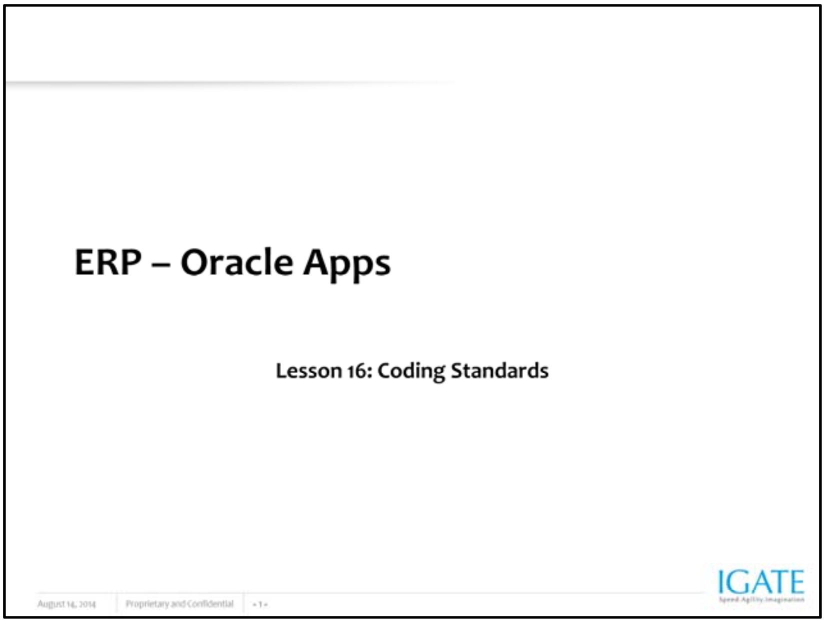
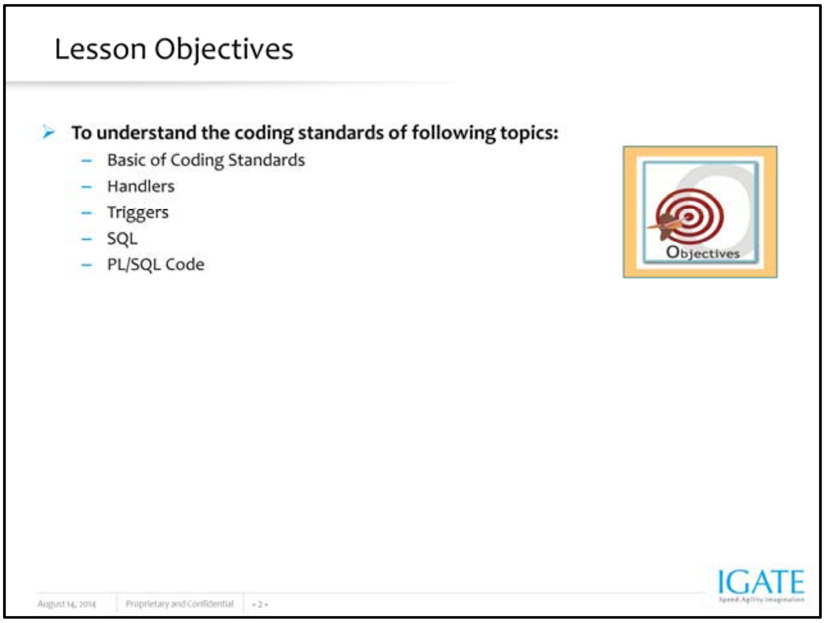


ERP – Oracle Apps

Lesson 16: Coding Standards

August 14, 2014 | Proprietary and Confidential | - 1 -

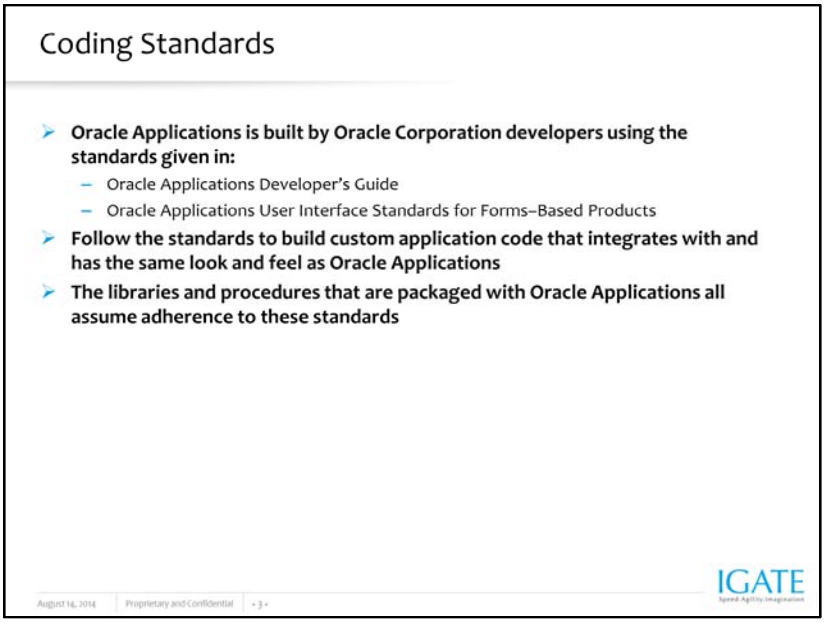


Lesson Objectives

- To understand the coding standards of following topics:
  - Basic of Coding Standards
  - Handlers
  - Triggers
  - SQL
  - PL/SQL Code

August 14, 2014 | Proprietary and Confidential | - 2 -

IGATE
Speed. Agility. Imagination



Coding Standards

- Oracle Applications is built by Oracle Corporation developers using the standards given in:
  - Oracle Applications Developer's Guide
  - Oracle Applications User Interface Standards for Forms–Based Products
- Follow the standards to build custom application code that integrates with and has the same look and feel as Oracle Applications
- The libraries and procedures that are packaged with Oracle Applications all assume adherence to these standards

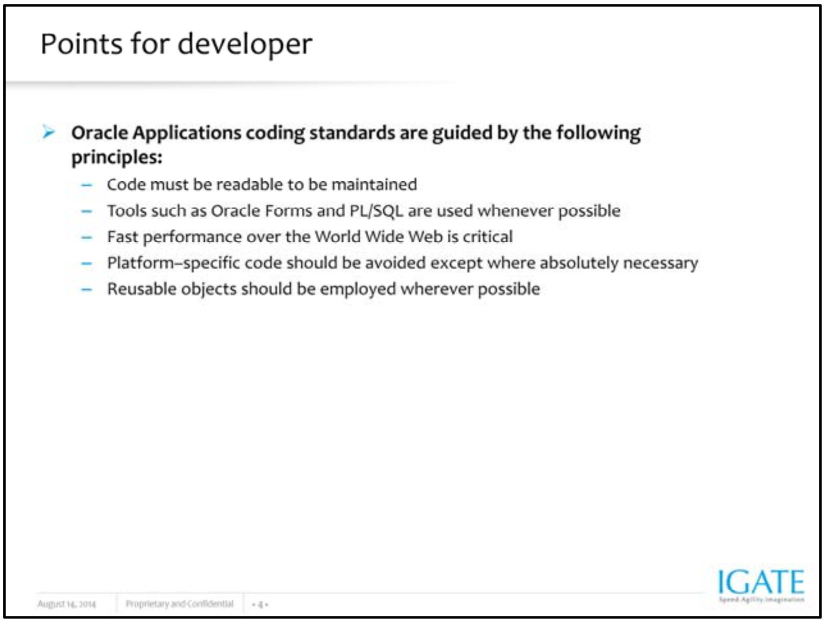
August 14, 2014 | Proprietary and Confidential | - 3 -

IGATE

Importance of these Standards

The coding standards described in Oracle Applications Developer's Guide, together with the user interface standards described in the Oracle Applications User Interface Standards for Forms–Based Products, are used by Oracle Corporation developers to build Oracle Applications. If you want to build custom application code that integrates with and has the same look and feel as Oracle Applications, you must follow these standards. If you do not follow these standards exactly as they are presented, you may not achieve an acceptable result.

The libraries and procedures that are packaged with Oracle Applications all assume adherence to these standards. In fact, since the behavior of Oracle Forms, the Oracle Applications standard libraries, and the standards are so tightly linked, a deviation from standards that appears to be minor may in fact have far–reaching and unpredictable results. Therefore, its recommend that when you develop custom application code, you follow the standards exactly as they are described in these manuals.



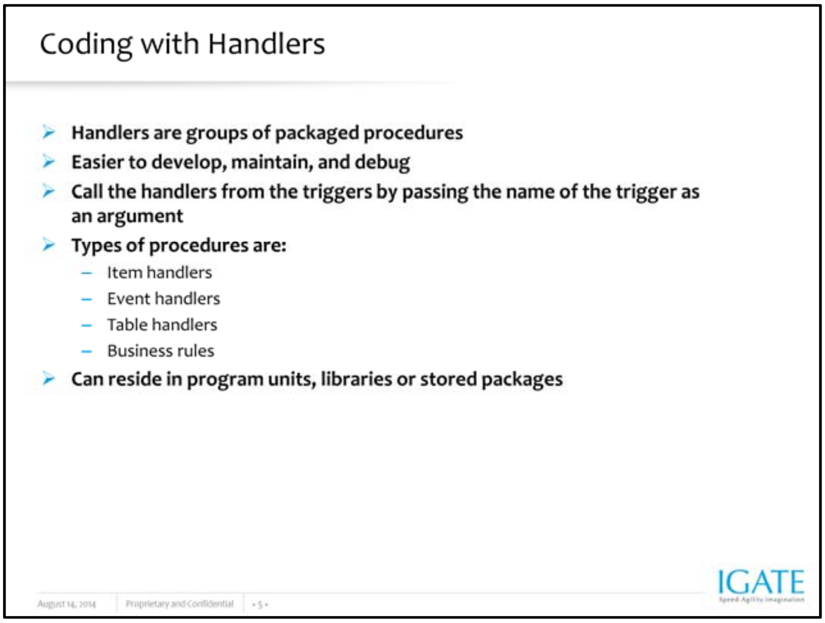
Points for developer

- Oracle Applications coding standards are guided by the following principles:
  - Code must be readable to be maintained
  - Tools such as Oracle Forms and PL/SQL are used whenever possible
  - Fast performance over the World Wide Web is critical
  - Platform–specific code should be avoided except where absolutely necessary
  - Reusable objects should be employed wherever possible

August 14, 2014 | Proprietary and Confidential | - 4 -

Oracle Applications coding standards are guided by the following principles:

- Code must be readable to be maintained
- Tools such as Oracle Forms and PL/SQL are used whenever possible (avoid complex user exits using other coding languages)
- Fast performance over the World Wide Web (the web) is critical
- Platform–specific code should be avoided except where absolutely necessary
- Reusable objects should be employed wherever possible



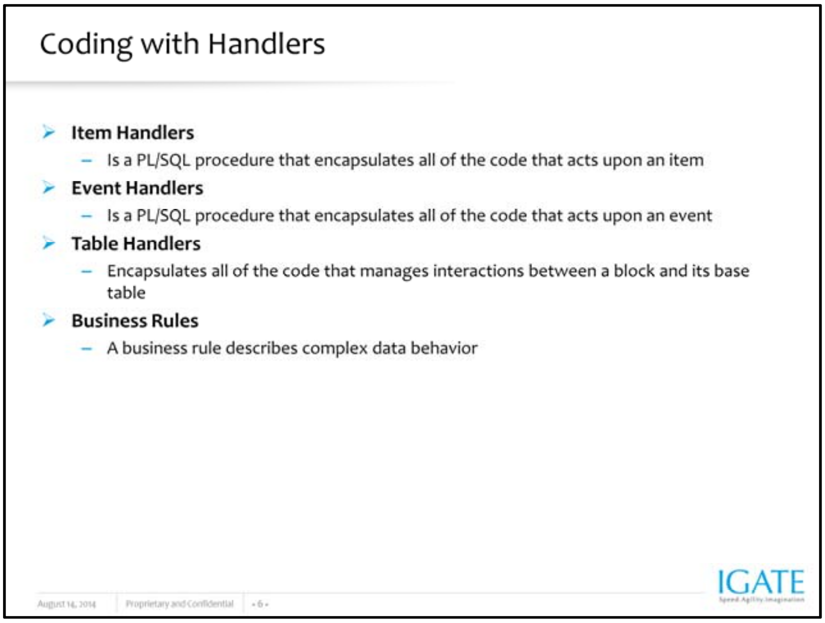
Coding with Handlers

- Handlers are groups of packaged procedures
- Easier to develop, maintain, and debug
- Call the handlers from the triggers by passing the name of the trigger as an argument
- Types of procedures are:
  - Item handlers
  - Event handlers
  - Table handlers
  - Business rules
- Can reside in program units, libraries or stored packages

August 14, 2014 | Proprietary and Confidential | - 5 -

IGATE Speed.Agility.Imagination

Oracle Applications uses groups of packaged procedures, called handlers, to organize PL/SQL code in forms so that it is easier to develop, maintain, and debug.
In Oracle Forms, code is placed in triggers, which execute the code when that trigger event occurs. Implementing complex logic may require scattering its code across multiple triggers. Because code in triggers is not located in one place, it cannot be written or reviewed comprehensively, making development, maintenance, and debugging more difficult. To determine what code and events affect a particular item, a developer must scan many triggers throughout the form. Code that affects multiple items can be extremely difficult to trace.
To centralize the code so it is easier to develop, maintain, and debug, place the code in packaged procedures and call those procedures from the triggers. Pass the name of the trigger as an argument for the procedure to process. This scheme allows the code for a single business rule to be associated with multiple trigger points, but to reside in a single location.
There are different kinds of procedures for the different kinds of code you write: item handlers, event handlers, table handlers, and business rules. Code resides in these procedures; do not put any code in the triggers other than calls to the procedures.
Handlers may reside in program units in the form itself, in form libraries, or in stored packages in the database as appropriate.

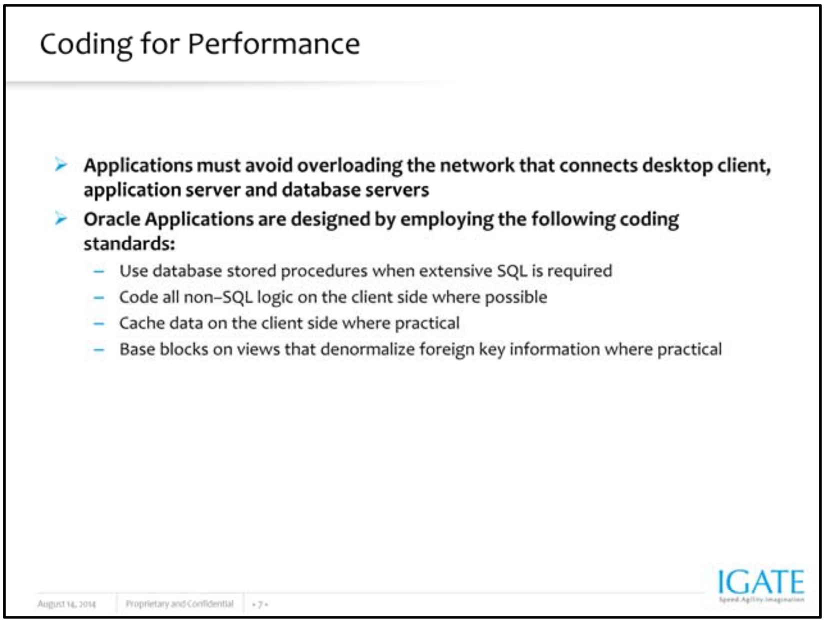


Coding with Handlers

- **Item Handlers**
  - Is a PL/SQL procedure that encapsulates all of the code that acts upon an item
- **Event Handlers**
  - Is a PL/SQL procedure that encapsulates all of the code that acts upon an event
- **Table Handlers**
  - Encapsulates all of the code that manages interactions between a block and its base table
- **Business Rules**
  - A business rule describes complex data behavior

August 14, 2014 | Proprietary and Confidential | - 6 -

Item Handlers
An item handler is a PL/SQL procedure that encapsulates all of the code that acts upon an item. Most of the validation, defaulting, and behavior logic for an item is typically in an item handler.
Event Handlers
An event handler is a PL/SQL procedure that encapsulates all of the code that acts upon an event. Usually event handlers exist to satisfy requirements of either Oracle Forms or the Oracle Applications User Interface Standards for Forms–Based Products, as opposed to particular business requirements for a product.
Table Handlers
A table handler encapsulates all of the code that manages interactions between a block and its base table. When an updatable block is based on a view, supply procedures to manage the insert, update, lock and delete. Referential integrity checks often require additional procedures. Table handlers typically reside in the database but may also reside on the forms server depending on size and the amount of interaction with the database.
Business Rules
A business rule describes complex data behavior. For example, one business rule is: ”A discount cannot be greater than 10% if the current credit rating of the buyer is less than ’Good’.” Another business rule is: ”A Need–By Date is required if a requisition is made for an inventory item.”



Coding for Performance

- Applications must avoid overloading the network that connects desktop client, application server and database servers
- Oracle Applications are designed by employing the following coding standards:
  - Use database stored procedures when extensive SQL is required
  - Code all non–SQL logic on the client side where possible
  - Cache data on the client side where practical
  - Base blocks on views that denormalize foreign key information where practical

August 14, 2014 | Proprietary and Confidential | • 7 •

Performance

Performance is a critical issue in any application. Applications must avoid overloading the network that connects desktop client, application server, and database servers, since often it is network performance that most influences users' perceptions of application performance.

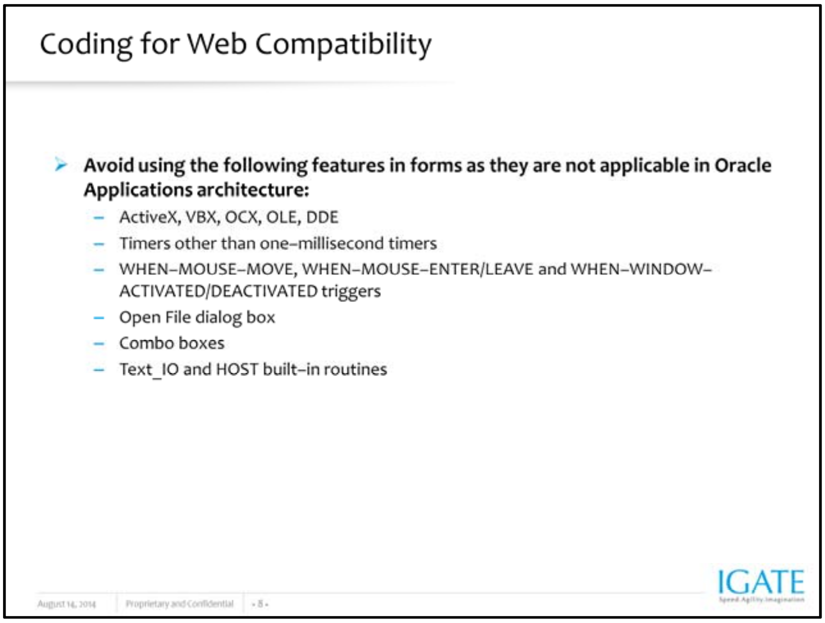
Oracle Applications are designed to minimize network traffic on all tiers. For example, they try to limit network round trips to one per user–distinguishable event by employing the following coding standards:

Use database stored procedures when extensive SQL is required

Code all non–SQL logic on the client side where possible

Cache data on the client side where practical

Base blocks on views that denormalize foreign key information where practical



Coding for Web Compatibility

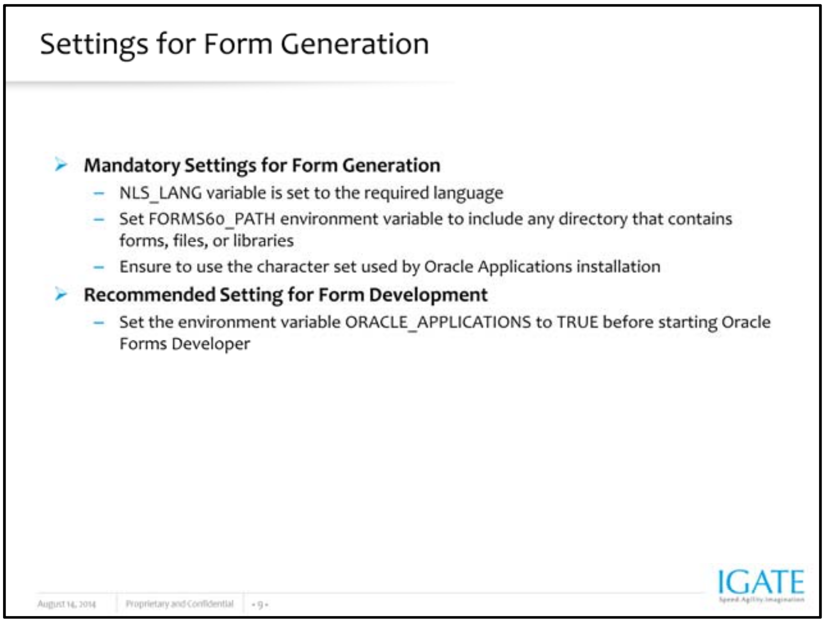
- Avoid using the following features in forms as they are not applicable in Oracle Applications architecture:
  - ActiveX, VBX, OCX, OLE, DDE
  - Timers other than one–millisecond timers
  - WHEN–MOUSE–MOVE, WHEN–MOUSE–ENTER/LEAVE and WHEN–WINDOW–ACTIVATED/DEACTIVATED triggers
  - Open File dialog box
  - Combo boxes
  - Text_IO and HOST built–in routines

August 14, 2014 | Proprietary and Confidential | - 8 -

IGATE
Speed. Agility. Imagination

Coding for Web Compatibility
Following Oracle Applications standards carefully will help ensure that your forms can be deployed on the Web.
You should avoid using the following features in your forms, as they are not applicable in this architecture:

ActiveX, VBX, OCX, OLE, DDE (Microsoft Windows–specific features that would not be available for a browser running on a Macintosh, for example, and cannot be displayed to users from within the browser)
Timers other than one–millisecond timers (one–millisecond timers are treated as timers that fire immediately)
WHEN–MOUSE–MOVE, WHEN–MOUSE–ENTER/LEAVE and WHEN–WINDOW–ACTIVATED/DEACTIVATED triggers
Open File dialog box – It would open a file on the applications server, rather than on the client machine (where the browser is) as a user might expect
Combo boxes – Our standards do not use combo boxes anyhow
Text_IO and HOST built–in routines – These would take place on the applications server, rather than on the client machine (where the browser is) as a user might expect

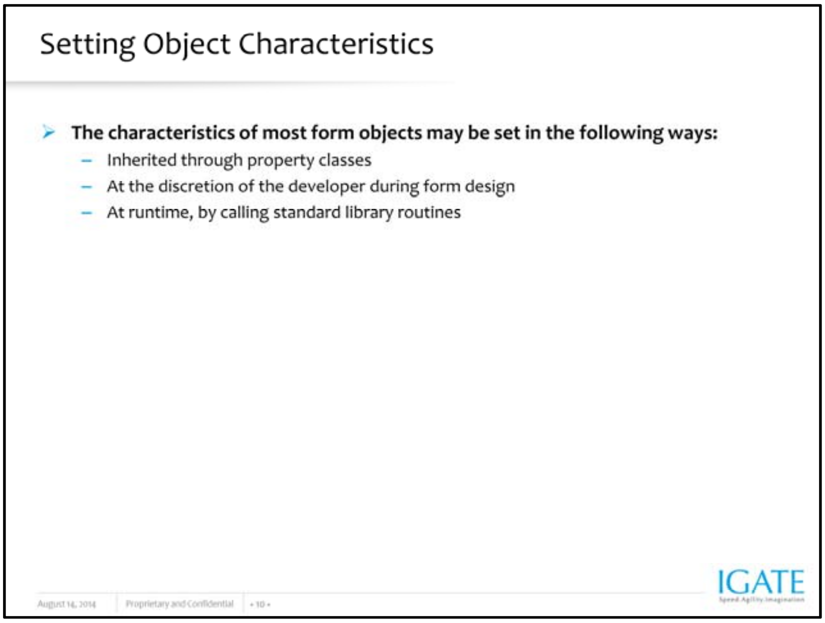


Settings for Form Generation

- **Mandatory Settings for Form Generation**
  - NLS_LANG variable is set to the required language
  - Set FORMS60_PATH environment variable to include any directory that contains forms, files, or libraries
  - Ensure to use the character set used by Oracle Applications installation
- **Recommended Setting for Form Development**
  - Set the environment variable ORACLE_APPLICATIONS to TRUE before starting Oracle Forms Developer

August 14, 2014 | Proprietary and Confidential | - 9 -

Mandatory Settings for Form Generation
At form generation time, make sure that NLS_LANG variable is set to the required language in the Windows NT registry or environment file (for Unix). Also ensure that the character set specified is the character set being used for Oracle Applications installation.
You must also set the value of FORMS60_PATH environment variable in environment file (Windows registry) to include any directory that contains forms, files, or libraries to be used to develop and generate forms.
Specifically, include a path to the <$AU_TOP>/forms/US directory to be able to find all referenced forms, and a path to the <$AU_TOP>/resource directory to be able to find the Oracle Applications library files you need.
Recommended Setting for Form Development
Oracle Forms Developer allows referenced objects to be overridden in the local form. Oracle Forms Developer also does not normally provide any indication that an object is referenced unless you set a special environment variable. Set the environment variable ORACLE_APPLICATIONS to TRUE before starting Oracle Forms Developer. This setting allows you to see the reference markers (little flags with an ”R” in them) on referenced objects so you can avoid changing referenced objects unintentionally. Any object referenced from the APPSTAND form must never be changed.



Setting Object Characteristics

- **The characteristics of most form objects may be set in the following ways:**
  - Inherited through property classes
  - At the discretion of the developer during form design
  - At runtime, by calling standard library routines

August 14, 2014 | Proprietary and Confidential | - 10 -

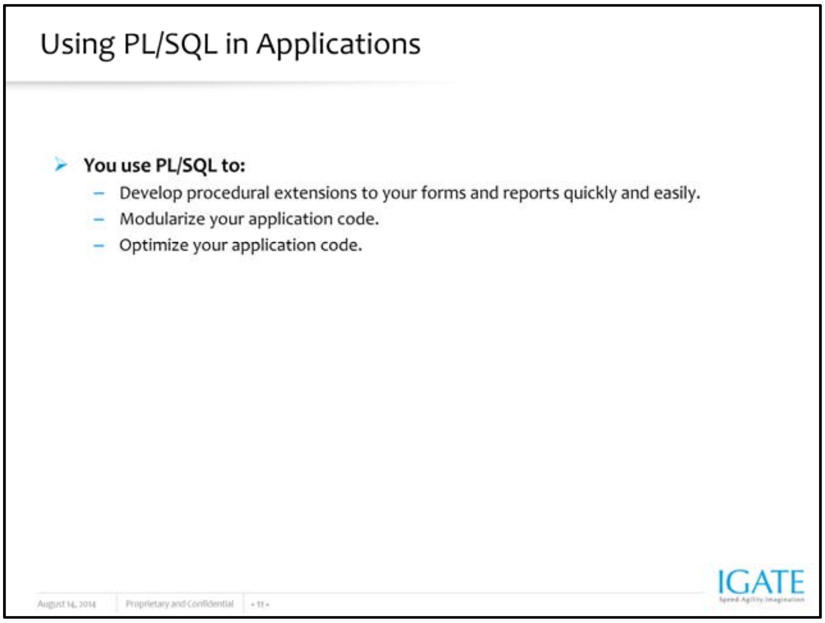
Setting Object Characteristics

The characteristics of most form objects, including modules, windows, canvases, blocks, regions, and items may be set in the following ways:

Inherited through property classes, which cause certain properties to be identical in all forms (such as canvas visual attributes)

At the discretion of the developer during form design (such as window sizes)

At runtime, by calling standard library routines (such as window positions)



Using PL/SQL in Applications

- **You use PL/SQL to:**
  - Develop procedural extensions to your forms and reports quickly and easily.
  - Modularize your application code.
  - Optimize your application code.

August 14, 2014 | Proprietary and Confidential | - 11 -

You can use PL/SQL procedures as part of an application that you build around Oracle Applications. By following the coding standards, you can create a PL/SQL procedure that integrates seamlessly with your application and with Oracle Applications.

You use PL/SQL to:

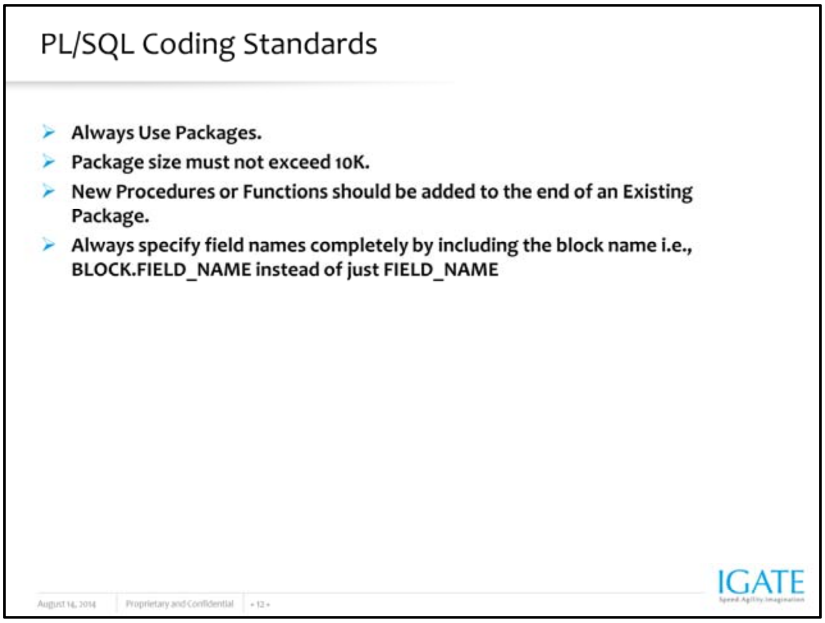
Develop procedural extensions to your forms and reports quickly and easily

Modularize your application code to speed development and improve maintainability

Optimize your application code to reduce network traffic and improve overall performance

You can use PL/SQL to develop procedural extensions to custom forms and reports you create with Oracle tools. For example, to develop a form that follows Oracle Applications standards, organize form code into PL/SQL business rule procedures, item handlers, event handlers, and table handlers. Put very little PL/SQL code directly into form triggers because those triggers do not represent a logical model; they are simply event points that Oracle Forms provides for invoking procedural code. If you put most of the code in packaged PL/SQL procedures, and then call those procedures from triggers, you will have modular form code that is easy to develop and maintain.

You can also use PL/SQL to develop concurrent programs or stored procedures that are called from concurrent programs.

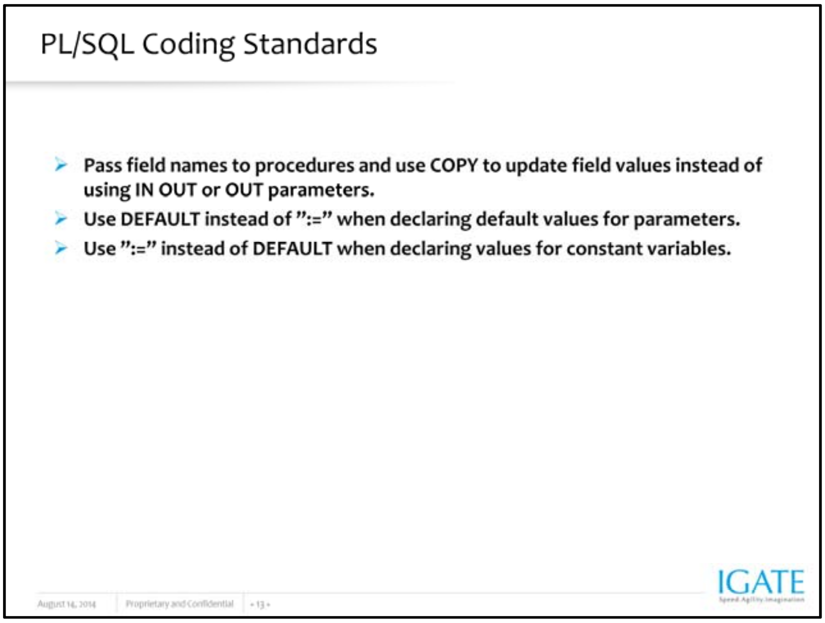


PL/SQL Coding Standards

- Always Use Packages.
- Package size must not exceed 10K.
- New Procedures or Functions should be added to the end of an Existing Package.
- Always specify field names completely by including the block name i.e., BLOCK.FIELD_NAME instead of just FIELD_NAME

August 14, 2014 | Proprietary and Confidential | - 12 -

Always Use Packages
PL/SQL procedures should always be defined within packages. Create a package for each block of a form, or other logical grouping of code. A client–side (Oracle Forms) PL/SQL program unit's (package specification or body or stand–alone procedure.) source code and compiled code together must be less than 64K. This implies that the source code for a program unit cannot exceed 10K.
Adding New Procedures to Existing Packages
When you add new procedures or functions to existing packages (either stored in the database or in Oracle Forms libraries), you should usually add them to the end of the package (and package specification). If you add new procedures to the middle of the package specification and package, you must regenerate every form that references the package, or those forms may get ORA–4062 errors.
Using Field Names in Client–Side PL/SQL Packages
Always specify field names completely by including the block name i.e., BLOCK.FIELD_NAME instead of just FIELD_NAME. Just specifying the field name makes Oracle Forms to scan through the entire list of fields for each block in the form to locate the specified field and check if its name is ambiguous, potentially degrading your form performance. If the block name is included, Oracle Forms searches only the fields in that block and stops when it finds a match. Moreover, if you ever add more blocks, existing code continues to work since you specified field names unambiguously.



PL/SQL Coding Standards

- Pass field names to procedures and use COPY to update field values instead of using IN OUT or OUT parameters.
- Use DEFAULT instead of ":=" when declaring default values for parameters.
- Use ":=" instead of DEFAULT when declaring values for constant variables.

August 14, 2014 | Proprietary and Confidential | - 13 -

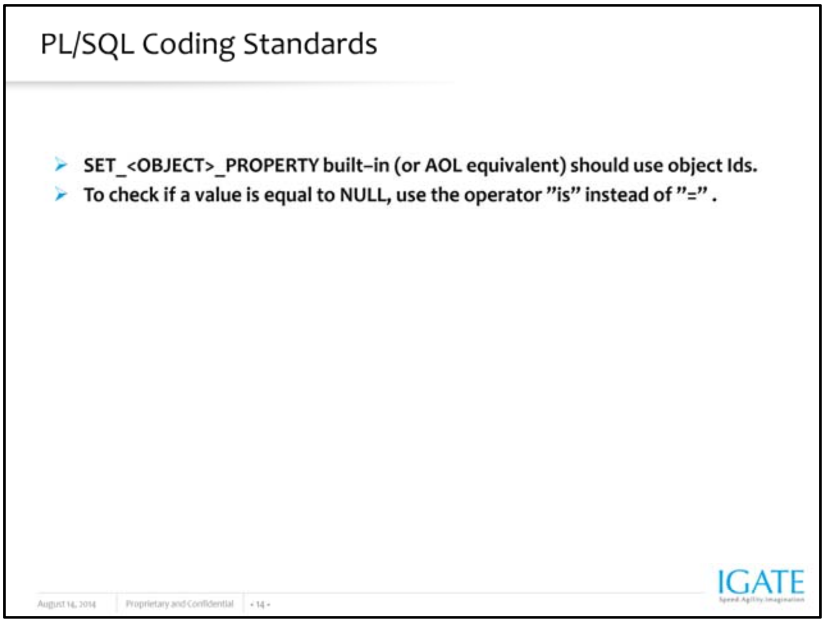
Field Names in Procedure Parameters
Pass field names to procedures and use COPY to update field values instead of using IN OUT or OUT parameters. This method prevents a field from being marked as changed whether or not you actually modify it in your procedure. Any parameter declared as OUT is always written to when the procedure exits normally.
For example, declare a procedure as

```
test(my_var VARCHAR2 IN)
```

and call it as test('block.field') instead of declaring the procedure as

```
test(my_var VARCHAR2 IN OUT)
```

and calling it as test(:block.field).
Using DEFAULT
Use DEFAULT instead of ":=" when declaring default values for your parameters. DEFAULT is more precise because you are defaulting the values; the calling procedure can override the values.
Conversely, use ":=" instead of DEFAULT when declaring values for your constant variables. Using ":=" is more precise because you are assigning the values, not defaulting them; the values cannot be overridden.

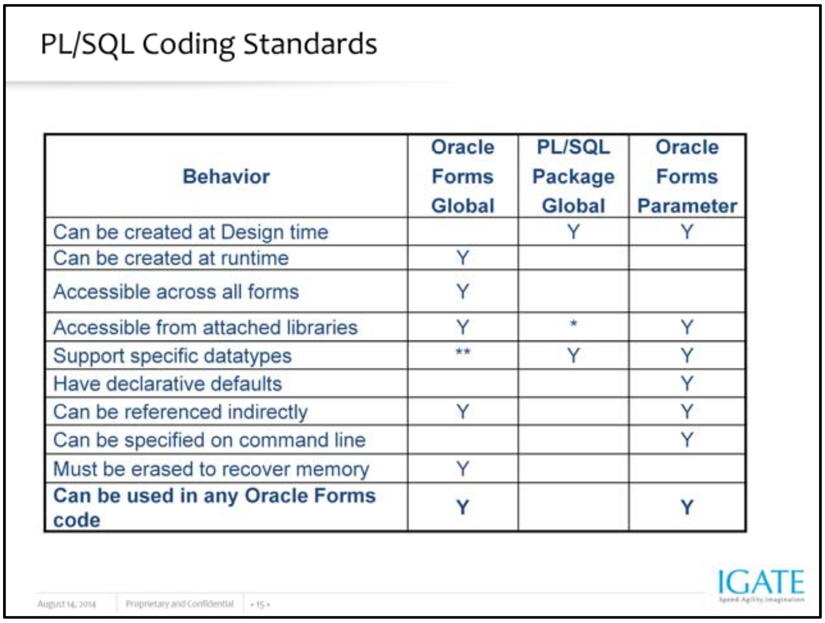


PL/SQL Coding Standards

- SET_<OBJECT>_PROPERTY built–in (or AOL equivalent) should use object Ids.
- To check if a value is equal to NULL, use the operator ”is” instead of ”=” .

August 14, 2014 | Proprietary and Confidential | - 14 -

Use Object IDs
Any code that changes multiple properties of an object using the SET_<OBJECT>_PROPERTY built–in (or the Oracle Application Object Library equivalent) should use object IDs. First use the appropriate FIND_<OBJECT> built–in to get the ID, then pass the ID to the SET_<OBJECT>_PROPERTY built–in. Consider storing the ID in a package global so that its retrieved only once while the form is running.
Handling NULL Value Equivalence
handle NULL values very carefully in PL/SQL. For example, if a := NULL and b := NULL, the expression (a = b) evaluates to FALSE. In any ”=” expression where one of the terms is NULL, the whole expression will resolve to FALSE.
For this reason, to check if a value is equal to NULL, you must use the operator ”is” instead. If you’re comparing two values where either of the values could be equal to NULL, you should write the expression like this: ((a = b) or ((a is null) and (b is null))



PL/SQL Coding Standards

Behavior	Oracle Forms Global	PL/SQL Package Global	Oracle Forms Parameter
Can be created at Design time		Y	Y
Can be created at runtime	Y		
Accessible across all forms	Y		
Accessible from attached libraries	Y	*	Y
Support specific datatypes	**	Y	Y
Have declarative defaults			Y
Can be referenced indirectly	Y		Y
Can be specified on command line			Y
Must be erased to recover memory	Y		
Can be used in any Oracle Forms code	**Y**		**Y**

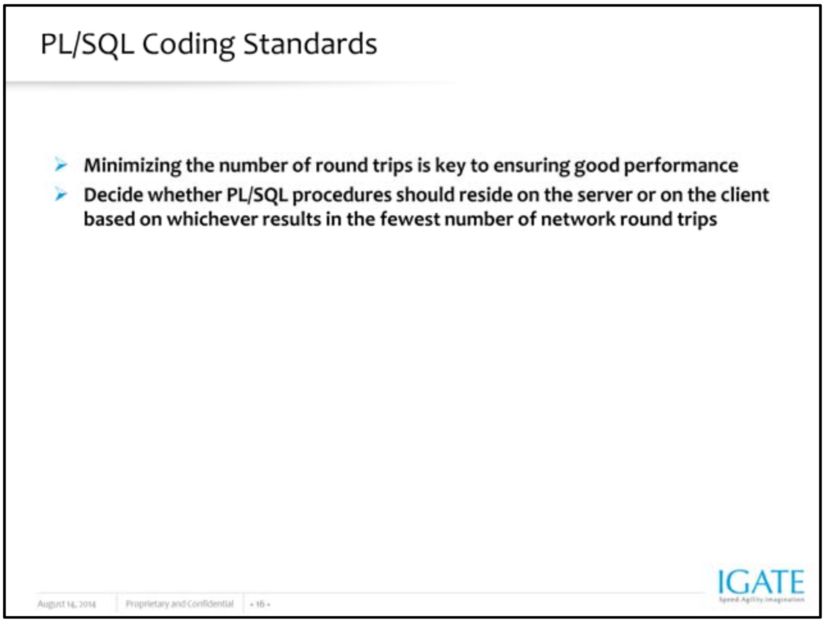
August 14, 2014 | Proprietary and Confidential | - 15 -

Global Variables
Oracle Forms Developer and PL/SQL support different types of global variables:

Oracle Forms Global: a variable in the ”global” pseudo–block of a form
PL/SQL Package Global: a global defined in the specification of a package
Oracle Forms Parameter: a variable created within the Oracle Forms Designer as a Parameter

The above table lists the characteristics of each type of variable, and enables you to select the type most appropriate for your code.

* A package variable defined in a form is not visible to any attached library; a variable defined in an attached library is visible to the form. (An Oracle Forms Global is visible to an attached library)
** Always CHAR(255)



PL/SQL Coding Standards

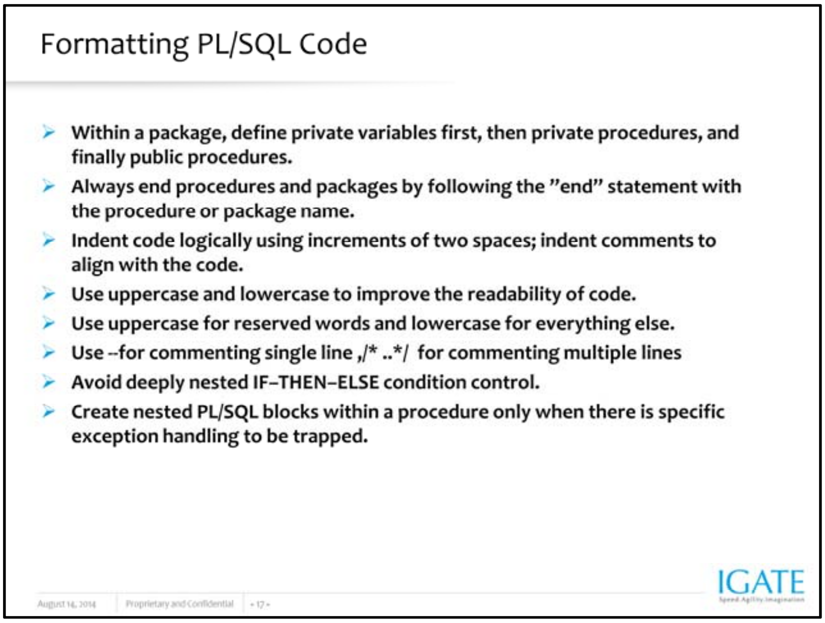
- Minimizing the number of round trips is key to ensuring good performance
- Decide whether PL/SQL procedures should reside on the server or on the client based on whichever results in the fewest number of network round trips

August 14, 2014 | Proprietary and Confidential | - 16 -

Database Server Side versus Client Side
Performance is a critical aspect of any application. Because network round trips are very costly in a typical internet environment, minimizing the number of round trips is key to ensuring good performance.
You should decide whether your PL/SQL procedures reside on the server or on the client based on whichever results in the fewest number of network round trips. Here are some guidelines:

Procedures that call Oracle Forms built–ins (more generally, client built–ins) must reside on the client
Procedures that reference fields directly, either as :block.field or via NAME_IN/COPY, must reside on the client. You can avoid referencing fields directly by accepting field values or names as parameters to your PL/SQL procedures, which also improves your code's modularity
If a procedure contains three or more SQL statements, or becomes very complicated, the procedure usually belongs on the server
Procedures that perform no SQL and that need no database access should reside wherever they are needed

If a procedure is called from the server, it must reside on the server.
If a procedure is called from both client and server, it should be defined in both places, unless the procedure is very complicated and double maintenance is too costly. In the latter case, the procedure should reside on the server.



Formatting PL/SQL Code

- Within a package, define private variables first, then private procedures, and finally public procedures.
- Always end procedures and packages by following the "end" statement with the procedure or package name.
- Indent code logically using increments of two spaces; indent comments to align with the code.
- Use uppercase and lowercase to improve the readability of code.
- Use uppercase for reserved words and lowercase for everything else.
- Use --for commenting single line ,/* ..*/ for commenting multiple lines
- Avoid deeply nested IF–THEN–ELSE condition control.
- Create nested PL/SQL blocks within a procedure only when there is specific exception handling to be trapped.

August 14, 2014 | Proprietary and Confidential | - 17 -

Formatting PL/SQL Code

This section contains recommendations for formatting PL/SQL code.

Within a package, define private variables first, then private procedures, and finally public procedures.

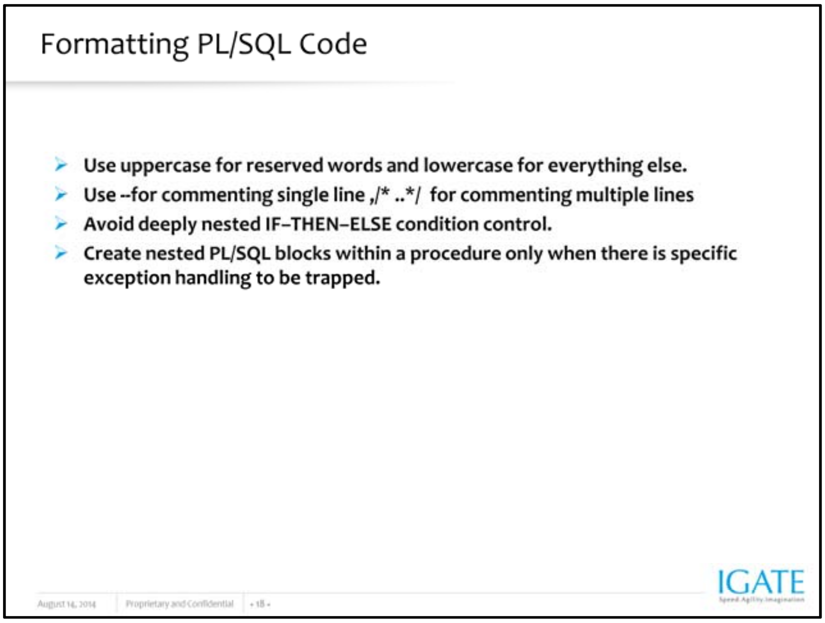
Always end procedures and packages by following the "end" statement with the procedure or package name to help delineate procedures.

Indent code logically. Using increments of two spaces provides an easy way to track your nested cases.

Indent SQL statements as follows:

Example

```
DECLARE
  CURSOR employees IS
    SELECT empno
    FROM emp
    WHERE deptno = 10
          AND ename IN ('WASHINGTON', 'MONROE')
          AND mgr = 2701;
```



Formatting PL/SQL Code

- Use uppercase for reserved words and lowercase for everything else.
- Use --for commenting single line ,/* ..*/ for commenting multiple lines
- Avoid deeply nested IF–THEN–ELSE condition control.
- Create nested PL/SQL blocks within a procedure only when there is specific exception handling to be trapped.

August 14, 2014 | Proprietary and Confidential | - 18 -

IGATE

Use ”– –” to start comments so that you can easily comment out large portions of code during debugging with ”/* ... */”

Indent comments to align with the code being commented

When commenting out code, start the comment delimiter in the leftmost column. When the code is clearly no longer needed, remove it entirely

Use uppercase and lowercase to improve the readability of your code (PL/SQL is case–insensitive). As a guideline, use uppercase for reserved words and lowercase for everything else

Avoid deeply nested IF–THEN–ELSE condition control. Use IF–THEN–ELSIF instead

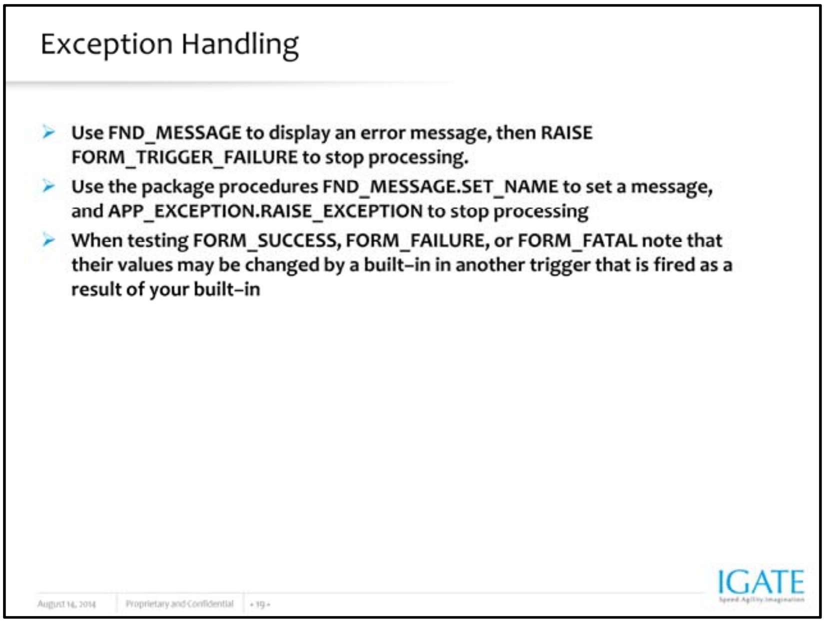
Example of Bad Style

```
IF ... THEN ... ELSE
          IF ... THEN ... ELSE
                              IF ... THEN ... ELSE
                              END IF
          END IF
END IF;
```

Example of Good Style

Only create nested PL/SQL blocks (BEGIN/END pairs) within a procedure when there is specific exception handling you need to trap

```
IF ... THEN ...
ELSIF ... THEN ...
ELSIF ... THEN ...
ELSIF ... THEN ...
ELSE ...
END IF;
```



Exception Handling

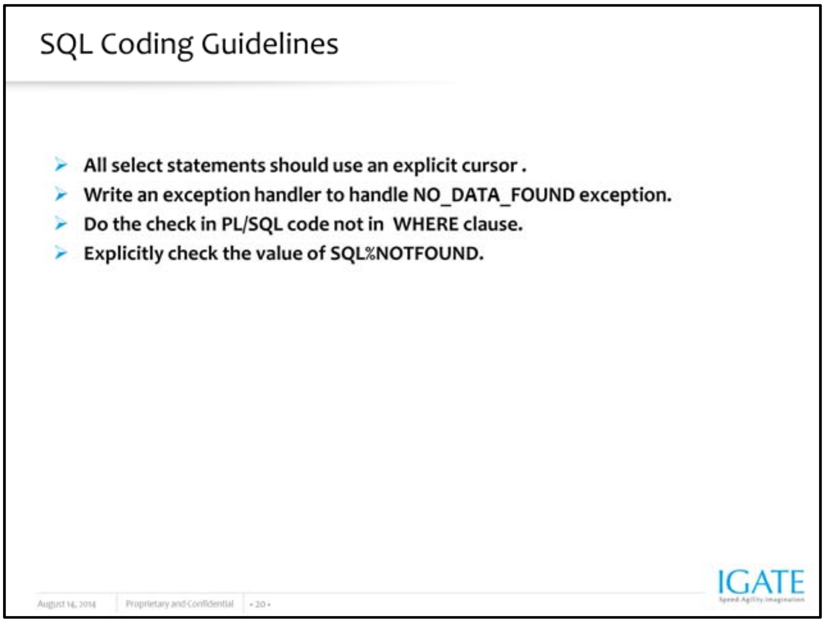
- Use FND_MESSAGE to display an error message, then RAISE FORM_TRIGGER_FAILURE to stop processing.
- Use the package procedures FND_MESSAGE.SET_NAME to set a message, and APP_EXCEPTION.RAISE_EXCEPTION to stop processing
- When testing FORM_SUCCESS, FORM_FAILURE, or FORM_FATAL note that their values may be changed by a built–in in another trigger that is fired as a result of your built–in

August 14, 2014 | Proprietary and Confidential | - 19 -

Errors in Oracle Forms PL/SQL
If a failure occurs in Oracle Forms PL/SQL and you want to stop further processing, use FND_MESSAGE to display an error message, then RAISE FORM_TRIGGER_FAILURE to stop processing:

```
IF (error_condition) THEN
  fnd_message.set_name(appl_short_name,
message_name);
  fnd_message.error;
  RAISE FORM_TRIGGER_FAILURE;
END IF;
```

Note: RAISE FORM_TRIGGER_FAILURE causes processing to stop quietly without error notification, you must display any messages yourself using FND_MESSAGE before raising the exception.



SQL Coding Guidelines

- All select statements should use an explicit cursor .
- Write an exception handler to handle NO_DATA_FOUND exception.
- Do the check in PL/SQL code not in WHERE clause.
- Explicitly check the value of SQL%NOTFOUND.

August 14, 2014 | Proprietary and Confidential | - 20 -

IGATE

Follow these guidelines for all SQL that you code:

Use ”select from DUAL” instead of ”select from SYS.DUAL”. Do not use SYSTEM.DUAL

All SELECT statements should use an explicit cursor. Implicit SELECT statements actually cause 2 fetches to execute: one to get the data, and one to check for the TOO_MANY_ROWS exception. Avoid this by FETCHing just a single record from an explicit cursor

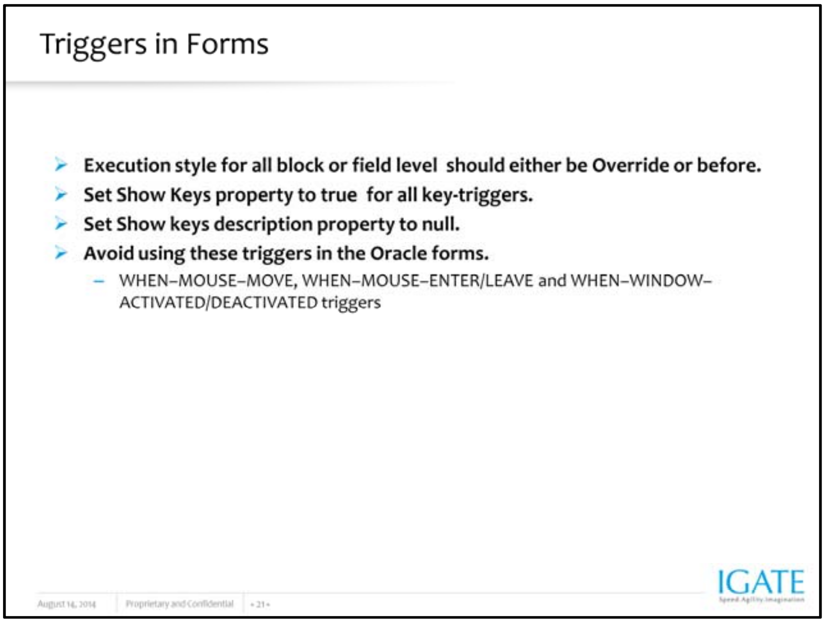
To SELECT into a procedure parameter, declare the parameter as IN OUT, whether or not the parameter value is referenced, unless the parameter is a field

A single–row SELECT that returns no rows raises the exception NO_DATA_FOUND. An INSERT, UPDATE, or DELETE that affects no rows does not raise an exception. Explicitly check the value of SQL%NOTFOUND if no rows is an error.

To handle NO_DATA_FOUND exceptions, write an exception handler. Do not code COUNT statements to detect the existence of rows unless that is your only concern.

When checking the value of a field or PL/SQL variable against a literal, do the check in PL/SQL code, not in a WHERE clause.

Do not check for errors due to database integrity problems. For example, if a correct database would have a table SYS.DUAL with exactly one row in it, you do not need to check if SYS.DUAL has zero or more than one row or if SYS.DUAL exists



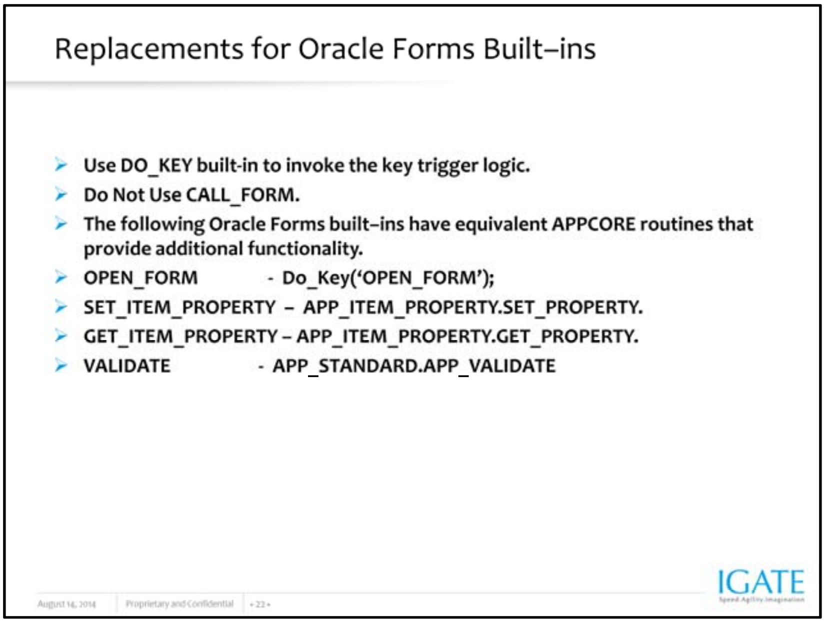
Triggers in Forms

- Execution style for all block or field level should either be Override or before.
- Set Show Keys property to true for all key-triggers.
- Set Show keys description property to null.
- Avoid using these triggers in the Oracle forms.
  - WHEN–MOUSE–MOVE, WHEN–MOUSE–ENTER/LEAVE and WHEN–WINDOW–ACTIVATED/DEACTIVATED triggers

August 14, 2014 | Proprietary and Confidential | - 21 -

Follow these general rules for triggers in the forms.
Execution Style
The 'Execution Style' for all block or field level triggers should either be Override or Before. In general, use style Before, since usually the form–level version of the trigger should also fire. The exception is if there is a flexfield call in the form–level POST–QUERY trigger, but you reset the query status of the block in the block level POST–QUERY. In that case, the block–level POST–QUERY should use Execution Style After.
KEY– Trigger Properties
Set the "Show Keys" property to True for all KEY– triggers you code, except those that you are disabling (which should have "Show Keys" set to False).
Always set the "Show Keys Description" property to NULL.
WHEN–CREATE–RECORD in Dynamic Query–Only Mode
The WHEN–CREATE–RECORD trigger fires even when the block does not allow inserts. Check if the block allows insert if you have logic in this trigger and your block may dynamically have insert–allowed "FALSE":

```
IF GET_ITEM_PROPERTY('<BLOCK>',
INSERT_ALLOWED) = FALSE THEN
  null;
ELSE
  <your logic here>;
END IF;
```



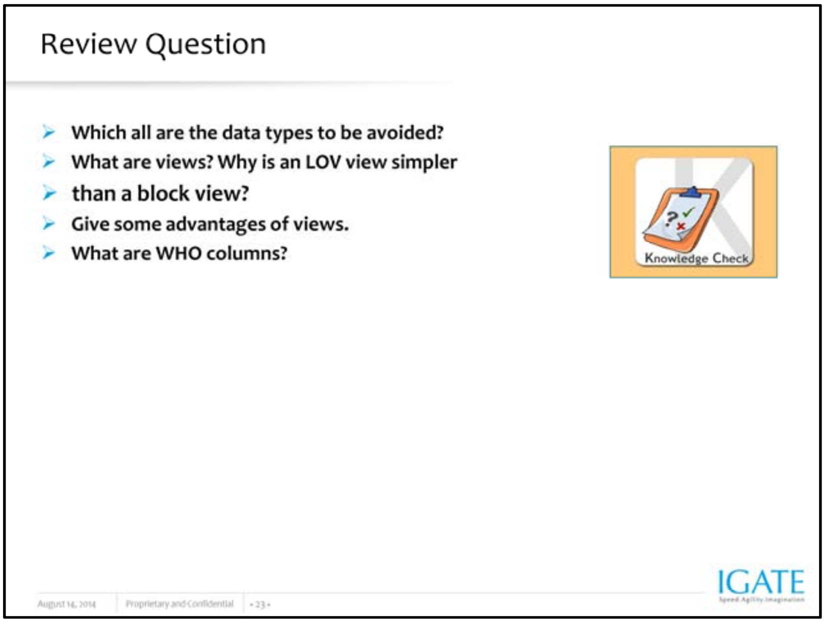
Replacements for Oracle Forms Built–ins

- Use DO_KEY built-in to invoke the key trigger logic.
- Do Not Use CALL_FORM.
- The following Oracle Forms built–ins have equivalent APPCORE routines that provide additional functionality.
- OPEN_FORM - Do_Key('OPEN_FORM');
- SET_ITEM_PROPERTY – APP_ITEM_PROPERTY.SET_PROPERTY.
- GET_ITEM_PROPERTY – APP_ITEM_PROPERTY.GET_PROPERTY.
- VALIDATE - APP_STANDARD.APP_VALIDATE

August 14, 2014 | Proprietary and Confidential | - 22 -

These standards require that certain built–ins be avoided entirely, or ”wrapper” routines be called in their place. For many built–ins, there are multiple methods of invocation. Call the built–in directly, to give the standard forms behavior. For some built–ins, there are standard Oracle Applications behaviors, which are invoked by calling APP_STANDARD.EVENT.
Many of these built–ins have a key and a KEY– trigger associated with them. If there is any additional logic which has been added to the KEY– trigger that you want to take advantage of, you can invoke the trigger by using the DO_KEY built–in. This is the same result you would get if the user pressed the associated key.

Do not use CALL_FORM Oracle Forms built–in.
This built–in is incompatible with OPEN_FORM, which is used by Oracle Applications routines. Use FND_FUNCTION.EXECUTE instead of either CALL_FORM or OPEN_FORM whenever a form to be opened programmatically. Using FND_FUNCTION.EXECUTE allow the form to open without bypassing Oracle Applications security, and takes care of finding the correct directory path for the form.



Review Question

- Which all are the data types to be avoided?
- What are views? Why is an LOV view simpler
- than a block view?
- Give some advantages of views.
- What are WHO columns?

Knowledge Check

August 14, 2014 | Proprietary and Confidential | - 23 -

IGATE Speed.Agility.Imagination