Advanced Techniques Using Oracle Development Tools with Rdb

Jeffrey S. Haidet

Cheryl P. Jalbert

JCC Consulting, Inc. Granville OH 43023

Abstract

Oracle (development) Tools enable you to provide powerful and easy access to your Rdb databases. There are, however, some challenges that deserve careful attention. This session draws from an array of solutions that we have developed at JCC to provide you with examples.

Topics will include:

- Using stored procedures and functions
- Controlling concurrency
- Forms performance tuning
- Approaches to date issues
- Extras

Using Stored Procedures and Functions

- When would you use stored procedures?
- How would you use stored procedures?

Stored Routines

- When would you use a stored procedure/function?
  - One good rule of thumb: Any time that the same task must be performed numerous times by multiple programs
- Why use a stored procedure/function over a DLL (Windows system file)?
  - Stored routines have benefit over file based programs:
    - Database resident
    - Stability
    - We already have developed stored routines
  - What do file based program have over stored routines?
    - Easier to fix if a bug occurs
      - A fix to a stored procedure may require exclusive database access
- Wouldn’t it be great to use our existing code?

Stored Routines

- The current production release of SQL*Net for Rdb (v1.0.2.5) does not include native interface stored routine support
  - It does however include Net8 support
    - You may use Developer 2000 Release 6.x with Rdb

- Even though a native interface for stored routines does not exist through the current version of SQL*Net for Rdb, you can still utilize your stored routines with:
  - Views
  - FORMS_DDL
  - Computed Fields

Stored Routines - Views

- Views are “windows” into the data
- Using the Oracle Tools, we are limited to 31 characters for column/table names
  - To get at the data stored in your current tables, you may need to create a view with shorter column/table names
    - Forms treats tables and views the same
      - NOTE: There is one major difference, in using the Oracle Toolset (specifically Forms), if you utilize a multi-table views, you MUST set the block and field properties such that they are non-updateable
        - Forms tries to return the ROWID from the database
        - The ROWID and the DBKey, though similar, are computed differently in multi-table joins
- Views are very powerful tools for data presentation

Stored Routines - Views

- How would a view help you utilize a stored procedure?
  - The view may be based upon the results of a stored routine
  - The view may return the result of a stored routine

```
create view my_view_example
(
  field_1, field_2, field_3
) as select a, b, c from my_table
where a = my_stored_function ( b );
```

Stored Routines - Views

- Once the view is created, reference it
  - Build data blocks that are built upon views
  - Use in-line SELECT INTO statements to retrieve values

NOTE: Referencing a view is NO different than referencing a table.

Stored Routines - FORMS_DDL

- FORMS_DDL is a forms built-in function that takes a string and passes it directly into the database

```
Begin
      forms_ddl (‘call my_procedure (1)’);
End;
```

  - Upside
    - You can pretty much do just about anything
      - Call stored routines, set transactions, trace statements, etc.
  - Downside
    - The database must compile the string before execution
    - You cannot directly pass information back to the client after execution
    - Limited error handling

Stored Routines - FORMS_DDL

- Consider using temporary tables to exchange the information between the database and the client
  - Will work across Oracle 8 and Rdb

```
Declare
  CURSOR my_result_set
  SELECT result from temporary_result_table;
Begin
  forms_ddl (‘call my_procedure (1)’);
  OPEN my_result_set;
  FETCH my_result_set into p_variable;
  CLOSE my_result_set;
  <add processing code depending on result>
End;
```

Stored Routines - FORMS_DDL

- By utilizing this approach, you get:
  - Error handling
    - Database resident exception handling
    - Client-side exception handling
  - The ability to call stored procedures/functions

- If you call a stored routine with FORMS_DDL without using a temporary table, you run a risk of your Forms application raising an unhandled exception
  - If your stored routine fails, then a FORMS_SUCCESS will be *FALSE*
    - As a programmer, all you know is that the stored routine call failed

Stored Routines - Computed Fields

- Execute stored functions by implementing Rdb computed fields
- Consider the following:
  - You have to write an Oracle Reports application that will print checks
  - You have the amount stored in the table, but you do not have the amount text stored
  - You have a stored function that will return the amount text, but how do you call it?

```
Alter table my_check_table
  add column check_amount_text
        computed by compute_amount_text (amount);
```

Stored Routines - Computed Fields

- The computed column is a virtual field, computed by Rdb on the fly
- The computed column is referenced as a normal column inside the Oracle Tools
- Computed fields allow for the database to compute values that are not stored, not the client
  - Makes for faster execution
  - All front ends can utilize the same back end
    - Don't have to re-code the routine in C, PL/SQL, Cobol, SQL, Fortran, etc

Stored Routines

- Leveraging your current stored procedure is a MUST
  - Speeds development time
  - Allows legacy application and new application to use the same code base
    - All users don’t have to be switched at once

- Even though SQL*Net for Rdb does not currently support a native interface for stored routines, we have been told, it is coming.

- Golden Rule of Programmers: *Don’t reinvent the wheel. Use what you know works.*

Controlling Concurrency

- Why do we care about concurrency in our application?
- How can we control concurrency?

Concurrency

- In large scale production applications, we do not want lock stalls to be visible to the users.
  - Users will become hostile
  - Data throughput will grind to a halt
- Legacy programs had a starting point and a series of steps required to perform a task
  - Transaction models were fairly simple
- In a multi-form application, how can we control the transaction model?
  - We don’t know what the user is going to do next
  - In event driven programs, users can issue events seemingly without pattern

Concurrency - Multi-Form Application Issues

- Only characteristic you know for sure is that the user must log in
  - To run a forms module, you must connect to a data source
- Users may utilize different forms depending upon their job function
  - Accounting
    - Payables/update account balances
  - Engineering
    - Set up services
  - Customer Service
    - Answer questions concerning account balances
  - Anyone
    - Research an unexpected situation

Concurrency - Multi-Form Application Issues

- In a true PC application, users can:
  - Multi-task
  - Perform similar operations in different ways
    - Click from window to window
    - Press a button to view different data
    - Use pop-up windows

- Even though in a multi-form PC application there are countless ways to perform tasks, we must (as application developers and designers) control the concurrency from within the application.

Controlling Concurrency

- There are two main issues to consider when attempting to solve the concurrency issues
  - Locking
    - You don’t want users to get stuck waiting on others
    - You don’t want mission critical applications to get stuck waiting on users
      - How about a Quiet-Point Backup, for example
  - “Stale” data
    - You don’t want your users stuck inside the same read-only transaction for the entire life of the application
      - Users would be making decision based on old data
- How do you solve these issues?

Controlling Concurrency

- In a database that the users must have the ability to update/insert data…
  - Configure SQL Services for Rdb to implicitly start read committed transactions
    - Closest we can get to read only transactions with Oracle model
  - Configure SQL Services for Rdb to utilize cursors with hold
    - Cursors remaining open across transactions
      - These are implicit in the Oracle model
  - Use forms_ddl to toggle the transaction model
    - Use READ-ONLY transactions when users query
    - Switch to READ-WRITE transactions right before the update/insert

Controlling Concurrency

- WHEN-NEW-FORM-INSTANCE trigger and also KEY-ENTQRY trigger

```
Begin
   -- get rid of current transaction
   -- lessens risk for stale data
   forms_ddl (‘rollback’);
   -- Start new transaction
   forms_ddl (‘set transaction read only’);
End;
```

Controlling Concurrency

- KEY-COMMIT trigger

```
Begin
    -- get rid of current transaction
    forms_ddl (‘rollback’);
    -- Start new transaction
    forms_ddl (‘set transaction read write’);
End;
```

- Note: If you allow for in-line inserts/updates, you will need to toggle the transaction from read-only to read-write or else you will raise an exception.
  - Cannot update/insert during a read-only transaction

Controlling Concurrency

- There is a downside to this design
  - The application is starting and ending a lot of transactions
    - At least one per form
    - At least one per query
    - Possibility of starting and ending a transaction without doing any work
      - Reason: Forms implicitly starts transactions. If we do not specify the read-only transaction, then the possibility exists to get “stuck” in a read-write transaction
- Overhead to I/O root file
  - TANSTAAFL
    - *“There ain’t no such thing as a free lunch”*

Controlling Concurrency - Deadlocks

- In a complex production environment, sometimes deadlocks occur
  - How do we handle deadlocks?
    - When a deadlock occurs, the INSERT/UPDATE triggers raise an exception.
    - Use SQLERRM to retrieve the error message from the database
    - If the message contains ‘DEADLOCK’, then re-issue the COMMIT_FORM
      - This will re-save all records to the database

NOTE: Deadlocks need to be investigated by both the application designers as well as DBAs. The above is a suggestion that will minimize confusion for the user.

Controlling Concurrency - Results

- This design has been implemented by JCC on a 300+ form application
  - Locking issues (caused by the application) have disappeared
    - We are now able to get quiet-point AIJ backups without causing a “log jam” in the database
    - The Forms application is no longer the critical piece that causes legacy code to stall
  - Stale data is less of a factor
    - Each query is a new transaction

Controlling Concurrency - Results

- If you get control of the transaction model
  - Less locking and database contention
  - Less risk of a user viewing stale data
  - A “free” approach to designing PC applications
    - Who cares in what order the user does their tasks?
  - Happier DBAs
  - Most importantly, happier users

Forms Performance Tuning

- The database can run great, but with a bad design, the application could crawl.
- Can we as developers do anything to speed up the application?

Performance Tuning - Objects

- Referencing Objects
  - Multiple ways to reference Objects at runtime
    - ‘*block.item*’
      - **Type - Character String (varchar2)**
    - Internal IDs
      - **Type - Item, Canvas, Block, etc. (internal record structure)**
    - Examples

```
GO_ITEM (‘my_block.my_item’);
GO_ITEM ( item_id );
SET_ITEM_PROPERTY ( ‘block.item’, VISIBLE,
  PROPERTY_TRUE);
SET_ITEM_PROPERTY ( item_id, VISIBLE,
  PROPERTY_TRUE);
```

Performance Tuning - Objects

- Rule: If you are going to reference an Object more than one time, obtain the ID first, then reference the Object via this ID each time

  – When you reference an item at runtime via ‘block.item’ this causes the Forms runtime engine to search through all the Objects in the form (sequentially) until it finds the item

    - The search occurs each time the program references the item

  – When you reference an item at runtime via the internal ID, this forces the Forms runtime engine to perform a direct lookup in memory for the Object

    - No search is required

Performance Tuning - Objects

- Helpful hints for runtime Object referencing:
  - The first time a form is referenced, store all internal IDs
    - Store Ids in:
      - Non-Database Blocks
      - Record Groups
      - Variables
    - Use `WHEN-NEW-FORM-INSTANCE` trigger
    - For any Object referenced, use the internal IDS
- How does this help and is all this really needed?
  - In a true PC application, the “life” of a form depends upon how the user uses the application
    - Objects may be referenced once, or multiple times
    - Resulting code easier to maintain

Performance Tuning - Global Variables

- We’ve all been taught in our computer science classes, AVOID GLOBAL VARIABLES...

Performance Tuning - Global Variables

- In reality…
  - Global variables are a MUST
  - Global variables are a life-saver

- What implications are there in using global variables in our forms applications?

Performance Tuning - Global Variables

- Global variables are treated like Objects
- Referencing an un-initialized global variable raises a runtime exception
    - Use default_value function before referencing any global variable

```
default_value (‘*’,’global.my_global_variable’);
:my_block.my_item := :global.my_global_variable;
```

- Every global variables use 255 characters of space in memory
    - You can address a lot of memory without trying very hard
        - No matter if you store 1 character or 255 characters, you are still using the same amount of memory
    - You cannot store more than 255 characters in a global variable

Performance Tuning - Global Variables

- What should we do?
  - Let’s face it, global variables are needed
  - The more global variables you use, the more performance implications arise

- Rule: Use global variables whenever you have no other alternative.

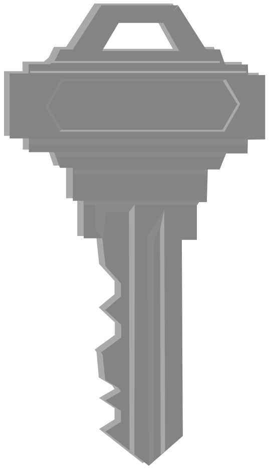
Be smart and plan
your application carefully

Performance Tuning - “Global” Hints

- When is it appropriate to use a global variable?
  - When passing data between two forms
  - To store application specific runtime settings
    - Username
    - Password
    - Database connect string
    - Application Version
- Use record groups with global scope to store large amounts of data that must be shared between forms
  - One item to reference in memory
  - A record group can contain a lot of information
- Use parameter lists to pass data from form to form

Performance Tuning - Application Security

- Rdb does not currently support Developer 2000 built-in roles
  - This is a security mechanism support by the Oracle “n” database
  - Implementation just beginning in Rdb8
- Some degree of security is needed in almost every production application
  - How do we implement security in our Rdb/Forms applications?
  - Can we tune this implementation to run fast?



Performance Tuning - Security

- Assume the following for discussion:
  - A task is defined as a business role
  - A user can be assigned to business roles depending upon job description
- In a multi-form application, we are forced to apply security to the form at initial form instantiation
  - We do not know what form(s) a user will access in a given session
  - Object properties are only good for the life of the form
- Do we have to query the database for each new form to apply the security?

Performance Tuning - Security

- Instead of issuing extra queries to apply application security at runtime, cache ALL the security information in memory on the client
  - At application login time, download the security information into global record groups
  - Upon form instantiation, loop through the security cache to apply security roles
    - In a 300+ form application, we found that 30-40% of the queries were being issued against the security tables
    - By caching this information, we removed the excess queries
    - Result was a “significant” performance boost in the application and a bunch of really happy users

Performance Tuning - Security Cache

- Example of caching information from the database

```
query_text := ’select user_name, user_role, form_name,
  block_name, query_flag, update_flag, insert_flag,
  delete_flag’ ||' from security_view where user_name =
  ||quote||p_username||quote;

security_record_group := CREATE_GROUP_FROM_QUERY
                    (
                      'SECURITY_RECORD_GROUP'
                     ,query_text
                     ,GLOBAL_SCOPE
                     ,25
                    );
```

Performance Tuning - Security Cache

- Instead of hitting the database for the cached information, use the record group

```
row_count := Get_Group_Row_Count ( security_record_group );
FOR i IN 1..row_count
LOOP
   < process through the record group >
   < use built-in functions Get_Group_Char_Cell,
        Get_Group_Date_Cell, Get_Group_Number_Cell to access
        the data >
END LOOP;
```

- Tip: Call security modules from each form
  - Consider the `WHEN-NEW-FORM-INSTANCE` trigger

Performance Tuning - Implicit vs. Explicit Cursors

- How you perform “in-line” queries can also impact the performance of your application
- There are two methods to fetch and process information in your PL/SQL blocks
  - Explicit Cursors
  - Implicit Cursors

Performance Tuning - Cursors

- Explicit Cursors
  - This is a select statement that is *explicitly* defined in declaration section of the PL/SQL block

```
declare
       CURSOR my_cursor
       SELECT field_a
       FROM   my_table
       WHERE  field_b = p_variable;
begin
       <some code>
       p_variable := 1;
       OPEN my_cursor;
       <process cursor>
       CLOSE my_cursor;
end;
```

Performance Tuning - Cursors

- An implicit cursor is a little tougher to define.
  - Each time you perform a SELECT INTO PL/SQL statement, the runtime engine automatically generates, opens, fetches from, and closes the cursor
    - "Great! The runtime engine does it all without me having to do anything!"
  - For a single query, the forms runtime engine performs two fetches from the database
    - First, obtain the row from the database
    - Second, determines if the TOO_MANY_ROWS exception should be raised
      - The second query does not take much time….

But…………...

Performance Tuning - Cursors

- Consider the following example:
  - You issue a SELECT INTO statement in a POST-QUERY trigger to stored certain values
  - The query returns 15 rows
  - The POST-QUERY trigger will fire once for each row retrieved from the database
  - You will have the extra overhead of 30 extra queries!
    - 15 fetches for the SELECT INTO to obtain the data rows
    - 15 fetches to determine if the TOO_MANY_ROWS exception is to be raised

- Rule: Avoid using implicit cursors whenever possible.

Performance Tuning - Helpful Hints

- Avoid issuing queries for ALL records
  - Done by EXECUTE_QUERY (all_records);
  - The database engine must return all the requires to the client that are a result of the query
    - More network traffic
    - More database work
- Avoid setting the block property query all records to ‘Yes’
  - Same implication as above
- Avoid using client-side (Forms) computed fields
  - All rows must be retrieved for this condition to be met
  - Same implication as above

Performance Tuning - Helpful Hints

- Make sure that the queries that are being issued into the database are behaving as they should
  - Turn on the BLR in the SQL Server log files
  - Review the queries
  - Disk space is cheap
- With such a powerful tool, you will be unable to tune all possible queries
  - Work closely with your DBA to tune the most important queries
  - Talk to your users to ensure the tasks they performed are well supported

Performance Tuning - Helpful Hints

- Try to limit the size of the FMX (compiled forms modules) to under 1 megabyte
  - The larger the form, the longer it takes to load into memory
  - To speed the loading of the form
    - Partition the application
      - Try doing more work on the server
    - Move form resident program units into libraries
- Avoid using default values for items that would require an extra query to the database
  - $$DBDATE$$
  - $$DBDATETIME$$
- Use database triggers to populate audit tables rather than writing the information from the application

Date Issues

- Conditional formatting (especially with dates) is essential
  - Do you want your users to see 17-NOV-1858?
  - Is it possible to hide the 17-NOV-1858 but allow other dates to be viewed?
  - Is it possible to alter the format for viewing/date entry?
- An issue to consider
  - If you programmatically delete data from a database field, the runtime engine marks the record as needing updating

Masking Dates

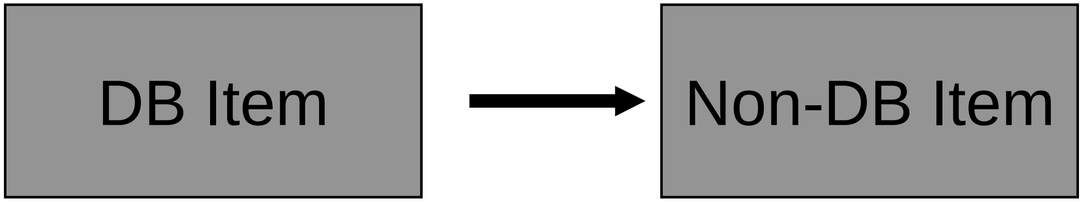
- Since the runtime engine marks blocks as updated if we programmatically remove data from fields
  - “Hide” the actual database field
  - Add a non-database field that is displayed
  - Mask/Unmask the field just before and just after the query
    - Pre-Query Trigger
    - Post-Query Trigger
    - Copy the data back and forth between the two items
      - This will not mark the database block as needing updating

- This will work for other datatypes

Masking Date - Example

- Database Item

```
Begin -- POST-QUERY Trigger
   if :system.cursor_item <>
               to_date (‘17-Nov-1858’,’DD-MON-YYYY’)
   then
       :block.display_date := :system.cursor_item;
   else
       :block.display_date := ‘’;
   end if;
End;
```



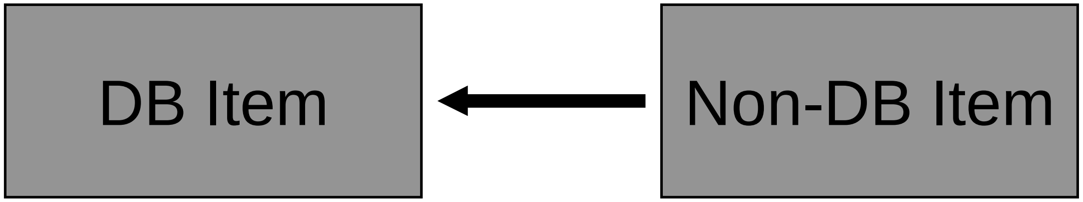
DB Item

Non-DB Item

Un-Masking Date - Example

- Non-Database Item

```
Begin -- PRE-QUERY Trigger
   if :system.cursor_item <>
               to_date (‘17-Nov-1858’,’DD-MON-YYYY’)
   then
        :block.database_date := :system.cursor_item;
   else
        :block.database_date := ‘’;
   end if;
End;
```



Formatting Dates

- Do you users like the 01-NOV-1999 date format?
- What about formatting data for heads-down data entry?
- Is it possible to speed data entry and also format the data in a reasonable way?

Formatting Dates

- Can a data field be formatted?
  - Answer, Yes. It is an item property
  - Instead of 01-NOV-1999 display 11/1/1999
    - Set the format mask to MM/DD/RRRR
      - The RRRR means that all 4 digits for the year are required
  - Must the users enter the “/” or “-” when entering data?
    - Unless you write some code, then answer is “*yes*”
    - Consider programmatically altering the format mask
      - Use SET_ITEM_PROPERTY (itm_id, FORMAT_MASK, mask_str)
      - When the user navigates into the field, remove the extra characters
      - When the user navigates out of the field, add the extra characters back into the string
      - This allows for FAST heads-down data entry

Extras

- I couldn’t resist some extra goodies
  - PL/SQL Libraries
  - Object Libraries
  - Building Your Application
  - Leveraging the Windows API
  - Logging Event
  - Fail Over

Extras - PL/SQL Libraries

- PL/SQL libraries
  - These are program units that are stored outside of form modules
  - Attached to form modules at design time
- Use PL/SQL libraries whenever possible
  - Upon forms compilation, references to the libraries are created
  - The program units are linked at runtime
- Cannot use libraries all the time
  - Cannot reference form module Objects from within a PL/SQL library

Extras - PL/SQL Libraries

- What do you put into a PL/SQL library?
  - Application security code
  - Date masking routines
  - Primary Key generation code
    - Software that generates unique values
      - JCC Get-ID Software
- Benefits
  - The form modules don’t need to include all this extra code
  - Partition forms load faster
    - The compiled library module (.PLX) is loaded into memory once
    - Code contained in the PL/SQL library module is not copied into the forms runtime module, making the modules smaller

Extras - Object Libraries

- An object library is a collection of form specific objects that are referenced at design time
- Upon compilation, the form module pulls in copies of the referenced objects
  - No linking is done dynamically
  - If you find a bug, you can fix it in one spot, but all form modules that use this object must be regenerated
  - Since the form modules copy the information into the FMX at compilation time, the form modules are not smaller

Extras - Object Libraries

- What do you put into an Object Library?
  - Common components
    - Alerts
    - Toolbars
    - Canvases
    - Blocks
    - Object Groups
    - Program Units
    - Etc.
- Benefits
  - All objects in one spot
  - Automatically, re-linked at design time

Extras - Building Your Application

- Now that you have created your application, how do you build it?
- “Batch” File Approach
  - Loop through all files compiling them with f50gen/f60gen
  - This requires attaching and disconnecting from an OCI server process for each form
  - This approach works on multiple platforms
- Developer 2000 API
  - You can compile all form modules with a single connection to a OCI server process
  - In a 300+ form application, this cut the time required to build the application from 3 hours to just around 15 minutes

Extras - Utilizing the Windows API

- You can reference DLLs/EXEs on the Windows platform in multiple ways
- The easiest is D2KWUTIL
  - This is a built-in package supplied by Oracle
  - You can attach it to your forms as an attached library
  - It contains the code required to call the most common operating system functions
    - Reading from/Writing to the Windows registry
    - Obtaining screen inactivity
    - Opening File Dialog boxes
    - Selecting a printer
    - Etc

Extras - Using the Windows API

- Consider an application in which you want to log out a user after 15 minutes of idle time
  - Use the WIN_API_SESSION.get_inactive_time function
    - Returns the inactive time of the parent window
    - This function leverages the Windows API to determine the amount of inactive time

```
<after setting the timer interval>
-- WHEN-TIMER-EXPIRED trigger
IF WIN_API_SESSION.get_inactive_time > 15
THEN
   logout;
END IF;
```

Extras - Logging Events

- Wouldn’t it be great to track what the users are actually doing?
  - What screens are accessed the most?
  - How often to timeouts occur?
  - What work flow does a specific user use?

- This can all be accomplished through logging Form events

Extras - Logging Events

- Create a stored procedure that issues trace statements out to the SQL Server log files
- Add Object Library code that can be inherited through all form modules
  - Track which forms are accessed
    - PRE-FORM and POST-FORM triggers
  - Track application timeouts
    - WHEN-TIMER-EXPIRED trigger
  - Etc.
- Review the SQL Server log files
  - Parse the log files for trace statements and dump results into the database for analysis

Extras - Fail Over

- Fail Over is the process by which we switch from the primary database server to a secondary database server
  - Reasons for fail over
    - Hardware Failure
    - Software Failure
    - Network Failure
    - Maintainence Window
- How can we implement fail over?

Extras - Fail Over

- Designing your application
  - Log activity to the database
    - A stored procedure that issues trace statements will suffice
    - Log each operation
      - Form initialization
      - Form termination
      - Query start
      - Query end
      - Exceptions raised
  - Logging assists in more than just fail over
    - Debugging
    - Determine user work flow
    - Gather query statistics

Extras - Fail Over

- How can you detect that the connection to the database is broken?
  - If the SQL*Net client software loses the connection with the database, then exceptions are raised
    - Built in function DBMS_ERROR_CODE in (-3121,-3113,-3114)
- Automatic switch over to secondary database
  - Detect DBMS_ERROR_CODE
  - Retry primary database
    - Number of times determined by application
  - Connect to secondary database
    - Download new security model
      - Decide if updates/inserts should occur in secondary database
  - Notify user of switch over

Object Orientation

- Rome wasn’t built in a day
  - By leveraging reusable objects, we have made implementing new features rather easy
    - Alter the code in either the Object or PL/SQL libraries
      - Propagated to all forms
    - Completely re-design functionally in short periods of time
      - Transaction control model - 2 days
      - Date formatting - 1 day
      - Security Cache - Less than 1 day

- Last Rule: Use the power of the tool to your advantage. Leverage the objects to make your life easier.

Questions