche successive.

Per poter implementare un qualsiasi piano di sicurezza è essenziale innanzitutto conoscere le componenti di un sistema e in alcune circostanze pensare come un possibile intrusore.

2.2 Tablespace

Oracle utilizza i file nel suo schema organizzativo, ma nonostante ciò la sua struttura va oltre lo stesso concetto di file. Si definisce tablespace un'area del disco costituita da uno o più file che può contenere al suo

interno molte tabelle, indici o cluster. Un tablespace ha una dimensione prefissata quindi nel momento in cui si riempie (spazio insufficiente) può essere ampliato da colui o coloro che hanno la giusta autorità. L'espansione viene realizzata creando un nuovo file sul disco e aggiungendolo al tablespace oppure estendendo i file di dati (datafile) esistenti. Uno o più tablespace insieme costituiscono un vero e proprio database.

2.3 Oggetti

Nel contesto Oracle è possibile interpretare il termine “oggetto” nel senso generico di entità creata da un utente ed include tabelle, sinonimi, viste, indici, procedure, trigger e via discorrendo.

Seguirà una descrizione di alcune componenti base di un sistema Oracle.

2.3.1 Tabelle

Una tabella è la componente base essenziale per la memorizzazione delle informazioni all'interno del database. A livello concettuale è possibile immaginarla come un file creato e mantenuto all'interno dei file di dati assegnati al database.

Nel momento in cui un utente ha la necessità di creare una tabella è indispensabile che specifichi alcuni parametri quali il nome della tabella, il nome delle colonne, il loro tipo di dato e la loro lunghezza. Nella dichiarazione (create) viene anche normalmente specificato un tablespace affinché la tabella sia creata nell'opportuno tablespace. Se lo tablespace non viene specificato la tabella sarà creata nello tablespace di default dell'utente. Può essere poi specificata un'iniziale quantità di spazio per i dati della tabella utilizzando il parametro initial della clausola storage. Quando lo spazio risulta però essere insufficiente verrà allocato ulteriore spazio facendo riferimento al valore impostato con il parametro next nella clausola storage.

Per maggiore chiarezza riporto di seguito un esempio di codice per la creazione di una tabella.

```
CREATE TABLE dipendenti
```

```
(num_dipendente NUMBER(6) NOT NULL,  
 nome_dipendente VARCHAR(20),  
 nome_manager VARCHAR(20))  
TABLESPACE data01  
STORAGE (  
INITIAL 275K  
NEXT 50K  
MAXEXTENTS UNLIMITED  
PCTINCREASE 0 );
```

2.3.2 Trigger

Un trigger è uno speciale programma memorizzato che il database deve eseguire nel momento in cui si verifica un determinato evento. L'esecuzione del trigger è totalmente trasparente all'utente.

Le azioni vengono intraprese dal database quando specifici comandi di manipolazione dei dati vengono eseguiti su determinate tabelle. Possono essere dei comandi sql insert, update e delete.

Per poter creare un trigger su una tabella è indispensabile essere in grado di modificare la tabella stessa. Questo significa che è necessario possedere la tabella, avere il privilegio alter o il privilegio alter any table.

Infine è necessario disporre del privilegio di sistema create trigger.

Per scopi di sicurezza i trigger possono essere utilizzati per tracciare o prevenire attività che cambiano, alterano e/o modificano i dati.

In alcune circostanze, per questioni di sicurezza oppure per scopi di monitoraggio, le tabelle sono create con dei campi extra per "catturare" username, tempo ed azioni di modifica.

2.3.3 Viste

Oracle permette di conservare una definizione nel dizionario dei dati che descrive come i dati devono essere recuperati da una o più tabelle.

Questa definizione logica è chiamata "vista". Le viste definiscono esclusivamente come i dati devono essere recuperati e le restrizioni per recuperare i dati.

È possibile creare viste utilizzando praticamente qualsiasi restrizione si desideri o qualsiasi calcolo nelle colonne, e poi concedere ad altri utenti l'accesso alla vista, invece che alle tabelle sottostanti. Gli utenti possono così vedere solo le informazioni presentate dalla vista. Le viste presentano i seguenti vantaggi:

1. Forniscono un livello aggiuntivo di sicurezza, permettendo di restringere l'accesso ad alcune colonne o record della tabella;
2. Riducono la complessità di una query;
3. Rendono possibile eseguire interrogazioni che altrimenti sarebbe impossibile effettuare con un'unica istruzione;
4. Garantiscono l'indipendenza logica dei dati;
5. Permettono di visualizzare i dati in modo differente a come sono memorizzati.

2.3.4 Programmi memorizzati

Programmi scritti in PL/SQL¹ possono essere memorizzati in forma compilata all'interno del database. Questi programmi si riferiscono sia a funzioni sia a procedure. Una funzione per definizione restituisce un valore mentre una procedura non ha un valore di ritorno.

A differenza dei trigger, le procedure e le funzioni sono eseguite da una chiamata esplicita (non c'è quindi trasparenza):

```
EXECUTE nome_funzione(param);
```

Dal punto di vista della sicurezza le procedure memorizzate possono essere uno strumento ulteriore di controllo della sicurezza dei dati. È infatti possibile limitare le operazioni che l'utente può eseguire sugli oggetti del database permettendo di accedere ai dati solo per mezzo di procedure e funzioni, eseguite con i privilegi del proprietario (colui che ha creato la

¹Linguaggio proprietario di Oracle per lo sviluppo di applicazioni di database relazionali

procedura o funzione).

È possibile ad esempio concedere ad un utente, l'accesso ad una procedura che aggiorna una tabella ma non alla tabella stessa.

Quando un utente invoca una procedura viene eseguita con i privilegi del proprietario (default). Per fare in modo che una procedura possa essere eseguita con i diritti del chiamante (invoker-rights) è sufficiente aggiungere un parametro opzionale AUTHID CURRENT USER alla sua dichiarazione. In quest'ultimo caso, all'utente che invoca la procedura, devono essere espressamente concessi i privilegi per eseguire select, update, delete, o qualsiasi altra operazione.

2.3.5 Utenti e Ruoli

Ciascun utente in un sistema Oracle è riconosciuto da un nome utente ed una password e possiede le tabelle, le viste e le altre risorse da lui create.

Un ruolo consiste in una serie di privilegi.

È possibile concedere particolari privilegi a dei ruoli e assegnare i ruoli agli utenti appropriati. Un utente può concedere direttamente privilegi ad altri utenti. I privilegi di sistema del database consentono di eseguire particolari serie di comandi. Il comando grant any privilege permette ad esempio la concessione di un qualsiasi privilegio di sistema.

Oracle viene fornito con due utenti già creati, system e sys. Per poterne creare altri occorre autenticarsi come utente system che ha questo privilegio. L'amministratore crea quindi in fase di installazione un utente per se stesso:

```
CREATE USER username Identified {by password|externally}
```

Per connettere username a password del sistema del proprio computer nella sicurezza Oracle in modo che serva una sola identificazione occorre specificare externally invece della password. Per una sicurezza maggiore un amministratore di sistema dovrebbe optare per la password. Quando si decide di cambiare la password oppure è necessario (perchè scaduta) modificarla è possibile utilizzare il comando sql:

```
ALTER USER username identified by nuova_password
```

Pertanto un DBA crea un ruolo e concede a questo alcuni privilegi. Questi privilegi possono essere sia di sistema che di oggetto. Oracle fornisce tre ruoli standard: connect, resource, dba.

Connect viene concesso ad utenti occasionali oppure ad utenti che non hanno la necessità di creare tabelle.

I privilegi di sistema per il ruolo connect sono: alter, create cluster, create database link, create sequence, create session, create synonym, create table, create view.

Resource è un ruolo che concede diritti per creare proprie tabelle, sequenze, procedure, trigger, indici e cluster.

I privilegi di sistema per il ruolo resource sono: create cluster, create procedure, create sequence, create table, create trigger, create type.

Il ruolo DBA (amministratore del database) ha tutti i privilegi di sistema, comprese quote di spazio illimitate e la capacità di concedere tutti i privilegi ad altri utenti. System è creato per essere utilizzato da un utente DBA.

In Oracle al DBA sono concessi i ruoli exp_full_database e imp_full_database che a loro volta possiedono i privilegi per l'esportazione e l'importazione dell'intero database Oracle.

2.3.6 Profili

Un profilo è un meccanismo offerto da Oracle per il controllo dell'allocazione e l'utilizzo delle risorse nel database. Qualora non vengano creati dei profili verranno utilizzati profili predefiniti (default), i quali definiscono delle risorse illimitate per tutti gli utenti. È possibile identificare due tipi di profili disponibili: profili di prodotto e profili di risorse di sistema. I profili di prodotto permettono di bloccare specifici comandi (uno, un gruppo oppure tutti) o prodotti Oracle. Il secondo tipo permette di definire un controllo sull'utilizzo delle risorse da parte degli utenti. Per poter imporre dei limiti a specifici utenti è indispensabile creare un profilo in cui definire le restrizioni e assegnarlo poi agli utenti. Per tutti quei parametri che non vengono specificati nel profilo personalizzato si assumeranno i valori del profilo di DEFAULT. Riporto un esempio per la

creazione di un profilo utente (è necessario avere il privilegio di sistema CREATE PROFILE):

```
CREATE PROFILE std_user LIMIT
SESSIONS_PER_USER          3
CPU_PER_SESSION            UNLIMITED
IDLE_TIME                  30
LOGICAL_READS_PER_SESSION DEFAULT;
```

Come specificato è poi necessario associarlo ad un determinato utente o gruppo di utenti per fare in modo che siano applicate le corrette restrizioni. Nell'esempio che segue osserviamo come associare un profilo utente:

```
CREATE USER          mary
IDENTIFIED BY       abc75!d
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp
PASSWORD            expire
PROFILE             std_user
```

2.3.7 Sinonimi

Un sinonimo è un modo diverso per esprimere lo stesso concetto. I sinonimi li usiamo ogni giorno nella nostra vita quotidiana. In Oracle i sinonimi vengono utilizzati per fornire trasparenza di locazione di un oggetto e del proprietario. L'oggetto in questione potrà essere una tabella, una vista, una procedura memorizzata, e via dicendo. I sinonimi possono essere sia pubblici sia privati. Qualora un utente crei un sinonimo senza la parola chiave public allora sarà privato e potrà essere utilizzato esclusivamente dall'utente che ha creato il sinonimo.

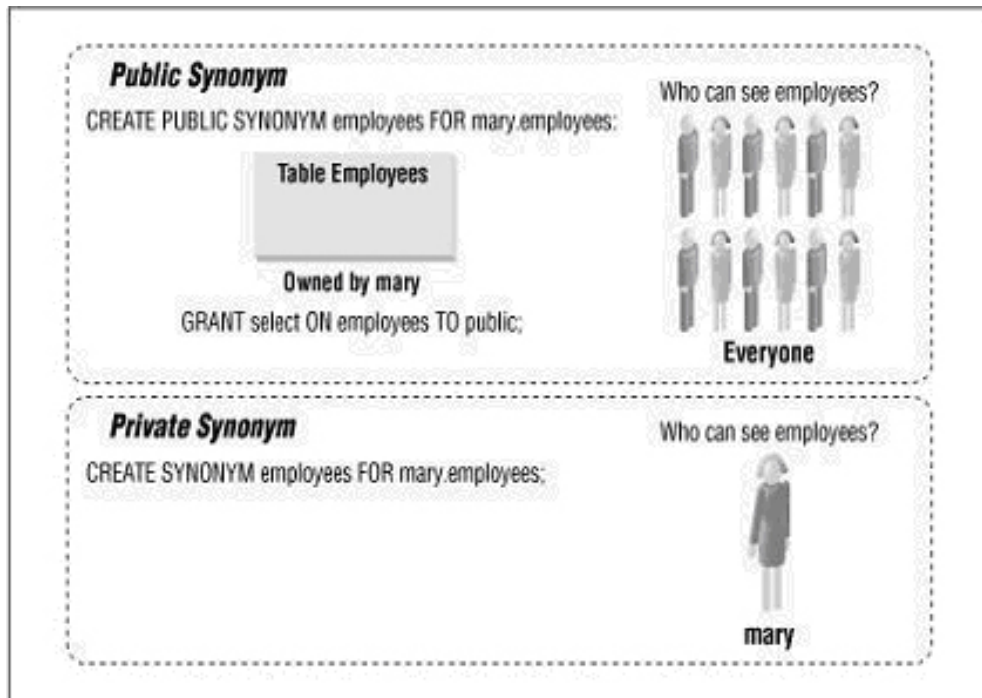


Figura 2.1: Sinonimi pubblici vs Sinonimi privati

La ragione fondamentale per cui si fa uso dei sinonimi è quella di evitare di dover conoscere il proprietario dell'oggetto. Se Scott possiede una tabella dipendenti e Bill vuole selezionare dei dati da essa allora può eseguire un comando del tipo:

```
SELECT * FROM Scott.dipendenti;
```

Se Bill non conosce che Scott ha una tabella, Bill avrà delle difficoltà ad accedere. D'altra parte, Scott può creare un sinonimo pubblico:

```
CREATE PUBLIC SYNONIM dipendenti  
FOR Scott.dipendenti
```

In questo modo si evita di scrivere applicazioni che contengano query in cui sia specificato il proprietario di una tabella. Riprendendo l'esempio precedente, dopo la creazione del sinonimo pubblico, sarà sufficiente scrivere una query del tipo per accedere ai dati della tabella:

```
SELECT * FROM dipendenti;
```

Si conclude che utilizzare sinonimi risulta essere di enorme vantaggio nel momento in cui sia necessario spostare una o più tabelle (quindi non è necessario modificare le query all'interno delle applicazioni).

2.3.8 Privilegi

Un DBA può creare un account utente ma fino a quando non concede il privilegio di sistema `create session` l'utente non può effettuare la procedura di login e connettersi al database. I privilegi sono suddivisi in due categorie principali: privilegi di sistema e privilegi di oggetto. I primi permettono all'utente di connettersi al database e creare o manipolare oggetti mentre i privilegi di oggetto permettono all'utente l'accesso ai dati contenuti nell'oggetto o permettono all'utente di eseguire un programma memorizzato.

Sintassi per concedere un privilegio di sistema:

```
GRANT system_privilege TO {user_name | role | PUBLIC}  
[WITH ADMIN OPTION];
```

Con l'opzione `WITH ADMIN OPTION` si può concedere all'utente non solo il privilegio ma anche la possibilità di concedere lo stesso ad altri utenti. Naturalmente questa eventualità è da evitare il più possibile, in quanto si perde il controllo del privilegio.

Alcuni esempi di privilegi di sistema sono `create session`, `create table`, `alter session`. Il primo è l'unico requisito richiesto affinché un utente si possa connettere al database. In un sistema sicuro, questo dovrebbe essere il solo concesso all'utente.

Consideriamo l'esempio in cui un ipotetico utente Scott abbia il privilegio di sistema `create any view` che permette di creare una vista in una qualsiasi area di un altro utente. Immaginiamo quindi che crei una vista nello spazio di proprietà di Sara. Questo non implica assolutamente che abbia diritto di `select` per quella vista. Scott ha solamente i privilegi per la creazione della vista.

I privilegi che un utente può concedere:

1. **Sulle tabelle e viste:** insert, update, select, delete;
2. **Sulle tabelle:** alter, references, index, all;
3. **Sulle procedure, funzioni, package:** execute.

Questi privilegi offrono al beneficiario della concessione la possibilità di intraprendere alcune azioni sugli oggetti.

Sintassi:

```
GRANT {object_priv|ALL} [ (column1,...) ] ON object  
TO {user|role|PUBLIC} [WITH GRANT OPTION]
```

```
GRANT select ON oggetto to user
```

Così come avviene con l'opzione with admin option, con with grant option si offre l'opportunità all'utente specificato di poter assegnare a sua volta i privilegi sull'oggetto che ha ricevuto da altri utenti. Se per esempio un utente ipotetico Caio concede privilegi with grant option su una propria tabella ad un utente Tizio, questo potrà a sua volta fare concessioni sulla tabella ad altri utenti (solamente i privilegi concessi).

2.4 Meccanismi di autenticazione

L'autenticazione degli utenti è di fondamentale importanza per la sicurezza del database, soprattutto in un ambiente distribuito. È indispensabile identificare gli utenti prima di poter determinare i privilegi e i diritti di accesso.

Il più comune metodo di autenticazione è la password (discusso in maggiore dettaglio nel paragrafo successivo) ma non è il solo. Oracle Advanced Security² fornisce infatti diversi sistemi di autenticazione (di terze parti) e l'uso di SSL e certificati digitali.

Oracle Advanced Security offre i seguenti metodi:

²Suite Oracle che fornisce il software per la crittografia, l'autenticazione e i protocolli di sicurezza

1. **Secure Socket Layer (SSL)**: è un protocollo standard per la sicurezza delle connessioni di rete. Permette di autenticarsi, crittografare i dati, garantisce integrità, PKI. È possibile configurare SSL per permettere l'autenticazione del server, del solo client oppure entrambi.
2. **Kerberos e CyberSafe**: Kerberos è un protocollo di autenticazione dei servizi di rete creato dal MIT che si serve della crittografia a chiave segreta, evitando quindi di inviare la password attraverso la rete. Autenticare mediante Kerberos impedisce agli utenti non autorizzati di intercettare le password inviate attraverso la rete. Lo scopo principale di Kerberos è quello di eliminare la trasmissione delle informazioni di autenticazione attraverso la rete. Il corretto utilizzo di Kerberos vi permette di ridurre drasticamente la possibilità di intercettazione da parte dei packet sniffer³.
3. **Smart Cards**: dispositivi fisici simili ad una carta di credito. Possiedono una memoria ed un processore e vengono letti da un lettore smart-card situato nella workstation client. Si basano su un tipo di autenticazione a due fattori. La smart card può essere bloccata e solamente l'utente che possiede la carta e il corretto PIN (personal identifier number) può sbloccarla.
4. **Biometric**: verifica l'identità di un utente per una sua caratteristica fisica univoca quale l'impronta digitale o la voce.

2.4.1 Password

Per garantire una determinata sicurezza Oracle mette a disposizione un meccanismo di password per proteggere l'accesso al database. Il DBA ha il pieno controllo della politica di gestione delle password e la possibilità di definire come deve essere una password dal punto di vista fisico. Inoltre Oracle mette a disposizione un meccanismo di verifica della complessità delle password che permette di controllare che ciascuna password definita dagli utenti sia abbastanza complessa (protezione ragionevole) contro

³analizzatore di pacchetti di rete

possibili attacchi.

Questa verifica è fornita da Oracle attraverso una funzione PL/SQL per cui è possibile aggiungere ulteriore sicurezza scrivendone una propria (sarebbe una buona norma). Per scrivere una funzione PL/SQL è necessario rispettare alcune regole base quali lunghezza minima della password e richiede inoltre che siano presenti nella password uno o più caratteri alfabetici, numeri e segni di punteggiatura.

Riassumo le regole base:

1. Contenere almeno quattro caratteri;
2. Non deve essere lunga quanto l'username;
3. Contiene almeno un carattere alfabetico, un carattere numerico e un segno di punteggiatura;
4. Differisce dalla password precedente definita dall'utente per almeno tre caratteri.

Com'è possibile osservare l'ultima regola sopra descritta è un ulteriore ricerca di sicurezza e si cerca di evitare che un utente cambi la password di un solo carattere per volta. Naturalmente, come già succede per la sicurezza del sistema operativo, è buona regola non utilizzare parole o nomi che per noi assumono un particolare significato.

Alcune password che adempirebbero alle regole di complessità sopra citate sarebbero: du4_ck e qwpx2#1.

In Oracle le password "scadono" e il sistema provvede un periodo di proroga in cui l'utente ha la notifica che la propria password deve essere cambiata. Ma qualora la password non venga cambiata durante tale periodo, l'account utente viene bloccato e non avrà l'accesso al database fino a quando il DBA non lo permetterà. Ulteriore caratteristica è quella di poter bloccare un utente che superi un determinato numero di tentativi di autenticazione falliti.

```
CREATE PROFILE studenti LIMIT  
FAILED_LOGIN_ATTEMPTS 4  
ACCOUNT_LOCK_TIME 30  
PASSWORD_GRACE_TIME 3;
```

Oracle permette poi di avere una storia delle password attraverso l'uso di parametri quali `password_reuse_time` e `password_reuse_max` per definire il periodo che deve essere trascorso prima che si possa riutilizzare un password. È chiaro che riutilizzare una password (sarebbe buona norma non farlo) può risultare un enorme rischio per la sicurezza. Le password sono criptate e memorizzate nel dizionario dei dati.

```
CREATE PROFILE studenti LIMIT
PASSWORD_REUSE_TIME 60
PASSWORD_REUSE_MAX UNLIMITED;
```

```
CREATE PROFILE studenti LIMIT
PASSWORD_REUSE_TIME UNLIMITED
PASSWORD_REUSE_MAX 3;
```

2.5 Data Dictionary

Il data dictionary del database è di notevole importanza per la sicurezza dell'intero sistema. Un amministratore potrà utilizzarlo per ricavare informazioni di vitale importanza per l'implementazione della sicurezza. Sarà utile conoscerlo per esaminare gli utenti e l'assegnamento della quota, per conoscere la locazione dei data-file ed altre informazioni utili per eseguire il proprio lavoro.

Il data dictionary è fondamentalmente una registrazione interna dello stato di tutti gli oggetti del database: tabelle, utenti, indici, sequenze, viste, link a database, ecc. Possiamo definirlo come un dizionario di "metadati", cioè dati che descrivono gli oggetti nel database. Questi sono valori dinamici che possono cambiare qualora cambino gli oggetti. Ad esempio, nel momento in cui viene creata una tabella, il nome di tutti i campi vengono registrati nel data dictionary. Ed esso verrà inoltre aggiornato nel momento in cui la tabella venga modificata o cancellata.

Il data dictionary di Oracle può essere suddiviso in due livelli: le tabelle che formano il reale data dictionary ed alcune viste che permettono l'accesso ai dati.

Il data dictionary quindi risulta essere molto importante dal punto di

vista della sicurezza in quanto offre funzioni di controllo utili per tenere traccia delle azioni degli utenti.

Le viste del dizionario sono divise fondamentalmente in quattro insiemi:

1. Viste contenenti informazioni sugli oggetti di proprietà dell'utente Oracle con cui si connessi, definite USER%;
2. Viste con informazioni su tutti gli oggetti accessibili per l'utente, ALL%;
3. Viste con informazioni sugli oggetti accessibili dagli utenti DBA, dunque tutto ciò che esiste in ogni utente Oracle (DBA%);
4. Le dynamic performances tables, vale a dire le viste che contengono a runtime informazioni su tutto quello che sta avvenendo nel DB (V\$%).

Sebbene il data dictionary permetta di ricavare numerose informazioni sullo stato del database, le viste qui sotto presentate sono particolarmente importanti per ottenere informazioni sulla sicurezza:

View Name	Type of Information Available
DBA_PROFILES	Profiles and their associated resource and time limits
DBA_ROLES	All roles that exist in the database
DBA_ROLE_PRIVS	Roles granted to users and other roles
DBA_SYS_PRIVS	System privileges granted to users and roles
DBA_TAB_PRIVS	Privileges like SELECT, INSERT, UPDATE, etc. that each user or role has per object
DBA_USERS	Who has an account for the database; also, which profile is assigned to the user
ROLE_ROLE_PRIVS	Roles granted to roles
ROLE_SYS_PRIVS	System privileges granted to roles
ROLE_TAB_PRIVS	Table privileges granted to roles
USER_ROLE_PRIVS	

Tabella 2.1: Viste di particolare importanza

Per consultare i privilegi concessi agli utenti sulle tabelle del database sono disponibili le viste:

1. USER_TAB_PRIVS
2. USER_TAB_PRIVS_MADE
3. USER_TAB_PRIVS_RECD

La prima mostra, per ogni grant concessa, il nome della tabella, dell'owner, dell'utente che ha concesso la grant e di quello che l'ha ricevuta. Sulla seconda vista si vedono solo le grant su oggetti di proprietà dell'utente connesso; sulla terza solo le grant concesse all'utente connesso. Stessa logica seguono, per le colonne, le viste:

1. USER_COL_PRIVS
2. USER_COL_PRIVS_MADE
3. USER_COL_PRIVS_RECD

Le viste USER_ROLE_PRIVS e USER_SYS_PRIVS contengono informazioni, rispettivamente, sui ruoli e sui privilegi di sistema concessi all'utente connesso. Se, ad esempio, si esegue la seguente istruzione SQL si ottiene il un risultato simile:

```
select * from user_role_privs;

USERNAME GRANTED_ROLE ADM DEF OS_
-----
NOME_UTENTE CONNECT NO YES NO
NOME_UTENTE DBA NO YES NO
NOME_UTENTE RESOURCE NO YES NO
```

Si può osservare che all'utente è stato attribuito il ruolo DBA, CONNECT e RESOURCE

Vista	Utilizzo
V\$DB_OBJECT_CACHE	Mostra gli oggetti contenuti all'interno della memoria comune (shared memory) e mostra le loro statistiche, come la memoria utilizzata, il numero di volte che sono stati caricati in memoria ed il numero di volte che sono stati eseguiti. È così possibile accedere alle query e stored procedure più utilizzate e conoscerne le performances.
V\$SYSSTAT	Visualizza le operazioni basilari del database ed il loro numero di esecuzioni dal momento in cui l'istanza è attiva.
V\$RECOVERY_STATUS	Permette di accedere alle informazioni sullo stato dei backup.
V\$LIBRARYCACHE	Visualizza le statistiche in merito al contenuto della shared pool library cache, il buffer degli oggetti comuni. Un informazione molto importante è la colonna GETHITRATIO, che indica il rapporto fra RELOADS e PINS: bisogna essere allarmati qualora tale valore non fosse inferiore a 1.
V\$DBFILE,V\$LOGFILE	Contengono informazioni sui file di database e di redo log.

V\$SESSION	Visualizza informazioni riguardo alle sessioni attualmente aperte. Per ogni connessione è possibile conoscere il nome dell'utente di rete (oltre all'utente Oracle), l'indirizzo di rete e l'applicazione ed il modulo utilizzati per la connessione.
V\$TRANSACTION	Permette di accedere alle informazioni sulle transazioni attualmente in corso.

Tabella 2.2: Dynamic Performances Tables

2.6 Oracle e rete

Oracle, per l'accesso ai database e ai relativi servizi in ambiente distribuito, offre un'infrastruttura indipendente dalla piattaforma. Questo prodotto in passato sql*net, ora è chiamato net8.

L'infrastruttura di rete Oracle segue un modello di computazione client-server. La funzione principale di net8 è quella di stabilire sessioni di rete e trasferire dati tra una macchina client e il server oppure tra due server (un server in questo caso agisce come client).

Le sessioni sono stabilite per mezzo di un listener. Un listener è un processo che risiede nel server e che riceve richieste di connessione dei clienti ed inoltra la richiesta al server. Un "listener" è caricato in ogni macchina che ospita un database o qualsiasi servizio che deve poter essere acceduto da un host differente.

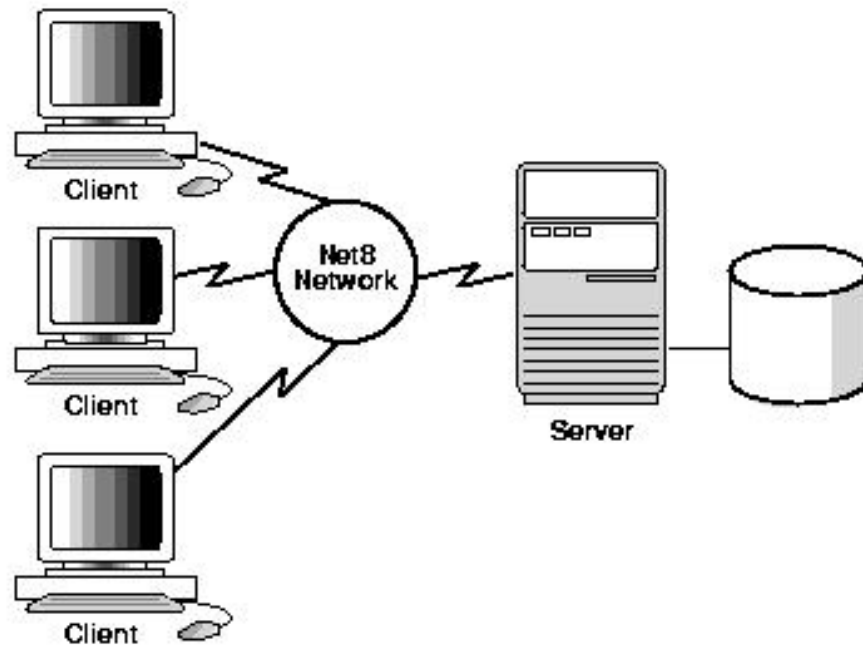


Figura 2.2: Connessione client/server

In questo modo, nel momento in cui l'applicazione cliente ha necessità di accedere ad un database (host remoto), stabilisce un collegamento con il listener dell'host remoto (che indirizza il client verso il servizio richiesto).

Una volta che la negoziazione è completata, il cliente comunica direttamente con il database (o servizio richiesto).

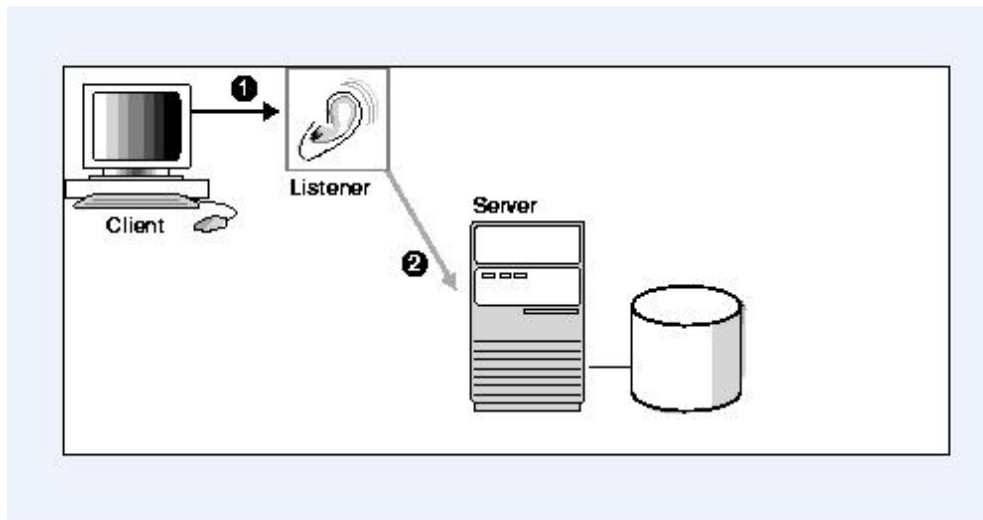


Figura 2.3: Listener - richiesta di connessione

Net8 può supportare diversi protocolli quali TCP/IP, SPX, DECnet e reti eterogenee. Purtroppo net8 da solo non basta per rendere la comunicazione sicura in rete. Esiste per questo motivo un pacchetto aggiuntivo “Advanced Security” che permette di migliorare sotto questo aspetto net8. I dati con net8 sono trasferiti in “chiaro” ed inoltre non è possibile rilevare se un pacchetto di rete sia stato intercettato e/o modificato durante il transito in rete.

Naturalmente è chiaro come le limitazioni di net8 possano essere importanti in termini di sicurezza e privacy: cosa succederebbe se i dati fossero modificati durante il transito in rete?.

Senza “Advanced Security” net8 accetterà qualsiasi richiesta sulla porta della macchina in ascolto indipendentemente da chi effettua la richiesta. Con Advanced Security qualsiasi applicazione client che richieda un collegamento può specificare una lista di metodi di crittografia che può supportare (in ordine di preferenza) e se la crittografia è richiesta, accettata o rifiutata.

Ogni listener può fare altrettanto (negoziazione).

Se l'una o l'altra parte rifiuta la crittografia e le due parti non possono concordare su un metodo di crittografia, fallisce la connessione.

Advanced Security abilita inoltre il database a modi d autenticazione differenti dalla semplice username e password:kerberos, SecurID ..ecc.

2.6.1 Configurare il listener net8: accettare o rifiutare richieste da specifici indirizzi IP

Una caratteristica interessante si presenta con la rete TCP/IP. Net8 infatti può essere configurato per accettare o rifiutare richieste che provengono da un elenco di indirizzi IP. Ogni listener è specificato in un file "listener.ora". Questo file specifica il nome del listener, una lista di porte e protocolli, e una lista di database e altre servizi disponibili. Come specificato in precedenza se si sta utilizzando il protocollo TCP/IP , è possibile specificare una lista di indirizzi ip dai quali le richieste saranno accettate o rifiutate. Per abilitare questa caratteristica all'interno del file protocol.ora (o sqlnet.ora a seconda della versione di Oracle) è necessario inserire il seguente codice:

```
tcp.validnode_checking=yes
tcp.invited_nodes=(address1, address2, ...)
tcp.excluded_nodes=(address1, address2, ...)
```

Gli indirizzi possono essere nomi oppure indirizzi ip numerici.

Una volta fatto ciò è sufficiente far ripartire o ricaricare il listener (i cambiamenti avranno effetto). Da quel momento in poi qualsiasi richiesta proveniente da un nodo escluso (tcp.excluded_nodes) sarà ignorata.

Capitolo 3

Auditing

Questo capitolo vuole approfondire il concetto di auditing già espresso nel capitolo introduttivo.

Come già precisato la sicurezza è un processo dinamico che si sviluppa nel tempo ed è pertanto opportuno che l'amministratore o il semplice utente che accede al database abbia una visione globale del sistema e conosca come proteggere le proprie risorse e/o oggetti.

3.1 Auditing

Oracle stesso fornisce alcune funzioni di controllo della sicurezza del sistema. Tra queste consideriamo la funzione di `AUDIT TRAIL` che permette la registrazione di qualsiasi attività, di nostro particolare interesse, svolta all'interno del database. Possiamo infatti definire con auditing quel processo che offre la possibilità di monitorare e registrare le attività all'interno del database. Alcune azioni che possono ad esempio essere monitorate sono le seguenti:

- La visualizzazione, la modifica o la cancellazione di informazioni dalle tabelle;
- La creazione o la rimozione di oggetti quali tabelle, viste, procedure, triggers, ecc;

- Esecuzione di programmi.

È utile precisare immediatamente che per mezzo di funzionalità standard di Oracle non è possibile effettuare un monitoraggio a livello riga. In altre parole, attraverso lo standard Oracle, è possibile ad esempio controllare le azioni che sono state effettuate su una tabella ma non cosa sia effettivamente cambiato in una specifica tabella. Nel secondo caso è opportuno scrivere del codice appropriato.

Quindi possiamo dire che le caratteristiche dell'auditing sono principalmente utili per la sicurezza e per poter valutare chi è connesso in un determinato istante, da dove è connesso, poter effettuare statistiche degli accessi, monitorare gli accessi ai dati e potenziali attività sospette.

Normalmente l'auditing non è attivato di default. Quindi esistono molte scuole di pensiero a riguardo. Se all'interno dell'azienda è stato installato un firewall oppure altre misure di sicurezza è possibile credere che il sistema sia abbastanza sicuro e non sia necessario attivare l'auditing. L'auditing potrebbe essere attivato a meno che o fino al momento in cui si sia verificato un problema di sicurezza dovuto ad un intrusore esterno oppure un dipendente che ha oltrepassato il limite. Potreste quindi abilitare l'auditing come conferma che qualcosa sia "anormale". Questo approccio "retro-attivo" naturalmente non è consigliato.

In altre situazioni ancora si opta per una scelta di auditing "pesante". Anche questo approccio ha i suoi svantaggi e quindi non sempre può essere considerata una scelta corretta, in quanto ogni oggetto sotto controllo porta con sé un costo potenziale sia per quanto riguarda le risorse sia di prestazioni. Pertanto è opportuno scegliere con attenzione le forme di auditing in modo da ridurre al minimo indispensabile le risorse sotto controllo.

3.2 Perché auditing

Il primo passo nello sviluppo di un piano di auditing è determinare il motivo oppure le ragioni per cui si ha la necessità di controllare il database. Al livello generale possono esistere due principali motivi:

1. **Sicurezza:** determinare se qualcuno sta tentando di “entrare” nel sistema. Quindi l’auditing è una risorsa utile per confermare possibili abusi (sospetti). Ad esempio è possibile confermare se dei dati siano stati cancellati da una tabella del database semplicemente abilitando l’auditing per tracciare le operazioni di cancellazione da quella specifica tabella. È opportuno e possibile cominciare con un controllo generale per poi restringere il campo di controllo per poter individuare meglio la sorgente del problema.
2. **Prestazioni:** Perché il sistema è così lento? Come viene utilizzato il database?

Una volta definito lo scopo sarà più facile decidere gli oggetti da controllare. Determinare le ragioni aiuterà inoltre a restringere l’ambito per evitare di raccogliere troppe informazioni inutili.

3.3 Abilitare l’auditing

Una volta definita qualche forma di controllo è necessario decidere il luogo in cui saranno memorizzate le informazioni di auditing. A livello pratico è possibile intervenire su determinati parametri del file di configurazione INIT.ORA di Oracle:

1. `AUDIT_FILE_DEST`: specifica ad Oracle il nome della directory in cui le informazioni saranno memorizzate;
2. `AUDIT_TRAIL`: abilita o disabilita l’auditing . Valori specifici “NONE”, “OS” o “DB”. Con NONE (default)l’auditing non sarà abilitato. Con valore OS i risultati saranno scritti in un file presente nella directory `AUDIT_FILE_DEST`. Con DB i risultati di auditing saranno conservati nella tabella `SYS.AUD$`.

Una osservazione obbligatoria è opportuna qualora si specifichi come valore di `AUDIT_TRAIL`, “DB”. In questo caso infatti l’utente SYS è proprietario della tabella `SYS.AUD$` e i valori registrati sono conservati nel tablespace di sistema. Se le opzioni di auditing sono “costose” in termini di registrazioni di attività è possibile che si generi frammentazione.

Pertanto sarebbe buona regola spostare ad un altro tablespace la tabella SYS.AUD\$. Per ottenere ciò, una volta connesso come SYS:

1. Creare un nuovo tablespace;

```
SQL> connect sys/<appropriate_sys_password>
Connected.
CREATE TABLESPACE audit_storage
DATAFILE 'E:\ORANT\DATABASE\AUDSTOR01.ORA'
SIZE 40M;
```

2. Creare una tabella temporanea nel nuovo tablespace usando il comando AS SELECT * FROM SYS.AUD\$;

```
SQL>
CREATE TABLE audit_temp
TABLESPACE audit_storage
STORAGE (
INITIAL 10M
NEXT 1M
MINEXTENTS 1
MAXEXTENTS 10
PCTINCREASE 1)
AS SELECT *
FROM sys.aud$;
```

3. Cancellare e rinominare la tabella temporanea;

```
SQL> DROP TABLE sys.aud$;
Table dropped.
```

```
SQL> RENAME audit_temp TO aud$;
Table renamed.
```

Naturalmente è compito dell'amministratore controllare che il tablespace assegnato per l'auditing non diventi "saturo". Qualora accada l'auditing si interrompe e le informazioni sono perdute.

3.4 Risultato dell'auditing

Ciascuna azione sotto controllo, qualora venga eseguita può generare uno o più record nella tabella SYS.AUD\$ del data dictionary. Interrogando direttamente questa tabella oppure le viste create a partire da questa tabella è possibile ottenere un riassunto delle informazioni di particolare interesse per la sicurezza del sistema. Le colonne sono numerose pertanto ne riporto alcune significative:

1. Azione compiuta;
2. Privilegio usato per compierla;
3. L'utente che l'ha compiuta;
4. L'oggetto su cui è stata compiuta;
5. Il giorno e l'ora in cui è stata compiuta;
6. ecc.

Il modo più semplice per ottenere la descrizione delle viste di auditing è quello di effettuare una select del tipo:

```
SQL>
COLUMN table_name FORMAT a25
COLUMN comments FORMAT a45 WORD
SELECT table_name, comments
FROM dictionary
WHERE table_name LIKE '%_AUDIT%';
```

TABLE NAME	COMMENTS
ALL_DEF_AUDIT_OPTS	Auditing options for newly created objects
DBA_AUDIT_EXISTS	Lists audit trail entries produced by AUDIT NOT EXISTS and AUDIT EXISTS
DBA_AUDIT_OBJECT	Audit trail records for statements concerning objects, specifically: table, cluster, view, index, sequence, [public] database link, [public] synonym, procedure, trigger, rollback segment, tablespace, role, user
DBA_AUDIT_SESSION	All audit trail records concerning CONNECT and DISCONNECT
DBA_AUDIT_STATEMENT	Audit trail records concerning with grant, revoke, audit, noaudit and alter system
DBA_AUDIT_TRAIL	All audit trail entries
DBA_OBJ_AUDIT_OPTS	Auditing options for all tables and views
DBA_PRIV_AUDIT_OPTS	Describes current system privileges being audited across the system and by user
DBA_STMT_AUDIT_OPTS	Describes current system auditing options across the system and by user
USER_AUDIT_OBJECT	Audit trail records for statements concerning objects, specifically: table, cluster, view, index, sequence, [public] database link, [public] synonym, procedure, trigger, rollback segment, tablespace, role, user
USER_AUDIT_SESSION	All audit trail records concerning CONNECT and DISCONNECT

USER_AUDIT_STATEMENT	Audit trail records concerning with grant, revoke, audit, noaudit and alter system
USER_AUDIT_TRAIL	Audit trail entries relevant to the user
USER_OBJ_AUDIT_OPTS	Auditing options for user's own tables and views

Tabella 3.1: Viste di auditing

3.5 Opzioni di auditing

Oracle permette tre tipi di controllo rivolti a:

- statement SQL specifici (connect, create table,);
- privilegi (system grant, ecc);
- operazioni (select, insert, alter, ecc) sugli oggetti dei Oracle.

Possono essere controllate le azioni che hanno restituito errore e/o quelle seguite con successo specificando `WHENEVER NOT SUCCESSFUL` o rispettivamente `WHENEVER SUCCESSFUL` nella dichiarazione del comando `AUDIT`. Inoltre i record di audit possono riferirsi ad ogni singola azione (`BY ACTION`) oppure essere raggruppate in un solo record per sessione (`BY SESSION`).

Sintassi del comando `AUDIT`:

```
AUDIT [option | ALL]
ON [username.]objectname
[BY [SESSION|ACCESS]]
[WHENEVER [NOT] SUCCESSFUL]
```

3.5.1 Statement

Questa forma di audit permette di controllare comandi SQL eseguiti dagli utenti.

AUDIT TABLE;

Questo comando di audit permette di mantenere sotto controllo tutte le CREATE TABLE, ALTER TABLE, DROP TABLE, TRUNCATE TABLE.

AUDIT SELECT TABLE;

Controlla tutte le possibili select.

3.5.2 Privilegi

Questa forma di audit controlla l'utilizzo dei privilegi di sistema quali CREATE TABLE oppure SELECT ANY TABLE Esempio:

AUDIT CREATE TABLE

3.5.3 Oggetti

È possibile con quest'ultima forma di audit controllare le referenze ai seguenti tipi di oggetto: tabelle, viste, sequenze, stored procedures, funzioni, ecc. Esempio:

AUDIT SELECT ON tablename

Controlla tutte le SELECT eseguite sulla tabella specificata (tablename).

3.6 Privilegi per l'auditing

Affinché un utente possa abilitare l'auditing è obbligatorio che possieda il privilegio AUDIT_SYSTEM.

Comunque sia anche senza privilegi concessi agli utenti esistono alcune azioni che saranno registrate in un file del sistema operativo sia nel caso in cui l'auditing sia abilitato sia che non lo sia. Quando viene creato un database viene infatti creato un file normalmente denominato "alert log" (la cui locazione dipende dal sistema operativo). Le azioni conservate in questo file sono:

1. Startup e shutdown del database;
2. Ogni connessione stabilita da un utente con privilegi di sistema;
3. Qualsiasi cambiamento alla struttura del database.

3.7 Auditing e Triggers

Come specificato in precedenza le caratteristiche standard di auditing di Oracle non permettono un controllo a livello riga. I triggers quindi compensano questa carenza e sono molto utili nel momento in cui si abbia necessità di controllare i cambiamenti dei dati di una o più tabelle. In altre parole i triggers offrono la possibilità di registrare il valore della riga sia precedentemente alla modifica sia successivamente.

3.8 Esempi di utilizzo

3.8.1 Tentativi di autenticazione falliti

```
SQL>
select count(*),username,terminal,
to_char(timestamp,'DD-MON-YYYY')
from dba_audit_session
where returncode<>0
group by username,terminal,
to_char(timestamp,'DD-MON-YYYY');
```

COUNT(*)	USERNAME	TERMIN	TO_CHAR(TIM
1	SCOOT	pts/3	15-APR-2004
3	FRANK	pts/3	17-APR-2004
4	PAMELA	pts/1	18-APR-2004

Dai risultati ottenuti possiamo verificare la presenza di due possibili tentativi di accesso non autorizzati. Ciò potrebbe significare che un

utente ha semplicemente dimenticato la propria password oppure potrebbe essere valida l'ipotesi per cui qualcuno abbia provato ad indovinare la password degli utenti sospetti (nel caso specifico Frank e Pamela). Per questioni di sicurezza sarebbe opportuno verificare ogni giorno possibili tentativi di accesso non autorizzato e valutare di volta in volta la causa. Naturalmente maggiore è la frequenza con la quale un utente non riesce ad autenticarsi al sistema maggiore è la possibilità che si tratti di possibile intrusione non autorizzata.

3.8.2 Tentativi di accesso al database con utenti che non esistono

Un altro interessante esempio riguarda la verifica di possibili autenticazioni mediante username inesistenti. Riporto un semplice codice SQL:

```
SQL>
select username,terminal,
to_char(timestamp,'DD-MON-YYYY HH24:MI:SS')
from dba_audit_session
where returncode<>0
and not exists (select 'x'
from dba_users
where
dba_users.username=dba_audit_session.username)
```

USERNAME	TERMIN	TO_CHAR(TIMESTAMP, 'D
FRED	pts/3	09-APR-2003 17:31:47
FRED	pts/3	09-APR-2003 17:32:02
FRED	pts/3	09-APR-2003 17:32:15
BILL	pts/3	09-APR-2003 17:33:01

Come possiamo osservare dal risultato sopra riportato possiamo affermare che si tratta di un possibile abuso. Qualcuno ha provato ad

autenticarsi utilizzando uno username che non esiste. Anche in questo caso è opportuno controllare ogni giorno possibili tentativi di intrusione e valutare oppure andare a fondo per valutarne la reale causa.

3.8.3 Tentativi di accesso ad ore non usuali

Questo terzo semplice esempio permette di verificare che non ci siano tentativi di accesso fuori dall'orario di lavoro. Si potrebbe trattare di lavoro straordinario oppure di manutenzione ma nonostante questo è importante conoscere la causa di questi tentativi di accesso.

```
SQL>
select
username,terminal,action_name,
to_char(timestamp,'DD-MON-YYYY HH24:MI:SS'),
from dba_audit_session
where to_date(to_char(timestamp,'HH24:MI:SS'),'HH24:MI:SS')<
to_date('08:00:00','HH24:MI:SS')
or to_date(to_char(timestamp,'HH24:MI:SS'),'HH24:MI:SS')>
to_date('19:30:00','HH24:MI:SS')
```

USERNAME	TERMIN	ACTION_N	TO_CHAR(TIMESTAMP,'D
SYS	pts/1	LOGOFF	09-APR-2003 20:10:46
SYSTEM	pts/5	LOGOFF	09-APR-2003 21:49:20
ZULIA	pts/5	LOGON	09-APR-2003 21:49:50
EMIL	APOLLO	LOGON	09-APR-2003 22:49:12

I risultati offrono una visione degli accessi al sistema avvenuti prima delle 8.00 della mattina e dopo le 19.30 di sera. Ogni connessione dovrebbe essere investigata. Inoltre è opportuno capire, qualora succeda, come mai ci siano accessi privilegiati provenienti dell'esterno.

Capitolo 4

Backup

Oramai è chiaro quanto i dati siano di notevole importanza per ciascuno di noi. Tutto ruota attorno alle informazioni memorizzate nel database. Perdere dei dati potrebbe ripercuotersi negativamente all'interno di un'azienda.

Per questo motivo è opportuno definire una strategia di backup e di recupero funzionale che permetta di recuperare in breve tempo la “disponibilità” dei dati e così perdere il numero minore di transazioni non completate.

Quando gestiamo un database è buona regola ricordarsi che software e hardware possono essere sostituiti ma i dati, in assenza di copie, non possono essere ripristinati o quanto meno non completamente.

4.1 Frequenza di backup

Il backup è essenzialmente una copia dei dati. I backup possono essere divisi in due categorie principali: backup fisici e backup logici. I backup fisici sono copie dei file fisici del database mentre i backup logici contengono dati esportati usando comandi sql e memorizzati in un file binario. Generalmente quindi i backup logici sono supplementari ai backup fisici. Un amministratore potrebbe eseguire un backup giornalmente, settimanalmente oppure mensilmente ma è corretto precisare che esistono alcuni fattori che dovrebbero essere presi in considerazione:

1. Requisiti di disponibilità del database;
2. Importanza dei dati del database;
3. Possibilità o meno di arrestare il database per effettuare copie dei dati;
4. Dimensione totale dei file logici da copiare.

4.2 Archivelog

Ogni volta che occorre un cambiamento nel database, Oracle genera un record (cambiamento) in un “redo log”. Vengono memorizzati tutti i cambiamenti, quindi sia quelli che hanno avuto effetto sia quelli che non hanno avuto esito positivo.

Oracle costantemente registra i redo log negli “online redo log” che sono sul disco. Attivando la modalità archivelog, il sistema può memorizzare queste informazioni copiando gli online redo log in altre destinazioni sul disco.

Per abilitare la modalità archivelog è necessario:

1. Togliere il commento alle seguenti righe all’interno del file di configurazione init.ora:

```
#Uncommenting the line below will cause automatic
#archiving if archiving has
#been enabled using ALTER DATABASE ARCHIVELOG.
#log_archive_start = true
#log_archive_dest = %ORACLE_HOME%\database\archive
#log_archive_format = %%%ORACLE_SID%%T%TS%S.ARC
```

2. STARTUP MOUNT:

```
ALTER SYSTEM ARCHIVE LOG START;
ALTER DATABASE OPEN;
```

4.3 Forme di backup

Oracle mette a disposizione diverse possibilità per effettuare backup. Presento qui di seguito alcune opportunità. Ciascuna soluzione è più o meno raffinata rispetto alle altre, comunque sia tutte sono ugualmente funzionali ed utili:

1. **Export/Import:** due utility di Oracle che permettono di esportare ed importare l'intero contenuto logico (tabelle, viste, ecc...) di uno o più schemi in un unico file;
2. **Off-line backup:** si effettua quando si arresta il database. Si effettua una copia fisica di tutti i file;
3. **On-line backup:** Con database attivo si mettono i tablespace nello stato di backup e si effettua la copia dei loro file di redo log e control.

4.3.1 Export e Import

Export e Import sono due utility Oracle che permettono rispettivamente di esportare ed importare il contenuto logico di un database verso e da un unico file. Contenuto logico significa che non sono esportati file fisici (datafile, log file, ecc) bensì il loro contenuto (tabelle, viste, ecc...).

Naturalmente export è un utility che permette di creare un output a partire da un ambiente di Oracle mentre Import è un utility che può leggere solo file generati da export.

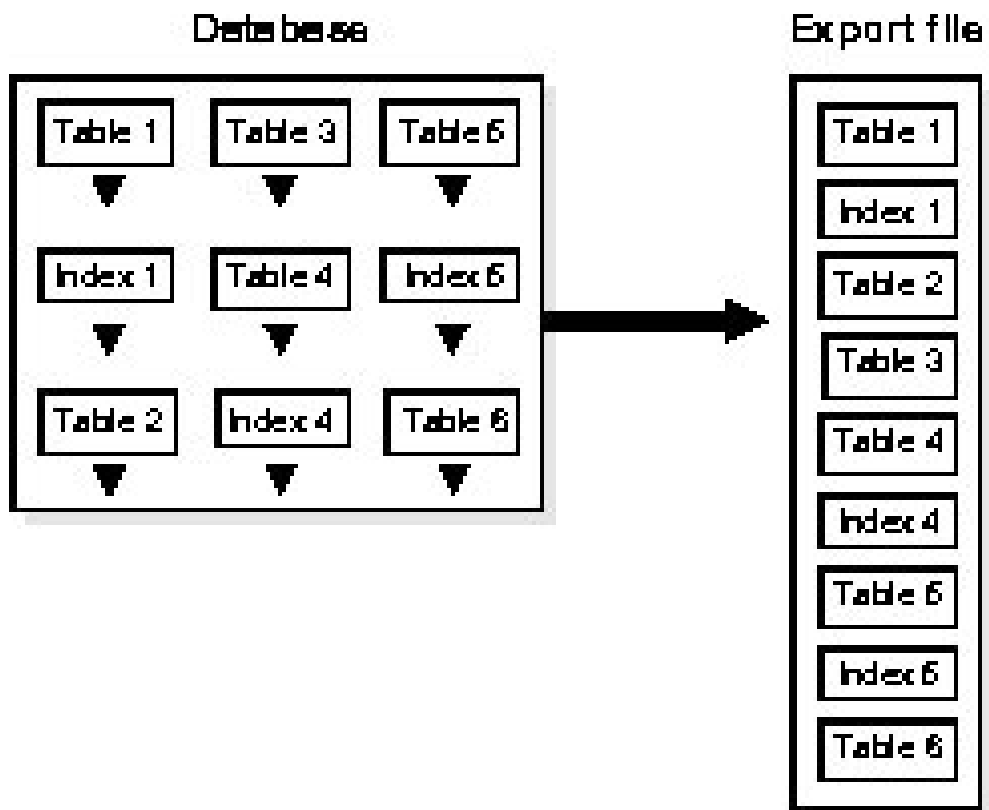


Figura 4.1: Esportare un database

Un vantaggio di avere un export è quello di poter facilmente ripristinare una singola oppure un insieme di specifiche tabelle. Le due utility possono essere utilizzate comunemente per eseguire i seguenti compiti:

1. Backup e recovery (per piccoli database);
2. Riorganizzare i dati;
3. Rilevare corruzione del database ed assicurare che tutti i dati possano essere letti;
4. Trasportare tablespace da un database ad un altro.

Ci sono differenti forme di export che possono essere eseguite. Queste si distinguono utilizzando un parametro "INC_TYPE"

1. COMPLETE: per un completo export del database;
2. INCREMENTAL: "cattura" ciò che è cambiato dall'ultimo incremento;
3. CUMULATIVE: "cattura" ciò che è cambiato dall'ultimo export completo.

4.3.2 Backup a freddo

Backup off-line può essere definito anche backup a freddo. Per poter effettuare questo tipo di backup è necessario arrestare il database tramite Sql*Plus (oppure Enterprise Manager) e successivamente copiare nelle apposite unità di recupero tutti i file che costituiscono il database. Una volta fatto ciò è possibile far ripartire il database. A seguire i file fisici che devono essere copiati.

Datafiles:

```
select name from v$datafile
```

NAME

```
-----  
F:\ORACLE\ORADATA\DBPROVA\SYSTEM01.DBF  
F:\ORACLE\ORADATA\DBPROVA\RBS01.DBF  
F:\ORACLE\ORADATA\DBPROVA\USERS01.DBF  
F:\ORACLE\ORADATA\DBPROVA\TEMP01.DBF  
F:\ORACLE\ORADATA\DBPROVA\TOOLS01.DBF  
F:\ORACLE\ORADATA\DBPROVA\INDX01.DBF  
F:\ORACLE\ORADATA\DBPROVA\DR01.DBF
```

Selezionate 7 righe.

Control file:

```
select name from v$controlfile
```

NAME

```
-----  
F:\ORACLE\ORADATA\DBPROVA\CONTROL01.CTL  
F:\ORACLE\ORADATA\DBPROVA\CONTROL02.CTL  
F:\ORACLE\ORADATA\DBPROVA\CONTROL03.CTL
```

On-line redo logs:

```
select member from v$logfile
```

MEMBER

```
-----  
F:\ORACLE\ORADATA\DBPROVA\REDO03.LOG  
F:\ORACLE\ORADATA\DBPROVA\REDO02.LOG  
F:\ORACLE\ORADATA\DBPROVA\REDO01.LOG
```

Questa strategia è totale per cui significa che viene effettuato un backup completo del database e la sola mancanza di uno dei file potrebbe portare all'inutilità della copia.

Lo svantaggio di adottare questo tipo di strategia è la difficoltà che occorre qualora si voglia recuperare una oppure un insieme di specifiche tabelle.

4.3.3 Backup a caldo

Backup on-line può essere definito anche backup a caldo. È possibile effettuare copie di salvataggio senza dover arrestare il database.

La procedura corretta richiede di mettere i tablespace nello stato di backup, effettuare la copia, e poi riportarli in stato normale.

```
ALTER TABLESPACE tablespace_name BEGIN BACKUP
```

È opportuno seguire i seguenti passi per eseguire un backup a caldo:

1. Essere sicuro di eseguire

```
ALTER TABLESPACE tablespace_name END BACKUP
```

dopo il completamento della copia dei file

2. Salvare il corrente log file :

```
ALTER SYSTEM SWITCH LOGFILE
```

3. Salvare tutti gli archive logs dal momento in cui il backup comincia sino a quando viene completato

4. Copiare i control file :

```
ALTER DATABASE BACKUP CONTROLFILE TO filespec
```

Capitolo 5

Row Level Security

Con questo capitolo si conclude la panoramica di alcune caratteristiche del sistema Oracle in termini di sicurezza del sistema e dei dati.

Fin qui si è potuto constatare come mediante diversi sistemi di autenticazione, ruoli, privilegi ed altre caratteristiche descritte nei precedenti capitoli, si possano gestire gli utenti e le azioni che essi possono compiere all'interno del database.

In tutte le versioni di Oracle è sempre stato possibile concedere oppure negare l'accesso a particolari oggetti del database, purtroppo questi privilegi sono definiti al livello di oggetto, per un'intera tabella, non per una o più specifiche righe di una tabella. In alcuni casi può essere sufficiente questo tipo di approccio, in altre circostanze è opportuno un maggiore controllo.

Row Level Security è una caratteristica di Oracle, introdotta con la versione Oracle 8i, che permette di restringere l'accesso a sole specifiche righe di una tabella in base ai privilegi sulla tabella stessa. Questa funzionalità è stata anche spesso descritta come “*fine grained access control*” o “*virtual private databases*”.

5.1 Cosa significa

Per garantire un maggior controllo sulla sicurezza dei dati, Oracle permette di definire delle funzioni di politica di sicurezza (strettamente connesse

ad una tabella oppure una vista del database) eseguite ogni volta che i dati della tabella o vista vengono visualizzati e/o modificati.

Questa funzione restituisce un “pezzo” addizionale di codice sql definito “predicato” che viene “attaccato” alla clausola where della dichiarazione sql originale. Ciò significa che la funzione controlla quali righe dei dati della tabella sono restituite all’utente che ha effettuato la richiesta.

Il database impone quindi le politiche di sicurezza, indipendentemente dal metodo utilizzato per accedere ai dati. Consente di gestire con un unico database i dati di utenti differenti nello stesso modo in cui fossero fisicamente residenti su database diversi tra loro.

È possibile inoltre implementare la sicurezza una sola volta, nel server dati, anziché in tutte le applicazioni che accedono ai dati.

Pertanto per mezzo di questa funzionalità possiamo:

1. Aumentare la sicurezza dei dati in maniera dinamica, evitando di dover mantenere delle viste statiche;
2. Impostare delle politiche di sicurezza differenti per select, insert, update, delete.

5.2 Esempio

Per comprendere bene il significato di “row level security” proviamo ad analizzare un semplice esempio di utilizzo definendo tutti i passaggi necessari per implementare questo tipo di sicurezza dei dati.

È necessario eseguire le seguenti operazioni (descritte successivamente):

1. Creare un contesto;
2. Creare un package che generi un predicato;
3. Eseguire il package DBMS_RLS per creare una politica di sicurezza;
4. Creare un’interfaccia di sicurezza del package.

5.2.1 Assunzioni

Creiamo un utente generico “test” e gli concediamo i privilegi per creare tabelle ed aggiungere dati, creare procedure, creare un contesto ed accedere ai packages di row level security e di sessione (tutti i privilegi necessari per i nostri scopi):

```
create user test identified by test default
tablespace users temporary tablespace temp;
```

```
grant create session to test;
```

```
grant create any context to test;
```

```
grant create table to test;
```

```
grant unlimited tablespace to test;
```

```
grant create procedure to test;
```

```
grant execute on dbms_ols to test;
```

```
grant execute on dbms_session to test;
```

A questo punto possiamo definire la tabella che ci permetterà di analizzare e comprendere meglio il funzionamento:

```
create table paziente(
id_paziente NUMBER,
nome varchar2(100),
cognome varchar2(100),
data date,
tipo_intervento varchar2(100)
);
```

Assumiamo di aver inserito dei dati per i test, per cui eseguendo la richiesta seguente si ottiene il seguente risultato:

```
select * from paziente;
```

ID_PAZIENTE	NOME	COGNOME	DATA	TIPO_INTER
1	Paolo	Rossi	01-GIU-04	visita
2	Marco	Valentini	13-APR-04	ricovero
3	Alessio	Fumagalli	14-FEB-04	visita

Assumiamo vere queste condizioni per il nostro semplice esempio:

1. Un ipotetico direttore ha i privilegi per vedere tutte le registrazioni;
2. Un dottore ha solo il privilegio di analizzare i dati dei pazienti ricoverati;
3. Un infermiere può visualizzare i dati dei pazienti visitati.

Naturalmente queste condizioni sono “irreali” ma ci permettono di capire come Oracle definisca la sicurezza a livello riga.

5.2.2 Context security

La funzionalità si basa su un concetto preciso di “session context”, definito come uno spazio di memoria nel quale sono memorizzate variabili di sessione per una particolare applicazione web. In altre parole un insieme di coppie nome/valore globali alla sessione.

Il contesto è associato al package che sarà usato per impostare i valori in questo contesto per gli utenti.

```
create or replace context test_prova using set_test_context;
```

Per il nostro obiettivo definiamo un package che ci permetta di definire il ruolo associato all’utente in una sessione (a livello reale dovrebbe essere impostato in fase di autenticazione).

```
create or replace package set_test_context
is
procedure s_dottore;
```

```
procedure s_infermiere;  
procedure s_direttore;  
end;  
/
```

E il corpo del package:

```
create or replace package body set_test_context  
as  
procedure s_dottore  
is  
begin  
dbms_session.set_context('test_prova','ruolo','dottore');  
end;  
--  
procedure set_infermiere  
is  
begin  
dbms_session.set_context('test_prova','ruolo','infermiere');  
end;  
--  
procedure set_direttore  
is  
begin  
dbms_session.set_context('test_prova','ruolo','direttore');  
end;  
end;  
/
```

5.2.3 Dynamic Access Predicates

Questa è la parte fondamentale dell'implementazione di row level security. La funzione che viene definita verifica il contesto per l'utente corrente e restituisce un predicato, quindi un "pezzo" di codice sql appeso alla clausola where.

```

create or replace package test_policy
as
function test_predicate(schema_name in varchar2,
object_name in varchar2)
return varchar2;
end;
/

create or replace package body test_policy
as
function test_predicate(schema_name in varchar2,
object_name in varchar2)
return varchar2
is
lv_predicate varchar2(1000):='';
begin
if sys_context('test_prova','ruolo') = 'direttore' then
lv_predicate:=''; -- allow all access
elsif sys_context('test_prova','ruolo') = 'dottore' then
lv_predicate:='tipo_visita=''ricovero''';
elsif sys_context('test_prova','ruolo') = 'infermiere' then
lv_predicate:='tipo_visita=''visita''';
else
lv_predicate:='1=2'; -- block access
end if;
return lv_predicate;
end;
end;
/

```

5.2.4 DBMS_RLS package

Una volta definita la funzione è necessario registrarla con il database e specificatamente con la tabella che deve essere controllata.

```
begin
```

```

dbms_ols.add_policy(
object_schema => 'test',
object_name => 'paziente',
policy_name => 'test_prova_policy',
function_schema => 'test',
policy_function => 'test_policy.test_predicate',
statement_types => 'select, insert, update, delete',
update_check => TRUE,
enable => TRUE);
end;
/

```

La politica `test_prova_policy` comincia a monitorare la tabella “paziente” per qualsiasi `select`, `insert`, `update` o `delete` eseguita su quella specifica tabella (`statement_types`).

5.2.5 Test

A questo punto, dopo aver seguito passo-passo tutti i passaggi qui sopra riportati possiamo passare al test della nostra politica di sicurezza. Innanzitutto accediamo al database (ad es. tramite `sql*plus`) come il proprietario della tabella e visualizziamo il contenuto della tabella `paziente`:

```

connect test/test

select * from paziente;

no rows selected

```

Come possiamo osservare dal risultato, l’utente in questione non vede nessun record della tabella. Per quale motivo?. La risposta è semplice. Non è ancora stato impostato il ruolo dell’utente in sessione. Impostiamo quindi un ruolo:

```

exec set_test_context.s_infermiere;

```

```

SQL> column nome format a10
SQL> column cognome format a10
SQL> column data format a10
SQL> column tipo_intervento format a10
SQL> select * from paziente;

```

ID_PAZIENTE	NOME	COGNOME	DATA	TIPO_INTER
1	Paolo	Rossi	01-GIU-04	visita
3	Alessio	Fumagalli	14-FEB-04	visita

Lo stesso test può essere effettuato impostando prima il ruolo dottore e successivamente il ruolo direttore (s_dottore e s_direttore).

Abbiamo potuto osservare che in base al ruolo dell'utente la stessa tabella può restituire risultati differenti senza impostare esplicitamente la clausola "where". Questo scopo viene raggiunto aggiungendo automaticamente una condizione in ogni statement sql.

Capitolo 6

Security Check-List

Obiettivo di questa sezione è quello di mettere in evidenza alcune linee guida che un amministratore (soprattutto uno alle prime armi) dovrebbe seguire per rendere il sistema il più sicuro possibile.

I punti a seguire possono essere pertanto un primo aiuto alla configurazione **post-installazione**:

6.1 Bloccare gli utenti di default

In fase di installazione Oracle crea un numero prefissato di utenti di default (può variare a seconda della versione), è vivamente consigliato bloccarli. Il DBCA (Database Client Administration Tool) blocca automaticamente gli utenti di default eccetto i seguenti: SYS, SYSTEM, SCOTT, DBSNMP, OUTLN, 3utenti JSERV (vedi documentazione della propria versione installata). Ma ciò non avviene se l'installazione viene effettuata manualmente. In quest'ultimo caso quindi non bisogna dimenticarsi di bloccarli, soprattutto se inutilizzati, altrimenti possono essere sfruttati per conseguire l'accesso non autorizzato ai dati o interrompere le operazioni del database oltre ad essere un enorme rischio per l'integrità del database:

```
ALTER USER username ACCOUNT LOCK;
```

6.2 Cambiare le password degli utenti di default

Cambiare immediatamente le password degli amministratori: SYS e SYSTEM (change_on_install e manager). Cambiare le password di tutti gli altri utenti.

```
ALTER USER username IDENTIFIED BY {new_password};
```

6.3 Controllo delle password

È raccomandabile che siano applicate le regole base per la gestione delle password e che sia richiesto agli utenti del database di cambiare periodicamente la propria password. Utilizzare i profili.

6.4 Proteggere il data dictionary

È importante proteggere il data dictionary poiché gli utenti con un privilegio di sistema “ANY” potrebbero potenzialmente leggere, eseguire, o modificare oggetti nel data dictionary. È possibile restringere i privilegi ponendo a **false** il parametro 07_DICTIONARY_ACCESSIBILITY nel file di configurazione init.ora. Per la versione 9i è false di default. In questo caso si permette solo ad utenti sysdba di gestire il data dictionary.

6.5 Privilegi e ruoli

Evitare di concedere privilegi quali SELECT/INSERT ANY TABLE. Usare i ruoli per poter gestire meglio i privilegi. Non concederli a specifici utenti. I privilegi di sistema dovrebbero essere concessi esclusivamente ad amministratori o amministratori della sicurezza.

6.6 Restringere l'accesso alla rete

1. Utilizzare un firewall;
2. Prevenire l'amministrazione non autorizzata del listener impostando una password ben definita.

Per impostare la password, al prompt `lsnrctl`:

```
set password xxxxxxxxxx;
```

È possibile inoltre prevenire l'amministrazione non autorizzata del listener oracle impostando ad 'on' il parametro seguente nel file `listener.ora`:

```
ADMIN_RESTRICTIONS_listener_name=ON;
```

3. Verifica dell'indirizzo ip:
permettere o negare l'accesso ai processi del server Oracle da parte di specifici client (ip). Configurare i seguenti parametri nel file `protocol.ora`:

```
tcp.validnode_checking=YES  
tcp.excluded_nodes={list of ip address}  
tcp.invited_nodes={list of ip address}
```

4. Password criptate:
impostare a `TRUE` i parametri `ORA_ENCRYPT_LOGIN` sul client e `DBLINK_ENCRYPT_LOGIN` sul server per permettere una trasmissione "criptata" delle password.
5. Utilizzare Advanced Security (enterprise edition)

6.7 Auditing

Attivare l'auditing per favorire un controllo più agevole della sicurezza del sistema specialmente quando è notevole l'importanza dei dati conservati nel database. Come già detto precedentemente può essere notevole l'impatto sulle prestazioni (incremento della dimensione della tabella

SYS.AUD\$). L'amministratore ha il compito di decidere a quale dettaglio effettuare il monitoraggio. Si può cominciare con un monitoraggio generale seguito successivamente da un controllo più specifico quando l'origine viene individuata. Nonostante ciò sarebbe buona norma controllare almeno le seguenti situazioni: connessioni, operazioni del DBA (se non è il responsabile della sicurezza), attività sull'audit_trail. È inoltre possibile restringere il monitoraggio delle operazioni ad un singolo utente. Per attivare l'auditing è necessario configurare il seguente parametro audit_trail nel file di configurazione init.ora, di **default** impostato a **none**:

```
AUDIT_TRAIL={DB|OS|NONE}
```

6.8 Backup

Definire, usare e verificare una completa strategia di backup.

6.9 Applicare patch di sicurezza

Periodicamente visitare il seguente indirizzo per valutare possibili vulnerabilità ed applicare le opportune patch:

<http://otn.oracle.com/deploy/security/alert>

Capitolo 7

Conclusioni

La conoscenza e la consapevolezza sono alla base della sicurezza.

Garantire la sicurezza assoluta del sistema e dei dati che conserviamo nel nostro database può cadere nell'ipocrisia.

Nonostante ciò è nostro dovere fare in modo che i problemi di sicurezza possano, nel limite del possibile, essere sempre rilevati e risolti.

È ormai chiaro come siano più di una le operazioni indispensabili per rendere un sistema “sicuro” e come sia importante non tralasciare nessun dettaglio.

Oracle richiede una configurazione (orientata alla sicurezza) sin dalla fase iniziale di installazione (ad es. cambiare le password e/o bloccare gli utenti non autorizzati). Successivamente è necessario monitorare con continuità le azioni che compiono gli utenti (ruoli, privilegi, auditing, ... ecc) e configurare una comunicazione sicura in rete. Inoltre è opportuno (quasi obbligatorio) prevenire, mediante strategie di backup, perdite accidentali dei dati.

Infine è responsabilità dell'amministratore restare al passo con eventuali aggiornamenti (patch rilasciate dal produttore) contro vulnerabilità riscontrate nel sistema.

Un ulteriore approfondimento di questa tesi può coinvolgere un'analisi della suite Advanced Security e quindi delle tematiche quali integrità e riservatezza dei dati (crittografia).

Ringraziamenti

Un sentito ringraziamento al mio relatore Prof. Renzo Orsini per la disponibilità e l'attenzione concessami.

Ringrazio inoltre i miei genitori Matteo e Antonina che mi hanno permesso di concludere questo primo ciclo di studio universitario.

Appendice A

DBMS_RLS package

Il package DBMS_RLS di Oracle offre un insieme di procedure per amministrare le politiche di sicurezza. Utilizzando questo package è possibile aggiungere, eliminare, abilitare, disabilitare oppure aggiornare le politiche create:

Program	Description
ADD_POLICY procedure	Creates or registers a fine grained access control policy for a table or view
DROP_POLICY procedure	Drops a fine grained access control policy from a table or view
ENABLE_POLICY procedure	Enables or disables a fine grained access control policy
REFRESH_POLICY procedure	Causes all the cached statements associated with the policy to be reparsed

Tabella A.1: DBMS_RLS programs


```

policy_function IN VARCHAR2,
statement_types IN VARCHAR2 := NULL,
update_check   IN BOOLEAN := FALSE,
IN BOOLEAN := TRUE);

enable

-- drop_policy - drop a row level security policy from a table or view
--
-- INPUT PARAMETERS
-- object_schema - schema owning the table/view, current user if NULL
-- object_name   - name of table or view
-- policy_name   - name of policy to be dropped

PROCEDURE drop_policy(object_schema IN VARCHAR2 := NULL,
object_name   IN VARCHAR2,
policy_name   IN VARCHAR2);

-- refresh_policy - invalidate all cursors associated with the policy
-- if no argument provides, all cursors with
-- policies involved will be invalidated
--
-- INPUT PARAMETERS
-- object_schema - schema owning the table/view, current user if NULL
-- object_name   - name of table or view
-- policy_name   - name of policy to be refreshed

PROCEDURE refresh_policy(object_schema IN VARCHAR2 := NULL,
object_name   IN VARCHAR2 := NULL,
policy_name   IN VARCHAR2 := NULL);

-- enable_policy - enable or disable a security policy for a table or view
--
-- INPUT PARAMETERS
-- object_schema - schema owning the table/view, current user if NULL
-- object_name   - name of table or view
-- policy_name   - name of policy to be enabled or disabled
-- enable       - TRUE to enable the policy, FALSE to disable the policy

PROCEDURE enable_policy(object_schema IN VARCHAR2 := NULL,

```

```
object_name IN VARCHAR2,
policy_name IN VARCHAR2,
enable      IN BOOLEAN);

END dbms_ols;
/
DROP PUBLIC SYNONYM dbms_ols
/
CREATE PUBLIC SYNONYM dbms_ols FOR sys.dbms_ols
/

--
-- Grant execute right to EXECUTE_CATALOG_ROLE
--
GRANT EXECUTE ON sys.dbms_ols TO execute_catalog_role
/
```

Appendice B

DBMS_SESSION package

Il package DBMS_SESSION di Oracle associato con la procedura “set_context” permette di impostare il valore per un attributo in un contesto:

Parameter	Description
namespace	The name of the context
attribute	The attribute name
value	The value to be assigned to that attribute in the current session

Tabella B.1: set_context parameters

Bibliografia

- [1] *<http://www.oracle.com>*
- [2] George Koch e Kevin Loney *La guida completa Oracle 8* McGraw-Hill, 1998.
- [3] Marlen Theriault *Oracle Security* O'Reilly,1998
- [4] Pete Finnigan *Introduction to Simple Oracle Auditing by Pete Finnigan*, URL=<http://www.securityfocus.com/infocus/1689>.
- [5] *Oracle Backup and Recovery FAQ*,
URL=<http://www.orafaq.com/faqdbabr.htm>
- [6] *Securing Oracle Network Traffic*,
URL=<http://www.dbspecialists.com>
- [7] *La sicurezza delle banche dati nel nuovo codice della privacy*,
URL=<http://www.forumpa.it>
- [8] *Deploying Fine-Grained Access Control*,
URL=http://www.hk8.org/old_web/oracle/guide8i/ch08_01.htm
- [9] *Internet Security With Oracle Row-Level Security*,
URL=<http://www.interealm.com/robby/technotes/8i-rls.html>
- [10] *Keeping Information Private with VPD*,
URL=http://otn.oracle.com/oramag/oracle/04-mar/o24tech_security.html

- [11] *Authenticating Users to the Database,*
URL=<http://www.lc.leidenuniv.nl/awcourse/oracle/network.920/a96582/authuser.htm#1008116>
- [12] *Legge 31 dicembre 1996 n. 675 Tutela delle persone e di altri soggetti
rispetto al trattamento dei dati personali,*
URL=http://www.interlex.it/testi/l675_96.htm
- [13] *Viaggio al centro di Oracle: il Data Dictionary,*
URL=<http://www.programmazione.it>
- [14] *A security check-list for Oracle,*
URL=http://otn.oracle.com/deploy/security/oracle9i/pdf/9i_checklist.pdf