مقدمة في الـ Triggers
Introduction to Triggers

❖ **Trigger :** هي عبارة عن مجموعة أكواد برمجية يتم تنفيذها عند حدوث حدث معين لأداء وظيفة ما وأسم Trigger يدل علي الوقت أو الحدث الذي سيتم عنده تنفيذ الأكواد البرمجية المكتوبة داخل هذا الـ Trigger وكل Trigger يكون مرتبط بحدث معين وهذه الأحداث يتم تعريفها من قبل برنامج Form Builder وهذه الأحداث تغطي العمليات التالية

1. الأحداث المتعلقة بالاستعلام
2. أحداث إدخال البيانات والتحقق منها
3. التنقل بين مكونات البرنامج Module
4. حركة الماوس
5. عمليات التفاعل بين العناصر في البرنامج Module
6. الأحداث الداخلية داخل البرنامج Module
7. الأخطاء والرسائل

❖ **المكونات الأساسية لـ Trigger :**

- نوع Trigger : فيه يتم تحديد الحدث الذي سيتم عنده تنفيذ هذه الأكواد البرمجية
- كود Trigger : فيه يتم كتابة الكود البرمجي الذي سيتم تنفيذه
- مدي Trigger : لتحديد المدى الذي سيتم تنفيذه علي عنصر محدد فقط أو علي بلك بيانات محددة فقط أو علي البرنامج Module كله

❖ **مدي Trigger :** هذا المدى أو المجال يؤثر علي المستويات التالية

1. Form : يتم تنفيذه في أحداث خاصة بالـ Form ويؤثر في كل مكونات الـ Form
2. Block : يتم تنفيذه في أحداث خاصة بهذا البلك فقط بمعني عندما يكون هذا البلك هو البلك الحالي
3. Item : يتم تنفيذه في أحداث خاصة بهذا الـ Item فقط بمعني عندما يكون هذا الـ Item هو الـ Item الحالي حيث بعض Triggers تكون خاصة بمستويات معينة ولا يمكن تطبيقها علي مستويات أخري مثل Post-Query Trigger لا يمكن تنفيذه علي مستوي الـ Item لأن هذا الحدث يتم تنفيذه بصفة عامة علي بلك محدد أو علي Form

❖ **أنواع Trigger** : إن نوع الـ Trigger يحدد الحدث الذي سيحدث فيه الـ Trigger حيث يوجد أكثر من 100 نوع Built in Trigger تم إنشاؤهم مسبقا أي جاهزة داخل برنامج Form Builder كما أن كل Built in Trigger المركب داخليا يكون مرتبط بالحدث وغالبا ما يحتوي اسم الـ Trigger علي الرمز (_) ويكون الجزء الأول من الاسم يدل علي نوع Trigger كما يلي

- **key_** : هذا النوع ينطلق عند الضغط علي مفتاح من مفاتيح الوظائف حيث يمكن توظيف مفاتيح من لوحة المفاتيح لأداء وظائف محددة في الـ Trigger
- **On _**: هذا النوع ينطلق أثناء حدوث الحدث
- **Pre _**: هذا النوع ينطلق قبل الحدث مباشرة
- **Post _**: هذا النوع ينطلق بعد الحدث مباشرة
- **When _**: هذا النوع ينطلق بعد الحدث

❖ **كود Trigger :** يحدد الوظائف التي سيقوم بها الـ Trigger عند انطلاقه حيث يكتب هذا الكود باستخدام محرر PL/SQL والجمل البرمجية
التي يتم كتابتها داخل الـ Trigger تكون كما يلي

1. جمل PL/SQL القياسية مثل جمل التحكم وجمل التخصيص وغيرها
2. جمل SQL (Statement)

3. برامج استدعاء أسماء المستخدمين (User subprograms)
4. برامج استدعاء البرامج الفرعية التي تم إنشاؤها مسبقا وهي تسمي Built in subprogram

❖ **استخدام الـ Built in Subprograms :**

إن برنامج الـ Form Builder يوفر مجموعة من البرامج الفرعية المعرفة مسبقا وكل مجموعة من هذه البرامج تسمي Package وهذه البرامج الفرعية يمكن أن تكون في شكل إجراء Procedure أو دالة Function والبرامج الفرعية هي نوعان كما يلي

1. **Standard Extensions Package : هذا النوع من البرامج الفرعية يمكن استدعائها مباشرة وكتابتها من دون أن يسبقها Package**
2. **الأنواع الاخري من الـ Package الداخلية : هذا النوع من البرامج الفرعية يتطلب اسم الـ Package الذي ينتمي إليه هذا البرنامج الفرعي**

❖ **أين ومتي يمكن استخدام البرامج الفرعية الداخلية Built in Subprograms :**

يمكن استدعاء أي برنامج فرعي داخلي في أي Trigger ولكن بعض البرامج الفرعية تكون وظيفتها غير متاحة في موضع معين وفي Trigger معين ولذلك تم تقسيمها إلي مجموعتين

1. **Unrestricted Built-ins : هذا النوع من البرامج الفرعية يستخدم في الـ Triggers التي تستخدم في التنقل في الفورم وفي أي نوع من أنواع الـ Triggers**
2. **Restricted Built-ins : هذا النوع من البرامج الفرعية لا يستخدم في الـ Triggers التي تستخدم في التنقل في الفورم ويستخدم في أي نوع من أنواع الـ Triggers الأخرى**

❖ **استخدام البرامج الفرعية الداخلية : يتم كما يلي**

1. **Open PL/SQL code**
2. **Select built-in**
3. **Past names or arguments**
4. **Modify pasted code**

❖ **كيفية استدعاء البرامج الفرعية الداخلية Built in Subprograms :**

عند كتابة أي Trigger أو وحدة برمجية فإن برنامج Form Builder يمكننا من استخدام البرامج الفرعية الداخلية ويمكن نسخ اسمها أو اسمها مع متغيراتها إلي الـ Trigger أو الوحدة البرمجية كما يلي

1. **نضع مؤشر الكتابة في المكان الذي سيتم نسخ اسم البرنامج الفرعي فيه داخل الـ Trigger أو الوحدة البرمجية ثم ننتقل إلي شاشة Object Navigator وننتقل إلي أيقونة Built in Package ثم نختار اسم البرنامج الفرعي أو الدالة المراد نسخها ثم من قائمة Navigator نختار أمر Past name في حالة نسخ اسم البرنامج الفرعي فقط أو نختار أمر Past Argument لنسخ اسم البرنامج مع المتغيرات الموجودة فيه سنلاحظ أن اسم البرنامج الفرعي تم كتابته في المكان المحدد له مسبقا**
2. **يمكن استدعاء البرامج الفرعية ونحن في شاشة PL/SQL من قائمة Program ثم نختار أمر Syntax Palette ثم نختار صفحة Built In ومن ثم نحدد اسم البرنامج الفرعي وبعد ذلك نضغط علي مفتاح Insert**

❖ **أمثلة علي البرامج الفرعية والدوال الأكثر استخداما :**

1. **ENTER_QUERY procedure : يستخدم هذا البرنامج الفرعي لإدخال الاستعلام حيث يجعل المستخدم يكتب شرطا للاستعلام**
2. **EXECUTE_QUERY procedure : يستخدم هذا البرنامج الفرعي لتنفيذ الاستعلام**
3. **EXIT_FORM procedure : يستخدم هذا البرنامج الفرعي للخروج من الفورم الحالية**
4. **GET_ITEM_PROPERTY function : تستخدم هذه الدالة لاستدعاء قيمة خاصية معينة وإمكانية السؤال عن قيمة خاصية معينة**
5. **GO_BLOCK procedure : يستخدم هذا البرنامج الفرعي للانتقال إلي بلك بيانات Data Block معين**
6. **GO_ITEM procedure : يستخدم هذا البرنامج الفرعي للانتقال إلي Item أو عنصر معين**
7. **SHOW_VIEW procedure : يستخدم هذا البرنامج الفرعي لإظهار Canvas معينة**
8. **HIDE_VIEW procedure : يستخدم هذا البرنامج الفرعي لإخفاء Canvas معينة**
9. **MESSAGE procedure : يستخدم هذا البرنامج الفرعي لإظهار رسالة معينة**
10. **SET_ITEM_PROPERTY procedure : يستخدم هذا البرنامج الفرعي لتعديل قيمة خاصية معينة لعنصر محدد**
11. **SHOW_LOV procedure : يستخدم هذا البرنامج لإظهار LOV معينة والمؤشر موجود في أي عنصر من عناصر البرنامج**
12. **LIST_VALUE procedure : يستخدم هذا البرنامج لإظهار LOV معينة عند نقل المؤشر إلي العنصر المحدد له LOV مسبقا**
13. **show_alert**

❖ **أمثلة على الـ Triggers :**

مثال 1

```
declare

        id number;



begin

        id:=show_alert('al1');
        if id=alert_button1 then

        DELETE_RECORD;
        commit;
        else

                null;
                end if;

        end;
```

Level of triggers

1 - form **2** - block **3** - item

اى (Built in (Procedure / function يجب معرفة التالى عنها

١ – عملها ٢ – هل هى Procedure أم Function
٣ – لو Function ما هى نوعية الـ Data type اللى هترجع
٤ – عدد الـ argument وكذلك نوع بياناتها
٥ – معرفة الجملة Syntax وكذلك مثال عليها
٦ – هل يوجد عليها قيود Restricted (لا تعمل فى الـ Enter_query) أم العكس

ملحوظة هامة : يمكن عمل زر وكتابة Trigger بداخلة وذلك بان
نضغط على الزر R- click ثم نختار smart trigger ثم نختار النوع المناسب
Enter_query , execute_query , exit_query , commit , clear_form

ملحوظة هامة

إذا تم عمل أزرار لكلا من ... , enter_query , save
فأننا لا نكرر الكود مرة أخرى ولكن نكتب تحت كل زر الـ Bulti in الخاصة بة

Save : do_key (‘commit_form ‘);
Exit : do_key (‘exit_form ‘);
Execute : do_key (‘execute_query‘);
Enter : do_key (‘enter_query‘);

ويجب أن تكون الـ mouse navigator = no لهذه الأزرار bush button

1. Block Processing Triggers

1.1 When-Clear-Block

Fires just before Oracle Forms clears the data from the current block. Note that the **`When-Clear-Block`** trigger does not fire when Oracle Forms clears the current block during the **`CLEAR_FORM`** event.

1.2 When-Create-Record

Fires whenever Oracle Forms creates a new record. For example, when the operator presses the [Insert] key, or navigates to the last record in a set while scrolling down, Oracle Forms fires this trigger.

1.3 When-Remove-Record

Fires whenever the operator or the application clears or deletes a record.

2 Interface Event Triggers

2.1 When-Button-Pressed

Fires when an operator selects a button, either by way of a key, or by clicking with a mouse.

2.2 When-Checkbox-Changed

Fires whenever an operator changes the state of a check box, either by clicking with the mouse, or through keyboard interaction.

2.3 When-List-Changed

Fires when an operator selects a different element in a list item or de-selects the currently selected element. In addition, if a **`When-List-Changed`** trigger is attached to a combo box style list item, it fires each time the operator enters or modifies entered text.

2.4 When-Radio-Changed

Fires when an operator selects a different radio button in a radio group, or de-selects the currently selected radio button, either by clicking with the mouse, or by way of keyboard selection commands.

2.5 When-Timer-Expired

Fires when a timer expires.

Note: Timers are created programmatically by calling the **CREATE_TIMER** built-in procedure.

3 Navigational Triggers

3.1 Post-Block

Fires during the Leave the Block process.

3.2 Post-Form

Fires during the Leave the Form process, when a form is exited.

3.3 Post-Record

Fires during the Leave the Record process. Specifically, the Post-Record trigger fires whenever the operator or the application moves the input focus from one record to another. The Leave the Record process can occur as a result of numerous operations, including **INSERT_RECORD**, **DELETE_RECORD**, **NEXT_RECORD**, **CREATE_RECORD, NEXT_BLOCK**, **PREVIOUS_BLOCK**, etc.

3.3 Post-Text-Item

Fires during the Leave the Item process for a text item. Specifically, this trigger fires when the input focus moves from a text item to any other item.

3.3 Post-Insert

Fires during the Post and Commit Transactions process, just after a record is inserted. It fires once for each record that is inserted into the database during the commit process

3.4 Pre-Block

Fires during the Enter the Block process, during navigation from one block to another.

3.5 Pre-Form

Fires during the Enter the Form event, at form startup.

3.6 Pre-Record

Fires during the Enter the Record process, during navigation to a different record.

3.7 Pre-Text-Item

Fires during the Enter the Item process, during navigation from an item to a text item.

3.8 When-Form-Navigate

Fires whenever any peer form navigation takes place.

3.9 When-New-Block-Instance

Fires when the input focus moves to an item in a block that is different than the block that previously had input focus. Specifically, it fires after navigation to an item, when Oracle Forms is ready to accept input in a block that is different than the block that previously had input focus.

3.10 When-New-Form-Instance

At form start-up, Oracle Forms navigates to the first navigable item in the first navigable block. A When-New-Form-Instance trigger fires after the successful completion of any navigational triggers that fire during the initial navigation sequence. This trigger does not fire when control returns to a calling form from a

called form. In a multiple-form application, this trigger does not fire when focus changes from one form to another.

3.11 When-New-Item-Instance

Fires when the input focus moves to an item. Specifically, it fires after navigation to an item, when Oracle Forms is ready to accept input in an item that is different than the item that previously had input focus.

3.12 When-New-Record-Instance

Fires when the input focus moves to an item in a record that is different than the record that previously had input focus. Specifically, it fires after navigation to an item in a record, when Oracle Forms is ready to accept input in a record that is different than the record that previously had input focus.

Fires whenever Oracle Forms instantiates a new record

4 Query-Time Triggers

4.1 Post-Query

When a query is open in the block, the Post-Query trigger fires each time Oracle Forms fetches a record into a block. The trigger fires once for each record placed on the block's list of records.

4.2 Pre-Query

Fires during Execute Query or Count Query processing, just before Oracle Forms constructs and issues the **`SELECT`** statement to identify rows that match the query criteria.

5- Transactional Triggers

5.1 On-Commit

Fires whenever Oracle Forms would normally issue a database commit statement to finalize a transaction. By default, this operation occurs after all records that have been marked as updates, inserts, and deletes have been posted to the database

5.2 On-Delete

Fires during the Post and Commit Transactions process. Specifically, it fires after the Pre-Delete trigger fires and before the Post-Delete trigger fires, replacing the actual database delete of a given row. The trigger fires once for each row that is marked for deletion from the database.

5.3 On-Insert

Fires during the Post and Commit Transactions process. Specifically, it fires after the Pre-Insert trigger fires and before the Post-Insert trigger fires, when Oracle Forms would normally insert a record in the database. It fires once for each row that is marked for insertion into the database.

5.4 On-Logon

Fires once per logon when Oracle Forms normally initiates the logon sequence.

5.5 On-Logout

Fires when Oracle Forms normally initiates a logout procedure.

5.6 On-Update

Fires during the Post and Commit Transactions process. Specifically, it fires after the Pre-Update trigger fires and before the Post-Update trigger fires, when Oracle Forms would normally update a record in the database. It fires once for each row that is marked for update in the form.

5.7 Post-Change

Fires when any of the following conditions exist:

• The Validate the Item process determines that an item is marked as Changed and is not NULL.

• An operator returns a value into an item by making a selection from a list of values, and the item is not NULL.

• Oracle Forms fetches a non-NULL value into an item. In this case, the When-Validate-Item trigger does not fire. If you want to circumvent this situation and

effectively get rid of the Post-Change trigger, you must include a Post-Query trigger in addition to your When-Validate-Item trigger. See "Usage Notes" below.

5.8 Post-Delete

Fires during the Post and Commit Transactions process, after a row is deleted. It fires once for each row that is deleted from the database during the commit process.

5.9 Post-Insert

Fires during the Post and Commit Transactions process, just after a record is inserted. It fires once for each record that is inserted into the database during the commit process.

5.10 Post-Update

Fires during the Post and Commit Transactions process, after a row is updated. It fires once for each row that is updated in the database during the commit process.

5.11 Pre-Update

Fires during the Post and Commit Transactions process, before a row is updated. It fires once for each record that is marked for update.

5.12 Pre-Delete

Fires during the Post and Commit Transactions process, before a row is deleted. It fires once for each record that is marked for delete. Note: Oracle Forms creates a Pre-Delete trigger automatically for any master-detail relation that has the Master Deletes property set to Cascading.

Pre-Insert Fires during the Post and Commit Transactions process, before a row is inserted. It fires once for each record that is marked for insert.

5.13 Pre-Commit

Fires once during the Post and Commit Transactions process, before Oracle Forms processes any records to change. Specifically, it fires after Oracle Forms determines that there are inserts, updates, or deletes in the form to post or commit. The trigger does not fire when there is an attempt to commit, but validation determines that there are no changed records in the form.

5.14 Pre-insert

6- Validation Triggers

6.1 When-Validate-Item

Fires during the Validate the Item process. Specifically, it fires as the last part of item validation for items with the New or Changed validation status.

6.2 When-Validate-Record

Fires during the Validate the Record process. Specifically, it fires as the last part of record validation for records with the New or Changed validation status.

Built-in Subprograms Tables

Alert Built-ins

SET_ALERT_BUTTON_PROPERTY

3.2.1.4 SET_ALERT_PROPERTY

3.2.1.5 SHOW_ALERT

Example

```
declare


v number;
begin
set_alert_property('go',title,'fffffffffff');
set_alert_property('go',alert_message_text,'fffffffffff');
set_alert_button_property('go',alert_button1,label,'ok');

set_alert_button_property('go',alert_button2,label,'cancel');

v:=show_alert('go')'
end;
```

3.2.2 Application Built-ins

3.2.2.2 GET_APPLICATION_PROPERTY

Example

```
declare

a varchar2(100)
b number
begin

a:=get_application_property(current_form);
message(a);
b:=instr(a,'\',-1);
a:=substr(a,1,b);
message(a);
new_form(a||'it3.fmx');
end
```

3.2.3 Block Built-ins

3.2.3.4 GET_BLOCK_PROPERTY

3.2.3.5 GO_BLOCK

3.2.3.9 SET_BLOCK_PROPERTY

3.2.4 Canvas and View Built-ins

3.2.4.5 HIDE_VIEW

3.2.4.11 SHOW_VIEW

3.2.5 Form Built-ins

3.2.5.3 CALL_FORM

CALL_FORM(formmodule_name);
CALL_FORM(formmodule_name, display);
CALL_FORM(formmodule_name, display, switch_menu);
CALL_FORM(formmodule_name, display, switch_menu, query_mode)
CALL_FORM(formmodule_name, display, switch_menu, query_mode, paramlist_id);
CALL_FORM(formmodule_name, display, switch_menu, query_mode, paramlist_name);

3.2.5.11 EXIT_FORM

Syntax: XIT_FORM; E
EXIT_FORM(commit_mode);
EXIT_FORM(commit_mode, rollback_mode);

3.2.5.19 NEW_FORM

Syntax: NEW_FORM(formmodule_name);
NEW_FORM(formmodule_name, rollback_mode);
NEW_FORM(formmodule_name, rollback_mode, query_mode);
NEW_FORM(formmodule_name, rollback_mode, query_mode, paramlist_id);
NEW_FORM(formmodule_name, rollback_mode, query_mode, paramlist_name);

3.2.5.20 OPEN_FORM

Syntax: OPEN_FORM(form_name);
OPEN_FORM(form_name,activate_mode);
OPEN_FORM(form_name,activate_mode,session_mode);
OPEN_FORM(form_name,activate_mode,session_mode, paramlist_name);
OPEN_FORM(form_name,activate_mode,session_mode, paramlist_id);

activate_mode:-activate, NO_activate
session_mode:- session, no_session

3.2.11 Multiple-form Application Built-ins

3.2.11.1 CLOSE_FORM

Syntax: CLOSE_FORM(form_name);
CLOSE_FORM(form_id);
Built-in Type: restricted procedure
Enter Query Mode: no

الـ Function المستخدمة لاستدعاء الـ Forma من اى مكان بالهارد

```
FUNCTION  GET_PATH RETURN VARCHAR2
IS
A VARCHAR2(50);
B NUMBER;
BEGIN
A := GET_APPLICATION_PROPERTY(CURRENT_FORM);
B := INSTR(A,'\',-1);
A := SUBSTR(A,1,B);
  RETURN (A) ;
END;
```

ولتنفيذها نكتب الأمر ;NEW_FORM(GET_PATH||'form name.fmx')

الـ get_application_property تقوم **بإرجاع المكان الموجود بة الـForm المفتوحة** وترجع null إذا تم فتح الـform الذى نناداى عليها من مكان أخر (تم فتحها مباشرة)
يمكن تنفيذ الـ procedure السابقة بكتابة الأمر **;call_FORM(GET_PATH||'form name.fmx')**
وذلك حسب ما نريد
تعنى فتح الـ Forma الجديدة بنافذة جديدة وغلق نافذة الـ forma الأصلية مفتوحة **new_form**
بينما الـ **call_form** تعنى فتح الـ Forma الجديدة بنافذة جديدة **وترك** نافذة الـ forma الأصلية مفتوحة

3.2.6 Item Built-ins

3.2.6.13 GET_ITEM_PROPERTY

Syntax: GET_ITEM_PROPERTY(item_id, property);
GET_ITEM_PROPERTY(item_name, property);

3.2.6.14 GET_RADIO_BUTTON_PROPERTY

Syntax: GET_RADIO_BUTTON_PROPERTY(item_id, button_name, property);
GET_RADIO_BUTTON_PROPERTY(item_name, button_name, property);

3.2.6.15 GO_ITEM

Syntax: GO_ITEM(item_id);
GO_ITEM(item_name);

3.2.6.25 SET_ITEM_PROPERTY

Syntax: SET_ITEM_PROPERTY(item_id, property, value);
SET_ITEM_PROPERTY(item_name, property, value);
SET_ITEM_PROPERTY(item_id, property, x, y);
SET_ITEM_PROPERTY(item_id, property, x);

SET_ITEM_PROPERTY(item_name, property, x, y);
SET_ITEM_PROPERTY(item_name, property, x);

3.2.6.26 SET_RADIO_BUTTON_PROPERTY

Syntax: SET_RADIO_BUTTON_PROPERTY(item_id, button_name, property, value);
SET_RADIO_BUTTON_PROPERTY(item_id, button_name, property, x, y);
SET_RADIO_BUTTON_PROPERTY(item_name, button_name, property, x, y);
SET_RADIO_BUTTON_PROPERTY(item_name, button_name, property,value);

3.2.7 List Item Built-ins

3.2.7.1 ADD_LIST_ELEMENT

Syntax: ADD_LIST_ELEMENT(list_name, list_index, list_label, list_value);
ADD_LIST_ELEMENT(list_id, list_index, list_label, list_value);

3.2.7.2 CLEAR_LIST

Syntax: CLEAR_LIST(list_id);
CLEAR_LIST(list_name);

3.2.7.3 DELETE_LIST_ELEMENT

Syntax: DELETE_LIST_ELEMENT(list_name, list_index);
DELETE_LIST_ELEMENT(list_id, list_index);

3.2.7.4 GET_LIST_ELEMENT_COUNT

Syntax: GET_LIST_ELEMENT_COUNT(list_id);
GET_LIST_ELEMENT_COUNT(list_name);

3.2.7.5 GET_LIST_ELEMENT_LABEL

Syntax: GET_LIST_ELEMENT_LABEL(list_id, list_name, list_index);
GET_LIST_ELEMENT_LABEL(list_name, list_index);

3.2.7.6 GET_LIST_ELEMENT_VALUE

Syntax: GET_LIST_ELEMENT_VALUE(list_id, list_index);
GET_LIST_ELEMENT_VALUE(list_name, list_index);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes

3.2.7.7 POPULATE_LIST

Syntax: POPULATE_LIST(list_id, recgrp_id);
POPULATE_LIST(list_id, recgrp_name);
POPULATE_LIST(list_name, recgrp_id);

POPULATE_LIST(list_name, recgrp_name);
Built-in Type unrestricted procedure
Enter Query Mode: yes

Example

```
declare
RR recordgroup;
n number;

begin

RR:=create_group_from_query('QQ','select  dname,to_char(deptno)from dept');
n:=populate_group(RR);
populate_list('dept.deptno','QQ');
GO_BLOCK('EMP');
EXECUTE_QUERY;

end;
```

```
2
 DECLARE
WHR   VARCHAR2(100);
XYZ   VARCHAR2(10);
BEGIN
WHR := ' ';
if :t = '1'   then
          set_block_property('EMP',default_where,WHR);
else
          :PARAMETER.WHR := :B.T||'%';
          WHR := 'EMP.ename LIKE :PARAMETER.WHR';
  set_block_property('EMP',default_where,WHR);
end if;
go_block('emp');
END;
```

3.2.14 Query Built-ins

3.2.14.1 ABORT_QUERY

Syntax: ABORT_QUERY;
Built-in Type: unrestricted procedure
Enter Query Mode: yes

3.2.14.3 ENTER_QUERY

Syntax: ENTER_QUERY;
ENTER_QUERY(keyword_one);
ENTER_QUERY(keyword_two);
ENTER_QUERY(keyword_one, keyword_two);

ENTER_QUERY(keyword_one, keyword_two, locking);

3.2.14.4 EXECUTE_QUERY

Syntax: EXECUTE_QUERY;
EXECUTE_QUERY(keyword_one);
EXECUTE_QUERY(keyword_two);
EXECUTE_QUERY(keyword_one, keyword_two);
EXECUTE_QUERY(keyword_one, keyword_two, locking);

Level of triggers

1 - form **2** - block **3** - item

اى (Built in (Procedure / function يجب معرفة التالى عنها

١ – عملها ٢ – هل هى Procedure أم Function
٣ – لو Function ما هى نوعية الـ Data type اللى هترجع
٤ – عدد الـ argument وكذلك نوع بياناتها
٥ – معرفة الجملة Syntax وكذلك مثال عليها
٦ – هل يوجد عليها قيود Restricted (لا تعمل فى الـ Enter_query) أم العكس

ملحوظة هامة : يمكن عمل زر وكتابة Trigger بداخلة وذلك بان
نضغط على الزر R- click ثم نختار smart trigger ثم نختار النوع المناسب
Enter_query , execute_query , exit_query , commit , clear_form

كتابة الـ Triggers
Producing Triggers

- **إنشاء Trigger باستخدام Smart Trigger** : باستخدام الزر الأيمن للماوس سوف تظهر قائمة فرعية تحتوي علي أمر Smart Triggers

وهذا الأمر يظهر فيه قائمة فرعية أخري تحتوي علي الـ Triggers الممكن استخدامها والمناسبة لهذا العنصر (object) وعندما نضغط علي أي Trigger فإن برنامج الـ Form Builder سيقوم بإنشاء هذا الـ Trigger ويفتح شاشة PL/SQL Editor لكتابة كود الـ Trigger

- **إنشاء Trigger جديد** : عندما لا يوجد نوع الـ Trigger الذي نريد إنشاؤه في الـ Smart Trigger فإن نستخدم إحدى الطرق التالية

1. في شاشة Object Navigator : نختار أيقونة الـ Trigger في الـ Form أو في بلك البيانات Data Block أو في Item المراد عمل Trigger فيه وهذا يتوقف علي مدي (Scope) الـ Trigger ثم نضغط مفتاح Create ستظهر نافذة تحتوي علي كل الـ Triggers لنختار منها الـ Trigger المراد إنشاءه
2. في شاشة PL/SQL : نضغط مفتاح New ليتم استدعاء نافذة تحتوي علي كل الـ Trigger
3. في شاشة Layout Editor : نختار العنصر Object المراد عمل Trigger عليه ثم نضغط الزر الأيمن للماوس فتظهر قائمة فرعية نختار منها أمر PL/SQL Editor لتظهر نافذة تحتوي علي كل الـ Triggers لنختار الـ Trigger المناسب ثم تظهر شاشة PL/SQL Editor لكتابة كود الـ Trigger وإذا كان هذا الشيء (Object) موجود علية Trigger فإنه في هذه الحالة سيظهر محرر PL/SQL وفيه الأكواد المكتوبة في هذا الـ Trigger

- **شاشة محرر PL/SQL** : هي الشاشة التي سيتم كتابة كود الـ Trigger فيها وهي تحتوي علي
  - Type : ليتم ضبطها علي الـ Trigger
  - Object : ليتم تحديد مدي الـ Trigger
  - Item : ليتم تحديد مدي الـ Trigger إذا كان علي مستوي الـ Item حيث يتم تحديد هذا الـ Item
  - Name : ليتم تحديد نوع (اسم) الـ Trigger
  - Source Pane : هذه المساحة الخالية ليتم كتابة الأكواد البرمجية فيها

- **مميزات استخدام نافذة PL/SQL** :
  1. التنسيق الآلي وتغيير ألوان الأكواد البرمجية
     - تغيير ألوان الأكواد البرمجية وترك مسافات بينها حسب الحاجة
     - عمليات السحب والإلقاء للنصوص البرمجية مما يسهل عملة كتابة الأكواد البرمجية
     - عمليات التراجع والتكرار غير محددة
  2. إمكانية تقسيم النافذة حتى أربعة نوافذ فرعية
  3. لوحة الصيغ Syntax Palette : تقوم بعرض الصيغ العامة لجمل SQL مع إمكانية نسخ هذه الصيغ إلي محرر PL/SQL ولاستدعاء هذه اللوحة من قائمة Program نختار منها أمر Syntax Palette
  4. إمكانية البحث في كافة الـ Trigger عن كود محدد مع إمكانية استبدال هذا الكود بكود آخر ولاستدعاء مربع البحث والاستبدال من قائمة Program نختار منها أمر Find and Replace
  5. عند كتابة كود جديد أو تغيير كود مكتوب الـ Trigger لا يتم تنفيذه إلا عند عمل Compile
  6. الـ Trigger التي سيتم عمل Compile لها والتي تحتوي علي أوامر SQL يشترط أن تكون متصلة بقاعدة البيانات أولا
  7. الـ Trigger التي لم يتم عمل Compile لها عند تنفيذ Run للبرنامج فإنه يتم عمل Compile لها

- **كتابة أكواد في الـ Trigger** : الكود في الـ Trigger يتكون من ثلاثة أجزاء هي
  1. التعريفات Declaration : حيث يتم تعريف المتغيرات والثوابت وهو جزء اختياري وهذا الجزء يبدأ بكلمة Declare
  2. الجمل التنفيذية Executable Statement : وهو جزء مطلوب ويحتوي علي الجمل التنفيذية المراد تنفيذها في هذا الـ Trigger وفي حالة كانت هنالك جزء تعريفات Declaration فإن هذا الجزء يبدأ بكلمة Begin وينتهي بكلمة End
  3. معالجة الخطاء Exception Handlers : وهو جزء اختياري وهذا الجزء يبدأ بكلمة Exception وفيه يتم تحديد الإجراء الذي سيتم اتخاذه في حالة حدوث أخطاء

أمثلة :

1. إذا كان الـ Trigger لا يتطلب جمل تعريفات فإن جملتي Begin و End يكون كتابتهما اختياريا كما في الـ Trigger التالي When_Validate_ Item ويكون الكود كما يلي

```
S_item.price  IS NULL THEN :   IF     ;S_item.price  :=: S_item.stdprice :     ; END IF ; Calculate_total
```

2. إذا كان الـ Trigger يتطلب جمل التعريفات فإن جملتي Begin و End يكون كتابتهما إجبارية كما في الـ Trigger التالي When_Button_Pressed ويكون الكود كما يلي

```
DECLARE
Vn_descount      number ;
```

```
BEGEN
Vn_descount := calculate_descount ( :S_item.product_id , S_item.quantity ;
( MESSAGE ( ' Descount : ' || to_char ( vn_descount ;))
```

3. ولمعالجة الأخطاء التي من الممكن أن تظهر نستخدم الجزء Exception كما في الـ Trigger التالي

```
Insert Into Log_Tab ( Log_Val , Log_User ) Values ( : S_Dept.Id , : Global.Username ; )
Exception   When Others Then   MESSAGE ( ' Error ! ' , || Sqlerrm ; )
```

❖ **استخدام المتغيرات في برنامج Form Builder :**

في البرامج الفرعية والـ Trigger إن برنامج الـ Form Builder يقبل نوعين من المتغيرات لتخزين البيانات

1. متغيرات PL/SQL : وهذا النوع من المتغيرات Variables يجب أن يتم تعريفة في جزء الـ Declaration وعند استخدامه داخل البرنامج يكتب مباشرة ولا يسبق اسمه بعلامة colon (:) وإذا تم تعرف هذا المتغير في برنامج فرعي فإن هذا المتغير يمكن استخدامه في كل الـ Trigger التي تستدعي هذا البرنامج الفرعي
2. متغيرات Form Builder : هذا النوع من المتغيرات يعرف من قبل برنامج Form Builder ويتم التعامل مع هذا المتغير داخل الـ PL/SQL علي أنه متغير خارجي لذا يجب أن يسبق هذا المتغير علامة colon (:) وأيضا للتمييز بين هذا النوع من المتغيرات ومتغيرات PL/SQL كما أن متغيرات Form Builder لا يتم الإعلان عنها في جزء الـ Declaration ويمكن أن توجد هذه المتغيرات خارج الـ PL/SQL Block

❖ **أنواع المتغيرات الـ Form Builder :**

1. متغيرات Item (text , list , check box , and son on) : وهي عبارة عن العناصر المكونة لبلوك البيانات Data Block
   - مدي هذا النوع هو الفورم الحالية والقائمة المتاحة فيها
   - يستخدم هذا النوع من المتغيرات للعرض أو التفاعل مع المستخدم

مثال : يجب هنا أن يسبق اسم العنصر Item اسم البلك Block الذي يوجد فيه هذا الـ Item كما يلي

```
: Block3.Product_id :=: Block2.Product_id ;
```

حيث هنا تم تخزين القيمة الموجودة في الـ Product_id الموجود في Block2 في المتغير Product_id الموجود في Block3

2. متغيرات Global Variable :
   - مدي هذا النوع من المتغيرات هو كل البرامج في المجال الحالي
   - يستخدم هذا النوع من المتغيرات في كافة البرامج بمحتوياته من Modules

مثال : يجب هنا أن تسبق كلمة Global اسم المتغير كما يلي

```
: Global.Customer_id :=: Block1.id
```

حيث هنا تم تخزين القيمة الموجودة في العنصر Id الموجود في Block1 في المتغير Customer_id ويمكن استخدام هذا المتغير في كافة البرامج المتاحة

❖ **إدخال قيمة ابتدائية لمتغير من النوع Global** : يمكن استخدام جملة Default_Value لتحديد قيمة ابتدائية لمتغير من النوع Global

مثال :

```
Default_value ( ' Turkey ' , ' Global.country ' )
```

حيث في هذا المثال تم تعريف متغير من النوع Global باسم Country وفي هذا المتغير تم تخزين القيمة Turkey

3. متغيرات System Variables :
   - مدي هذا النوع من المتغيرات هو الفورم الحالية والقائمة المتاحة فيها
   - يستخدم هذا النوع من المتغيرات لتغيير حالة الفورم والتحكم فيها

مثال : هنا يجب أن تسبق كلمة System اسم المتغير كما يلي

```
IF : System.mode := ' Normal ' then   Ok_to_leave_block := true ;   End If ;
```

حيث هنا تم السؤال عن متغير النظام Mode هل يساوي كلمة Normal أم لا

4. متغيرات Parameter :
   - مدي هذا النوع من المتغيرات هو الفورم الحالية
   - يستخدم هذا النوع من المتغيرات لتبادل قيم المتغيرات داخل الفورم

مثال : هنا يجب أن تسبق كلمة Parameter اسم المتغير كما يلي

```
IF : Parameter.Sterting_point = 2  then ; ( Go_block ( ' Block2' )  End If ;
```

حيث هنا تم السؤال عن المتغير Sterting_point هل يساوي 2 أم لا

حيث إن المتغير من النوع Parameter تم إنشاءه في شاشة الـ Object Navigator من خلال وضع المؤشر علي أيقونة Parameter ثم الضغط علي مفتاح Create ثم تغيير اسم المتغير Parameter إلي الاسم الجديد