

How to Setup & Use Toad in a Secure Environment

Let's face it - it's a rough and scary computer world we live in today. We have people routinely losing notebook computers with massive local copies of confidential records, computer hackers aggressively attempting to break into restricted systems as a marquee of accomplishment (and as a possible way to land a high paying security industry job), and of course all the new regulatory laws to try to stem the pandemic. So it's up to the DBA to stand as a first bastion of defense in protecting their company's most valuable asset – the data within their databases.

Database security threats can initially be broken down into two categories: external and internal. This paper will focus on the latter issue as it relates to ***Toad*** users. Like many software packages which access a database, Toad was never designed to function as your company's sole means of database security. Toad can very easily be configured to work within whatever security practices your company chooses to implement. However Toad in of and by itself can not, and should not, be your only means of database security. That may sound funny at first – but see what you think when you reach the end of this paper ☺

Remember that Toad offers a plethora of user options – some of which are very clearly security related – but most of which are set by the individual users. Therefore, Toad user education and training is paramount to success – with ongoing monitoring for compliance. However with minimal DBA setup and user efforts Toad can function well within even the most secure database environments. Below are ten easy recommendations that have proven quite effective. Note that there are often multiple ways to accomplish any given advice – this paper simply identifies an area and offers simple and centralized solutions.

#1 Follow Oracle's Security Checklist

The Oracle 9i Administrator's and 10g Database Security Guides offer very handy security checklists. I've [highlighted in blue](#) some of the more commonly occurring security compromises.

1. Install only what is required off the Oracle client CD
2. Lock and expire default user accounts
 - SYS
 - SYSTEM
 - SCOTT
 - DBSNMP
 - OUTLN
 - All those installed by DBCA as samples
3. Change default user password
4. Enable dictionary protection

- O7_DICTIONARY_ACCESSIBILITY = FALSE
5. Practice principle of least privilege
 - Grant necessary privileges only
 - Revoke unnecessary privileges from PUBLIC
 - Restrict permissions on run-time facilities
 6. Enforce access controls effectively
 - REMOTE_OS_AUTHENT = FALSE
 7. Restrict operating system access
 - Limit the number of operating system users
 - Limit the privileges of the operating system accounts on the Oracle server to the least required for the user
 8. Restrict network access
 - Utilize a firewall
 - Never poke a hole through a firewall
 - Prevent unauthorized administration of the Oracle Listener
 - Check network IP addresses
 - Encrypt network traffic
 - Harden the operating system
 9. Apply all security patches and work-arounds
 10. Report exceptions and/or issues to Oracle support

However look again at item #2 about locking built-in accounts. Yet numerous shops still routinely use SYS and SYSTEM for connecting to do DBA work! I've yet to see a Toad login screen at customer sites that don't have for either or both across their databases ☹

#2 Implement Oracle Roles

The first and best place to enforce database security is via Oracle itself – because Toad only allows users to do to the database whatever the DBA has granted to them. Contrary to popular misconception, Toad does not circumvent database security and permit people to do more than this. Sure, Toad has a pretty GUI that eliminates users needing to know cryptic syntax – but again, Toad only lets users do what the DBA has granted them. To quote the poetic Toad (like Poe's Raven) – “Users can do only this, and nothing more”.

So the first advice is to explicitly create your own database roles for the various classes of Toad users, and then assign all Toad users to only those roles. This is really nothing more than best practice for the above Oracle security checklist item #5 – practicing principle of least privilege. For example, here are 5 very basic Oracle role recommendations for Toad use:

- TOAD_DBA_SENIOR
- TOAD_DBA_JUNIOR
- TOAD_PLSQL_DEV_SENIOR
- TOAD_PLSQL_DEV_JUNIOR

- TOAD_READ_ONLY_DATA

The rationale is quite simple – you should not rely upon any pre-canned Oracle roles for a truly secure environment, as you may not be sure what they grant or when they’ll change. Let’s review three commonly abused pre-canned Oracle roles: CONNECT, RESOURCE, and DBA. Here’s a brief table to remind you just what you’re granting to database users when you grant these roles.

	8i	9i	10g
CONNECT	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW	CREATE SESSION
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	All system privileges WITH ADMIN OPTION	All system privileges WITH ADMIN OPTION	All system privileges WITH ADMIN OPTION

How many of you were aware of the change with CONNECT in 10g? And in some cases, 9i release two patches had the same change! How many of you knew that connect permits database links? Many shops where I’ve worked strictly forbid or closely manage database links, so again using CONNECT would not have been advisable. For those who may have forgotten Oracle 6, these words were simply the SQL syntax options when granting user privileges. They were basically provided and maintained in later Oracle releases for backwards compatibility. So it’s far past time to create, manage and use your own roles.

Here’s an example:

```
CREATE ROLE TOAD_PLSQL_DEV_SENIOR NOT IDENTIFIED;

GRANT CREATE SESSION TO TOAD_PLSQL_DEV_SENIOR;
GRANT DEBUG ANY PROCEDURE TO TOAD_PLSQL_DEV_SENIOR;
GRANT DROP ANY TRIGGER TO TOAD_PLSQL_DEV_SENIOR;
GRANT CREATE ANY TRIGGER TO TOAD_PLSQL_DEV_SENIOR;
GRANT DROP ANY PROCEDURE TO TOAD_PLSQL_DEV_SENIOR;
GRANT ALTER ANY PROCEDURE TO TOAD_PLSQL_DEV_SENIOR;
GRANT CREATE ANY PROCEDURE TO TOAD_PLSQL_DEV_SENIOR;

GRANT TOAD_PLSQL_DEV_SENIOR TO BERT;
```

#3 Implement Oracle Profiles

This one topic alone will probably incur the most negative user feedback. Toad is a great tool – and it's not uncommon for people to use it from sun-up to sun-down. But leaving any software running all the time on your desktop is inherently unsafe. Even with screen savers, it's still not entirely safe. Furthermore, we routinely have people who leave Toad running unattended overnight and over entire weekends – again, even though this too is highly insecure. There is no mechanism within Toad to force idle people to logout – but Oracle provides this capability and DBA's should strongly consider implementing this.

The problem here is that while this is good security advice – certain older versions of Toad behave rather poorly when left open, lose their connection, and then users try to resume working. Sometimes they get a series of error dialogs (one per open connection) and sometimes it requires using task manager to restart Toad. But that's not really a Toad problem per se – since restarting Toad generally fixes all the issues. But Toad 9.0 will have some additional fixes for any remaining quirks here – so that's yet another good reason to be on maintenance and to stay current with your Toad versions 😊

Now the real problem is user complaints (and do expect them). It's just human nature to resist change – and especially for things like this. Simply tell them to shutdown their PC's or applications over nights and weekends. This practice merely requires users to shut down and restart applications between long periods of inactivity to improve security. If they resist too hard, then just threaten to sick the security compliance people on them.

The other thing profiles can help with for very little additional effort is basic management of database resources for fair use practice. In the example below, I've further limited this class of Toad users to eight concurrent connections. Because Toad makes it so easy to do so many things at once – it's not unusual to see Toad users connected multiple times, and sometimes without even realizing it. And with Toad 9.0's new threading model – I expect to see this exasperated even further. But much more comprehensive recommendations regarding fair resource usage best practices will be handled later (i.e. resource groups).

Here's an example:

```
CREATE PROFILE TOAD_PLSQL_DEV_SENIOR
LIMIT
  SESSIONS_PER_USER 8
  IDLE_TIME 60;

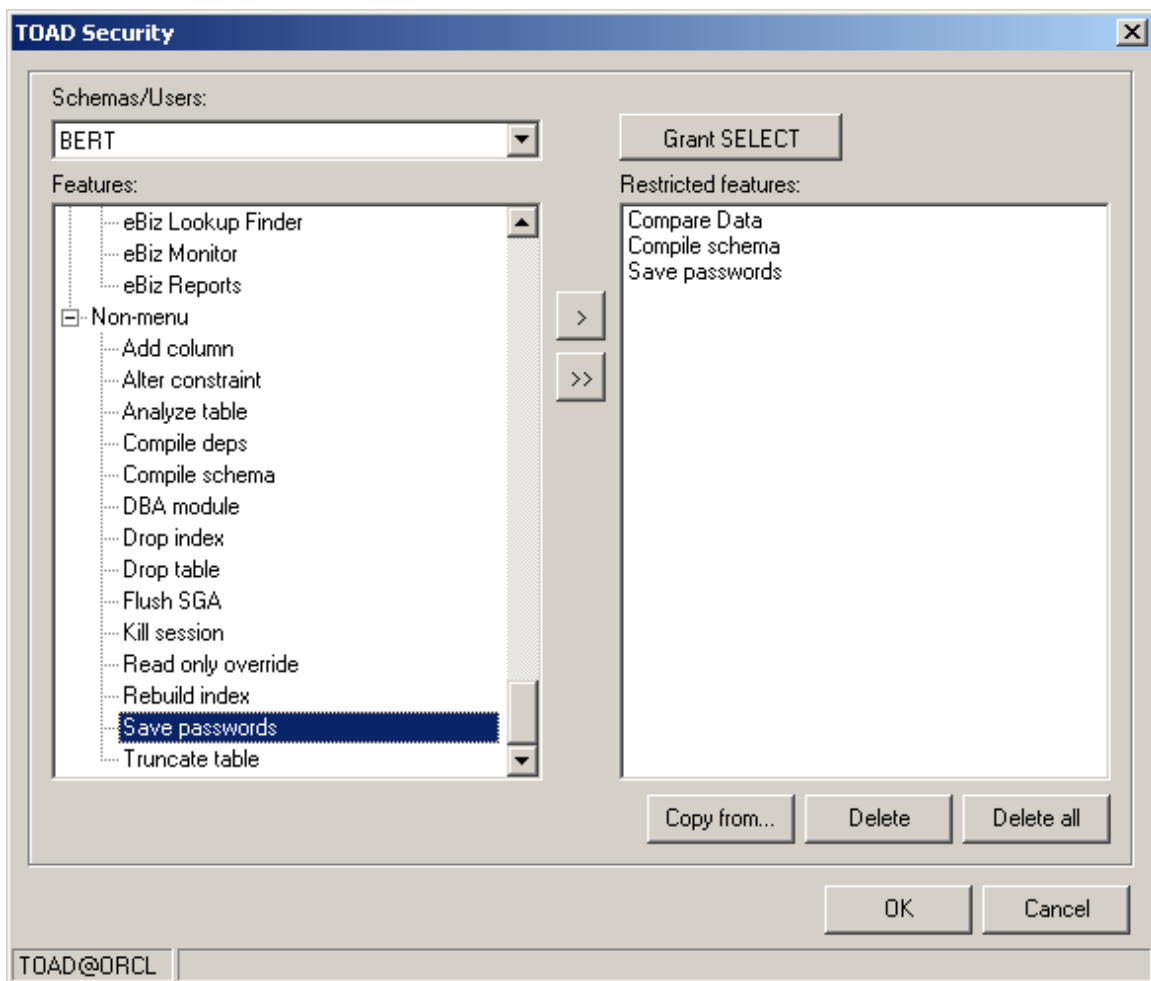
ALTER USER "BERT" PROFILE TOAD_PLSQL_DEV_SENIOR;
```

NOTE – this requires the following init.ora parameter:

- RESOURCE_LIMIT = TRUE (default is FALSE)

#4 Implement Toad Security

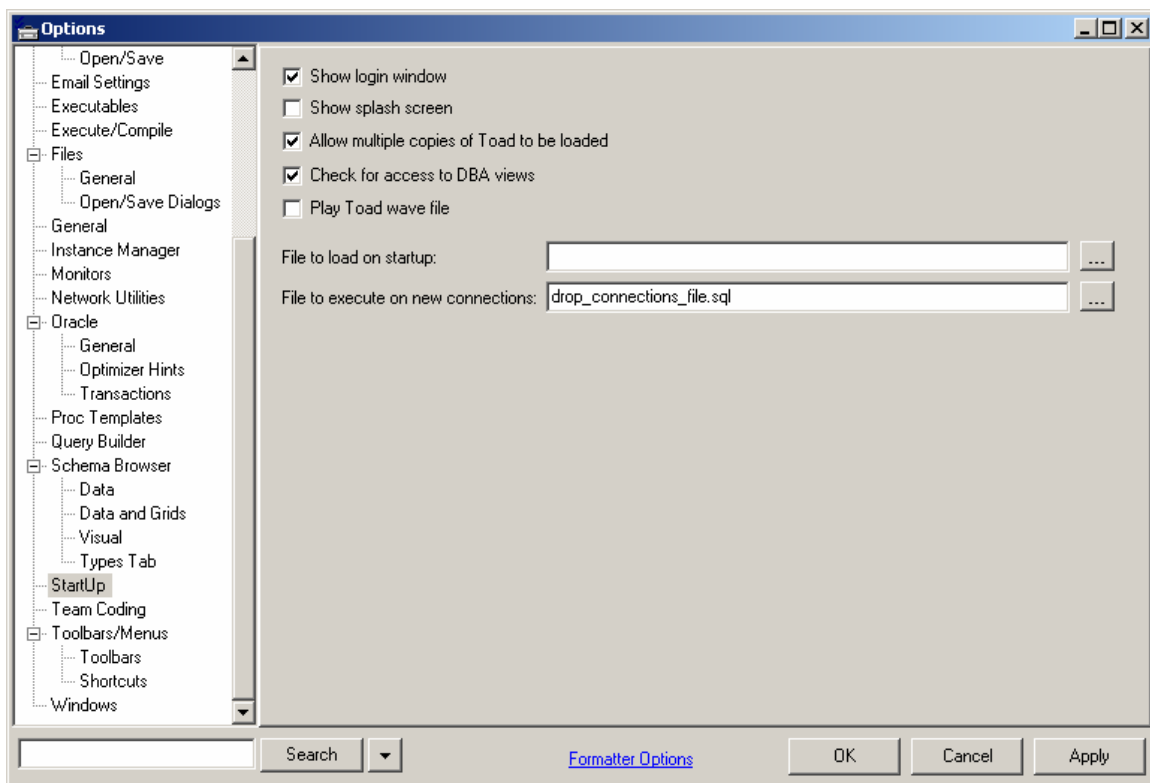
OK – let’s assume that you’ve done recommendations #2 and #3, and now simply want to refine these accomplishments a little further – but in much more Toad specific terms. Just for the sake of argument, let’s say that you want to eliminate the use of certain expensive database operations that Toad’s GUI just makes too easy and thus tempting to use – and that you further want to prevent those users from utilizing Toad’s optional ability to save passwords on their PC (the real security threat – although Toad 9.0 uses a very complex encryption algorithm that also ties the CONNECTIONPWDS.INI file to the user’s PC). If we simply use Toad’s Server Side Object Wizard to create the TOAD schema, then we can accomplish these goals as shown in the screen snapshot below for setting Toad Security settings:



Toad security allows you to easily augment your basic database security – but to do so in some very Toad relevant and specific terms. You’re essentially disabling or turning off Toad features for users regardless of their database grants – and this includes both menu driven and non-menu driven features. Once again we are doing nothing more here than the best practice for the above Oracle security checklist item #5 – practicing the principle

of least privilege (merely now at the application level rather than just the database). Furthermore, you should follow the Oracle Security checklist item #2 – lock the built-in accounts - for your Toad schema as well. Otherwise crafty people will be able to very easily circumvent these security settings.

Note – some customers desire to eliminate the ability to save Toad password file without having to use Toad Security. Right now the only work around is to have a batch file for launching Toad that automatically deletes the CONNECTIONPWDS.INI file, and having all users launch Toad via that script (rather than simply running the TOAD.EXE). You also could write a special startup SQL script to execute every time Toad makes a connection (and which executes a host command to delete the CONNECTIONPWDS.INI file), as shown below. But neither of these alternate techniques is very intrinsically secure.



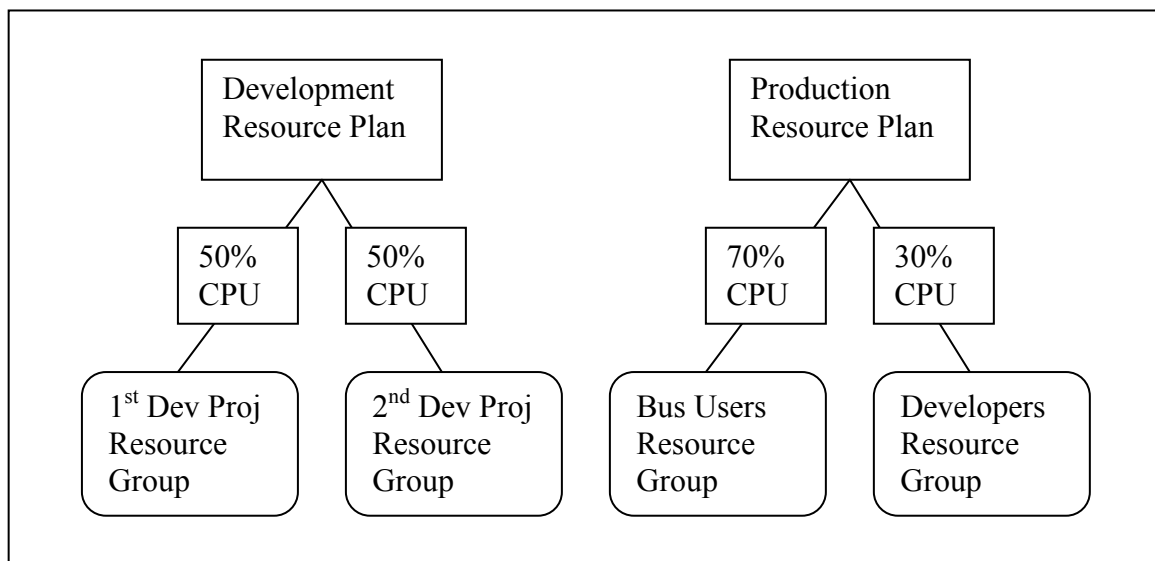
#5 Implement Resource Groups

This next recommendation probably has the least real security related value – but it's worth mentioning nonetheless since it builds on what we've discussed so far, and also leads to the next topic – which will be much more clearly security oriented. Other than possibly helping to prevent or minimize the effects of certain DOS (Denial of Service) type attacks against your database – it's really much more of a fair resource utilization control technique. Note too that there are no SQL commands per se for managing this.

Therefore unless you have the Toad DBA Module, you're going to need to learn some pretty obscure pre-canned PL/SQL packages ☹

Ever have some Toad users that you need to limit in terms of fair resource consumption, but with more granularity than Oracle Profiles? Think about those users who love to kick off Cartesian join queries or fetch multi-million row queries back to their PC on a regular basis. These guys are being unfair resource hogs – and once again Toad is making it too easy and non-obvious to them. It would be far too easy therefore for one user's action to have a negative impact on another user in terms of time – especially when Toad makes users fairly. Another way to say this is that as desktop PC's have increased in their speed and thus capabilities, so has their users' demands against the database servers they work with. So sometimes we need to reign-in certain Toad power users lest they run amuck.

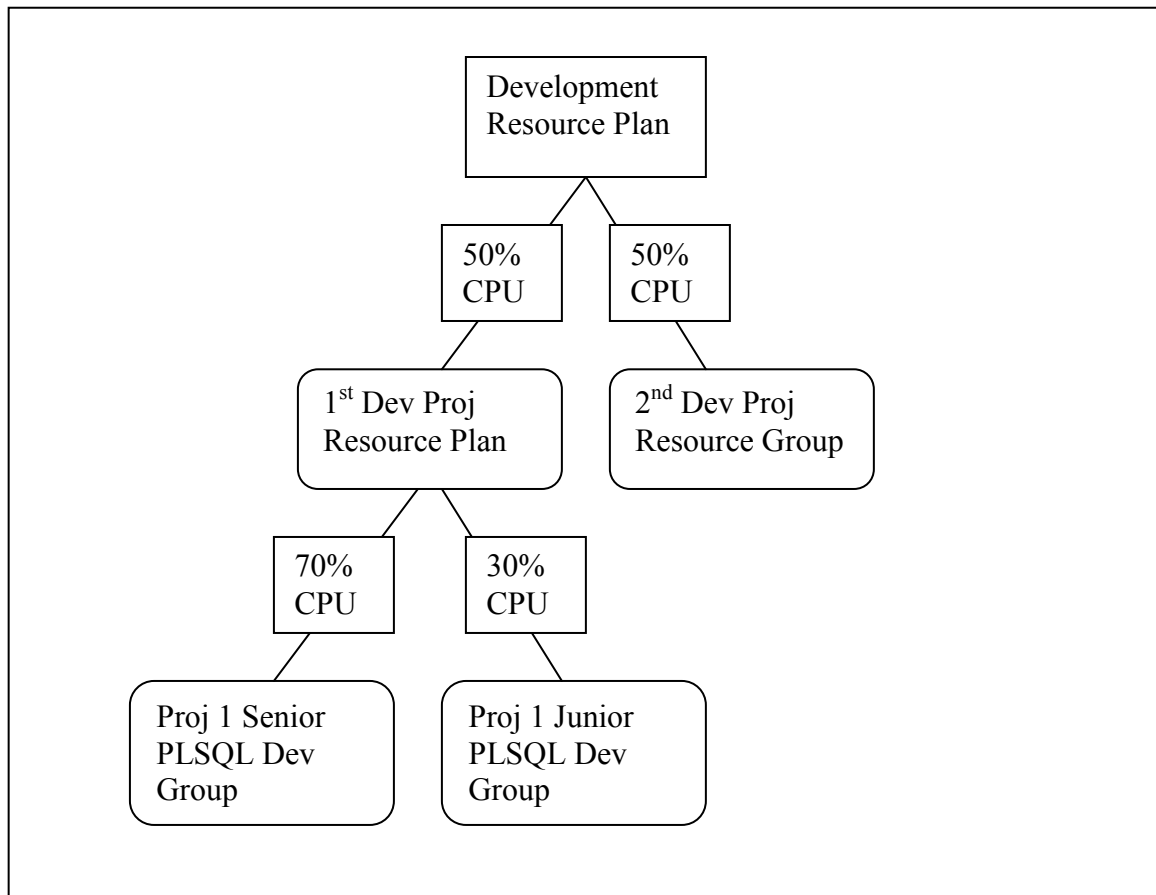
Let's look at Oracle Resource Groups. Suppose we have two development projects sharing a common database server and cannot afford for one group to use an unfair percentage of the shared servers' resources to the detriment of the other group's work and project schedule. Or suppose a production database server, maybe business users and processes need to have higher priority than Toad users who are developers. So here are the simple solutions to those two scenarios:



Two other aspects make resource management a worthwhile feature to use with Toad.

First, you can create hierarchies of plans. So look what we can do for the development machine example above reworked for further details below. We can now define that on project #1, senior developers should get slightly more resources than junior developers. Whereas with project #2, everyone gets the same limits. So let's now return to the idea of a DOS service security attack. Suppose someone makes such an attack using a username and password for a junior level developer from project #1, the resource limits will stop that attack from consuming more than 15% of the server's resources. That might make the difference between a slow machine that needs a reboot and being able to work on in

spite of such attacks. Plus, it also can be used to minimize Toad users from unknowingly stressing the database server from their killer PC's – where they multi-task to an extreme and ask for ever growing demands from the database server.



And second, you can define resource consumption limits within a group that automatically cause the users' processes to switch to another resource group. So in the example above, we could define project #1's senior developers resource group such that any process that consumes more than some threshold downgrades to a junior status job – and thus gets far less CPU time for that point forward.

Finally, almost everything you could do above with Profiles you can do much better with Resource Plans and Groups. For example both execution and idle time limits can be best managed via this technique – and with additional options and features. But once again, unless you have the Toad DBA Module and its screens for this – then you'll need to learn some overly-complex PL/SQL package calls like this example:

```
begin
  sys.dbms_resource_manager.clear_pending_area();
  sys.dbms_resource_manager.create_pending_area();
  sys.dbms_resource_manager.create_consumer_group (
    consumer_group => 'TOAD_PLSQL_NICE');
  sys.dbms_resource_manager.submit_pending_area();
```

```

end;

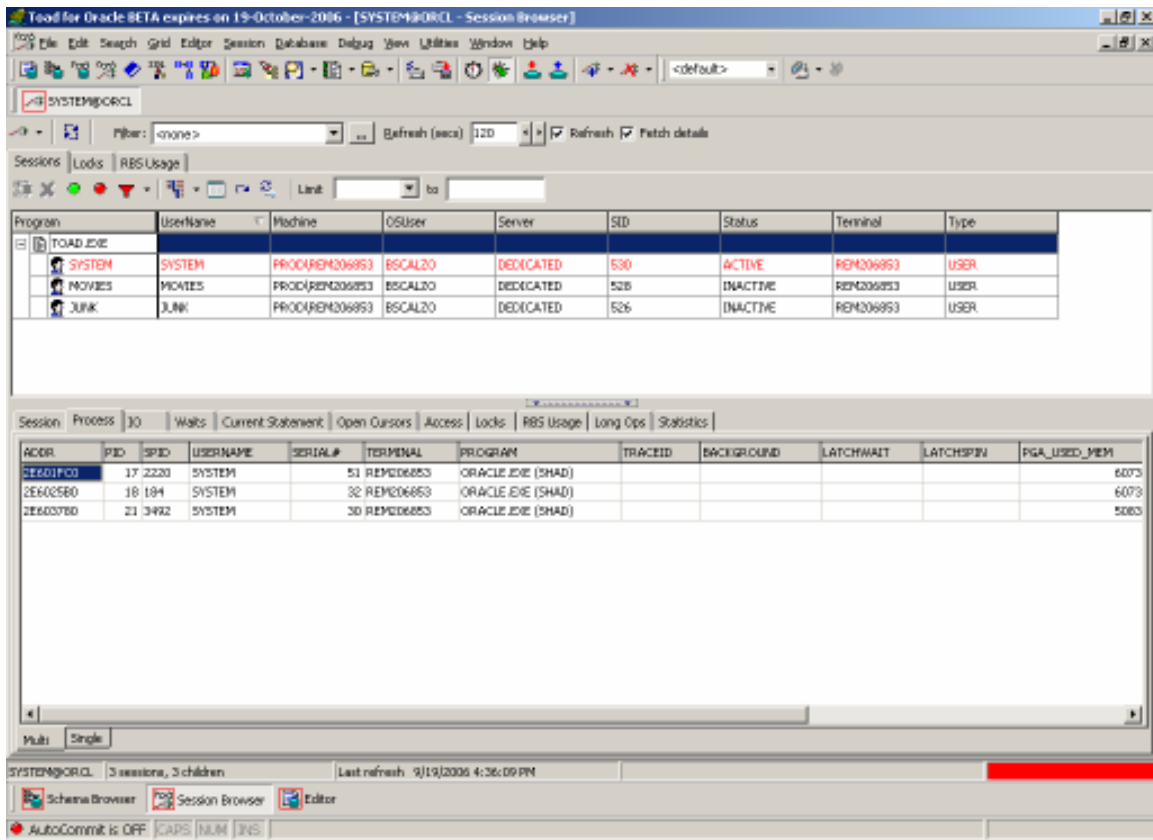
begin
sys.dbms_resource_manager.clear_pending_area();
sys.dbms_resource_manager.create_pending_area();
sys.dbms_resource_manager.create_plan (
    plan            => 'TOAD_PLSQL_NICE_1'
    ,cpu_mth        => 'EMPHASIS'
    ,active_sess_pool_mth => 'ACTIVE_SESS_POOL_ABSOLUTE'
    ,parallel_degree_limit_mth => 'PARALLEL_DEGREE_LIMIT_ABSOLUTE'
    ,queueing_mth   => 'FIFO_TIMEOUT');
sys.dbms_resource_manager.create_consumer_group (
    consumer_group => 'TOAD_PLSQL_NICE'
    ,comment        => "");
sys.dbms_resource_manager.create_plan_directive (
    plan            => 'TOAD_PLSQL_NICE_1'
    ,group_or_subplan => 'TOAD_PLSQL_NICE'
    ,cpu_p1         => 30
    ,switch_estimate => FALSE
    ,max_est_exec_time => 3000
    ,parallel_degree_limit_p1 => 1 );
sys.dbms_resource_manager.create_plan_directive (
    plan            => 'TOAD_PLSQL_NICE_1'
    ,group_or_subplan => 'OTHER_GROUPS'
    ,cpu_p1         => 70
    ,switch_estimate => FALSE );
sys.dbms_resource_manager.submit_pending_area();
end;
/

```

#6 Proactively Manage Toad Connections

There's a common security need to sometimes restrict database access from either certain users, class of users, or applications. For example, it's not uncommon for an Oracle DBA to issue an ALTER SYSTEM ENABLE RESTRICTED SESSION command right before doing critical DBA tasks that require a quiet database. While this command will prevent future logins, the DBA may then also need to kill already established sessions. The point is that there is a mechanism within the database for doing tasks like this. So what about Toad? How can we implement security controls to prevent Toad user logons or to limit the number of users to our ordered license count (in an attempt to comply with known licensing limits)?

Well actually it's not too hard since Toad nicely performs an Oracle database application registration (DBMS_APPLICATION_INFO.SET_MODULE) of itself whenever a Toad user connects to the database. Thus we automatically have a very easy way to identify and thus manage Toad users. Look at the screen snapshot below of Toad's Schema Browser. See how my 3 Toad users show up grouped by the EXE name. Now even if tricky users attempt to circumvent this by renaming their EXE file, note how the Module has been set to Toad. Did you happen to note that I broke a golden rule? I'm logged in as SYSTEM ☺



So how can we prevent Toad users from creating new connections? Because we can clearly kill the session shown above, but those pesky users will just try to reconnect. Below is a simple database level trigger to accomplish this task:

```
connect sys/mgr as sysdba
```

```
create or replace trigger restrict_toad after logon on database
```

```
declare
```

```
  v_sid number;
```

```
  v_module varchar2(30);
```

```
  v_isdba varchar2(10);
```

```
begin
```

```
  execute immediate 'select distinct sid from sys.v_$mystat' into v_sid;
```

```
  execute immediate 'select module from sys.v_$session where sid = :b1' into v_module using
```

```
  v_sid;
```

```
  if upper(v_module) LIKE 'TOAD%' then
```

```
    select sys_context('userenv','ISDBA') into v_isdba from dual;
```

```
    if v_isdba = 'FALSE' then
```

```
      raise_application_error (-20001,'TOAD Access for non DBA users restricted',true);
```

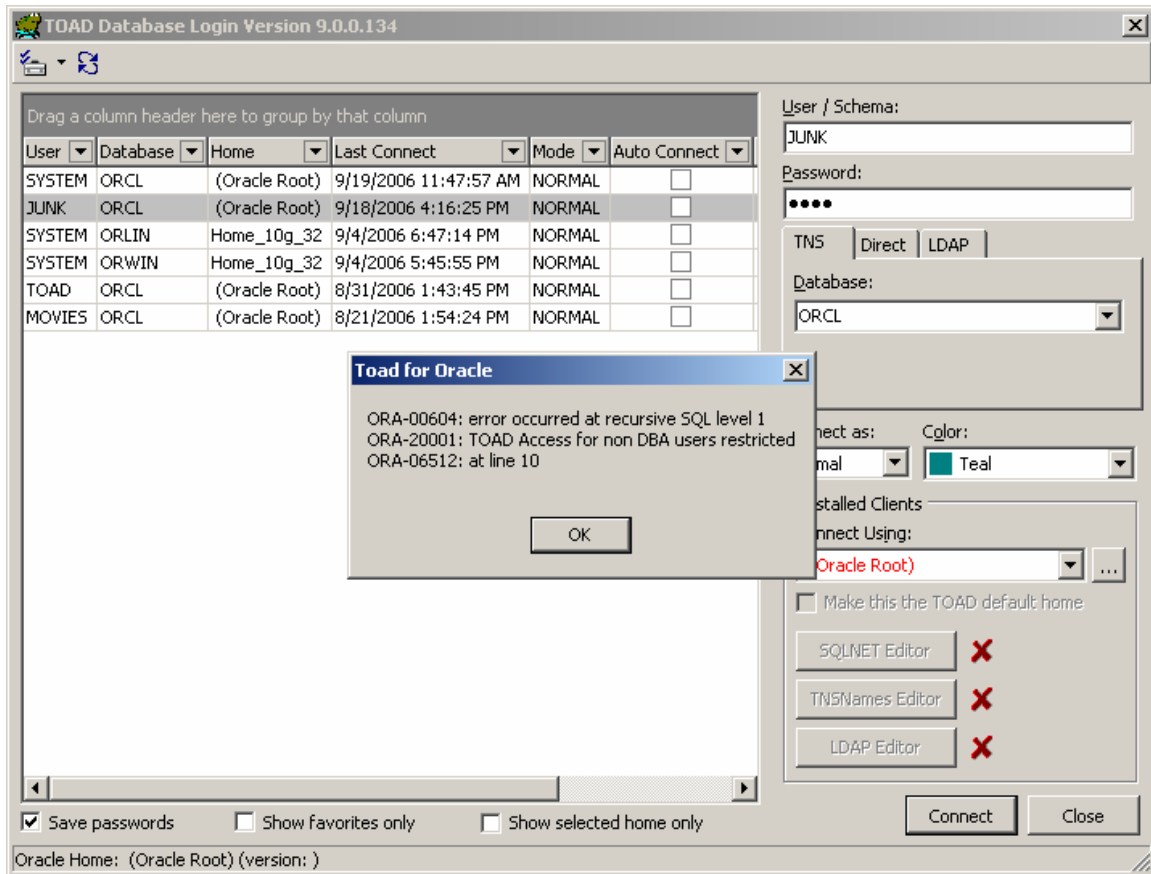
```
    end if;
```

```
  end if;
```

```
end;
```

```
/
```

This code functions much like the restricted session database command – in that it only allows Toad users who are DBA's to connect. When a non-DBA user tries to connect, they get the error message as shown below.



So let's return to the other proposed scenario – not allowing more than two concurrent Toad users since we only bought two licenses (of course the better solution is to simply buy more Toad licenses). Here's the code to do that:

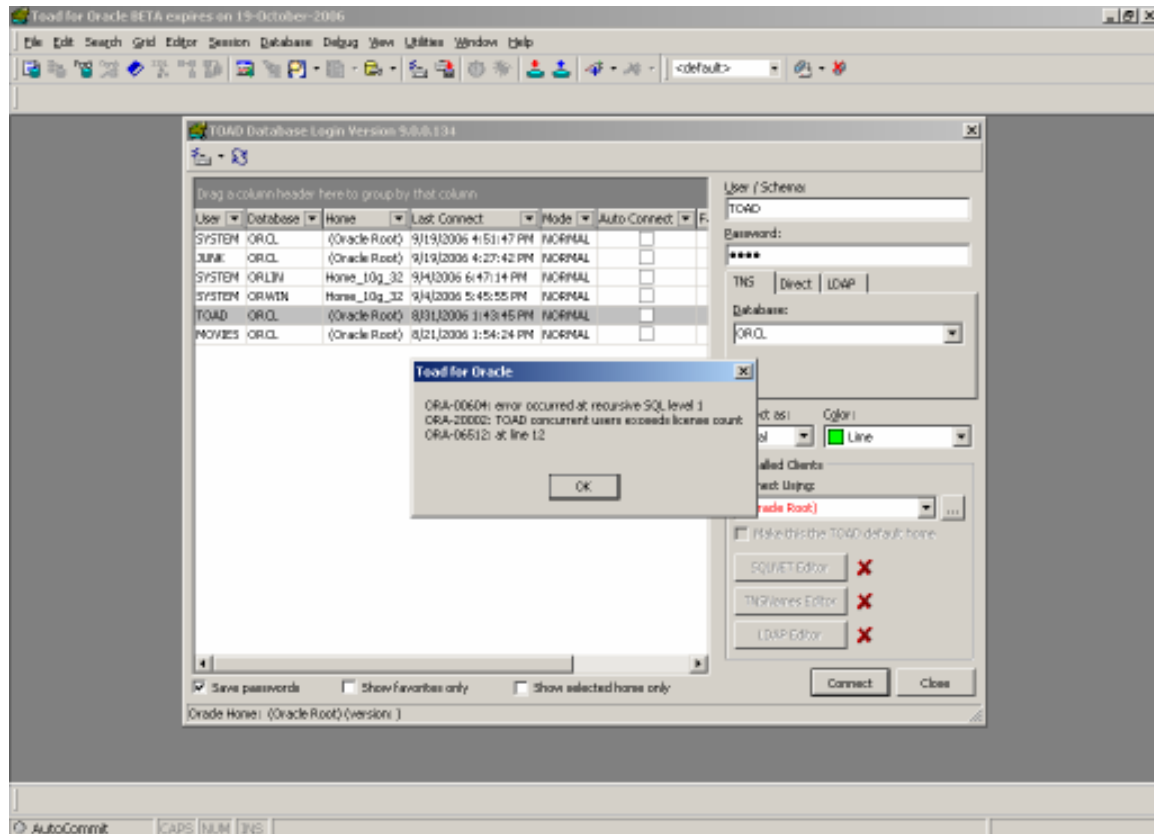
```

connect sys/mgr as sysdba

create or replace trigger limit_toad after logon on database
declare
  v_sid number;
  v_module varchar2(30);
  v_toad_count number;
begin
  execute immediate 'select distinct sid from sys.v_$mystat' into v_sid;
  execute immediate 'select module from sys.v_$session where sid = :b1' into v_module using
  v_sid;
  if upper(v_module) like 'TOAD%'
  then
    execute immediate 'select count(distinct machine) from sys.v_$session where sid <> :b1 and
  module like "TOAD%"' into v_toad_count using v_sid;
  if v_toad_count >= 2 then
    raise_application_error (-20002,'TOAD concurrent users exceeds license count',true);
  
```

```
end if;  
end if;  
end;  
/
```

But there is one major limit to this technique: it only works per database. But at least that's some progress. And it gives the nice licensing reminder message shown below:



#7 Use Password Aging and Complexity Verification

This security recommendation will probably incur the 2nd most negative user opposition (much like issue #3 above about using Oracle profiles for idle session timeouts). Toad is a superior productivity enhancer – and it's not uncommon for people to choose to save their passwords such that the connection screen takes but a quick look and a double click in order to connect to one's databases. That's exactly why it's unsafe – because anyone finding an unattended PC with Toad both installed and using saved passwords could have unfettered access to all your databases. This actually happens more than you think ☺

But even if you achieve a relatively secure stance: where profiles timeout idle sessions and users don't save passwords, a significant password security loophole yet exists – we should not rely on static (i.e. non-changing) database passwords. Furthermore, Oracle (and therefore Toad) passwords should not be set to either short nor common, and thus

easily guessable words. Nor should users be permitted to simply reuse their current password when prompted for a new one, nor to use relatively simple adjustments to the password (such as merely adding a suffix). Furthermore, passwords ought to be changed regularly and with some meaningful frequency – such as every thirty days.

Following such advice will help to mitigate the security risks in those shops who choose not to disable password saving and have users who save their passwords, because now would-be hackers watching Toad users for their passwords might not actually attempt to utilize them until after they've been changed. This technique all by itself may not sound like such a big deal, however when successfully used in conjunction with the other recommendations it nonetheless provides for an even more secure Oracle and Toad work environment. Plus like many of the earlier security recommendations, this one too is best achieved via the Oracle database itself ☺

Here's an example of a very basic password verification PL/SQL function (of course you can use as your basis either this one or the Oracle provided default contained in the script \$ORACLE_HOME/RDBMS/admin/utlpwdmg.sql):

```
CREATE OR REPLACE FUNCTION verify_password
(username  VARCHAR2,
new_password VARCHAR2,
old_password VARCHAR2)
RETURN    BOOLEAN
IS
  i      INTEGER;
  c      CHAR;
  isdigit  BOOLEAN := false;
  isletter  BOOLEAN := false;
  ispunct  BOOLEAN := false;
  differ   INTEGER;
  m        INTEGER;
BEGIN
  --Check if the password is same as the username
  IF (new_password = username) THEN
    raise_application_error(-20001, 'New password same as username');
  END IF;

  --Check for the new password = old password
  IF (new_password = old_password) THEN
    raise_application_error(-20002, 'New password same as old password');
  END IF;

  --Check for the minimum length of the password
  IF length(new_password) < 8 THEN
    raise_application_error(-20003, 'New password length less than 8 chars');
  END IF;

  --Check if the password is too common or simple
  IF NLS_LOWER(new_password) IN ('welcome', 'database', 'account', 'user',
    'password', 'oracle', 'computer', 'abcd', 'mgr', 'manager', 'tiger')
  THEN raise_application_error(-20004, 'New password too common or simple');
  END IF;
```

```

--Check if the password contains at least one
--letter, one digit, and one punctuation mark
FOR i IN 1 .. length(new_password) LOOP
  c := substr(new_password,i,1);
  IF (NOT isdigit) THEN
    isdigit := (INSTR('0123456789',c) > 0);
  END IF;
  IF (NOT isletter) THEN
    isletter := (INSTR('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ',c)
> 0);
  END IF;
  IF (NOT ispunct) THEN
    ispunct := (INSTR('!"#$%&()*+,-/;<=>?_!',c) > 0);
  END IF;
  EXIT WHEN isdigit AND isletter AND ispunct;
END LOOP;
IF (NOT isletter) OR (NOT isdigit) OR (NOT ispunct) THEN
  raise_application_error(-20005, 'Password should contain at least one digit, one character and
one punctuation');
END IF;

--Check if the password differs from the previous password by at least 3 letters
differ := length(old_password) - length(new_password);
IF abs(differ) < 3 THEN
  IF length(new_password) < length(old_password) THEN
    m := length(new_password);
  ELSE
    m:= length(old_password);
  END IF;
  differ := abs(differ);
  FOR i IN 1..m LOOP
    IF substr(new_password,i,1) != substr(old_password,i,1) THEN
      differ := differ + 1;
    END IF;
  END LOOP;
  IF differ < 3 THEN
    raise_application_error(-20006, 'Password should differ by at least 3 characters');
  END IF;
END IF;

--Everything is fine; return TRUE ;
  RETURN(TRUE);
END;
/

```

Note - The password verify function must be owned by SYS.

Then to enact this password security verification plus all of the above mentioned additional password security checks, one simply uses Oracle profiles as follows:

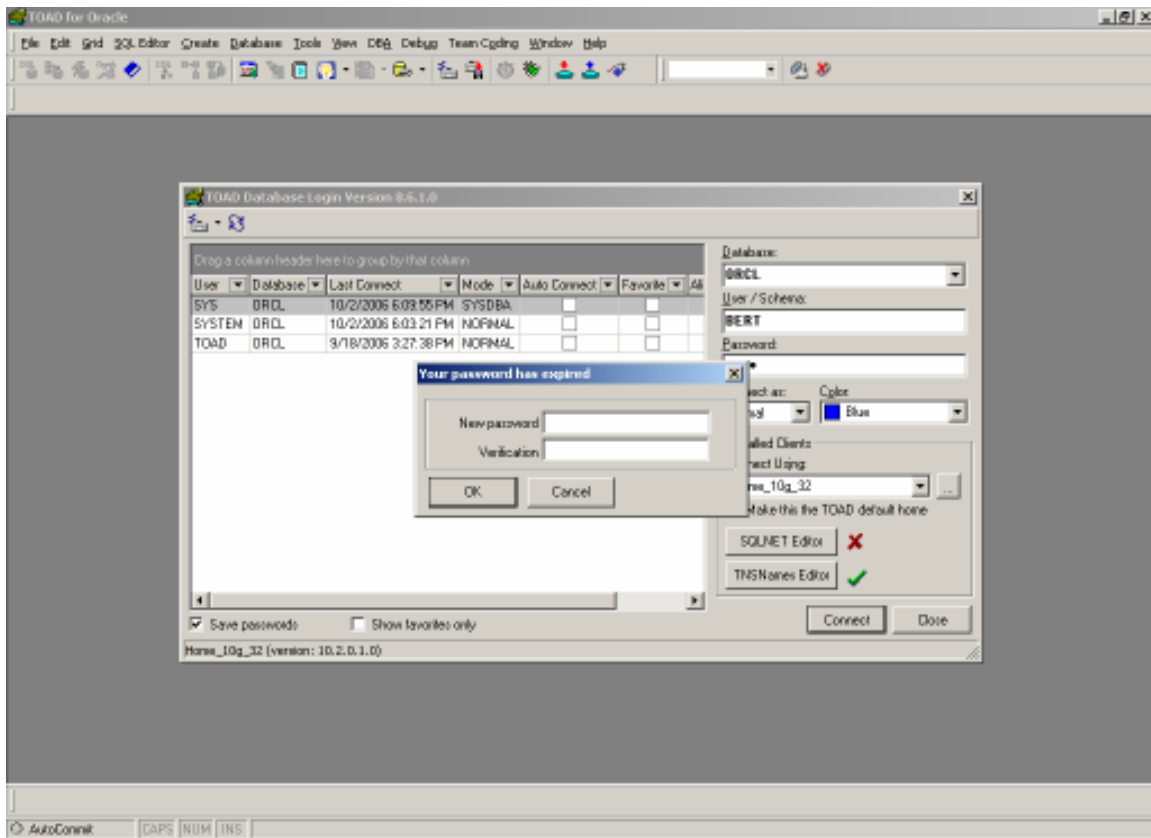
```

ALTER PROFILE TOAD_PLSQL_DEV_SENIOR
LIMIT
SESSIONS_PER_USER 8
IDLE_TIME 60

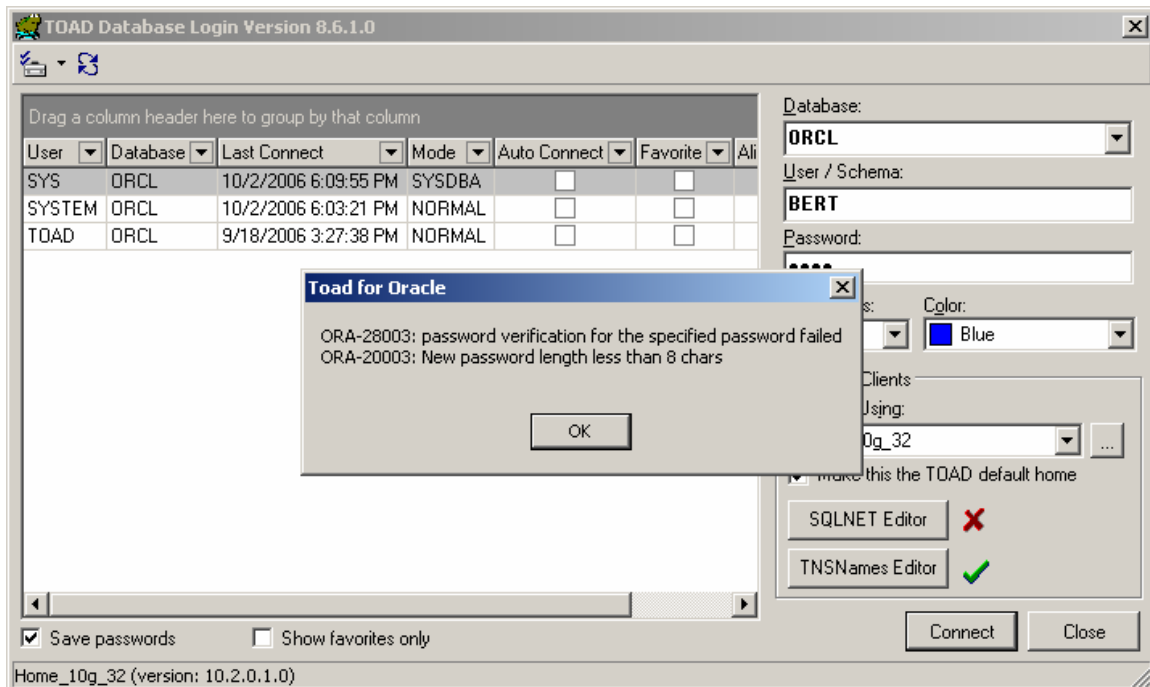
```

FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LIFE_TIME 30
PASSWORD_REUSE_TIME 365
PASSWORD_REUSE_MAX 1
PASSWORD_GRACE_TIME 1
PASSWORD_VERIFY_FUNCTION verify_password;

Thus if a user tries to log into Toad and their password has expired, they will be prompted to replace it as shown below:

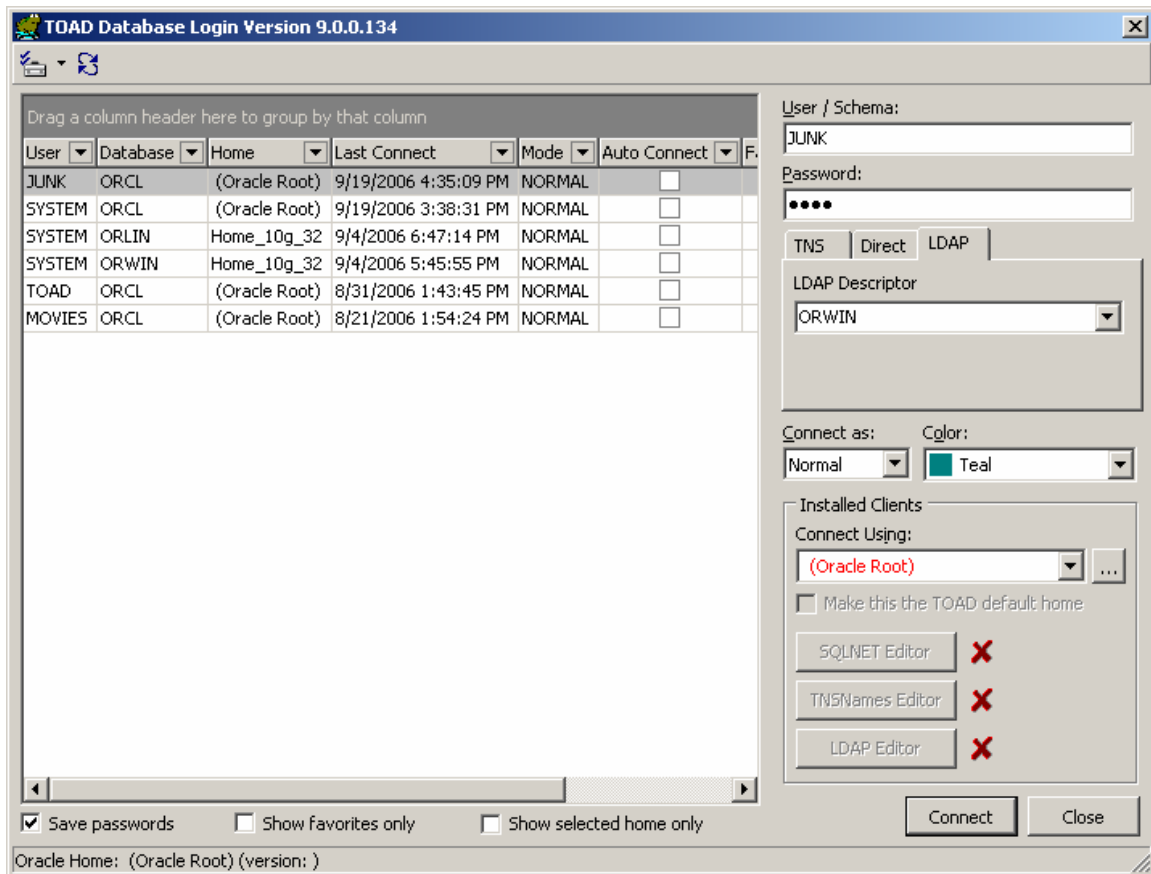


And if the Toad user provides a new Oracle password that fails verification, an error like the one below will display:



#8 Switch to LDAP (OID or Active Directory)

Great news – Toad 9.0 supports LDAP, both for Oracle’s Internet Directory (OID) and Microsoft’s Active Directory (when using Oracle 10g release 2). And while this clearly makes database instance management far easier (i.e. no need to duplicate 4000 copies of a correct tnsnames.ora file to every PC in your company), it also provides a heightened level of security. Because that also means 4000 less locations where someone might learn your server names and IP addresses. And even that lowly network information these days is being considered as sensitive data. So by simply having an ldap.ora file on each PC to assist software with sever and database name resolution, you’ve eliminated another major potential security threat. Plus it’s dirt easy to do now in Toad 9.0 as shown below. You simply choose the LDAP tab on the connection screen and provide the LDAP registered database alias. Then the LDAP server hand shakes with Toad to connect you to the right listener and database instance.



I'll refer interested readers to the following great web-site explaining how to set this up:

<http://www.dizwell.com/prod/node/23>

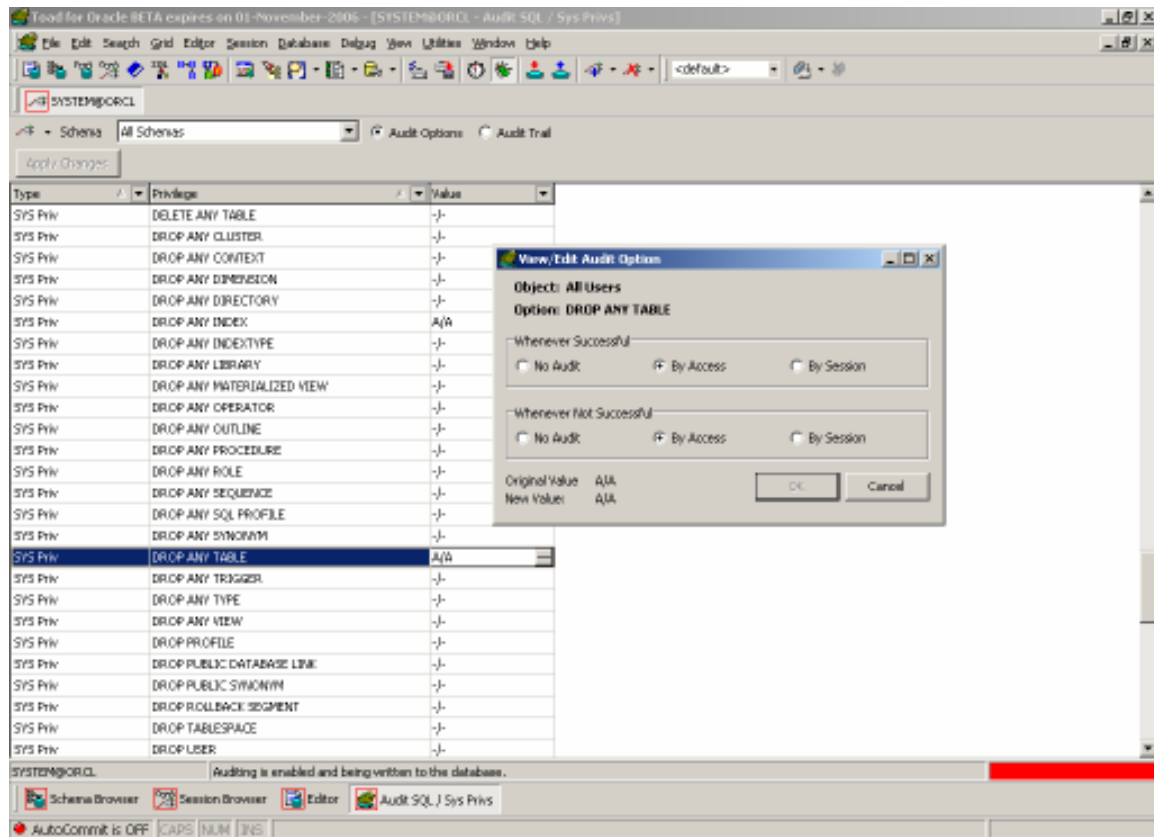
#9 Turn-On Auditing

Ever been called and told that someone – who shall remain anonymous – has accidentally either dropped a table or deleted all its rows. Suppose you needed to know who in order to meet security guidelines. Maybe the policy is that if you drop a table more than twice, then you lose that privilege. So how will the DBA or Toad Administrator know who did this? Of course you can eliminate all those people who don't have either the database privilege or for whom you've used Toad Security to remove that capability. But that still might leave quite a few people to choose from. Once again the Oracle database can solve this security problem as well – and Toad makes managing database auditing a breeze ☺

NOTE – this requires the following init.ora parameter:

- AUDIT_TRAIL = DB | TRUE (default is NONE | FALSE)

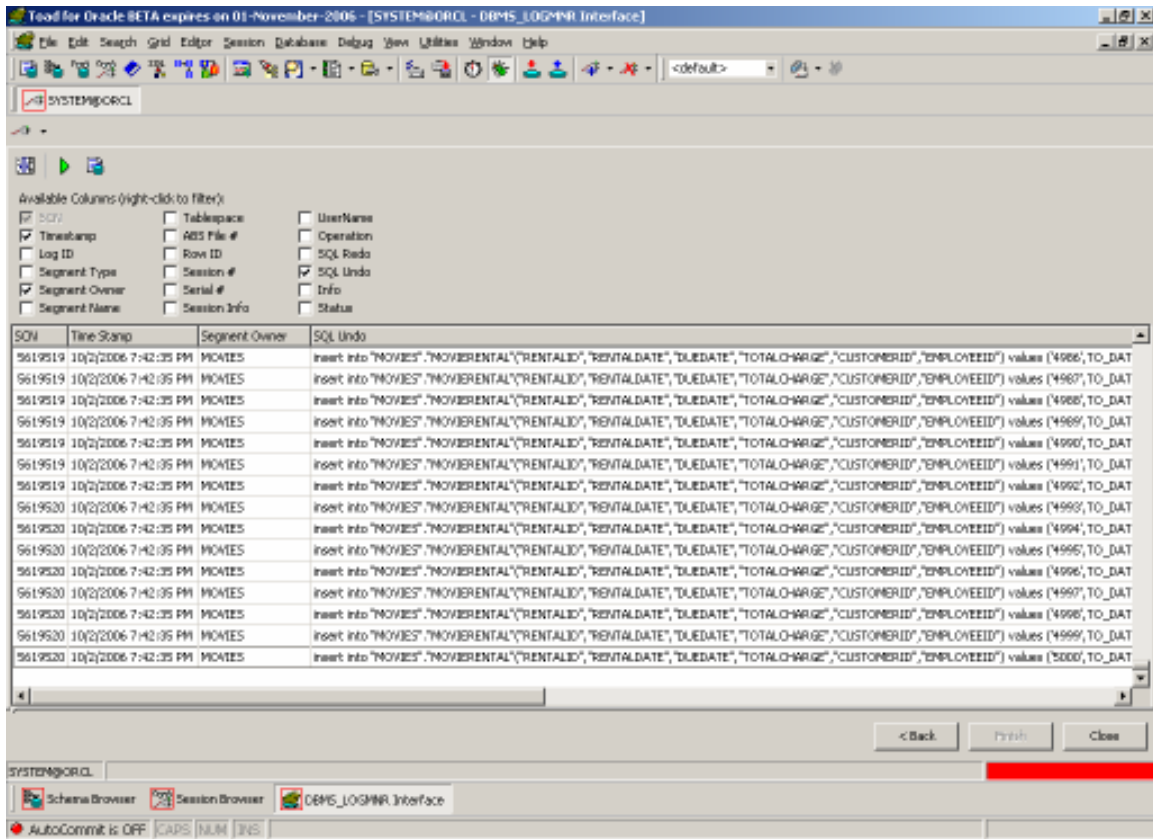
Now rather than learning a whole bunch of cryptic SQL syntax for enabling the numerous and various auditing options, it's far easier to use Toad with the DBA Module to manage all your audit requirements. Notice in the screen below, we've simply set database level audit trail events for any drop table or drop index (and delete from table as well, although that was much higher up on the scroll list and thus not visible in this screen snapshot).



So what happens now when someone calls to say that somebody deleted my table and/or its data – and it wasn't me. Well now Toad can show you exactly who did the dirty deeds as shown below (and regardless whether they used Toad, SQL Plus or some other tool). Looks like that "BERT" user is the guilty culprit – and funny, that's the same guy who called and said someone screwed up his database (and that it wasn't him – he swears it). So now he's nabbed 😊

OS_USERNAME	USERNAME	USERHOST	TERMINAL	TIMESTAMP	OWNER	OBJ_NAME	ACTION_NAME
SYSTEM	SYSMAN	REM206853	REM206853	10/2/2006 7:36:21 PM	SYSMAN	MGMT_JOB_END_STATUS_QUEUE	SESSION REC
SYSTEM	SYSMAN	REM206853	REM206853	10/2/2006 7:40:20 PM	SYSMAN	MGMT_JOB_END_STATUS_QUEUE	DELETE
SYSTEM	SYSMAN	REM206853	REM206853	10/2/2006 7:41:20 PM	SYSMAN	MGMT_JOB_END_STATUS_QUEUE	DELETE
SYSTEM	SYSMAN	REM206853	REM206853	10/2/2006 7:42:20 PM	SYSMAN	MGMT_JOB_END_STATUS_QUEUE	DELETE
BSCALZO	BERT	PROD/REM206853	REM206853	10/2/2006 7:42:24 PM	MOVIES	RENTALITEM	DROP TABLE
BSCALZO	BERT	PROD/REM206853	REM206853	10/2/2006 7:42:33 PM	MOVIES	MOVIERENTAL	DELETE

The best part is that now that you know this – you can also undo the database damage using Toad’s Redo Log Miner (also part of the Toad DBA Module). You simply invoke the Redo Log Miner screen, select the online redo log files to mine, choose to display the “Segment Owner” and “SQL Undo” columns, use column filters to locate the offending SQL commands (in this case deletes), and then select the corresponding undo commands (in this case inserts) as shown below. Then you can select to copy those Undo (i.e. insert) commands into the Toad SQL Editor and thus undo the deletes. Of course first you might want to revoke some privileges from the offending user before doing this work, so that you won’t have to worry about repeating these last few steps.



#10 Stay Current on Toad!!!

This may sound like a self serving sales pitch, but staying current with your Toad version is an easy and effective way to increase your database security. How many of you would risk running Windows 2000 with no service packs and no security updates? Yet we see Toad users running older versions all the time – many of which came out years before their database version. So not only are they running less secure software, but also less productive software for doing their primary job. So there's really no good reason to be several versions or years behind with your Toad version.

For example, people still using Toad versions prior to 8.0 are relying on software that pre-dates many updated compliance regulations, such as Sarbanes-Oxley and Basel II, plus pre-dates the more current Oracle releases, such as 9i and 10g. A perfect example of why this matters in Toad terms is simple, let's talk about saved passwords (although we already said this is often undesirable – and should be turned off via Toad security). Through feedback from large secure customer sites and security industry expert audits, Toad's password saving has evolved over the years as follows:

- Initially stored all connection information in the TOAD.INI file, plus offered option for saving passwords as plain text

- Later stored all connection information in the CONNECTIONS.INI file, removed the option for saving passwords as plain text, but only used the overly simplistic Caesar-Chiffre encryption algorithm – which was far too easily “crack-able”
- Still later stored all connection information in the CONNECTIONS.INI file, but switched to the more secure Advanced Encryption Standard (AES) encryption algorithm – which Network World says should take 149 trillion years to crack (<http://www.networkworld.com/details/597.html>)
- And then finally, separated the connection information into two separate files: CONNECTIONS.INI for the bulk of the connection information, and the new CONNECTIONPWDS.INI for just the encrypted passwords. Furthermore, the passwords’ file now additionally encrypts all passwords with machine specific information – so while the CONNECTIONS.INI file can be transported from machine to machine, the passwords file cannot.

As this one example so clearly demonstrates, staying current on Toad is clearly an easy and effective way to maintain the highest level of database security ☺

About the Author

[Bert Scalzo](#) is a Product Architect for [Quest Software](#) and a member of the [TOAD](#) team. He has worked extensively with TOAD's developers and designed many of its features - including the DBA module, Code Road Map and Code Xpert. Mr. Scalzo has worked with Oracle databases for well over two decades, starting with version 4. His work history includes time at Oracle Education and Oracle Consulting, plus he holds several Oracle Masters certifications. Mr. Scalzo also has an extensive academic background - including a BS, MS and PhD in Computer Science, an MBA and several insurance industry designations. Mr. Scalzo is an accomplished speaker and has presented at numerous Oracle conferences and user groups - including OOW, ODTUG, IOUGA, OAUG, et al. His key areas of DBA interest are Data Modeling, Database Benchmarking, Database Tuning & Optimization, "Star Schema" Data Warehouses and Linux. Mr. Scalzo has written articles for Oracle's Technology Network (OTN), Oracle Magazine, Oracle Informant, PC Week (eWeek), Dell PowerEdge Magazine, The Linux Journal, www.linux.com, and www.orafaq.com. He also has written four books: "[Oracle DBA Guide to Data Warehousing and Star Schemas](#)", "[TOAD Handbook](#)", "[TOAD Pocket Reference](#)" (2nd Edition), and "[Database Benchmarking: Practical methods for Oracle 10g & SQL Server 2005](#)". He also is working on a brand new book for early next year: "[Easy Data Modeling: Practical Methods for Oracle 10g & SQL Server 2005](#)". Mr. Scalzo can be reached via email at bert.scalzo@quest.com or bert.scalzo@yahoo.com.