بسم الله الرحمن الرحيم

بعون الله و توفيقه سأبدأ سلسلة من المحاضرات لتعليم أوراكل ديفلوبر ، بحيث يصبح الإخوة المتابعون لهذه السلسلة بعد إتمامها إن شاء الله ، مؤهلين لخلق نظام مكتمل باستخدام جميع الأدوات المختلفة في أوراكل ديفلوبر،واستيفاءً لهذا الهدف فقد تم اختيار نظام جامعة مبسط للتطبيق عليه..

مع العلم أن هذا الكتاب هو في الأصل محاضرات درستها في الجامعة وقمت بتطبيقها وترتيبها وجمعها في هذا الكتيب وجميع الحقوق محفوظة للأستاذ وديع القباطي .

جمع وترتيب أخوكم في الله...

م/ عبدالله اليحوي

لأي استفسار هذا ايميلي

a_alyahawi@hotmail.com

و لا أريد منكم سوى الدعاء لي في صلواتكم و سأكون سعيدا جدا بذلك

فهرس محاضرات Developer 6i

a_alyahawi@hotmail.com

المحاضرة الأولى

تعتبر الـ(SQL , PL/SQL) هي البُنية الأساسية لبرنامج Developer 6i ،

📖 والتي تمتلك أنواع من التعليمات :

- **لغة معالجة البيانات DML :**
والتي تتعامل مع محتويات الجداول لذلك يمكن التراجع عنها بعد تنفيذ تعليماتها لأنها لا تتعامل مع الهيكل نفسه ،

وتحوي التعليمات (select لإستعراض البيانات - insert لإدخالها - update لتعديلها - delete لحذفها) ،

- **لغة تعريف البيانات DDL :**
والتي تتعامل مع بُنية الـ(Database) لذلك لا يمكن التراجع بعد تنفيذ تعليماتها ،

وتحوي التعليمات (create لإنشاء الكائن - alter لتعديل مواصفات الكائن - drop لحذف الكائن نهائياً) ،

- **لغة التحكم بالبيانات DCL :**
وتحوي التعليمات (grant لمنح الصلاحية - revoke لسحب الصلاحية) .

📖 وعدد من الدوال القياسية Functions :

- **الدوال الحسابية :**

```
Sum( ) - Max( ) - Min( ) - Avg ( ) - Count( ) -

Abs( ) - Mod( ) - Sqrt( ) - Power( ) - Floor( ) - Ceil( )  - Round( ) -
                                                          Nvl( , )
```

- **الدوال المحرفية :**

```
Lower( ) - Upper( ) - Initcap( ) - Concat( , ) - Substar( , , ) -
Length( ) - Instr( , ) - Lpad( , , ) - Rpad( , , ) - Ltrim( ) - Rtrim(
                                                                  )
```

- **دوال التاريخ والوقت :**

```
Sysdate - Months_Between( , ) - Add_Months( ) - Next_Day( , ) -
                                                      Last_Day( )
```

📖 وعدد من عوامل التشغيل :

```
* , / , + , - , Not , And , Or , = , != , > , >= , < , <= , || , .. ,

In , Not In , Is Null , Is Not Null , Between X And Y , Not Between X
                                And Y ,   X Like Y , X Not Like Y
```

==========

على إفتراض أن أربعة محاسبين يعملون في مجالهم بشبكة عمل موحدة فإنه من الأفضل عدم إعطائهم جميعاً إسم User وحيد وكلمة المرور التابعة له وذلك بغرض معرفة الأخطاء ممن إرتكبت ومن هو المسئول المباشر عنها لذلك نحدد لكل شخص إسم User وكلمة مرور خاصة به وأمنحه كذلك الصلاحيات ذاتها ،

لذلك نجد أن تجميع صلاحيات المستخدمين داخل (Role) واحدة نقوم بإنشائها يوفر علينا الكثير من الجهد والوقت اللازمين عند تكرار نفس العمل ...

فعلى افتراض أنه يوجد لدينا 20 جدول وأردنا منح كل جدول

(صلاحية الإستعراض select - صلاحية الإدخال insert - صلاحية التعديل update - صلاحية الحذف delete)يعني أننا سنحتاج إلى منح **4 صلاحيات * 20 جدول = 80 صلاحية** للمستخدم الواحد ، أي **320 صلاحية** للمستخدمين الأربعة ،....

ولكن إذا قمنا بتجميع هذه الـ(**80 الصلاحية**) على الجداول داخل (Role) واحدة فإننا سنحتاج فقط إلى منح **84 صلاحية** بواقع 80 صلاحية إلى داخل (Role) ومن ثم 4 صلاحيات للمستخدمين (منح صلاحية الـ(Role) لكل مستخدم) ، والتفاصيل نفسها تنطبق عند عملية سحب الصلاحيات .

==========

❖ PL / SQL : وشكل البرنامج فيها كالتالي:

Declare

وفي هذا الجزء يتم الإعلان عن المتحولات والمؤشرات Cursor و الثوابت ونلاحظ هنا عند استخدامنا للمؤشرات أن جملة Select هذه لا تحتوي على معامل الإسناد into لأن طبيعة عمل المؤشرات يعتمد على إحضار جميع السجلات دفعة واحدة (بدون تزامن) ومن ثم معالجتها على سجل سجل .

Begin

وفي هذا الجزء يتم كتابة البرنامج بأوامر الـ(SQL, PL/SQL) ونلاحظ هنا أن جملة Select هذه تحتو ي على معامل الإسناد into ويشترط فيها كتابة شروط الربط بين الجداول Where لإرجاع سجل وحيد .

Exception

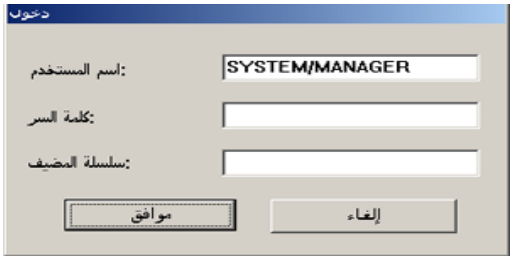
وفي هذا الجزء تتم معالجة الأخطاء المتوقع حدوثها أثناء تنفيذ البرنامج وهي تلك الأخطاء التي لا تكتشف أثناء الترجمة Compile (ليست أخطاء صيغة Syntax) ومن هذه الأخطاء (الاستثناءات) :

- no_data_found : وتستخدم عندما لا يعيد الإستعلام أي بيانات حسب الشرط ،
- too_many_rows : وتستخدم عندما يعيد الإستعلام أكثر من سجل حسب الشرط ،
- zero_devided : وتستخدم عندما تتم القسمة على صفر ،
- cursor_already_open : وتستخدم عندما يكون المؤشر مفتوحاً مرة أخرى دون إغلاقه ،
- others : وتستخدم من أجل ألا يتم إيقاف البرنامج لأي خطأ كان (من المفترض كتابتها دائماً) .

End ;

❖ إنشاء مستخدم جديد :

Start → Programs → Oracle - oracle → Application Development → SQL Plus



ولعمل ذلك نقوم أولاً بالدخول إلى برنامج **SQL Plus** حسب الخطوات السابقة لنلاحظ ظهور شاشة تطالب بإدخال إسم للمستخدم وكلمة المرور الخاصة به ، **(حالياً نكتب إسم المستخدم SYSTEM وكلمة المرور MANAGER)**

- نقوم بكتابة هذا الأمر أمام محث SQL للتأكد من المستخدم الحالي هو مدير النظام SYSTEM

```
SQL> show user ;
```

user is "SYSTEM"

- نقوم بإنشاء مستخدم جديد اسمه UST وكلمة السر TAIZ

```
SQL> create user ust identified by taiz ;
```

User created.

- نمنح الـ(Role [connect , resource]) للمستخدم UST

```
SQL> grant connect , resource to ust ;
```

Grant succeeded.

- نمنح الـ(Role [exp_full_database , imp_full_database]) للمستخدم UST

```
SQL> grant exp_full_database , imp_full_database to ust ;
```

Grant succeeded.

- نقوم الآن بالاتصال بالمستخدم UST

```
SQL> connect ust ;
```

Enter password : ****

Connected.

- نتأكد من أننا داخل المستخدم UST فعلياً

```
SQL> show user ;
```

user is "UST"

- نقوم بإنشاء الجداول التالية للمستخدم UST (بحسب المواصفات والقيود المذكورة لكل حقل)

Table name	Fields Name			
Section	Sec_No number(2) (P. K)	Sec_Name varchar2(50) (Not Null)		
Lvl	Lvl_No number(1) (P. K)	Lvl_Name varchar2(50) (Not Null)		
Stu	Stu_No number(3) (P. K)	Stu_Name varchar2(50) (Not Null)	Stu_Sec number(2) (F.K)	Stu_Lvl number(1) (F.K)
	Stu_Add varchar2(50)	Stu_BrthDt date	Stu_Avg number(5,2)	Stu_CertType number(1)
Subject	Sub_No number(3) (P. K)	Sub_Name varchar2(50) (Not Null)	Sub_Sec number(2) (F.K)	Sub_Lvl number(1) (F.K)
Mark	Mark_Sec number(2) (F.K)	Mark_Lvl number(1) (F.K)	Mark_Stu number(3) (F.K)	Mark_Sub number(3) (F.K)
	Mark_Yj number(5,2) (Check 0 .. 20)	Mark_Ht number(5,2) (Check 0 .. 20)	Mark_Ft number(5,2) (Check 0 .. 60)	

```
SQL> create table Section(Sec_No number(2) primary key ,
  Sec_Name varchar2(50) not null) ;
```

Table created.

```
SQL> create table Lvl(Lvl_No number(1) primary key ,
  Lvl_Name varchar2(50) not null) ;
```

Table created.

```
SQL> create table Stu(Stu_No number(3) primary key ,
  Stu_Name varchar2(50) not null ,
  Stu_Sec number(2) constraint stusecfk references Section(Sec_no) ,
  Stu_Lvl number(1) constraint stulvlfk references Lvl(Lvl_no) ,
  Stu_Add varchar2(50) , Stu_BrthDt date ,
  Stu_Avg number(5,2) , Stu_CertType number(1)) ;
```

Table created.

```
SQL> create table Subject(Sub_No number(2) primary key ,
  Sub_Name varchar2(50) not null ,
  Sub_Sec number(2) constraint subsec references Section(Sec_No) ,
  Sub_Lvl number(1) constraint sublvl references Lvl(Lvl_No) ,
  Sub_Stu number(3) constraint substu references stu(Stu_No) ) ;
```

Table created.

```
SQL> create table Mark(
  Mark_Sec number(2) constraint marksec references Section(Sec_No) ,
  Mark_Lvl number(1) constraint marklvl references Lvl(Lvl_No) ,
  Mark_Stu number(3) constraint markstu references stu(Stu_No) ,
  Mark_Sub number(2) constraint marksub references subject(Sub_No) ,
  Mark_Yj number(5,2) constraint chkYj check(Mark_Yj between 0 and 20)
,
  Mark_Ht number(5,2) constraint chkHt check(Mark_Ht between 0 and 20)
,
  Mark_Ft number(5,2) constraint chkFt check(Mark_Ft between 0 and 60))
;
```

Table created.

==========

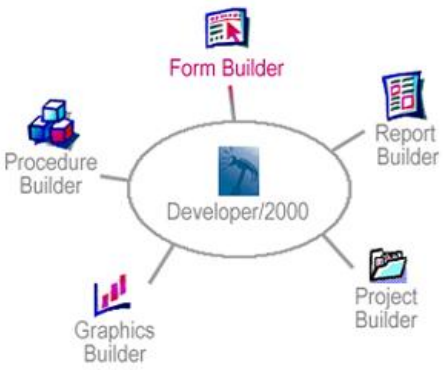
ملاحظات هامة :

a. عند إنشاء أي مستخدم جديد فإن ذلك يكون من داخل المستخدم SYSTEM وذلك لأنه قادر على الإنشاء من خلال صلاحية الـ([Role [DBA) التي يمتلكها .
b. لنتمكن من إتمام عملية الإتصال بالمستخدم UST بنجاح يجب منحه صلاحية إنشاء الجلسة Create Session من خلال الـ([Role [connect) لأن هذه الصلاحية مضمنة فيها .
c. عند إنشاء الجداول فإننا نقوم بالإتصال بالمستخدم UST وذلك من أجل أن تكون الجداول المنشئة تابعة له .

المحاضرة الثانية

❖ Developer 6i :

هو الواجهة التي تمكننا من عمل برامج يتخاطب بها الـ(User) مع قاعدة البيانات وهو يحتوي العديد من البرامج (الأدوات) المساعدة في أوراكل مثل :



Graphics Builder – Report Builder – Form Builder ... إلخ .

==========

❖ Form Builder :

أداة تستخدم لبناء النماذج التي يستخدمها الـ(User) بكل عناصرها .

==========

❖ تشغيل Form Builder :

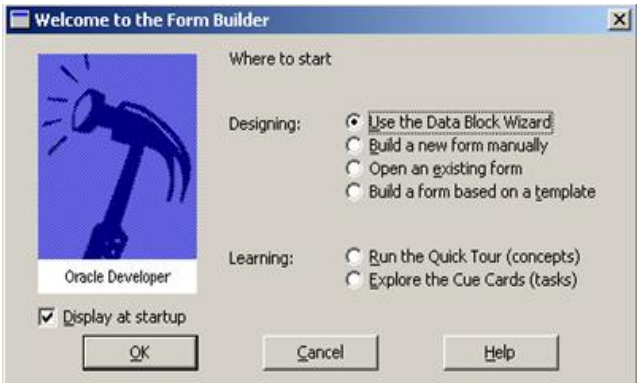
Start → Programs → Oracle Forms 6i-orantf → Form Builder

فتظهر أولاً شاشة ترحيبية يمكن من خلالها تحديد ما إذا كان المطلوب البدء بالتصميم أو بالتعليم ،فالتصميم أربع خيارات وهي :

- إستخدام معالج كتلة البيانات Data Block ،
- بناء نموذج جديد يدوياً ،
- فتح نموذج موجود مسبقاً ،
- بناء نموذج يعتمد على قالب .

وللتعليم خياران وهما :

- عرض الجولة السريعة (مفاهيم) ،
- إستكشاف كروت التلميحات (مهام) .



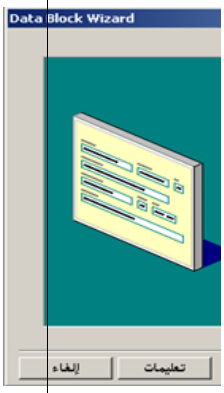
(حالياً سنختار البدء بالتصميم بإستخدام معالج كتلة البيانات Data Block ثم نضغط OK)

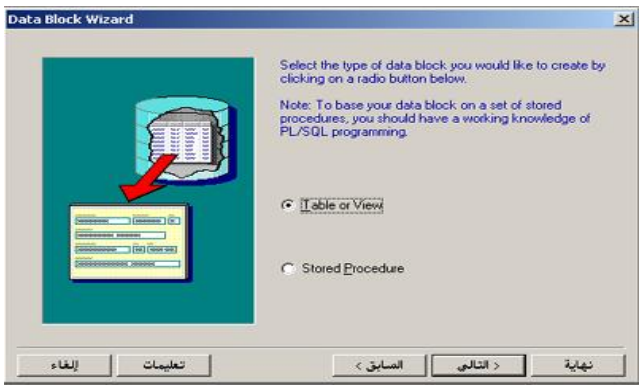
لتظهر بعد ذلك شاشة ترحيبية خاصة بمعالج كتلة البيانات Data Block **(فنضغط التالي)**

ثم تظهر شاشة تحديد نوع كتلة البيانات [illegible]

- إجراء مخزن .

(حالياً سنختار جدول أو مشهد ثم نضغط التالي)





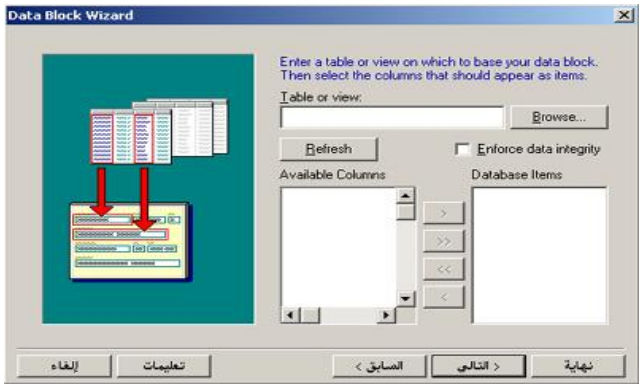
بعد ذلك يطلب إسم الجدول أو المشهد الذي ستؤخذ منه كتلة البيانات Data Block ،

ولعمل ذلك لابد من الإرتباط أولاً بقاعدة البيانات **وذلك بالضغط على الزر Browse...**

وكتابة إسم الـ(User) والـ(Password) بشكل صحيح **ومن ثم الضغط على الزر connect**

<u>فتحديد طريقة العرض :</u>

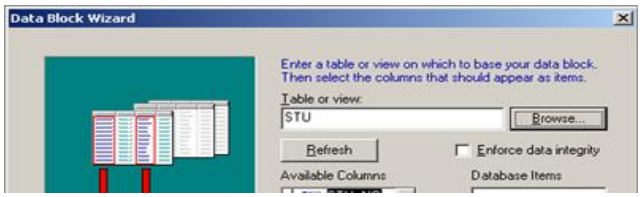
- هل ستكون لما يتعلق بالمستخدم الحالي (الذي تم الإرتباط من خلاله) أم لمستخدمين آخرين أم لكليهما ،
- وتحديد ما نريد تضمينه من الجداول والمشاهد والمرادفات ،
- فإختيار الجدول أو المشهد أو

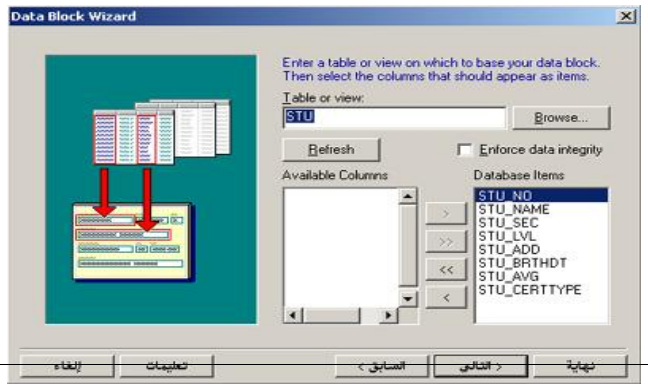


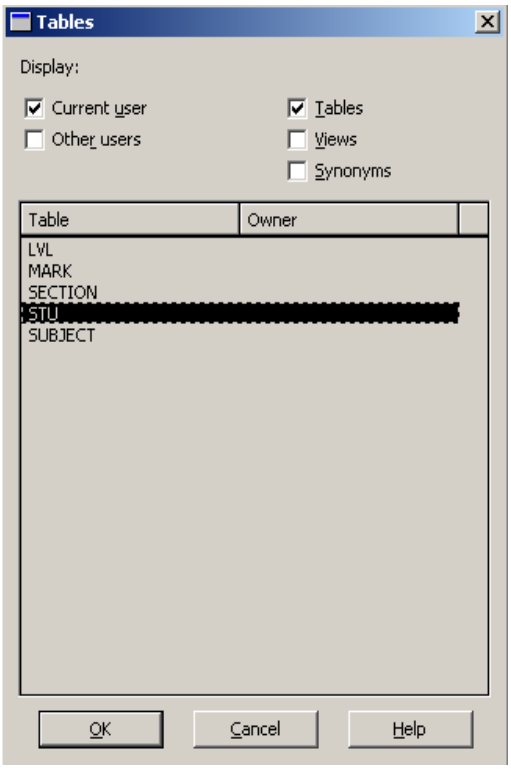
المرادف المطلوب .

(حالياً سنختار إسم المستخدمUST وكلمة السرTAIZ ، ومن جداول هذا المستخدم الحالي نختار الجدول STU)

وبالتالي تظهر الحقول في الـ **(List)** فنختار منها الحقول المطلوب التعامل معها بواسطة الأزرار < ، << ، > ، >> **(حالياً سنختار << ثم نضغط التالي)**







تظهر شاشة التهاني والتبريكات congratulations لتعلن عن إتمام معالج كتلة البيانات Data Block بنجاح ، وقبل أن نضغط Finish نرى وجود الخيارين التاليين :

- إنشاء كتلة البيانات Data Block بعد الإتصال بمعالج التصميم (المخطط Layout) ،
- إنشاء كتلة البيانات فقط .

(حالياً سنختار إنشاء كتلة البيانات بعد الإتصال بمعالج التصميم Layout ثم نضغط نهاية)

لتظهر بعد ذلك شاشة ترحيبية خاصة بمعالج التصميم (المخطط Layout) **(فنضغط التالي)**

فتظهر شاشة تحديد نوع الـ(الورقة أو القماش Canvas) وهناك خمسة أنواع

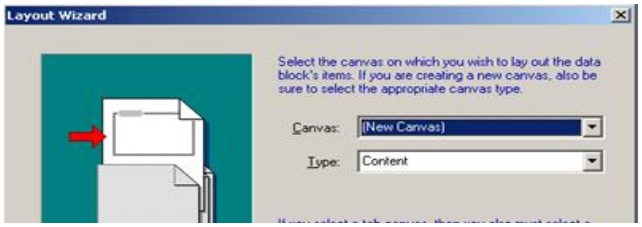
وهي :

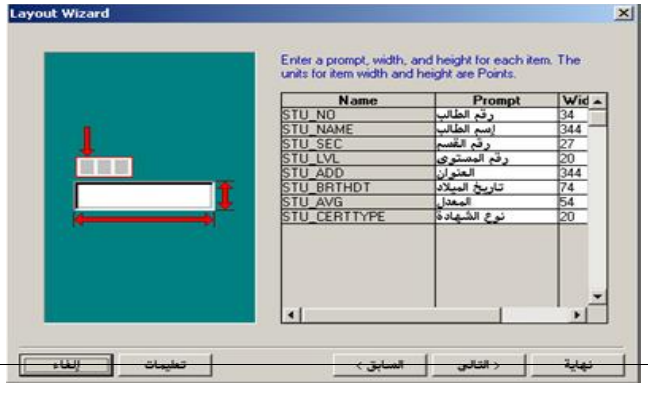
(سندرسها لاحقاً بالتفصيل [المحاضرة الثامنة])

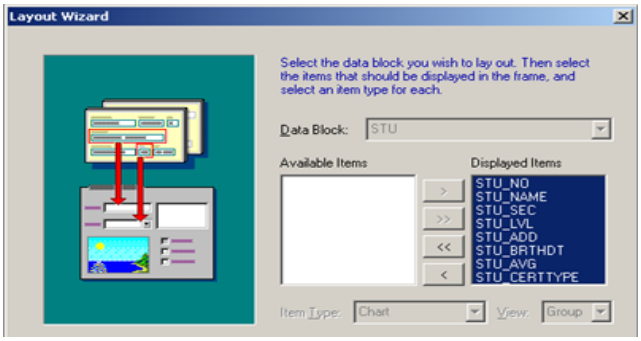
- Content
- Stacked
- Vertical Toolbar
- Horizontal Toolbar
- Tab

(حالياً نجعلها على النوع الإفتراضي Content ثم نضغط التالي)

في الشاشة التالية لها تظهر الحقول في الـ **(List)** فنختار منها الحقول المطلوب إظهارها في النموذج بواسطة الأزرار > ، >> ، < ، << **(يفضل هنا إختيار جميع الحقول وإذا أردت فيما بعد إخفاء أحد هذه الحقول فيتم ذلك عن طريق إعطاء القيمة No لخاصية الرؤية Visible التابعة له ثم نضغط التالي)**





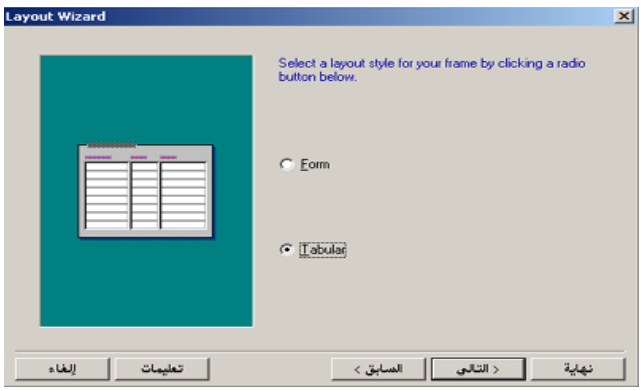


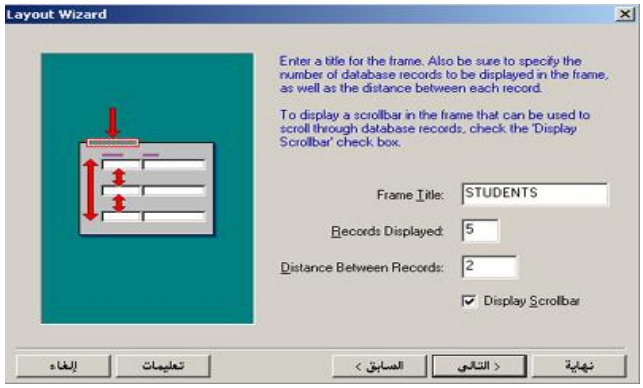
يمكننا في هذه الشاشة تغيير العناوين الظاهرة لأسماء الحقول Prompt وكذلك عرضها W وطولها H **(ثم نضغط التالي)**

تظهر شاشة تحديد طريقة عرض كتلة البيانات هل ستكون :

- Form : عندما نريد أن يحتوي النموذج على عدد من الكائنات مثل Text ، Display item ، List item ، Image item ، Radio Group ، ... إلخ
- Tabular : عندما نريد أن تظهر البيانات على شكل سجلات .

(حالياً سنختار Tabular ثم نضغط التالي)





في هذه الشاشة نحدد :

- إسم للـ(Frame) ،
- وعدد السجلات الظاهرة (الإفتراضي 1) ،
- والمسافة بين السجلات مقدرة بالبيكسل Pixcel (الإفتراضي 0) .

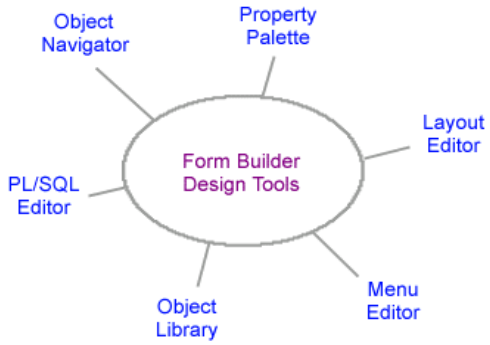
وهل نريد عرض شريط التمرير بجوار البيانات أم لا ؟ (**حالياً سنقوم بكتابة Students كإسم للإطار ، وتحديد عدد سجلات ظاهرة بـ(5) ، وتحديد المسافة بين السجلات بـ(2) بيكسل ، وتنشيط عرض شريط التمرير ، ثم نضغط التالي)**

بعدها تظهر شاشة التهاني والتبريكات congratulations لتعلن عن إتمام معالج التصميم (المخطط) بنجاح ، **(فنضغط نهاية)**

وبذلك نكون قد قمنا بتشغيل Form Builder .

a_alyahawi@hotmail.com

❖ الشاشات الرئيسية للـ(Form Builder) :



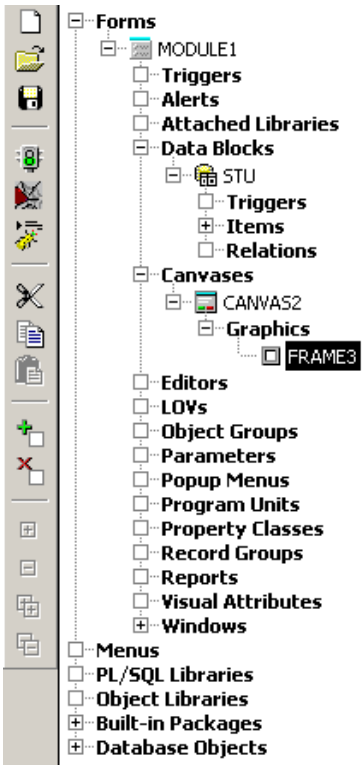
1. Object Navigator :
ويحتوي عناصر النموذج كاملة Form وكذلك الكائنات Objects ، ومن الممكن الحصول على هذه الشاشة من خلال الضغط على الزر **F3** أو من القائمة Tools نختار الأمر Object Navigator .

2. Layout Editor :
ويحتوي الورقة أو القماش Canvas والعناصر المتوضعة عليه ، ومن الممكن الحصول على هذه الشاشة من خلال الضغط على الزر **F2** أو من القائمة Tools نختار الأمر Layout Editor .

3. Property Palette :
وتحتوي خصائص العناصر والكائنات Objects ، ومن الممكن الحصول على هذه الشاشة من خلال الضغط على الزر **F4** أو من القائمة Tools نختار الأمر Property Palette .

==========

❖ نبذة مُبسطة عن محتويات Object Navigator :



Forms :

1. **[حالياً MODULE1] :** إسم الـ (Form) الإفتراضي ويأخذ إسم البرنامج دوماً ،
2. **Triggers :** وهي عبارة عن (SQL , PL/SQL) ينفذ عند حدث معين ،
3. **Alerts :** وهي عبارة عن الرسائل ،
4. **Attached Libraries :** وهي عبارة عن مكتبات PL/SQL ،
5. **Data block :** وهي تحتوي الكتل والتي إما أن تكون DB وهي التي تعتمد على جداول أو مشاهد من الـ(Database) أو قد تكون NonDB وهي كتل التحكم التي لا تعتمد على الـ(Database) ،
6. **Canvases :** وهي الخلفية أو القماشية (الورقة التي ترسم عليها العناصر) ،
7. **Editor :** ويمكنك من خلاله إختيار محرر النصوص الذي تريد إستخدامه ،
8. **LOVs :** وهي عبارة عن قائمة مركبة تقيم حقل معين من الـ(Database) ،
9. **Object Groups :** وهي عبارة عن تجميع للكائنات في مجموعة واحدة لنتجنب بذلك التكرار أثناء العمل خاصة في عمليات النسخ واللصق ،
10. **Parameters :** مهم جداً عند إرسال قيم مع (النموذج Form) إلى نموذج آخر ،
11. **Popup Menus**
12. **[illegible]**
13. **[illegible]**

14. **Record Groups :** نحتاج لإستخدامها مع LOVs ،
15. **Reports :** لإنشاء التقارير ،
16. **Visual Attributes :** يستخدم لإنشاء مجموعة خصائص مرئية ،
17. **Windows :** يستخدم للتحكم بالنوافذ ،

Menus : نحتاجها عند التعامل مع القوائم (حيث تخزن في ملف مستقل) ،

PL/SQL Libraries : وهي عبارة عن مكتبات PL/SQL ،

Object Libraries : وهي عبارة عن مكتبات تتبع الكائنات الموجودة في Oracle ،

Built-in Packages : وهي عبارة عن الحزم المدمجة مع الـ(Oracle) ،

Database Objects : حيث تقوم بعرض جميع مستخدمي قاعدة البيانات

❖ الحفظ والتنفيذ في برامج Developer 6i :

الكائن	عند عملية	يتولد ملف	ذو إمتداد	يشغله برنامج
Forms النماذج	Save الحفظ	Sourceمصدري	*.FMB	Form Builder
	Compileالترجمة	Executeتنفيذي	*.FMX	Forms Runtime
Menu القوائم	Save الحفظ	Sourceمصدري	*.MMB	Form Builder
	Compileالترجمة	Executeتنفيذي	*.MMX	Forms Runtime
Reports التقارير	Save الحفظ	Sourceمصدري	*.RDF	Report Builder
	Compileالترجمة	Executeتنفيذي	*.REP	Reports Runtime

طبعاً يعطى الزبون الملف التنفيذي Execute ويبقى الملف المصدري Source مع المبرمج ،

حيث يستطيع التعديل في البرنامج ولا يستطيع التعديل في التنفيذي .

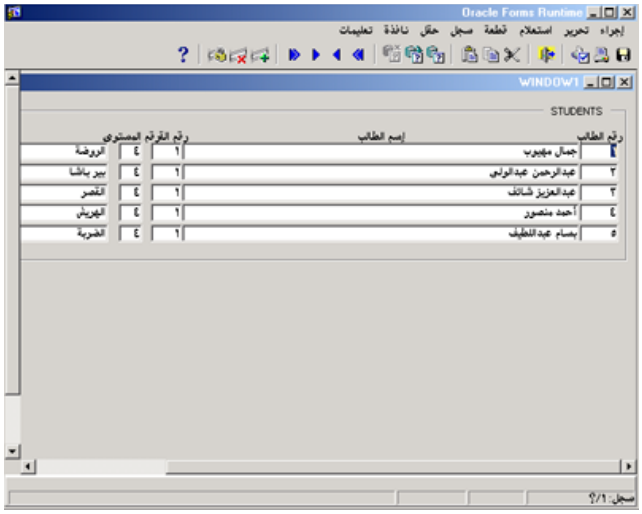
a_alyahawi@hotmail.com

<u>ملاحظات هامة :</u>

a. عند رسم أي أداة (كائن) لابُد أن يوضع على Canvas وإلا فإن المترجم سيعطي رسالة خطأ بأنه غير قادر على التنفيذ .
b. عند تنفيذ البرنامج <u>نلاحظ وجود شاشتين :</u>
   - Forms Runtime ،
   - WINDOW1 .
c. في وضع التنفيذ يمكن إجراء (إدخال الإستعلام) وكذلك (تنفيذ الإستعلام) والتنقل بين السجلات .
d. لترجمة البرنامج نتبع الخطوات التالية :

File → Administration → Compile File

وبالتالي يتم الحصول على النسخة التنفيذية [ملفات الـ(*.FMX)] .



a_alyahawi@hotmail.com

المحاضرة الثالثة

❖ مـــن خـــصـــائـــص MODULE :

General	
Name	MODULE1
Subclass Information	
Comments	
Functional	
Title	MODULE1
Console Window	WINDOW1
Menu Source	File
Menu Module	DEFAULT&SMARTBAR
Initial Menu	
Menu Style	Pull-down
Defer Required Enforcement	No
Menu Security	
Menu Role	
Navigation	
Mouse Navigation Limit	Form
First Navigation Data Block	<Null>
Records	
Current Record Visual Attribute Gr	<Null>
Database	
Validation Unit	Default
Interaction Mode	Blocking
Maximum Query Time	0
Maximum Records Fetched	0
Isolation Mode	Read Committed
Physical	
Coordinate System	
Use 3D Controls	Yes
Form Horizontal Toolbar Canvas	<Null>
Form Vertical Toolbar Canvas	<Null>
International	
Direction	Default
Compatibility	
Runtime Compatibility Mode	5.0

1. Menu Module :
وتعني تحديد القائمة الرئيسية التي تريد إستخدامها عند تنفيذ البرنامج ، (فمن المعلوم أن القوائم في Oracle تأخذ ملفاً مستقلاً كما ذكرنا ذلك سابقاً) ، والقيمة الإفتراضية لهذه الخاصية هي **DEFAULT&SMARTBAR** أي القائمة التي تظهر في شاشة Forms Runtime .

2. Defer Required Enforcement :
وتأخذ القيمة الإفتراضية No وعند جعلها Yes فإنها تقوم بتجاوز القيود المنشئة على الحقول Items .

3. Direction :
وتعني تحديد إتجاه القراءة والكتابة للنموذج Form وتأخذ هذه الخاصية القيم الثلاث الآتية :

a. Default : وتكون حسب اللغة المحددة في Registry .

Start → Run → RegEdit

وبعد الدخول إلى محرر الـ(Registry) نتبع التسلسل الآتي :

HKEY_LOCAL_MACHINE

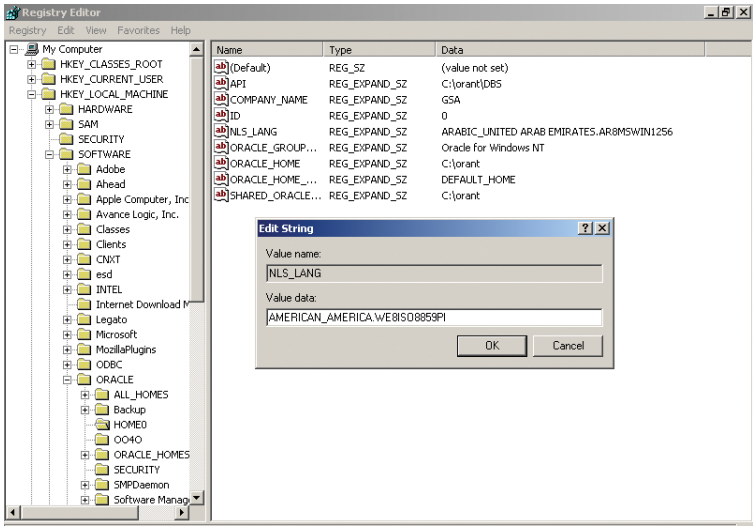
→ SOFTWARE

→ ORACLE

a_alyahawi@hotmail.com

→ HOME0

→ NLS_LANG → [National Language Standard_LANG]



فـقيمة الـ(Value Data) التابعة للـ(NLS_LANG) ستأخذ أحد القيمتين التاليتين :

- ARABIC_UNITED ARAB EMIRATES.AR8MSWIN1256
- ARABIC_SAUDI ARABIA.AR8MSWIN1256

مما يعني أن اللغة الإفتراضية هي العربية وبالتالي تصبح القيمة الإفتراضية هي (Right To Left) ،

مع ملاحظة أنه يتم تحدد نوع اللغة العربية المنتقاة (إماراتي – سعودي ... إلخ) أثناء عملية تثبيت Oracle ،

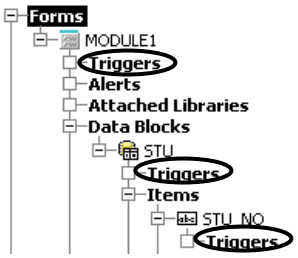
أو القيمة التالية :

- AMERICAN_AMERICA.WE8ISO8859PI

مما يعني أن اللغة الإفتراضية هي الإنجليزية وبالتالي تصبح القيمة الإفتراضية هي (Left To Right) .

b. **Left To Right :** من اليسار إلى اليمين .
c. **Right To Left :** من اليمين إلى اليسار .

❖ الأحـــداث Triggers :



وهي عبارة عن شفرة مصدرية Code تكتب بـ(SQL ; PL/SQL) وعلى ثلا

1. **مستوى النموذج Module) منها:**
   - WHEN-NEW-FORM-INSTANCE : ينفذ عند بداية تحميل الـ(Form
2. **مستوى كتلة البيانات Data Block) منها :**
   - WHEN-NEW-BLOCK-INSTANCE : ينفذ عند بداية الدخول على ال
   - WHEN-NEW-RECORD-INSTANCE : ينفذ عند بداية الدخول على
   - PRE-BLOCK : ينفذ قبل الدخول على الـ(Block) ،
   - POST-BLOCK : ينفذ بعد الدخول على الـ(Block) .
3. **مستوى العنصر Item) منها :**
   - WHEN-NEW-ITEM-INSTANCE : ينفذ عند بداية الدخول على الـ(Item) ،
   - WHEN-BUTTON-PRESSED : ينفذ عند النقر على العنصر PUSH BUTTON ،
   - WHEN_LIST_CHANGED : ينفذ عند التعديل على العنصر List Item ،
   - WHEN_IMAGE_PRESSED :ينفذ عند النقر على العنصر Image .

❖ الـــرســـائـــل :

بالطبع ليس المهم أن نكتب الرسالة ولكن المهم معرفة :

- ما هو غرض الرسالة (لتحديد نوعها هل هو message أو Alert) ،
- وأين سنكتبها (في أي حدث Trigger) .

❖ **أنواع الرسائل :**

1. **message)** وهي الرسائل التي تختص بالظهور على شريط المعلومات (شريط الحالة) ، وتأخذ الصيغة العامة التالية

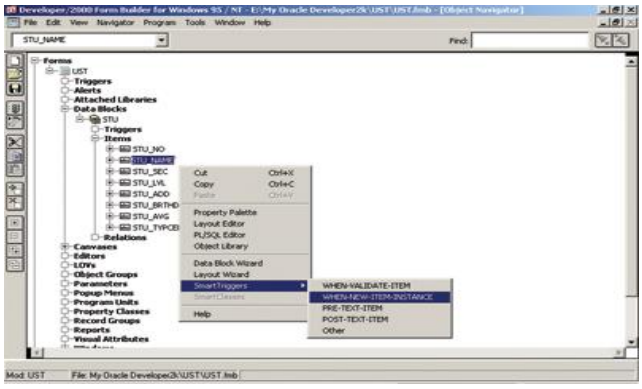
message('نص الرسالة; (')

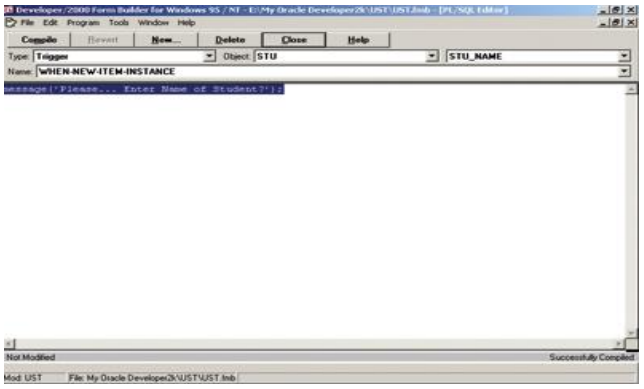
مثال على ذلك :

نفترض أنك أردت إظهار رسالة للمستخدم في شريط الحالة عند وصول المؤشر لحقل إسم الطالب تطالبه فيها بالإدخال ؟

ولعمل ذلك نحتاج أولاً لمعرفة الـ**(Trigger)** الذي نريد تنفيذ أمر إظهار الرسالة عنده فبعد تحديد **العنصر STU_NAME** نقوم بالنقر على زر الفأرة الأيمن لنختار الأمر Smart Trigger ومنه الحدث **WHEN-NEW-ITEM-INSTANCE** لنكتب فيه أمر الرسالة `message(' Please… Enter Name of Student? ') ;` ثم نضغط على الزر **Compile** لنتأكد من أنه تم ترجمة الأمر بشكل صحيح لهذا الحدث ،

ونلاحظ عند تكرار عمل الرسالة الخاصة بالـ(message) مرتين أنه قد تم تحويل الرسالة الثانية تلقائياً إلى ما يسمى بالـ(Alert) .





2. **Alert)** وهي الرسائل التي تظهر بشكل صندوق حوار
والتي قد تكون رسالة من النوع (إيقاف Stop أو تحذير Caution أو ملاحظة Note)

وقد تحتوي على (زر أو زرين أو ثلاثة على الأكثر) إلى غير ذلك من خصائص صناديق الحوار ... وكمثال على ذلك :

نفترض أننا نريد إظهار رسالة تأكيدية للمستخدم عند عملية حذف السجل كما يلي :

فإذا نقر الزر (إلغاء الأمر) يخرج بدون الحذف ، وإذا نقر الزر (موافق) يقوم بحذف السجل ويحفظ ؟

وفي Oracle يمكن عمل مثل هذه الرسائل ولكن على مرحلتين :

a. مرحلة إنشاء الرسالة :

حيث نقوم بتحديد الـ(Alert) الموجود في شاشة الـ(Object Navigator) ثم Create لينشئ لنا إسماً إفتراضياً للرسالة المنشئة فنحددها ثم نضغط F4 للتحكم بقيم الخصائص الهامة التابعة لها وكالتالي :

	Property	Value
1	Name	Del
2	Title	حذف سجل
3	Message	هل تريد حذف السجل بالتأكيد ؟
4	Alert Style	Stop
5	Button 1 Label	موافق
6	Button 2 Label	إلغاء الأمر
7	Default Alert Button	Button2

b. مرحلة طلب الرسالة والمعالجة:

ولعمل ذلك نحتاج لمعرفة الـ**(Trigger)** الذي نريد تنفيذ أمر إظهار الرسالة عنده فبعد تحديد الـ**Data (Block STU** لأنه وكما هو معلوم أن عملية حذف السجل تكون على مستوى كتلة البيانات Data Block ، نقوم بالنقر على زر الفأرة الأيمن لنختار الأمر Smart Trigger ومنه الحدث **KEY-DELREC** لنكتب فيه أوامر معالجة الرسالة وكالتالي :

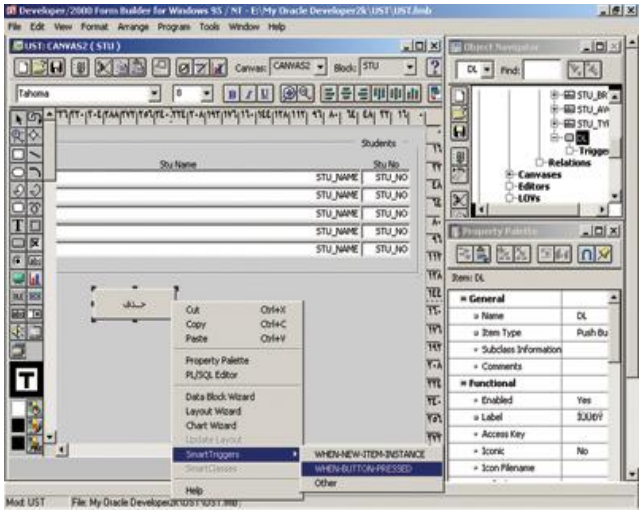
```
Declare
     N   Number ;
Begin
  N := Show_Alert('Del') ;
   If  N = Alert_Button1 Then
      Delete_Record ;
      Commit ;
   Else
      Null ;
   End If ;
End ;
```

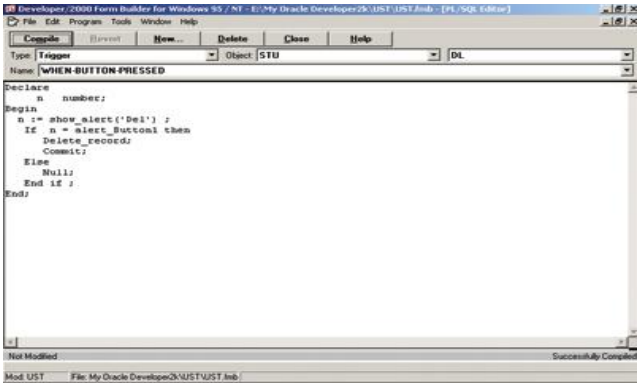
ثم نضغط على الزر **Compile** لنتأكد من أنه تم ترجمة الأمر بشكل صحيح لهذا الحدث ، وليتم التنفيذ من خلال القائمة التي تظهر في شاشة Form Runtime ، ولكن إذا أردنا تنفيذ أمر الحذف هذا من خلال **زر يقوم بعملية الحذف** فنتبع الخطوات التالية :

1. ننشئ زر ليقوم بعملية الحذف وذلك برسمه على الـ(Canvas) ونعطيه الخصائص :

	Property	Value
1	Name	DL
2	Label	حذف
3	Number of item displayed	1

2. نقوم بنسخ ما كتبناه في الحدث **KEY-DELREC** إلى الحدث الخاص بالـ(Push-Button) عن طريق إختيار الأمر **Smart Trigger** ومنه الحدث **WHEN-BUTTON-PRESSED** من خلال :
   a. تحديد العنصر DL من شاشة Object Navigator والنقر على زر الفأرة الأيمن ،
   b. تحديد الزر حذف في شاشة Layout Editor والنقر على زر الفأرة الأيمن ،
   c. عند تحديد العنصر DL أو تحديد الزر حذف من القائمة Program .





❖ كُـتـل الـبـيـانـات Data Blocks :

❖ **مكونات الـ(Data Block) الأساسية :**

a. **Triggers :** وهي الأحداث المتعلقة بالـ(Data Block) [المستوى الثاني من الـ(Triggers)] ،
b. **Items :** وتمثل العناصر التابعة لهذا الـ(Data Block) [مثلاً : حقول الجدول وعناصر التحكم التابعة له] ،
c. **Relations :** لربط هذا الـ(Data Block) مع أي (Data Block) آخر [مثلاً : ربط جدول مع جدول آخر] .

❖ **خصائص الـ(Data Block) :**

من الضروري تحديد [illegible]

- [illegible]

- وليكون العمل أكثر أمنية .

	الخاصية	شرح مبسط
1	**Name**	والذي سيتم التعامل من خلاله مع الـ(Data Block) برمجياً
2	**Navigation Style**	تحديد كيفية إنتقال المؤشر ولها ثلاثة قيم : **A) Same Record :** وتعني التنقل بين حقول نفس السجل من أول حقل إلى آخر حقل بحيث يظل المؤشر على نفس السجل وليتم التنقل بين حقول هذا السجل بضغط Enter ، **B) Change Record :** وتعني تغير السجل الحالي بحيث يتم انتقال المؤشر من آخر حقل للسجل الحالي إلى أول حقل للسجل التالي بضغط Enter ، **C) Change Data Block :** وتعني تغير الـ(Block) الحالي بحيث يتم انتقال المؤشر من أخر حقل للسجل الحالي في الـ(Block) الحالي إلى أول حقل في الـ(Block) التالي .
3	**Previous Navigation Data Block**	تحديد الـ(Block) السابق حسب الترتيب الإفتراضي حسب شاشة F3
4	**Next Navigation Data Block**	تحديد الـ(Block) التالي
5	**Number of Record display**	تحديد عدد السجلات المعروضة على الـ(Canvas)
6	**Records Orientation**	تحديد إتجاه السجلات عمودي أم أفقي (الإفتراضي عمودي)
7	**Database Data Block**	تحديد نوع الـ(Data Block) وتمتلك القيم(Yes , No) **1. Database (Yes) :** أي المرتبط مع قاعدة البيانات (الجداول والمشاهد والمرادفات ...إلخ) ، **2. Non Database (No) :** أي المعتمد على عناصر التحكم فقط و لا يعتمد على قاعدة البيانات (DUMMY) .
8	[illegible]	[illegible] التنفيذ وتمتلك القيم (Yes , No)

9	**Insert Allowed**	سماحية الإدخال للبيانات أثناء التنفيذ وتمتلك القيم (Yes , No)
10	**Update Allowed**	سماحية التعديل على البيانات أثناء التنفيذ وتمتلك القيم(Yes , No)
11	**Delete Allowed**	سماحية الحذف من البيانات أثناء التنفيذ وتمتلك القيم (Yes , No) ✍ **ملاحظة هامة :** صلاحية الـ(SQL) أقوى من صلاحية Developer وكمثال على ذلك : SQL> revoke delete on STU.Lvl from UST ; تعليمة الـ(SQL) هذه ستسحب صلاحية الحذف على الجدول Lvl من المستخدم ust في أي برنامج ، أما في Developer فإنه إذا تم إعطاء القيمة No للخاصية Delete Allowed فستمنع الحذف من الجدول Lvl ضمن البرنامج الحالي فقط وعند إنشاء برنامج آخر فإنه يمكن الحذف من الجدول Lvl إذا ما تم إعطاء القيمة Yes للخاصية Delete Allowed ما لم تكتب تعليمة الـ(SQL) أعلاه وإلا فلا جدوى من قيم هذه الخاصية من سماحية للحذف بـ(Yes) أو منعها بـ(No) .
12	**WHERE Clause**	شرط جلب السجلات من الجداول
13	**ORDER BY Clause**	تحديد الحقل المراد الترتيب من خلاله وكيفية الترتيب ،أمثلة ذلك : • Lvl_Name : يتم الترتيب حسب الحقل إسم المستوى تصاعدياً (الإفتراضي) • Lvl_Name desc : يتم الترتيب حسب الحقل إسم المستوى تنازلياً • Lvl_Name Asc : يتم الترتيب حسب الحقل إسم المستوى تصاعدياً

a_alyahawi@hotmail.com

❖ الـخـصـائـص الـمـرئـيـة Visual Attribute :

مثال : مطلوب إظهار السجل الحالي Current Record في جدول Stu بشكل مختلف عن بقية السجلات وكالتالي :

خلفية بلون Cyan ، خط نوعه Arial بحجم 12 وبلون Red ؟

وليتم ذلك لا بد من المرور بمرحلتين :

a. مرحلة إنشاء الخصائص المرئية :

حيث نقوم بتحديد الـ(Visual Attribute) الموجود في شاشة الـ(Object Navigator) ثم Create لينشئ لنا إسماً إفتراضياً للخصائص المرئية المنشئة فنحددها ثم نضغط F4 للتحكم بقيم الخصائص المطلوبة وكالتالي :

	Property	Value
1	Name	VA1
2	Visual Attribute Type	Common
3	Foreground Color	Red
4	Background Color	Cyan
5	Font Name	Arial
6	Font Size	12

b. مرحلة طلب الخصائص المرئية :

ولعمل ذلك نحدد **الـSTU (Data Block)** ثم نضغط F4 لنتحكم بالخصائص المرئية وكالتالي :

Property	Value
Current Record Visual Attribute Group	VA1

==========

❖ الـعـنـاصـر Items :

وتنقسم إلى نوعين :

a. عناصر إدخالية Input Items : مثل : Text_item ، List_Item ، Check_box ، Radio_Group .

b. عناصر لا إدخالية Non Input Items : مثل : Display_Item ، Push_Button ، Images .

تشترك هذه العناصر ببعض الخصائص وتختلف في الخصائص الأخرى

❖ خـــصـــائـــص Text_Item :

	الخاصية	شرح مبسط
1	**Name**	والذي سيتم التعامل من خلاله مع الـ(Text_item) برمجياً
2	**Item Type**	تحديد نوع العنصر المعروض
3	**Enabled**	منشط أم مبهت (Yes , No)
4	**Justification**	موقع ظهور الكتابة (Start ,End, Center ,Right, Left)
5	**Data Type**	نوع البيانات
6	**Maximum Length**	طول البيانات
7	**Initial Value**	القيمة الإبتدائية للعنصر
8	**Required**	قيمة العنصر مطلوب إدخالها (إجباري)
9	**Format Mask**	تنسيق العنصر الرقمي حسابياً (مثلاً :**999.99**)
10	**Database Item**	هل العنصر مرتبط بقاعدة البيانات أم غير مرتبط(Yes , No)
11	**Query Allowed**	السماحية بالإستعلام (Yes , No)
12	**Insert Allowed**	السماحية بالإضافة (Yes , No)
13	**Update Allowed**	السماحية بالتعديل (Yes , No)
14	**Visible**	هل هو مرئي أم مخفي (Yes , No)
15	**Canvas**	تحديد القماشية المتوضع عليها العنصر
16	**X Position**	تحديد موقع العنصر الأفقي
17	**Y Position**	تحديد موقع العنصر العمودي
18	**Width**	تحديد العرض
19	**Height**	تحديد الإرتفاع
20	**Font & Color**	التحكم بالخط والألوان
21	**Prompt**	التحكم بالـ(Label) الخاص بالـ(Text_item)
21	**Prompt Font & Color**	التحكم بألوان وخطوط الـ(Label) الخاص بالـ(Text_item)
23	**Hint**	[illegible] عند تفعيل العنصر
[illegible]	[illegible]	[illegible]

❖ خـــصـــائـــص List_Item :

	الخاصية	شرح مبسط
1	**Element in List**	لتحديد العناصر المحتواة مع قيمها
2	**Mapping of Other Values**	عند وجود قيمة مغايرة لقيم العناصر المحتواة

بقية الخصائص كما مر سابقاً

==========

❖ خـــصـــائـــص Check_Box :

	الخاصية	شرح مبسط
1	**Value When Checked**	قيمة العنصر عندما يكون محدد (منشط)
2	**Value When Unchecked**	قيمة العنصر عندما لا يكون محدداً
3	**Check box Mapping of Other Value**	عند وجود قيمة مغايرة لقيمتين السابقتين

وتكون القيمة المغايرة عادة موضوعة في**Initial Value**

بقية الخصائص كما مر سابقاً

==========

❖ خـــصـــائـــص Radio_Group :

وهي تحوي العديد من Radio_Button **أي أنها بمثابة الحاوية**Container **للخيارات المفردة**

	الخاصية	شرح مبسط
1	**Mapping of Other Values**	عند وجود قيمة مغايرة لقيم العناصر المحتواة
2	**Initial Value**	القيمة الإبتدائية للعنصر

بقية الخصائص كما مر سابقاً

❖ خـــصـــائـــص Radio_Button :

	الخاصية	شرح مبسط
	Label	

2	**Radio Button Value**	القيمة الخاصة بالعنصر

بقية الخصائص كما مر سابقاً

==========

❖ خـــصـــائـــص Display_Item :

كل الخصائص قد مرت سابقاً ولكن نستفيد من هذا العنصر أنه لا يمكن التعديل فيه ولا ينتقل إليه التركيز Focus

==========

❖ خـــصـــائـــص Push_Button :

	الخاصية	شرح مبسط
1	**Label**	العنوان الظاهر أمام المستخدم
2	**Iconic**	تحديد تنشيط وجود الأيقونات على الزر(Yes , No)
3	**Icon Filename**	تحديد الملف الذي ستكون أيقونته على الزر

بقية الخصائص كما مر سابقاً

==========

❖ خـــصـــائـــص Image :

	الخاصية	شرح مبسط
1	**Image Format**	نوع الصورة
2	**Image Depth**	عمق الصورة
3	**Sizing Style**	تحديد نمط التحجيم (قطع – ملائمة)

بقية الخصائص كما مر سابقاً

المحاضرة الرابعة

❖ التحكم بالخصائص برمجياً :

عن طريق التعليمة Set نستطيع التحكم بخصائص الـ(Window) أو الـ(Block) أو الـ(Item) أثناء تنفيذ البرنامج بينما

في لوحة الخصائص Property Palette نستطيع التحكم بالخصائص في مرحلة تصميم البرنامج .

والصيغة العامة كما يلي :

للتحكم بخصائص النافذة برمجياً Set_**Window**_Property(**إسم النافذة**, إسم الخاصية ,قيمة الخاصية) ;

للتحكم بخصائص الكتلة برمجياً Set_**Block**_Property(**إسم الكتلة**, إسم الخاصية ,قيمة الخاصية;)

للتحكم بخصائص العنصر برمجياً Set_**Item**_Property(**إسم العنصر**, إسم الخاصية ,قيمة الخاصية ;)

بالطبع هذه الخصائص تعدل في الحدث المناسب الذى يحدده الطلب والحاجة فمثلاً :

- عند تحميل البرنامج ؟ : حدث WHEN_NEW_FORM_INSTANCE ،
- في زر ... ؟ : حدث WHEN_BUTTON_PRESSED ... إلخ .

1. التحكم بخصائص النافذة Window برمجياً ؟

وكأمثلة على ذلك :

❖ **تكبير نافذة الـ(Forms Runtime) ؟**

```
Set_Window_Property(Forms_Mdi_Window , Window_State , Maximize);
```

لاحظ أن إسم نافذة الـ(Forms Runtime) إسم ثابت دائماً فهو يكتب Forms_Mdi_Window ومن غير إشارات الإقتباس .

❖ **تكبير نافذة البرنامج إلى حجم نافذة الـ(Forms Runtime) ؟**

```
Set_Window_Property('Window1' , Window_State , Maximize);
```

لاحظ أن إسم نافذة البرنامج إسم متغير فهو يكتب حسب إسم النافذة البرمجي ومع وجود إشارات الإقتباس .

❖ **تغيير عرض نافذة البرنامج بإرتفاع 600 وعرض 800 ؟**

```
Set_Window_Property('Window1' , Height , 600 , Width , 800);
```

لاحظ [illegible]

❖ **إعطاء عنوان ' برنامج الطلاب ' لنافذة الـ(Forms Runtime) ؟**

```
Set_Window_Property(Forms_Mdi_Window , Title , 'برنامج الطلاب');
```

لاحظ أن الخصائص عندما يكون لديها قيم محرفية فلا بد من وجود إشارات الإقتباس .

❖ **إعطاء عنوان ' بيانات المستوى ' لنافذة البرنامج ؟**

```
Set_Window_Property('Window1' ,Title , 'بيانات المستوى');
```

❖ **تصغير نافذة البرنامج وكذلك تصغير نافذة الـ(Forms Runtime)؟**

```
Set_Window_Property('Window1' , Window_State, Minimize);

Set_Window_Property(Forms_Mdi_Window , Window_State, Minimize);
```

2. التحكم بخصائص الكتلة Block برمجياً ؟

وكأمثلة على ذلك :

❖ **السماح بالإستعلام من جدول المستويات Lvl ؟**

```
Set_Block_Property('Lvl' , Query_Allowed , Property_True);
```

لاحظ إعطاء القيمة property_True عندما تكون قيمة الخاصية Yes .

❖ **منع الإستعلام عن جدول المستويات Lvl ؟**

```
Set_Block_Property('Lvl' , Query_Allowed , Property_False);
```

لاحظ إعطاء القيمة property_False عندما تكون قيمة الخاصية No .

❖ **ترتيب جدول المستويات Lvl بحسب الرقم ؟**

```
Set_Block_Property('Lvl' , Order_By , 'Lvl_No');
```

❖ **ترتيب جدول المستويات Lvl بحسب الإسم ؟**

```
Set_Block_Property('Lvl' , Order_By, 'Lvl_Name');
```

وكذلك الحال بالنسبة لبقية خصائص الكتلة Block مثل :

- `Insert_Allowed` : الخاصة بالإضافة ،
- `Update_Allowed` : الخاصة بالتعديل ،
- `Delete_Allowed` : الخاصة بالحذف ،
- `Defalut_Where` : الخاصة بالشرط Where

```
Set_Block_Property('Stu' , Defalut_Where , 'Stu_Sec >= 1');
```

- `Current_Record_Attribute` : الخاصة بالخصائص المرئية للكتلة Visual Attribute

```
Set_Block_Property('Stu' , Current_Record_Attribute , 'VA1');
```

... إلخ .

3. التحكم بخصائص الـ[illegible] Item برمجياً

وكأمثلة على ذلك :

❖ **إظهار حقل إسم المستوى في جدول المستويات Lvl ؟**

```
Set_Item_Property('Lvl_Name' , Visible , Property_True);
```

❖ **إخفاء حقل إسم المستوى في جدول المستويات Lvl ؟**

```
Set_Item_Property('Lvl_Name' , Visible , Property_False);
```

❖ **تنشيط (تمكين) حقل إسم المستوى في جدول المستويات Lvl ؟**

```
Set_Item_Property('Lvl_Name' , Enabled , Property_True);
```

❖ **تبهيت (عدم تمكين) حقل إسم المستوى في جدول المستويات Lvl ؟**

```
Set_Item_Property('Lvl_Name' , Enabled , Property_False);
```

وكذلك الحال بالنسبة لبقية خصائص الكتلة Block مثل :

- `Position` : الخاصة بتحديد موقع العنصر على نافذة البرنامج بـ(X , Y)

```
Set_Item_Property('MyButton' , Position , 100 , 100);
```

- `Visual_Attribute` : الخاصة بالخصائص المرئية للعنصر

```
Set_Item_Property('Stu_No' , Visual_Attribute , 'VA1');
```

... إلخ .

==========

❖ المتحولات العامة والمتحولات المحلية :

1. المتحولات العامة Global Variables :

تبدأ بـ(**Global.**:)

ولتعريف متحولات عامة بحيث نستطيع التعامل معها بأي Trigger أو أي Procedure أو أي Function

نعرف المتحول العام في الحدث **WHEN_NEW_FORM_INSTANCE** وحتى يصبح معروفاً لكامل الـ(Forms) .

والصيغة العامة للتعريف كما يلي :

عندما تكون القيمة الإبتدائية رقماً `:global.Variable := Value ;`

عندما تكون القيمة الإبتدائية نصاً محرفياً `:global.Variable := 'Value' ;`

بالطبع يفضل دائماً أن نعطي قيماً إبتدائية للمتحولات تلافياً لحدوث أية مشاكل

وكأمثلة على ذلك :

```
:Global.MyNo := 1 ;
:Global.MyName := 'Amerah';
```

2. [illegible]

تكون [illegible] بداخله فقط

يعني ضمن الـ(Begin) والـ(End) المعرف ضمنها .

والصيغة العامة للتعريف كما يلي :

عندما تكون القيمة الإبتدائية رقماً `Variable DataType := Value ;`

عندما تكون القيمة الإبتدائية نصاً محرفياً `Variable DataType := 'Value' ;`

بالطبع يفضل دائماً أن نعطي قيماً إبتدائية للمتحولات تلافياً لحدوث أية مشاكل

وكأمثلة على ذلك :

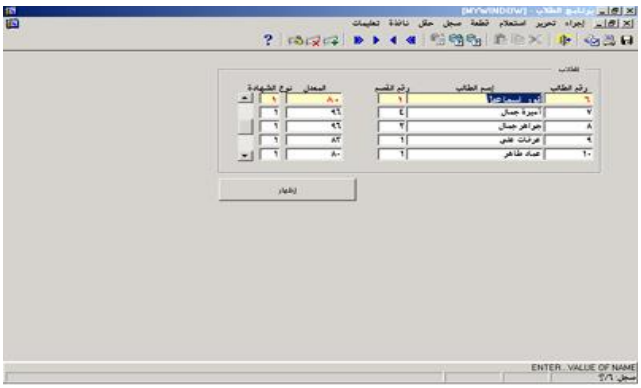
```
MyNo Number := 2 ;
MyName Varchar2(20) := 'Gawaher';
```

==========

❖ التحكم بالخصائص برمجياً عن طريق المتحولات العامة

:

مثال :



نريد إنشاء زر عنوانه ' إخفاء ' وعند النقر عليه يخفي إسم المستوى ويتغير عنوانه إلى ' إظهار '

وعند النقر عليه مرة أخرى يظهر إسم المستوى ويتغير عنوانه إلى ' إخفاء '

وهكذا تستمر العملية بين ' إظهار ' و ' إخفاء' ؟

تلميح :

(قم بربط قيم الخصائص البرمجية بمتحولات عامة)

ولعمل ذلك لا بد من :

- تعريف متحول عام (على مستوى النموذج Form) وليكن X حيث نقوم بإسناد القيمة ' إخفاء' إليه كقيمة إبتدائية ونتحكم بكتابة عنوان الزر من خلال قيمة المتحول العام ،
- نفحص قيمة المتحول العام في حدث (الضغط على الزر) من خلال جملة IF الشرطية فإذا كانت قيمة المتحول ' إخفاء' فإننا سنخفي هذا العنصر بخاصية Visible ونسند القيمة المغايرة ' إظهار' إلى المتحول العام والعكس صحيح ثم نتحكم بكتابة عنوان الزر من خلال قيمة المتحول العام الجديدة .

```
' :Global.X := 'إخفاء;

Set_Item_Property('MyButton' , Label , :Global.X) ;
```

2. WHEN_BUTTON_PRESSED

```
' :Global.X = 'إخفاء If  then'

   Set_Item_Property('Lvl_Name' , Visible , Property_False);

' :Global.X := 'إظهار;

Else

   Set_Item_Property('Lvl_Name' , Visible , Property_True);

' :Global.X := 'إخفاء;

End if ;

Set_Item_Property('MyButton' , Label , :Global.X);
```

==========

ملاحظات هامة :

- := تستخدم للتخصيص ،
- = تستخدم للمقارنة ،
- : تستخدم قبل العنصر لتعني أنها قيمته .

==========

❖ **الترقيم التلقائي :**

مطلوب : عمل ترقيم متسلسل لحقل رقم الطالب Stu_No في جدول STU ؟ مع إختيار الـ Trigger المناسب ؟

تلميح :

(يتم إفتراض أن قيمة الحقل للسجل الأول مهيئة بقيمة مدخلة من المستخدم) ؟

حل المطلوب :

من المعلوم أنه يمكن عمل ترقيم تلقائي على حقل من نوع Number ولتكن الصيغة المصدرية كالتالي :

(Trigger : KEY_CREREC)

حدث إنشاء السجل

```
Declare
  Counter  number ;
تعرف متحول العداد من نوع رقمي
Begin
  Select Max(Stu_No) into Counter From Stu ;
إختيار القيمة العظمى من حقل الطالب وإسنادها إلى متحول العداد
  :Stu_No := Counter + 1 ;
ترقيم حقل الطالب بقيمة المتحول مضافاً إليه واحد
End ;
```

المحاضرة الخامسة والسادسة

❖ العـــلاقـــات Relations :

هي تعبير يطلق على علاقة بين جدولين أحدهما هو الجدول الرئيسي **Master** والآخر هو الجدول التفصيلي **Details** ، **ولإتمام العلاقة بالشكل المطلوب فلا بد من المرور بالخطوات التالية :**

1. تحديد الجدول الرئيسي Master والجدول التفصيلي Details ،
2. تحديد حقول الربط بين الجدولين (وتسمى الحقول المخفية) ،
3. تحديد شروط الربط بين الجدولين **Join Condition** .

مثال ذلك :

ليكن لدينا الجدولين التاليين

Master

رقم الطالب	إسم الطالب	رقم القسم	رقم المستوى

Details

رقم المادة	إسم المادة	أعمال السنة	النصفي	النهائي

- نلاحظ من خلالهما أن جميع حقول الجدول الرئيسي متوفرة في جدول **STU** ، بينما نجد أن جميع حقول الجدول التفصيلي متوفرة في جدول **MARK** .

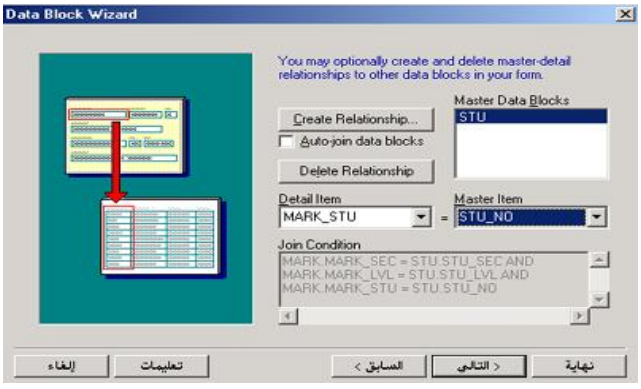
- يتم الربط من خلال الحقول (**رقم الطالب – رقم القسم – رقم المستوى**) ،
- شروط ربط الجدولين يحدد من خلال عدد الجداول لذلك سنحتاج إلى شرط ربط واحد على أقل تقدير (**n-1** ولكننا هنا نجد أن شروط الربط هي :

```
Mark.Mark_Sec = Stu.Stu_Sec And

Mark.Mark_Lvl = Stu.Stu_Lvl And

    Mark.Mark_Stu = Stu.Stu_No
```

الآن لدينا طريقتين لعمل العلاقة بين الجدولين :

1. **طريقة معالج Data Block Wizard :**

حيث نقوم بإنشاء Form تكون فيه الـ Data Block معتمدة على الجدول **STU** وليكن بشكل Tabular يحوي سجلاً وحيداً ، و كذلك إنشاء Data Block معتمدة على الجدول **MARK** وليكن بشكل Tabular يحوي خمسة سجلات ، وأثناء إنشاء Data Block الخاص بالـ MARK عرض [illegible]



كتلة البيانات Data Block Wizard نلاحظ ظهور الشاشة التالية :

فنقوم أولاً بإلغاء الربط التلقائي بين الجدولين

Auto-join Data Blocks

ثم نذهب للزر **Create Relationship...** لنحدد من خلاله الجدول الرئيسي STU أما الجدول التفصيلي MARK فهو محدد تلقائياً بسبب أننا قمنا بتحديد العلاقة من خلاله ونحدد شروط الربط من خلال إنتقاء عنصر من الجدول التفصيلي **Detail Item** وما العنصر الذي يقابله من الجدول الرئيسي **Master Item**

لنرى الشروط تكتب مباشرة أثناء عملية التحديد .

(نضغط التالي ونكمل بقية متطلبات المعالج الخاص بـData Block Wizard)

✍ شرح مبسط :

لو قمنا بتحديد عنصر العلاقة الجديد Relation (غالباً ينشئ بإسم إفتراضي يأخذ فيه إسم الجدولين)

ثم ضغطنا F4 لنرى أهم الخصائص سنجد أنها :

- خاصية شرط الربط **Join Condition** : ومنها تستطيع التعديل على شرط الربط الذي قد أنشئ فيما سبق ،
- خاصية سلوك حذف السجلات **Delete Record Behavior** والتي تمتلك الثلاثة القيم الآتية :

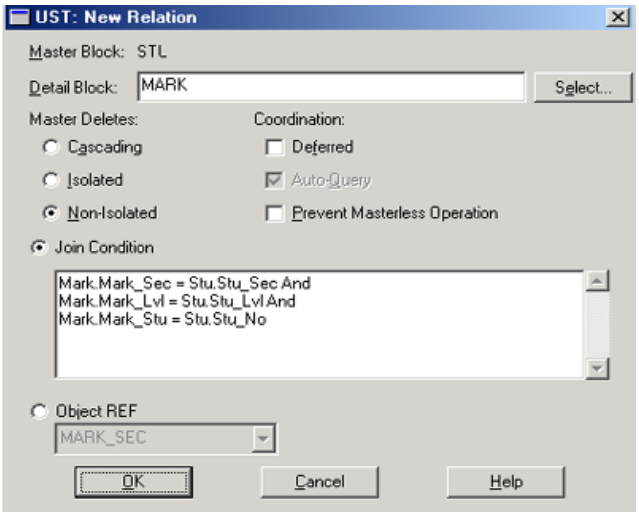
1. **Non_Isolated :** (اللا عزل وهو الإفتراضي)
ويعني أنه لن نستطيع الحذف من الجدول الـMaster إلا بعد حذف كل ما يتعلق به في الجدول الـ Details ،

2. **Isolated :**
ويعني وجود عزل بين الجدول الـMaster والجدول الـ Details بحيث يمكن الحذف من أحد الجدولين دون تأثر الآخر ،

3. **Cascading :**
فإذا حذفنا سجل من الجدول الـMaster فإنه سيحذف كل ما يتعلق بهذا السجل من الجدول الـDetails .

2. الطريقة اليدوية :

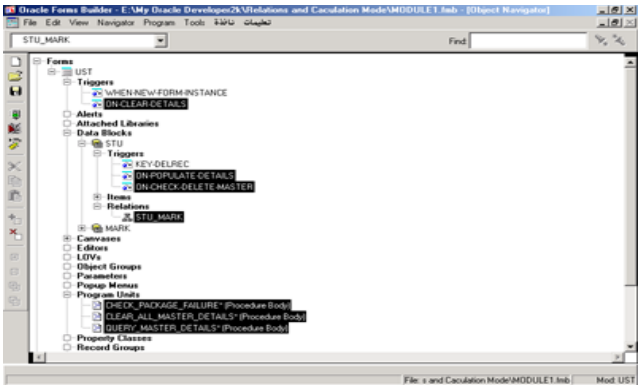
حيث نقوم بإنشاء Form تكون فيه الـ Data Block معتمدة على الجدول **STU** وليكن بشكل Tabular يحوي سجلاً وحيداً ، و كذلك إنشاء Data Block معتمدة على الجدول **MARK** وليكن بشكل Tabular يحوي خمسة سجلات ،



ثم نقوم بتحديد الـ(**Relations**) الموجود في شاشة الـ(Object Navigator) **والتابع للجدول الرئيسي STU** ثم Create لتظهر الشاشة التالية :

حيث نشاهد فيها أن الجدول الرئيسي STU محدد تلقائياً بسبب أننا قمنا بتحديد العلاقة من خلاله ، بينما تتم المطالبة بإختيار الجدول التفصيلي من خلال الزر **Select...** لنختار من LOVs التي تظهر الجدول MARK ؛ ولنلاحظ سلوك حذف السجلات من خلال **Master Deletes** حيث نجد الثلاث الخيارات :

a) Non_Isolated ،
b) Isolated ،
c) Cascading .
والتي تم شرحها في الفقرة السابقة ؛



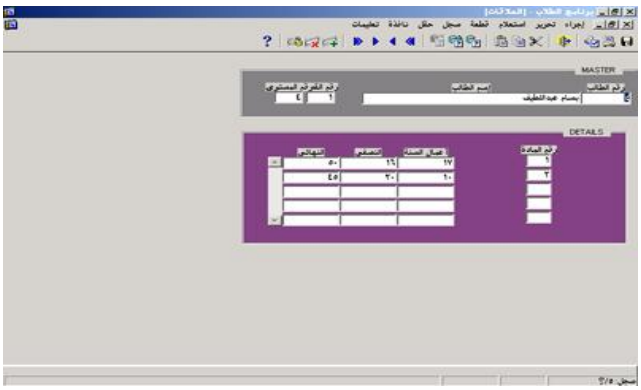
ونكتب شروط الربط في **Join Condition** والتي قد قمنا بذكرها قبل ذلك .

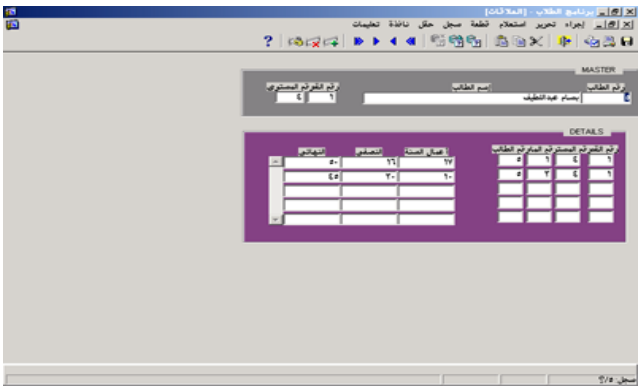
<u>التحديثات على البرنامج بعد إنشاء العلاقة Relation :</u>

1. يُضاف Trigger على مستوى الـ Form
2. يُضاف (2 Triggers) على مستوى كتلة البيانات Data Block للجدول الرئيسي Master .
3. تُضاف علاقة Relation إلى كتلة البيانات Data Block للجدول الرئيسي Master .
4. يُضاف (3 Procedure) إلى وحدات البرنامج Program Units .

ملاحظة هامة :

لا ننسى إخفاء حقول الربط بين الجدولين (**وذلك لعدم مواجهة مشكلة تكرار البيانات**) وذلك من خلال إعطاء القيمة **NULL** للخاصية **Canvas** لكل عنصر ربط ولتخفي العناصر أثناء التصميم والتنفيذ ، أما إذا قمنا بإعطاء القيمة **No** للخاصية **Visible** فإنها ستخفي العناصر أثناء التنفيذ فقط .





وهي هنا حقول :

1. رقم القسم Sec_NO ،
2. رقم المستوى Lvl_No ،
3. رقم الطالب Stu_No ،

a_alyahawi@hotmail.com

❖ الحـقـول الـحـسـابـيـة :

تكون الحقول الحسابية عادة عن طريق

a) عنصر Item غير مرتبط بقاعدة البيانات Non_DB على DataBlock مرتبط بقاعدة البيانات DB ،
b) عنصر Item غير مرتبط بقاعدة البيانات Non_DB على DataBlock غير مرتبط بقاعدة البيانات Non_DB والذي عادة ما يسمى بال(Dummy) .

3. حـقـول الـصـيـغـة Formula :

وهي صيغة رياضية تنتج عنها قيمة عددية .

مثال) مطلوب إنشاء عنصر يحوي مجموع درجات الطلاب (أعمال السنة والنصفي والنهائي)في كتلة البيانات Mark ؟

ننشئ عنصر غير مرتبط بقاعدة البيانات Non_DB وليكن **TDeg** على جدول الدرجات Mark بالخصائص الآتية :

	Property	Value
1	Name	TDeg
2	Item Type	Text Item
3	Enabled	No
4	Data Type	Number
5	Calculation Mode	Formula
6	Formula	NVL(:Mark_Yj , 0) + NVL(:Mark_Ht , 0) + NVL(:Mark_Ft , 0)
7	Database Item	No
8	Canvas	Canvas2
9	Prompt	الدرجة النهائية

4. حـقـول الـتـجـمـيـع Summary :

مثال) مطلوب إيجاد متوسط درجات الطلاب النهائية ؟ (بالإرتباط مع المثال السابق)

a. نقوم بإنشاء Block غير مرتبط بقاعدة البيانات Non_DB وليكن (**Dummy**) بالخصائص الآتية :

	Property	Value
1	Name	Dummy
2	Database Data Block	No

b. ثم ننشئ عنصر غير مرتبط بقاعدة البيانات Non_DB وليكن **Total** بالخصائص الآتية :

	Property	Value
1	Name	Total
2	Item Type	Text Item
3	Enabled	No

6	Summarized Block	Stu
7	Summarized Item	TDeg
8	Summary Function	Sum
9	Database Item	No
10	Canvas	Canvas2
11	Prompt	متوسط الدرجات

c. وليتم التنفيذ بتجنب أي رسالة خطأ
نقوم بتغير قيمة الخاصية Query All Record التابعة للـ(Block) بالدرجات Mark ،

ومن الـ(Block) التحكمي الذي يحوي عنصر التجميع Dummy نغير قيمة الخاصية Single Record .

	Data Block	Property	Value
1	**Mark**	Query All Record	Yes
2	**Dummy**	Single Record	Yes

❖ إيـــجـــاد الـــتـــقـــديـــر :

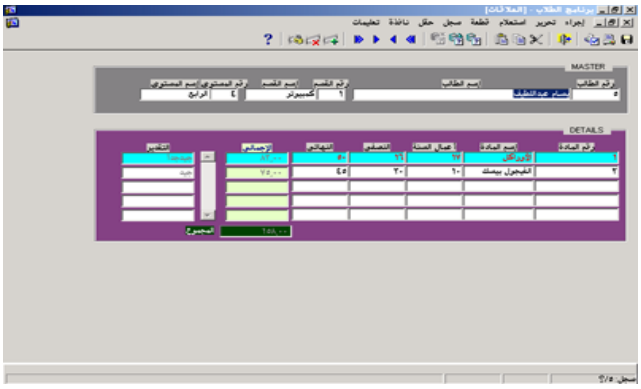
a) نقوم بإنشاء عنصر في كتلة البيانات MARK خصائصه كالتالي :

	Property	Value
1	Name	VALUATION
2	Database Item	No
3	Canvas	Canvas2
4	Prompt	التقدير
5	Prompt Justification	Center
6	Prompt Attachment Edge	Top
7	Prompt Alignment	Center

b) وبما أن التقدير يعتمد على معدل الدرجات لذلك يجب أن يتأثر هذا التقدير إذا قمنا بتعديل أي من الدرجات لذلك نقوم بكتابة هذه الشفرة المصدرية Code في حدث **Post_Change** ولكل من (**أعمال السنة – النصفي – النهائي**) وكالتالي :

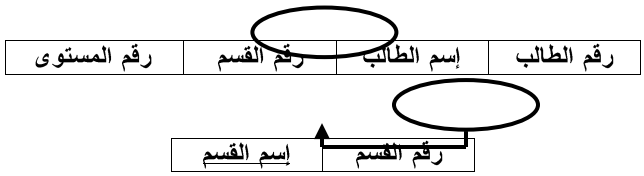
```
Declare
  N   Number;
Begin
  N := NVL(:Mark_Yj , 0) + NVL(:Mark_Ht , 0) + NVL(:Mark_Ft , 0);

  If N Between 90 And 100 Then
:Mark_Valuation := 'ممتاز';

  Elsif N Between 80 And 89 Then
:Mark_Valuation := 'جيدجداً';

    Elsif N Between 70 And 79 Then
:Mark_Valuation := 'جيد';

       Elsif N Between 50 And 69 Then
:Mark_Valuation := 'مقبول';
```

```
ضعيف';                    :Mark_Valuation := '

  End If ;


End ;
```



❖ ربـــط الـــجـــداول :

نحتاج في كثير من الأحيان الربط بين جدولين ليس بشكل Master & Details وإنما بشكل ربط طبيعي من أجل إستخلاص معلومات معينة فمثلاً جدول الطلاب **STU** يحتوي على (رقم القسم **Stu_Sec**) والذي يعتبر حقلاً ممثلاً لمفتاح غريب **Foreign Key** تابع لجدول الأقسام **SECTION** وعن طريق عملية الربط هذه نريد جلب أسماء الأقسام أي أننا نريد إظهار أسماء الأقسام في كتلة البيانات STU من خلال حقل رقم القسم Stu_Sec وبشرط أنه إذا تم إدخال قيمة خاطئة فإن البرنامج يتوقف حتى يدخل المستخدم القيمة الصحيحة .



ولعمل ذلك سنحتاج إلى :

a. إنشاء عنصر في كتلة البيانات STU خصائصه كالتالي :

	Property	Value
1	Name	SEC_NAME
2	Database Item	No
3	Canvas	Canvas2
4	Prompt	إسم القسم
5	Prompt Justification	Center
6	Prompt Attachment Edge	Top
7	Prompt Alignment	Center

b. ربط هذا العنصر (إسم القسم) من خلال الحدث بعد التعديل **Post_Change** التابع لرقم القسم الخاص بكتلة البيانات STU أي الحقل **STU_SEC** وكالتالي :

```
Declare


Begin

Select Sec_Name Into :Sec_Name From Section

Where Sec_No = :Stu_Sec;


Exception

When No_Data_Found Then
```

```
');الـرقـم  غير صحيـح'Message(

Raise Form_Trigger_Failure;

End ;
```

وبنفس الطريقة يتم إيجاد إسم المستوى المعتمد على رقم المستوى ... إلخ .

==========

تلميح :

من الممكن إنشاء رسالة Alert بدلاً من رسالة Message ولتكن خصائصها كالتالي :

	Property	Value
1	Name	Msgbox
2	Title	تنبيه
3	Message	الرقم غير صحيح
4	Alert Style	Caution
5	Button 1 Label	موافق
6	Default Alert Button	Button1

ثم بعد ذلك نقوم بإستدعاء الرسالة أثناء معالجة الأخطاء **Exception** وكالتالي :

```
N := Show_Alert('Msgbox') ;
```

وذلك بعد تعريف المتحول في جزء **Declare** وكالتالي :

```
N Number ;
```

من أجل أن الأوراكل تتطلب إسناد الدوال لمتحول أثناء إستدعائها .

ملاحظة هامة :

يفضل دوماً التحكم بكتلة بيانات DB عن طريق عناصر Non_DB بحيث تكون آتية من كتلة بيانات Non_DB والتي ما نرمز لها عادة بالرمز Dummy أي كتلة تحكم ، وفي مثال ربط الجداول السابق فإنه يفضل أن ننشئ عنصر إسم القسم SEC_Name في كتلة بيانات Non_DB ولتكن Dummy والذي من خصائصه :

Property	Value
Database Data Block	No

وإذا أنشئنا إسم القسم هذا من نوع List Item وليكن إسمه **List_Item_Sec** (راجع كيف تنشئ القائمة من المحاضرة السابعة) وأردنا أن تكون البيانات المعروضة فيه تؤثر على ما يعرض في كتلة البيانات STU بشكل فعال فإننا نقوم بكتابة الشفرة المصدرية في حدث (تعديل القائمة) وكالتالي :

(Trigger : WHEN_LIST_CHANGED)

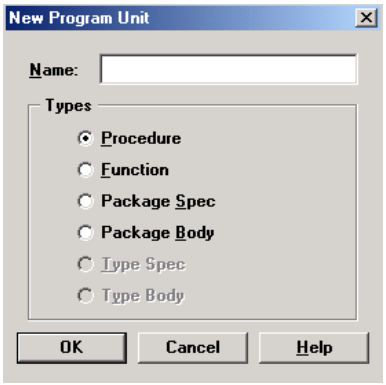
```
Set_Block_Property('Stu'    ,    'Default_Where'    ,    'Stu_Sec    =
                                        :Dummy.List_Item_Sec') ;
```

ومن أجل ألا نختار عنصر من القائمة List_Item_SEC ثم نضطر إلى الذهاب إلى كتلة البيانات STU لتنفيذ الإستعلام أي نجعل ظهور البيانات في كتلة البيانات STU مرتبطاً بإختيار القيمة من List_Item_SEC فنضيف في حدث (تعديل القائمة) بعد التعليمة السابقة ما يلي :

```
Go_Block('Stu') ;

Execute_Query ;
```

==========

❖ الوحدات البرمجية Program Units :



تستخدم الوحدات البرمجية لإنشاء الإجراءات Procedure أو الدوال Function أو الحزم البرمجية Package Spec ، وذلك من خلال بتحديد الـ(**Program Units**) الموجود في شاشة الـ(Object Navigator) ثم Create لتظهر الشاشة التالية :

فنكتب إسم الوحدة البرمجية المراد إنشائها داخل الـ **(Name)** ثم نختار نوع الوحدة البرمجية هل هي :

- **إجراء Procedure** ،
- **دالة Function** ،
- **حزمة برمجية Package Spec** ، (نحدد فيها أسماء الوحدات البرمجية)
- **جسم الحزمة Package Body** ،

(والذي يتم فيه كتابة الشفرة المصدرية Code للإجراءات والدوال التي تم تجميعها في Package Spec) .

a) مطلوب إنشاء عنصر في كتلة البيانات Mark يعرض مجموع درجات الطلاب (أعمال السنة والنصفيوالنهائي) ؟ (بإستخدام الإجراءات Procedure)

b) مطلوب إنشاء عنصر في كتلة البيانات Stu يعرض إسم القسم ؟ (بإستخدام الدوال Function)

حل المطلوب الأول

أولاً) ننشئ عنصر غير مرتبط بقاعدة البياناتNon_DB وليكن **TDeg** بالخصائص الآتية :

	Property	Value
1	Name	TDeg
2	Item Type	Text Item
3	Enabled	No
4	Data Type	Number
5	Database Item	No
6	Canvas	Canvas2
7	Prompt	مجموع الدرجات

ثانياً) نحدد الـ(**Program Units**) الموجود في شاشة الـ(Object Navigator) ثم Create لتظهر شاشة الوحدات البرمجية فنكتب في الـ **Name** إسم الإجراء وليكن **ProTDeg** ونختار نوع الوحدة البرمجية إجراء **Procedure** ثم نضغط **Ok** ، فيظهر محرر كتابة الشفرة المصدرية لنكتب التالي :

```
PROCEDURE ProTDeg IS

BEGIN

  :Tdeg := NVL(:Mark_Yj , 0) + NVL(:Mark_Ht , 0) + NVL(:Mark_Ft , 0);

END;
```

ثالثاً) وبما أن المجموع يتأثر إذا قمنا بتعديل أي من الدرجات لذلك نقوم بإستدعاء الإجراء Procedure من خلال كتابة هذه الشفرة المصدرية Code في حدث **Post_Change** ولكل من (**أعمال السنة – النصفي – النهائي**) وكالتالي :

```
ProTDeg ;
```

حل المطلوب الثاني

أولاً) ننشئ عنصر [illegible] الآتية :.

	Property	Value
1	Name	SEC_NAME
2	Item Type	Text Item
3	Enabled	No
4	Data Type	Char
5	Database Item	No
6	Canvas	Canvas2
7	Prompt	إسم القسم

ثانياً) نحدد الـ(**Program Units**) الموجود في شاشة الـ(Object Navigator) ثم Create لتظهر شاشة الوحدات البرمجية فنكتب في الـ**Name** إسم الإجراء وليكن**FunSec** ونختار نوع الوحدة البرمجية إجراء **Function** ثم نضغط **Ok** ، فيظهر محرر كتابة الشفرة المصدرية لنكتب التالي :

```
FUNCTION FunSec RETURN Char IS
  X Varchar2(50);
BEGIN
  Select Sec_Name Into X From Section
  Where Sec_No = :Stu_Sec ;

  Return X ;
END;
```

ثالثاً) ثم نقوم بإستدعاء الدالة Function من خلال كتابة هذه الشفرة المصدرية Code في حدث بداية الدخول على السجل وكالتالي :

(Trigger : WHEN_NEW_RECORD_INSTANCE)

```
:Sec_Name := FunSec ;
```

==========

ملاحظات هامة :

- يمكن إستدعاء الإجراء بكتابته مباشرة ، أما الدالة فيجب أن تسند لمتحول أثناء إستدعائها لأنه من الممكن أن يستخدم إسم الدالة مع العمليات الحسابية ،
- لا يتطلب من الإجراء إرجاع أي قيمة بينما يجب أن تعيد الدالة قيمة لذلك فإنه عند تحرير الشفرة المصدرية للدوال سنرى بداية محرر الشفرة المصدرية بهذه الطريقة :

FUNCTION FunSec RETURN _ IS

فنقوم بكتابة ما ترجعه الدالة بحيث نعدل محرر الشفرة المصدرية وبهذه الطريقة :

FUNCTION FunSec RETURN Char IS

❖ إنـــشـــاء عـــنـــصـــر List Item :

قبل إنشاء List Item يجب معرفة هل نريد أن تكون القيم الظاهرة فيها :

- **ثابتة (إستاتيكية) Static :** وبالتالي فهي تمتلك قيم ثابتة لا يمكن تعديلها ؛
- **متغيرة (ديناميكية) Dynamic :** وبالتالي فهي تمتلك قيم متغيرة تعكس صورة السجلات في قاعدة البيانات وجلب البيانات فيها سيكون بواسطة الإستعلام Query .

مع ملاحظة أن إنشاء الـ(List Item) يكون **يدوي دائماً .**

أمثلة على ذلك :

مطلوب عمل List Item في الكتلة الخاصة بالجدول Stu بحيث يكون :

a) **حقل رقم المستوى مرتبطاً مع حقل إسم المستوى (يعرض بيانات ثابتة) ،**
b) **حقل رقم القسم مرتبطاً مع حقل إسم القسم (يعرض بيانات متغيرة) ؟**

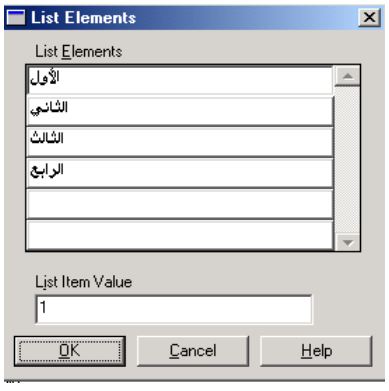
حل المطلوب الأول

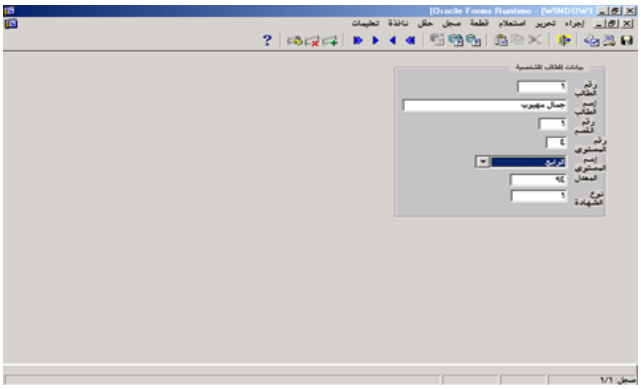
بعد إنشاء Form تكون فيه الـData Block معتمدة على الجدول Stu نلاحظ توفر الحقول :

(رقم الطالب Stu_No ، إسم الطالب Stu_Name ، رقم القسم Stu_Sec ، رقم المستوى Stu_Lvl ، المعدل Stu_Avg ، نوع الشهادة Stu_Certtype)

مما يستوجب توفير حقل إسم المستوى وذلك من خلال إنشاء عنصر جديد Item خصائصه كما يلي :

	Property	Value
1	Name	LVL_NAME
2	Item Type	List Item
3	Element in List	**More...**
4	Mapping of Other Values	4
5	Initial Value	2
6	Copy Value from Item	STU_LVL
7	Synchronize with Item	STU_LVL
8	Database Item	No
9	Canvas	Canvas2





لتظهر الشاشة أعلاه والتي من خلالها :

- ندخل قيم الحقل LVL_Name في **List Elements**
- نحدد القيم المقابلة لها في حقل رقم المستوى **List Item Values** ،

مع وجوب أن يكون إدخال القيم متناظر في جميع الأعمدة (الحقول) فمثلاً :

القيمة **الأول** في **List Elements** يقابلها القيمة **1** في **List Item Values** .

==========

✍ شرح مبسط :

- لاحظ عمل خاصية نسخ القيمة للعنصر إلى داخل قاعدة البيانات **Copy Value from Item** (إلى STU_LVL)

وذلك مفيد جداً عند إضافة سجل جديد (غالباً ما يتم إخفاء حقل STU_LVL بواسطة الخاصية Canvas

والإعتماد على LVL_NAME كمدخلات معطاة من المستخدم وهنا تأتي الفائدة العظمى لهذه الخاصية) ،

- لاحظ عمل خاصية التزامن مع العنصر **Synchronize with Item** والتي تعمل على إظهار قيم الـ List Item

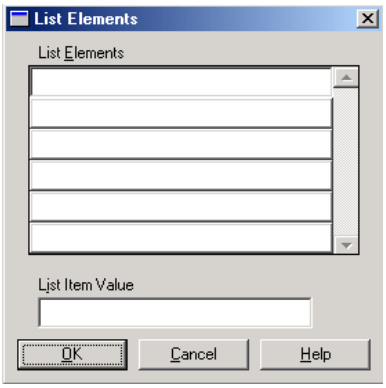
بعد التعديل على قيمة حقل رقم المستوى STU_LVL (شبيهة في طريقة العمل لـ : Trigger Post_Change)

على حقل STU_LVL) ،

- لاحظ أن List Item شبيهة في طريقة العمل للدالة **Decode** في بيئة SQL PLUS .

أولاً) توفير حقل إسم القسم وذلك من خلال إنشاء عنصر جديد Item خصائصه كما يلي :

	Property	Value
1	Name	LVL_NAME
2	Item Type	List Item
3	Element in List	**More...**
4	Mapping of Other Values	
5	Initial Value	
6	Copy Value from Item	STU_SEC
7	Synchronize with Item	STU_SEC
8	Database Item	No
9	Canvas	Canvas2

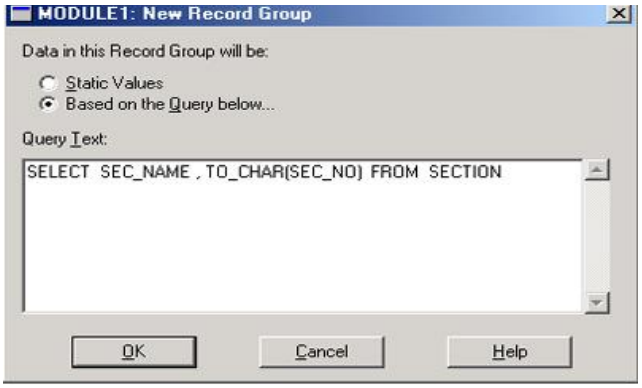


لاحظ كيف قمنا بتفريغ كلٍ من :

a) جميع القيم من Element In List ،
b) خاصيتي Mapping of Other Values & Initial Value .

ثانياً) ننشئ مجموعة سجل **Record Group** متغيرة (ديناميكية) Dynamic حيث نقوم بتحديد الـ(Record Group) الموجود في شاشة الـ(Object Navigator) ثم Create لتظهر الشاشة التالية فنختار خيار **Based on the Query Below...** ونكتب الإستعلام ثم نضغط على الزر OK

```
select Sec_Name , to_char(Sec_No) from section ;
```



إذن نقوم بجلب البيانات الخاصة بإسم القسم ورقم القسم (بعد تحويله من رقم إلى نص عبر الدالة to_char() وذلك ليكون مجموعة سجل Record Group قادراً على الإستعلام) ، ويجب أيضاً أن نجعل نوع البيانات لرقم القسم Char ولتتم عملية التطابق ،

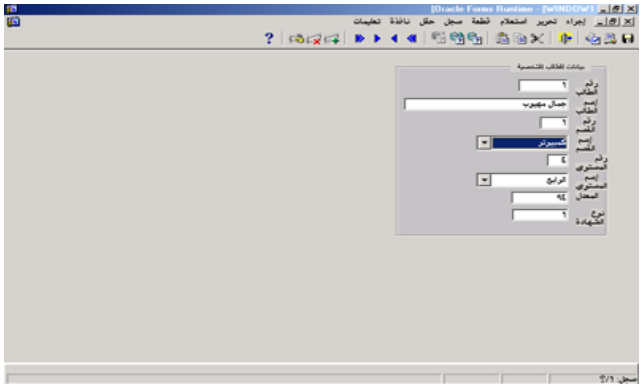
Property	Value
Data Type	Char

ولنسمي هذا الـ(Record Group) برمجياً من خلال الضغط على F4 وتحديد قيمة الخاصية Name بـ **RG_SEC_DYNAMIC** .

ثالثاً) نقوم بكتابة الشفرة المصدرية من List Item ولنملئ الـ Form عند حدث تحميل النموذج Code قاعدة البيانات :

(Trigger : WHEN_NEW_FORM_INSTANCE)

```
Declare
   N   Number;
Begin
   N := Populate_Group('RG_SEC_DYNAMIC');
   Populate_List('Sec_Name','RG_SEC_DYNAMIC');
End ;
```



❖ القائمة المركبة LOVs (List Of Values)

هي عبارة عن قائمة تحتوي مجموعة قيم نقوم بانتقاء إحداها وإعادتها إلى حقول في الـ (Form) ، وتتميز بأنها :

a
b ... إلخ ؛ حجمه وعنوانه

c) إمكانية البحث Search والتصفية (الفلترة) Filter .

==========

❖ إنـــشـــاء LOVs (List Of Values) :

قبل إنشاء قائمة مركبة LOVs يجب معرفة هل نريد أن تكون القيم الظاهرة فيها :

- **ثابتة (إستاتيكية) Static :** وبالتالي فهي تمتلك قيم ثابتة لا يمكن تعديلها ؛
- **متغيرة (ديناميكية) Dynamic :** وبالتالي فهي تمتلك قيم متغيرة تعكس صورة السجلات في قاعدة البيانات وجلب البيانات فيها سيكون بواسطة الإستعلام Query .

مع ملاحظة أن الـ(LOVs) لا تطبق عادة إلا على عنصر Text Item ،

وإنشائها إما أن يكون **يدوياً** أو عن طريق المعالج الخاص بها **LOVs Wizard** [جديد (Form Builder 6i)] .

وبشكل عام فإن إنشاء أيLOVs يتطلب إنشاء مجموعة سجلRecord Group

وبحسب نوع مجموعة سجلRecord Group يتحدد نوع الـLOVs

أمثلة على ذلك :

مطلوب عمل LOVs في الكتلة الخاصة بالجدول Mark بحيث يكون :

c) **حقل رقم المستوى مرتبطاً مع حقل إسم المستوى (يعرض بيانات ثابتة) بإستخدام الطريقة اليدوية Manually ،**
d) **حقل رقم القسم مرتبطاً مع حقل إسم القسم (يعرض بيانات متغيرة) بإستخدام المعالج Wizard ؟**

حل المطلوب الأول

بعد إنشاء Form تكون فيه الـData Block معتمدة على الجدول Mark نلاحظ توفر الحقول :

(رقم القسم Mark_Sec ، رقم المستوى Mark_Lvl ، رقم الطالب Mark_Stu ، رقم المادة Mark_Sub ،

أعمال السنة Mark_Yj ، النصفي Mark_Ht ، النهائي Mark_Ft) مما يستوجب :

أولاً) توفير حقل إسم المستوى وذلك من خلال إنشاء عنصر جديد Item خصائصه كما يلي :

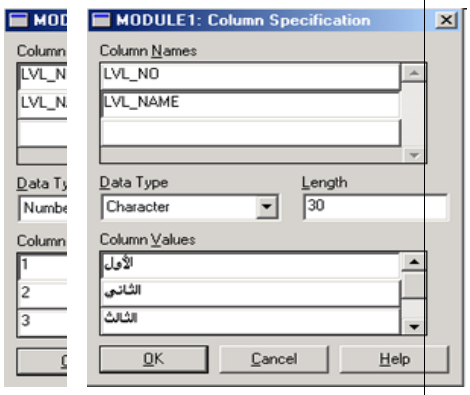
	Property	Value
1	Name	Lvl_Name
2	Database Item	No
3	Canvas	Canvas2
4	Prompt	إسم المستوى

ثانياً) [illegible] بتحديد الـ(Record
p [illegible] خيار **Static**
es [illegible] :

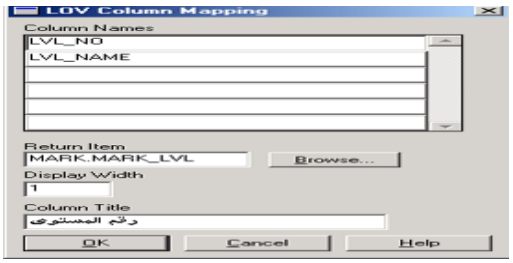
(ندخل إسم الحقل LVL_NO في **Column Names** ونحدد نوعه Number في **Data Type** أما طوله **Length** فهو محدد تلقائياً للأرقام ومن ثم ندخل جميع القيم الممكنة لهذا الحقل [1,2,3,4] **Column Values**) ،

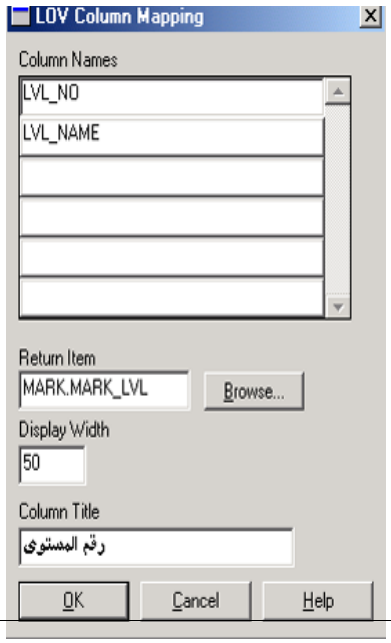
ونكرر ذلك مع حقل إسم المستوى حيث :

(ندخل إسم الحقل LVL_NAME في **Column Names** ونحدد نوعه Cheterar في **Data Type** وطوله **Length** هو 30 ومن ثم ندخل جميع القيم الممكنة لهذا الحقل [الرابع ,الثالث ,الثاني ,الأول] **Column Values**) ، مع وجوب أن يكون إدخال القيم متناظر في جميع الأعمدة (الحقول) فمثلاً القيمة **1** في رقم المستوى يقابلها القيمة **الأول** في إسم المستوى ، ولنسمي هذا الـ(Record Group) برمجياً من خلال الضغط على F4 وتحديد قيمة الخاصية Name بـ**RG_LVL_STATIC**



ثالثاً) ننشئ قائمة مركبة **LOVs** معتمدة على نوع Record Group السابق [ثابتة (إستاتيكية) Static] حيث نقوم بتحديد الـ(LOVs) الموجود في شاشة الـ(Object Navigator) ثم Create لتظهر شاشة تحديد طريقة الإنشاء هل هي بإستخدام المعالج أم يدوياً فنختار (الخيار اليدوي) **Build a new LOV manually** ، ولتكون الخصائص فيها كما يلي :





	Property	Value
1		
2		
3		

4	Column Mapping Properties	**More...**
5	Filter before Display	Yes
6	Automatic Display	Yes
7	Automatic Skip	Yes

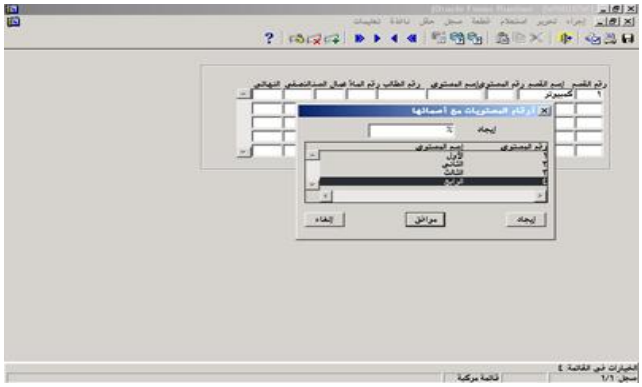
لتظهر الشاشة المجاورة والتي من خلالها :

(ندخل إسم الحقل LVL_NO في **Column Names** ونحدد إرجاعه إلى حقل MARK.MARK_LVL **Return Item** عن طرق الزر ...Browse وعرضه 50 أثناء عرض القيم في LOVs وعنوان هذا الحقل رقم المستوى **Column Title**) ، ونكرر ذلك مع حقل إسم المستوى حيث :

(ندخل إسم الحقل LVL_NAME في **Column Names** ونحدد إرجاعه إلى حقل MARK.LVL_NAME **Return Item** عن طرق الزر ...Browse وعرضه 100 أثناء عرض القيم في LOVs وعنوان هذا الحقل إسم المستوى **Column Title**) .

رابعاً) بعد تصميم الـLOVs نحدد ظهورها على **حقل رقم المستوى MARK_LVL** وكما يلي :

Property	Value
List of Values	LOV_LVL_STATIC



حل المطلوب الثاني

أولاً) توفير حقل إسم القسم وذلك من خلال إنشاء عنصر جديد Item خصائصه كما يلي :

	Property	Value
1	Name	Sec_Name
2	Database Item	No
3	Canvas	Canvas2
4	Prompt	إسم القسم

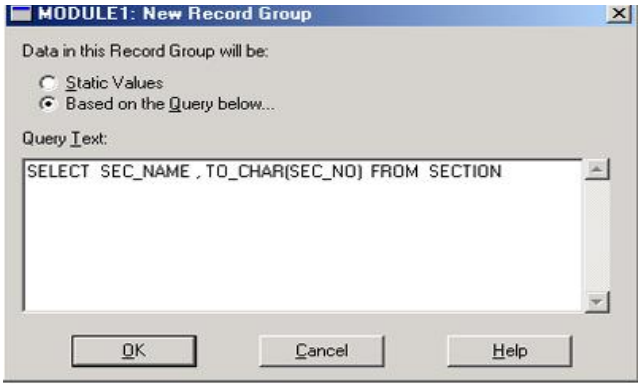
ثانياً) إنشاء الـ(Trigger) الذي [illegible] (**Trigger : Post_Change**)

[illegible] Mark.Sec_[illegible] وكالتالي :

```
Declare


Begin

Select Sec_Name Into :Sec_Name From Section

Where Sec_No = :Mark_Sec;


Exception

When No_Data_Found Then

');الـرقـم غير صحيـحMessage('

Raise Form_Trigger_Failure;

End ;
```

ثالثاً) ننشئ مجموعة سجل **Record Group** متغيرة (ديناميكية) Dynamic حيث نقوم بتحديد الـ(Record Group) الموجود في شاشة الـ(Object Navigator) ثم Create لتظهر الشاشة التالية فنختار خيار **Based on the Query Below...** ونكتب الإستعلام ثم نضغط على الزر OK

```
select Sec_Name , to_char(Sec_No) from section ;
```



إذن نقوم بجلب البيانات الخاصة بإسم القسم ورقم القسم (بعد تحويله من رقم إلى نص عبر الدالة to_char() وذلك ليكون مجموعة سجل Record Group قادراً على الإستعلام) ، ويجب أيضاً أن نجعل نوع البيانات لرقم القسم Char ولتتم عملية التطابق ،

Property	Value
Data Type	Char

ولنسمي هذا الـ(Record Group) برمجياً من خلال الضغط على F4 وتحديد قيمة الخاصية Name بـ **RG_SEC_DYNAMIC** .

ثالثاً) ننشئ قائمة مركبة **LOVs** معتمدة على نوع Record Group السابق [متغيرة (ديناميكية) Dynamic] حيث نقوم بتحديد الـ(LOVs) الموجود في شاشة الـ(Object Navigator) ثم Create لتظهر شاشة تحديد طريقة الإنشاء هل هي باستخدام المعالج أم يدوياً فنختار (خيار [illegible]

Use the LOV Wizard ،

لتظهر الشاشة الأولى في معالج LOVs والتي تقوم بالسؤال عن توفر إحدى متطلباتها وهو مجموعة سجل Record Group فهل :

- تريد إنشاء مجموعة سجل جديدة تعتمد على الإستعلام ،
- يوجد مجموعة سجل منشئة مسبقاً .

(حالياً سنختار الخيار الثاني [المنشئة مسبقاً] **ومنها بالتحديد RG_SEC_DYNAMIC ، ثم نضغط التالي)**

قد تتساءل لماذا لم نجد الشاشة الترحيبية كما هي العادة في بقية Wizards والسبب في ذلك يعود إلى عدم تنشيطها فإذا أردت ذلك فقم بالخطوات التالية (من خلال قائمة الأدوات) :

Tools → Preference... → Wizard →

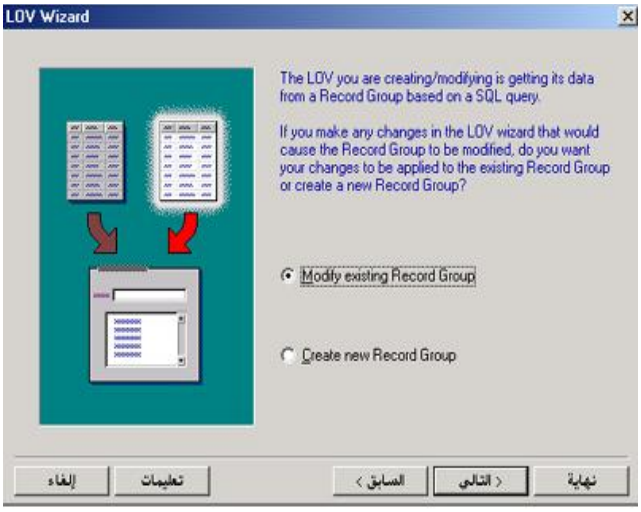
☑ LOV Wizard Welcome Page

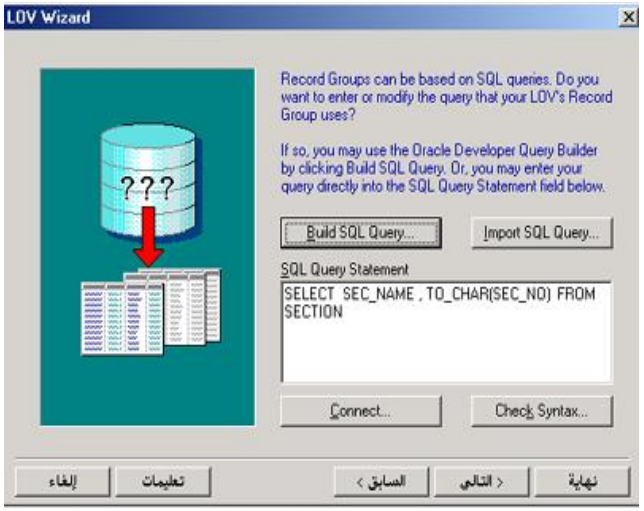
ثم تظهر شاشة تخبرك بأنك هل :

- تريد التعديل على مجموعة سجل المنشئةRecord Group ،
- تريد إنشاء مجموعة سجل جديدة .

(حالياً سنختار الخيار الأول ثم نضغط التالي)

لتظهر بعدها شاشة تأكيد مجموعة السجل Record Group حيث يمكنك تحريرها مباشرة في SQL Query Statement أو بناءها من جديد من خلال برنامج باني الإستعلام ...Build SQL Query أو إستيرادها من ملف ...Import SQL Query وهناك أيضاً زر الإتصال إن لم تكن متصلاً بالقاعدة ...Connect و زر فحص الصيغة التي

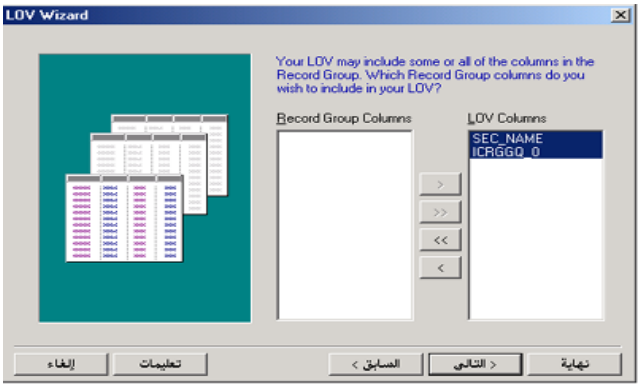


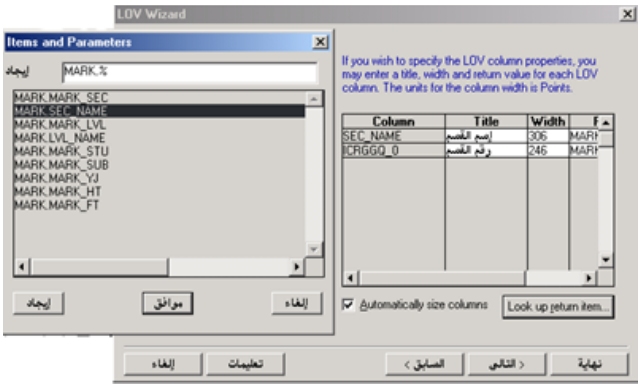


تمت كتابتها (لا تضع الفاصلة المنقوطة) ...Check Syntax **(حالياً نضغط التالي)**

بعدها تظهر شاشة لتختار الحقول التي جلبتها من إستعلام مجموعة سجل Record Group وما الذي تريد جلبه وإظهاره على القائمة المركبة LOVs **(حالياً سنختار جلب الكل الزر >> ثم نضغط التالي)**

ثم تظهر أهم شاشة من شاشات المعالج وهي شاشة تحديد العناوين الظاهرة للحقول في القائمة المركبة Title والحجم الخاص بالعرض Width والأهم ربط الحقول التي جلبت بالإستعلام وإرجاعها للحقول المراد إرجاع البيانات إليها **Lock Up Return Item... (حالياً سنقوم بإرجاع SEC_NAME إلى MARK.SEC_NAME & الحقل ICRGGQ_0 إلى MARK.MARK_SEC ثم نضغط التالي)**





لتظهر بعد ذلك شاشة نحدد من خلالها العنوان العام للقائمة المركبة Title وكذلك عرض القائمة width وإرتفاعها height وتحديد ما إذا كنت تريد إظهار القائمة المركبة في إحداثيات أتوماتيكية أو تقوم بتحديد الإحداثيات يدوياً , **(حالياً سنعطي العنوان** أرقام الأقسام مع أسمائها **ثم نضغط التالي)**

ثم تظهر شاشة لتحديد إمكانيات :

- عدد الصفوف الظاهرة ،
- إمكانية تحديث البيانات قبل عرضها في LOVs ،
- إمكانية الفلترة للمستخدم عند بداية عمل LOVs .

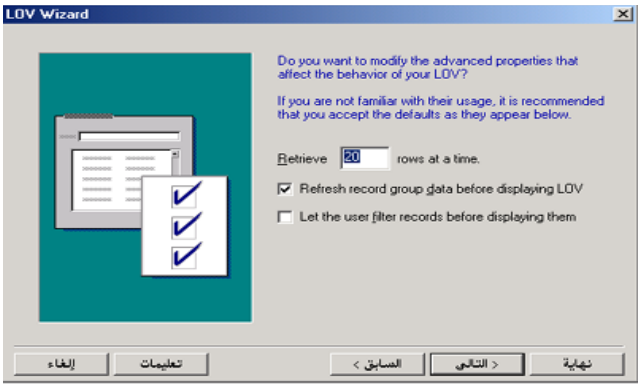
(حالياً نضغط التالي)

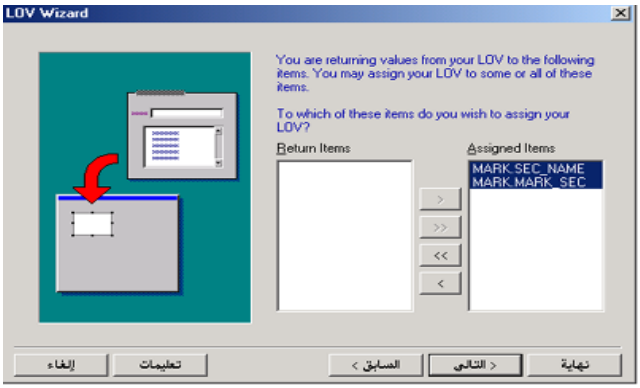
لتظهر شاشة التأكيد على حقول الإرجاع للتأشير عليها

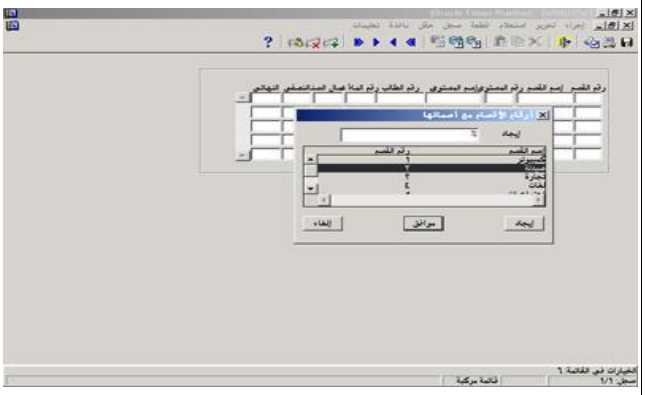
(حالياً سنختار جلب الكل الزر >> ثم نضغط التالي)

فينتهي بذلك معالج القائمة المركبة **(نضغط نهاية)**

a_alyahawi@hotmail.com







ولنسمي هذا الـ(LOVs) برمجياً من خلال الضغط على F4 وتحديد قيمة الخاصية Name بـ**LOV_LVL_STATIC** .

📖 **نلاحظ أنه قد تم من خلال المعالج ما يلي :**

- للخاصية Title القيمة **أرقام الأقسام مع أسمائها** في الـLOVs ،
- للخاصية Record Group القيمة **RG_SEC_DYNAMIC** في الـLOVs ،
- للخاصية Column Mapping Properties في الـLOVs تم **إرجاع القيم** للحقول المعنية بذلك .
- إعطاء خاصية List of Values القيمة **LOV_SEC_DYNAMIC** في الـMARK_SEC ،

وكذلك في الـSEC_NAME (ولتتجنب ظهور القائمة **مرتين** إجعل قيمة الخاصية في الـSEC_NAME تساوي **NULL**) .

==========

❖ **مـــن خـــصـــائـــص LOVs (List Of Values) :**

1. **الخاصية Filter before Display :** تستخدم عندما تكون السجلات كثيرة حيث تظهرLOVs عند التنفيذ جاهزة لإدخال جزء من قيم الحقل المراد إظهاره وذلك عند جعل الخاصيةYes .
2. **الخاصية Automatic Display :**

إذا جعلناها Yes فإنه سيتم إظهار الـLOVs مباشرة عند وصول المؤشر إلى الحقلMARK_SEC مثلاً وذلك بدون :

- الضغط على F9 لإظهار القائمة المركبة ،
- كتابة شفرة مصدرية Code كما في هذا المثال :

(Trigger : WHEN_NEW_ITEM_INSTANCE)

```
Declare

 J Boolean;

Begin

 J := Show_Lov('LOV_SEC_DYNAMIC');

End ;
```

3. **الخاصية Automatic Skip :** إذا كانت Yes فإن المؤشر ينتقل للحقل الذي بعد حقل الـLOVs بعد الإختيار وإذا كانت No يظل في نفس الحقل بعد الإختيار ، (بالطبع الزمن عامل مهم في قواعد البيانات) .

==========

✍ **ملاحظات هامة :**

a) يوجد تشابه بين **LOVs** و **List Item** ولكن تتميز الـLOVs بوجود الخاصية **Find** والتي تسهل إيجاد قيمة معينة إذا كانت القيم كثيرة .

b) يفضل إستخدام الـ**List Item** إذا كانت القيم ثابتة **Static Values** بينما يفضل إستخدام الـ**LOVs** إذا كانت القيم متغيرة **Dynamic Values** .

c) [illegible] من قاعدة البيانات

المحاضرة الثامنة

❖ إنشاء عنصر Radio Group & Radio Button :

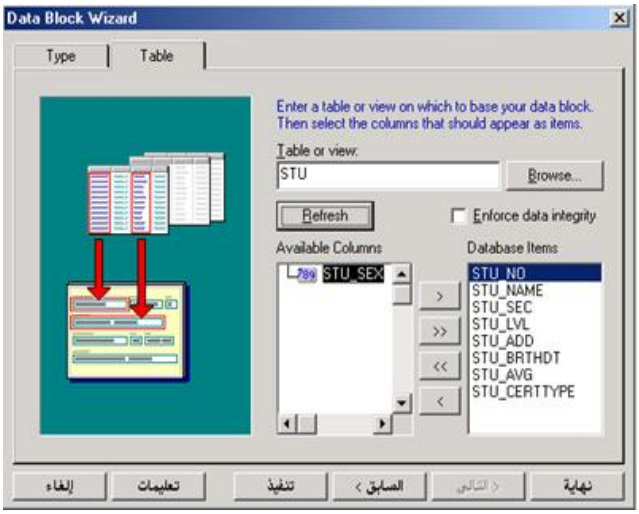
من المعلوم أن عناصر الإختيار المفرد لا تكون إلا لقيم ثابتة مثل (الجنس ، الحالة الإجتماعية ... إلخ) ،

وهذه العناصر يجب أن تجمعها حاوية **Container** حتى يمكن إختيار خيار واحد من كل مجموعة وهنا في Oracle

قد تم وجب تصميم الحاوية Container أولاً ثم إنشاء العناصر التابعة لها أي أنه :

يتم إنشاء Radio Gruop أولاً ثم العناصر التابعة لها Radio Buttons

مثال ذلك :



مطلوب إضافة حقل الجنس (ذكر/أنثى) لجدول الطلاب Stu ومن ثم عرض البيانات في نموذج Form ؟

ولعمل ذلك لا بد أولاً من إضافة هذا الحقل بأمر SQL وكالتالي :

```
SQL>    alter    Table    Stu
Add(Stu_Sex number(1));
```

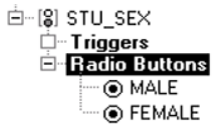
(إذا قمت بإنشاء الحقل بعد جلب البيانات إلى الـForm فقم بتحديد الـ **Data Block (STU)** ثم إنقر على زر الفأرة الأيمن لتختار **Data Block Wizard** ومنه التبويب **Table** ثم إضغط الزر **Refresh** وبعدها قم بجلب العنصر STU_SEX ثم إضغط **نهاية**)

أو نقوم بإنشاء Form تكون فيه الـData Block معتمدة على الجدول STU لنلاحظ أن الحقل STU_SEX قد أخذ الشكل الإفتراضي للعناصر وهو Text Item ، ولجعله بشكل Radio Buttons منتمياً إلى Radio Group نقوم بتعديل قيم الخصائص التالية :

	Property	Value
1	Name	STU_SEX
2	Item Type	Radio Group
3	Mapping of Other Values	

4	Initial Value	1

نلاحظ بعد ذلك إختفاء الحقل من شاشة التصميم Layout Editor لأنه قد تم تكوين Radio Group ، ولم يتكون أي عناصر إختيار Radio Buttons حتى الآن لذلك نقوم بتحديد الـ Radio Group (STU_SEX) ومنه **Radio Buttons** ثم Create لعنصرين خصائصهما كالآتي :

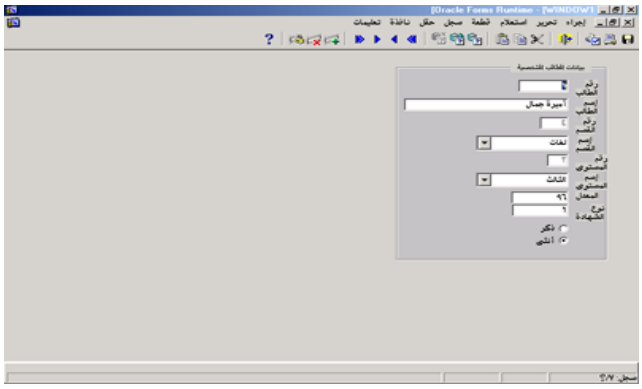


	Property	Value
1	Name	MALE

	Property	Value
1	Name	FEMALE

إذا لم يتم ظهور العنصرين فإذهب إلى خصائص Radio Group وأعطي القيمة Canvas2 للخاصية Canvas ،

لنلاحظ بعدها تنفيذ البرنامج وظهور القيم على شكل إختيارات مفردة بدلاً من القيم المدخلة كما في Text Item .



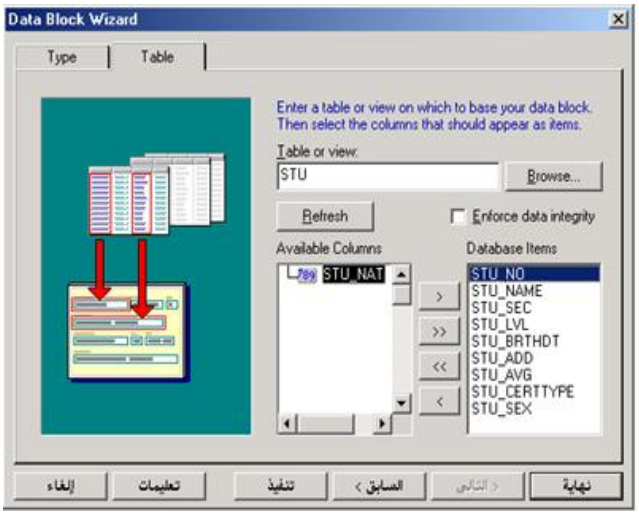
❖ إنـــشـــاء عـــنـــصـــر Check Box :

من المعلوم أن عناصر الإختيار المتعدد لا تكون إلا لقيم ثابتة مثل (الجنسية ، الحالة الإجتماعية ... إلخ) ،

ومن الممكن إختيار أكثر من خيار واحد .

مثال ذلك :

مطلوب إضافة حقل الجنسية (يمني/أجنبي) لجدول الطلاب Stu ومن ثم عرض البيانات في نموذج Form ؟



ولعمل ذلك لا بد أولاً من إضافة هذا الحقل بأمر SQL وكالتالي :

```
SQL>  alter  Table  Stu
      Add(Stu_Nat number(1));
```

(إذا قمت بإنشاء الحقل بعد جلب البيانات إلى الـForm فقم بتحديد الـ **Data Block (STU)** ثم إنقر على زر الفأرة الأيمن لتختار **Data Block Wizard** ومنه التبويب **Table** ثم إضغط الزر **Refresh** وبعدها قم بجلب العنصر STU_NAT ثم إضغط **نهاية**)

أو نقوم بإنشاء Form تكون فيه الـData Block معتمدة على الجدول STU لنلاحظ أن الحقل STU_NAT قد أخذ الشكل الإفتراضي للعناصر وهو Text Item ، ولجعله بشكل Check Box نقوم بتعديل قيم الخصائص التالية :

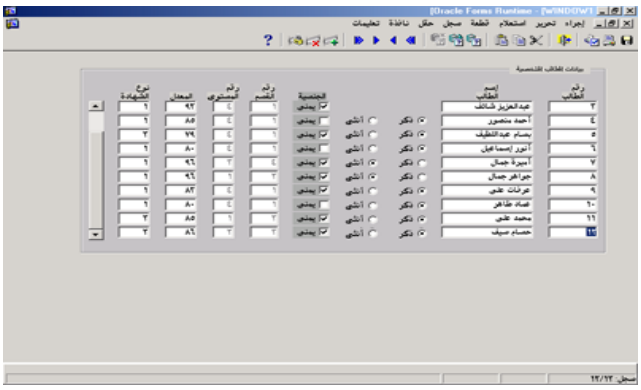
	Property	Value
1	Name	STU_NAT
2	Item Type	Check Box
3	Label	يمني
4	Value when Checked	1
5	Value when Unchecked	0
6	Check Box Mapping of Other Values	Checked
7	Initial Value	1

نلاحظ بعد ذلك أنه قد تم تكوين Check Box ،

وكذلك تنفيذ البرنامج وظهور القيم على شكل إختيارات متعددة بدلاً من القيم المدخلة كما في Text Item .

لاحظ عمل كل من :

- خاصية **Value when Checked** والتي ستقوم بإعطاء الإختيار النشط إذا كانت القيمة مساوية لحقل STU_NAT ،
- خاصية **Value when Unchecked** والتي ستقوم بإعطاء الإختيار الغير نشط إذا كانت القيمة غير مساوية القيمة لحقل STU_NAT ،
- خاصية **Check Box Mapping of Other Values** والتي ستقوم بإعطاء الخيار النشط **Checked** أو الإختيار غير النشط **Unchecked** (حسب تحديد المبرمج) إذا وجدت قيمة في حقل STU_NAT مغايرة لما ذكر في الخاصيتين السابقتين .



❖ أنـــواع الـــقـــمـــاشـــيـــة Canvases :

القماشية تعني الخلفية (الورقة) أو الأرضية التي تتوضع عليها العناصر Items ،

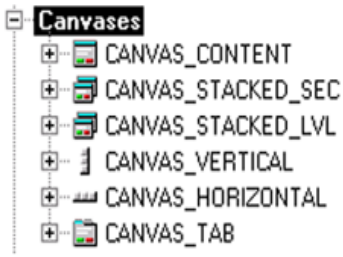
ولها عدة أنواع هي :

1. **Content :** وهي على مساحة النافذة التي تحتويها وهو النوع الإفتراضي حيث ينشأ عند إنشاء النموذج Form ويجب تعريف واحد من هذا النوع على الأقل لكل نافذة Window .
2. **Stacked :** ويظهر فوق الـ Content Canvas ويمكن التحكم بحجمه وكذلك إظهاره وإخفاؤه تلقائياً .
3. **Vertical Toolbar :** وتظهر بشكل شريط أدوات عمودي .
4. **Horizontal Toolbar :** وتظهر بشكل شريط أدوات أفقي .
5. **Tab:** ويظهر بشكل صفحات ونحتاج له في بعض الأحيان لحصر العناصر في مجموعات .

أمثلة على ذلك :

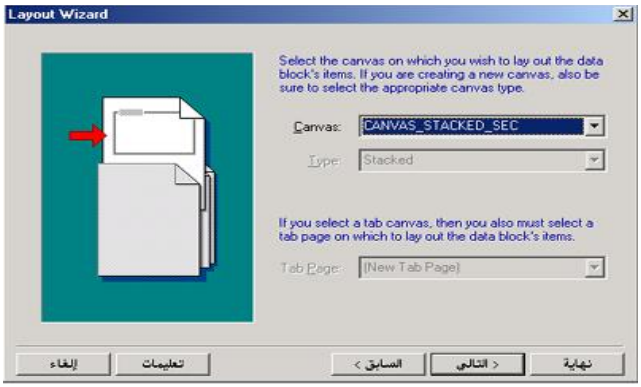
نقوم بإنشاء Form جديد يعتمد على Data Block هو الجدول **STU** لنلاحظ أنه سيقوم بإنشاء الـ Content Canvas (إفتراضياً) وستظهر العناصر عليه وإسم الـ Canvas (Canvas2) وليكن تسميته البرمجية **CANVAS_CONTENET** ،

بعد ذلك نقوم بتحديد الـ(Canvases) الموجود في شاشة الـ(Object Navigator) ثم Create ثم نضغط F4 للتحكم بقيم الخصائص التالية :



	Property	Value
1	Name	CANVAS_STACKED_SEC
2	Canvas Type	Stacked
3	Viewport X Position	25
4	Viewport Y Position	150
5	Viewport Width	375
6	Viewport Height	150

ثم نقوم الآن بجلب بيانات الأقسام Section إلى داخل هذا الـ Canvas وذلك بتحديد الـ(Data Block) الموجود في شاشة الـ(Object Navigator) ثم بزر الفأرة الأيمن نختار **Data Block Wizard** ونتبع نفس الخطوات المعهودة الـ[illegible]خطوة توضع العناصر فـ[illegible] **CANVAS_STACKED_SEC** ، لنلاحظ بعدها توضع العناصر [illegible]



وبنفس الخطوات السابقة ننشئ الـ Canvas الخاص ببيانات المستويات مع إختلاف :

a) التسمية البرمجية **CANVAS_STACKED_LVL** ،
b) جلب كتلة البيانات Lvl ،
c) توضع العناصر على هذا الـ Canvas الجديد .

نقوم الآن بإنشاء عنصرين من نوع **Push_Button** على الـContent Canvas مع مراعاة عدم وضعهما في أمكنة الـCanvases المراد إظهارهم ،

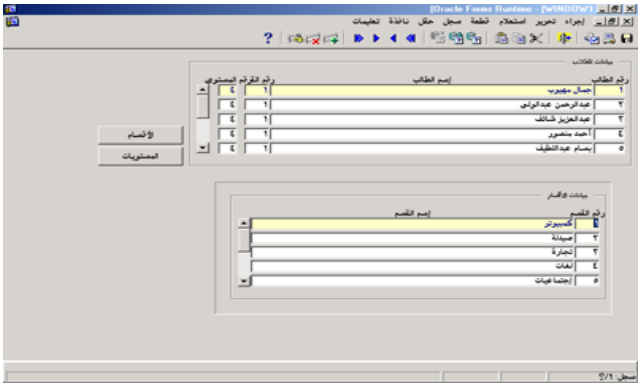
(**لأن الـ Canvas من نوع Satcked يتوضع على الـ Canvas من نوع Content كما ورد ذلك سابقاً**)
،

وذلك من أجل التحكم بإظهار أحد القماشيتين أو إخفاؤها ، حالياً نجعل خاصية Label للزر الأول (**الأقسام**) وللزر الثاني (**المستويات**) وقبل كتابة الشفرة الخاصة بكل منهما فإننا نقوم إحتياطاً بإخفاءهما عند بداية تحميل البرنامج لحين الطلب وكالتالي :

(Trigger : WHEN_NEW_FORM_INSTANCE)

```
Hide_View('CANVAS_STACKED_SEC') ;

Hide_View('CANVAS_STACKED_LVL') ;
```

ثم نكتب البرمجة الخاصة بكل زر منهما



ففي زر الأقسام نكتب :

(Trigger : WHEN_BUTTON_PRESSED)

```
Hide_View('CANVAS_STACKED_LVL') ;

Show_View('CANVAS_STACKED_SEC') ;

Go_Block('Section');

Execute_Query;
```

وفي زر المستويات نكتب :

(Trigger : WHEN_BUTTON_PRESSED)

```
Hide_View('CANVAS_STACKED_SEC') ;

Show_View('CANVAS_STACKED_LVL') ;
```

```
Go_Block('Lvl');

Execute_Query;
```

بالطبع بعد إظهار الـCanvas الذي نريد نذهب إلى Data Block التابع له ثن نجلب البيانات بالإستعلام .

الآن وبعد إنجاز هذه المرحلة نريد إنشاء عناصر تحكم (أزرار) على شريط الأدوات العمودي وكذلك شريط الأدوات الأفقي [**التي يفضل إنشاؤها في كتلة بيانات غير معتمدة على قاعدة البياناتNon_DB**] خاصة بالجدول **STU** وبالطبع لن يتم هذا أو ذاك إلا بتحديد الـ(Canvases) الموجود في شاشة الـ(Object Navigator) ثم Create مرتين مع مراعاة الخصائص التالية :

	Property	Value
1	Name	CANVAS_VERTICAL

	Property	Value
1	Name	CANVAS_HORIZONTAL

فمثلاً نجعل أزرار شريط الأدوات العمودي (**حفظ – إستعلام - خروج**) ونجعل أزرار شريط الأدوات الأفقي (**الأول – التالي – السابق – الأخير**) بعد تغيير Label الخاص بكل زر منهم بالإسم الظاهر عليه ونكتب الشفرة الخاصة في حدث **WHEN_BUTTON_PRESSED** بالأزرار كالتالي :

(PUSH_BUTTON_SAVE)

```
Commit_Form ;
```

(PUSH_BUTTON_QUERY)

```
Go_Block('Stu') ;

Execute_Query ;
```

(PUSH_BUTTON_FIRST)

```
Go_Block('Stu') ;

First_Record ;
```

(PUSH_BUTTON_NEXT)

```
Go_Block('Stu') ;

Next_Record ;
```

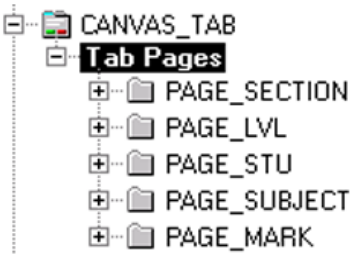
(PUSH_BUTTON_PREVIOUS)

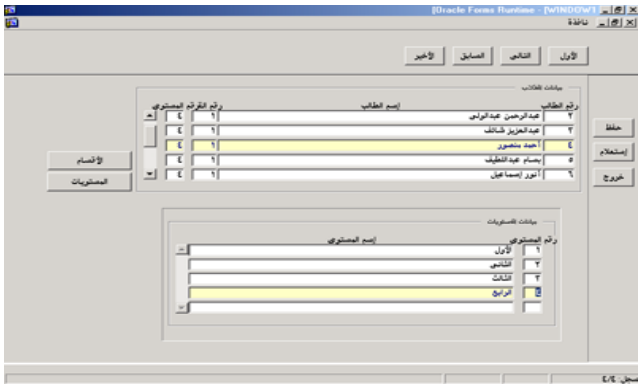
```
Go_Block('Stu') ;

Previous_Record ;
```

(PUSH_BUTTON_LAST)

```
Go_Block('Stu') ;

Last_Record ;
```

(PUSH_BUTTON_EXIT) `Exit_Form ;`		



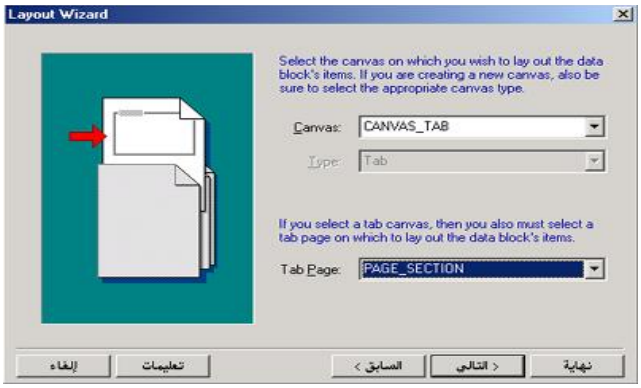


النوع الأخير من أنواع الـ Canvas هو النوع Tab وفي هذا النوع يتم أولاً تحديد الـ(Canvases) الموجود في شاشة الـ(Object Navigator) ثم Create ثم نضغط F4 للتحكم بقيم الخصائص التالية :

	Property	Value
1	Name	CANVAS_TAB
2	Canvas Type	Tab
3	Tab Attachment Edge	Top

لنلاحظ إنشاء ثلاثة صفحات إفتراضية ومن الممكن جعله يتكون من عدة صفحات (ضمن مجموعة) لذلك إذا أردنا إنشاء صفحة جديدة فنقوم بتحديد الـ(Canvases) الموجود في شاشة الـ(Object Navigator) ومنه نحدد الـ(**Tab Pages**) ثم Create ثم نضغط F4 للتحكم بقيم أهم الخصائص مثل [التسمية برمجية - التمكين وعدم التمكين - العنوان - الإظهار والإخفاء] .

	Property
1	Name
2	Enabled



ثم نقوم الآن بجلب بيانات الأقسام Section إلى داخل هذا الـ Canvas وذلك بتحديد الـ(Data Block) الموجود في شاشة الـ(Object Navigator) ثم بزر الفأرة الأيمن نختار **Data Block Wizard** ونتبع نفس الخطوات المعهودة إلا خطوة توضع العناصر فنختار الـ **CANVAS_TAB** ، وبما أنها عبارة عن صفحات فيجب أن نحدد Tab Pages بالقيمة **PAGE_SECTION** (وهو الإسم البرمجي لهذه الصفحة) لنلاحظ بعدها توضع العناصر على هذا الـ Canvas .

وبنفس الخطوات السابقة نجلب الـ Data Block الخاصة ببقية الجداول إلى الصفحة المنشئة مسبقاً على الـ Tab Canvas كلٌ بما يناسبها .

نلاحظ أن الـ Data Block قد يتم جلبها أكثر من مرة على إعتبار وجوب توضع عناصرها على Canvas جديدة لذلك قد يتغير إسم الـ Data Block مثلاً من **SECTION** إلى **SECTION1** وهذا لا يعني أن الـ Data Block المسماة SECTION1 ليست موجودة في قاعدة البيانات فهذه أسماء إختيارية ولكن المهم هو أن تكون الخصائص لها كالتالي :

	Property	Value
1	Database Data Block	Yes
2	**Query Data Source Name**	**SECTION**

وكذلك حقول هذه الـ Data Block يجب أن تكون الخصائص لها كالتالي :

	Property	Value
1	Database Item	Yes

	Property	Value
1	Database Item	Yes

وبمعرفتنا لهذه الخصائص يصبح من الممكن إنشاء Data Block تعتمد على قاعدة البيانات DB يدوياً

(وهذا ما لم نتحدث عنه في درس جلب بيانات الـ Data Block وأجلناه حتى ورود هذه الفقرة)

<u>ملاحظات هامة :</u>

- من الممكن التحكم بخصائص الـ Canvas برمجياً من خلال التعليمة التالية :
Set_Canvas_Property('إسم القماشية , 'إسم الخاصية , القيمة;)

- يمكن التحكم بخصائص الصفحات برمجياً خاصة إذا أردنا منع إظهار صفحة ما على مستخدم معين وذلك من خلال التعليمة التالية :
Set_Tab_Page_Property('إسم الصفحة , 'إسم الخاصية , القيمة;)

مثال :

```
Set_Tab_Page_Property( 'PAGE_MARK' , Visible , Property_False );
```

وفي هذه التعليمة إشارة إلى أنه لا يمكن أن يتكرر إسم صفحة ما حتى إذا كانت في Tab جديدة Canvas

- إستدعاء الـ Data Block على Canvas أخرى سيظهرها حتى إذا لم نكتب أمر الإظهار على الـCanvas .

- عند التنفيذ تتوضع أولاً Toolbar Canvas ثم تأتي بقية العناصر في الجزء المتبقي من النافذة لذلك يجب مراعاة **Width** في الـ Vertical Toolbar وكذلك الـ **Height** في الـ Horizontal Toolbar .

- إذا أنشئت أي عنصر Item في Canvas معينة وأردت نقلها إلى Canvas أخرى فما عليك سوى تغيير خاصية الـ Canvas لذلك العنصر بجعلها تشير إلى الـ Canvas التي تريد .

- من الممكن رسم أيقونة Icon على كل زر Push_Button بعد تعديل خصائصه <u>بحيث :</u>
  a. تعطى القيمة Yes لخاصية **Iconic**
  b. يعطى إسم لملف الأيقونة ذو الإمتداد ico.* للخاصية **Icon Filename** .

- خاصية **Tab Attachment Edge** تعمل على تحديد الموقع الذي ستظهر فيه التبويبات الخاصة بدخول الصفحات وتمتلك القيم (Top _ Bottom _ Left _ Right _ Start _ End) .

المحاضرة التاسعة

❖ Report Builder :

أداة تستخدم لبناء التقارير التي يستخدمها الـ(User) بكل عناصرها ، ومثل ما قمنا بكتابة البرامج والتحكم بصلاحيات المستخدمين وإنتقاء أفضل الواجهات والأساليب لكتابة البرامج وأداء العمليات المختلفة عليها فإن الثمرة التي يجب أن يجنيها البرنامج هي التقارير والتي لا بد من أن تكون ببرنامج الـ Report Builder ، وقد ذكرنا سابقاً أنه عند عملية :

- حفظ التقرير Save يتولد ملف مصدري Source ذو إمتداد *.RDF ويشغله برنامج Report Builder
- الترجمة Compile يتولد ملف تنفيذي Execute ذو إمتداد *.REP ويشغله برنامج Reports Runtime .

==========

❖ تشغيل Report Builder :

Start → Programs → Oracle Reports 6i-orantr → Report Builder

فتظهر أولاً شاشة ترحيبية يمكن من خلالها تحديد ما إذا كان المطلوب البدء بالتصميم أو بالتعليم ، فالتصميم ثلاث خيارات وهي :

- إستخدام معالج التقارير ،
- بناء تقرير جديد يدوياً ،
- فتح تقرير موجود مسبقاً ،

وللتعليم خياران وهما :

- عرض الجولة السريعة (مفاهيم) ،
- إستكشاف كروت التلميحات (مهام) .

(حالياً سنختار البدء بالتصميم بإستخدام معالج التقارير ثم نضغط OK)

لتظهر بعد ذلك شاشة ترحيبية خاصة بمعالج التقارير **(فنضغط التالي)**

ثم تظهر شاشة نعطي فيها العنوان العام الذي سيظهر على التقرير وكذلك نقوم بتحديد نوع التقرير هل هو :

- جدولي ،
- يشبه النموذج ،
- عنوان بريدي ،
- خطاب نموذج ،
- تجميع لليسار ،
- تجميع للأعلى ،
- مصفوفة ،
- [illegible]

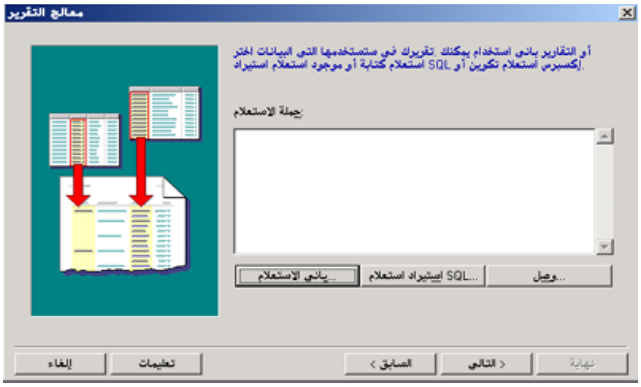
(حالياً سنختار سنكتب العنوان تقرير بيانات الطلاب ونختار نوع التقرير جدولي ثم نضغط التالي)

ثم تظهر شاشة تحديد نوع الإستعلام هل هو من :

- جملة SQL ،
- إستعلام Express (عندما يكون على شبكة Server/Client) .

(حالياً سنختار نوع الإستعلام من جملة SQL ثم نضغط التالي)

لتظهر بعدها شاشة نكتب من خلالها جملة الإستعلام والتي بناءاً عليها ستظهر البيانات في التقرير حيث يمكنك تحريرها مباشرة في SQL Query Statement أو بناءها من جديد من خلال برنامج باني الإستعلام ...Build SQL Query أو إستيرادها من ملف ...Import SQL Query وهناك أيضاً زر الإتصال إن لم تكن متصلاً بالقاعدة ...Connect

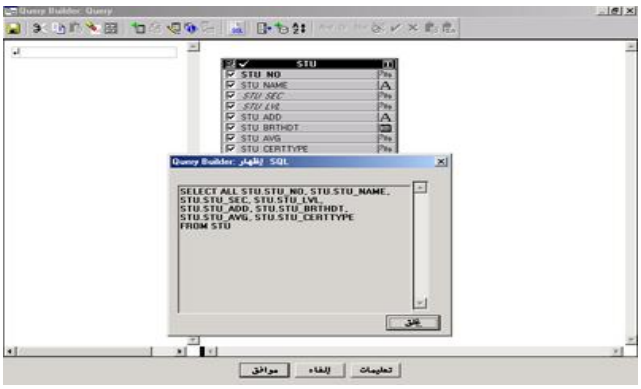


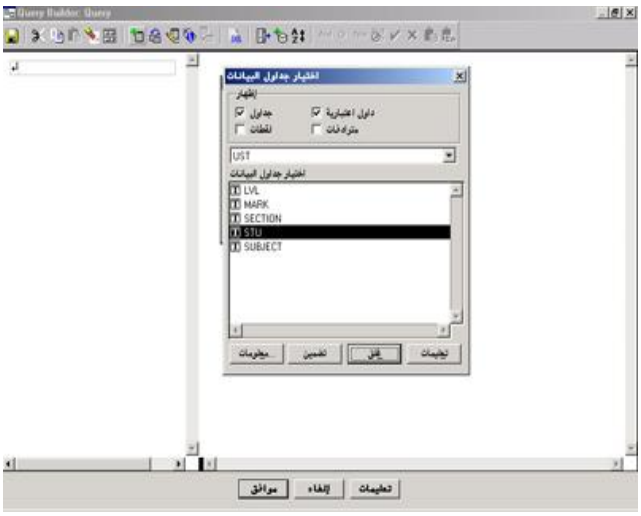
(حالياً سنقوم ببناء جملة الإستعلام من خلال الضغط على البرنامج المصغر باني الإستعلام...)

حيث لابد من الإرتباط أولاً بقاعدة البيانات وكتابة إسم الـ(User) والـ(Password) بشكل صحيح **ومن ثم الضغط على الزر وصل** ولتظهر الشاشة الخاصة ببرنامج باني الإستعلام ومنها شاشة إختيار جداول البيانات المراد التعامل معها **(حالياً سنختار الجدول STU)** ونقوم بالتأشير أمام كل حقل نريد جلبه أما إذا أردنا جلب جميع الحقول فنؤشر على الخيار المتعدد أعلى الجدول ، لاحظ على شريط الأفقي لبرنامج باني الإستعلام وجود الأيقونات :

- **إختيار جدول البيانات :** عندما نريد إظهار هذه الشاشة مرة أخرى لإضافة جدول آخر ،

- **إظهار SQL :** لنشاهد من خلالها تكون أمر الإستعلام .

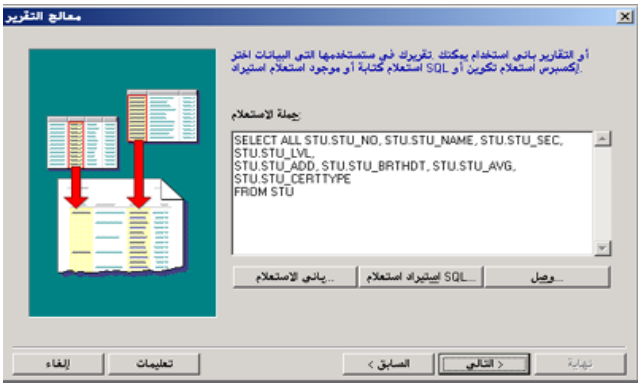




(نضغط موافق ولنرى تكون جملة الإستعلام مباشرة في SQL Query Statement فنضغط التالي)

وبالتالي تظهر الحقول في الـ**(List)** فنختار منها الحقول المطلوب التعامل معها بواسطة الأزرار > ، >> ، < ، <<

(حالياً سنختار >> ثم نضغط التالي)



لتظهر شاشة **حساب إجماليات الحقول** التي ترغب بإيجادها مثل (المجموع– المتوسط – العد – الحد الأدنى – الحد الأقصى – الإجمالي) وذلك من خلال تحديد الحقل ثم ضغط الزر الذي تريد تطبيق الحساب المراد إجرائه عليه ،

(حالياً نضغط التالي)

ثم تظهر شاشة تمكننا من تغيير العناوين الظاهرة لأسماء الحقولPrompt وكذلك عرضها W وطولها H

(حالياً نضغط التالي)

لتظهر شاشة تحديد القالب Template الذي
تريد عرض التقرير به فيمكنك أن تختار :

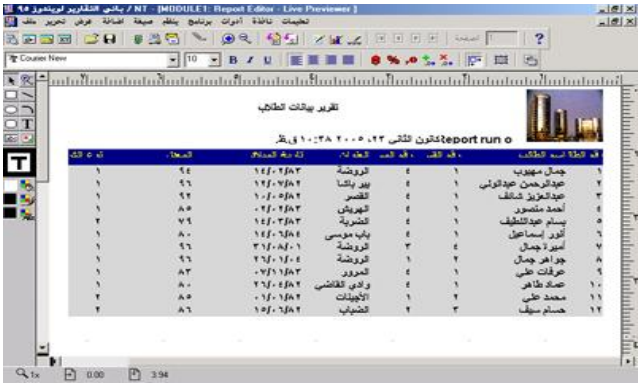
- قالب محدد مسبقاً ،
- تحديد قالب من ملف مخزن على جهاز الكمبيوتر الخاص بك ،

ظهور التقرير بدون أي قالب . **(حالياً نجعله على القالب الإفتراضي ثم ضغط التالي)**

بعدها تظهر شاشة التهاني والتبريكات congratulations لتعلن عن إتمام معالج التصميم (المخطط) بنجاح ،
(فنضغط نهاية) وبذلك نكون قد قمنا بتشغيل Report Builder .

ملاحظات هامة :

- يتم تشغيل التقرير من خلال جلب البيانات من الـ Database سواءً أكانت قاعدة البيانات موجودة على الـ Server أو على الـ Client **(لاحظ هنا تنشيط الـ Client Activity والذي يعني جلب البيانات من قاعدة البيانات الموجودة على الـ Client) ،**
- في حالة عدم ظهور الخط بالعربي قم بتغيير الخط وذلك بالضغط على **Ctrl+A** لتحديد الكل ثم من شريط الأدوات الأفقي قم بإختيار نوع الخط وليكن Arial(Arabic) وحجم خط وليكن 12 ،
- بالإمكان تغيير الصورة الظاهرة في هذا القالب بصورة أخرى (مثلاً شعار الجامعة) وذلك بإختيارنا الأمر **إستيراد** ومنه **صورة** من القائمة **ملف** .



a_alyahawi@hotmail.com

❖ الشاشات الرئيسية للـ(Report Builder) :

1. Object Navigator :
ويحتوي عناصر التقرير كاملة Report وكذلك الكائنات Objects ، ومن الممكن الحصول على هذه الشاشة من خلال الضغط على الزر **F3** أو من القائمة Tools نختار الأمر Object Navigator .

2. Data Model :
نموذج البيانات وسنأتي إلى شرحه بالتفصيل ، بإمكانك إعتباره شبيه الـ Data Block Wizard في برنامج Form Builder .

3. Layout Model :
نموذج النسق وسنأتي إلى شرحه بالتفصيل ، بإمكانك إعتباره شبيه الـ Layout Wizard في برنامج Form Builder .

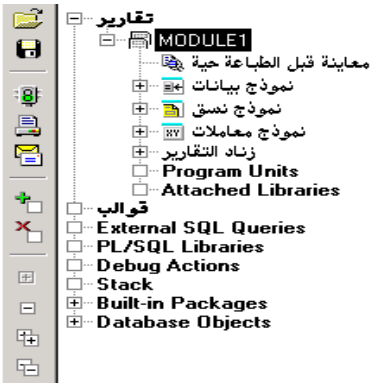
4. Property Palette :
وتحتوي خصائص العناصر والكائنات Objects ، ومن الممكن الحصول على هذه الشاشة من خلال الضغط على الزر **F4** أو من القائمة Tools نختار الأمر Property Palette .

✍ ملاحظة هامة :

نموذج البيانات Data Model ونموذج النسق Layout Model هما مكوني ما يسمى بالـ **Report Editor** .

==========

❖ نبذة مُبسطة عن محتويات Object Navigator :



Reports :

1. **[حالياً MODULE1] :** إسم الـ (Report) الإفتراضي ويأخذ إسم البرنامج دوماً ،
2. **Live Preview :** وهي لمعاينة التقرير قبل الطباعة ،
3. **Data Model :** نموذج البيانات وسنأتي إلى شرحه بالتفصيل ،
4. **Layout Model :** نموذج النسق وسنأتي إلى شرحه بالتفصيل ،
5. **Parameter Form :** نموذج المعاملات وهو عبارة عن Form يظهر قبل التقرير بمعاملات منشئة من المستخدم أو أنها منشئة من النظام .
6. **Reports Trigger :** وهي عبارة عن (SQL , PL/SQL) ينفذ عند حدث معين قد يكون هذه الحدث :
   - قبل نموذج المعاملات Before Parameter Form ،
   - بعد نموذج المعاملات After Parameter Form ،
   - قبل التقرير Before Report ،
   - [illegible]
7. [illegible]

8. **Attached Libraries :** وهي عبارة عن مكتبات PL/SQL ،
Templates : عندما تنشئ تقريراً بشكل قالب ،

External SQL Queries : إستعلامات SQL الخارجية ،

PL/SQL Libraries : وهي عبارة عن مكتبات PL/SQL ،

Debug Actions : للتنقيح ،

Stack : إستخدام المكدس ،

Built-in Packages : وهي عبارة عن الحزم المدمجة مع الـ(Oracle) ،

Database Objects : حيث تقوم بعرض جميع مستخدمي قاعدة البيانات .

==========

ملاحظات هامة :

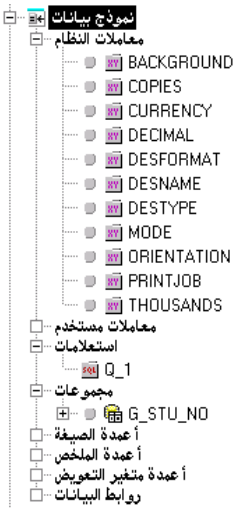
- أثناء عملية الحفظ **File → Save** تظهر شاشة خيارات الحفظ والتي من خلالها يمكنك تحديد إذا ما كنت تريد الحفظ في قاعدة البيانات أو في ملف (الإفتراضي) وكذلك هل تريد الحفظ لعرض التقارير (الإفتراضي) أم القوالب أم الإستعلامات أم المكتبات التابعة للـ PL/SQL أم حفظ الكل ،
(حالياً نضغط موافق ليتم حفظ التقرير على ملف نحدد مساره)

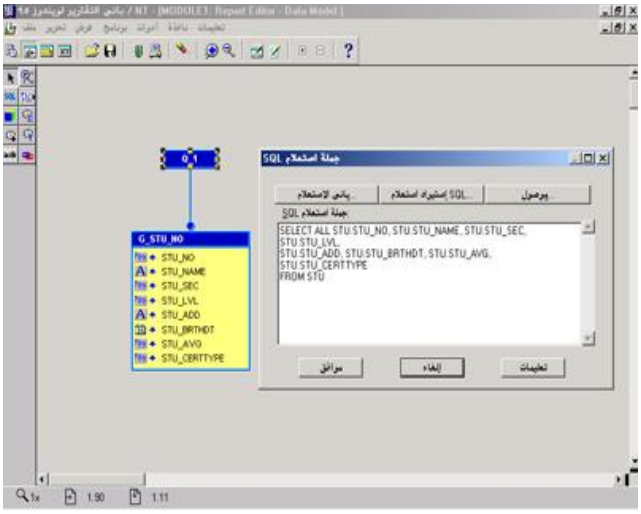
- وبنفس المنوال تظهر شاشة شبيهة أثناء عملية الفتح **Open** لتقرير ما .

a_alyahawi@hotmail.com

❖ نـــمـــوذج الـــبـــيـــانـــات Data Model :

حيث نقوم بتحديد الـ(Data Model) الموجود في شاشة الـ(Object Navigator) ونضغط F2 لتظهر نافذة فيها الإستعلامات الموجودة للتقرير (حالياً **Q_1**) وإذا نقرنا عليها نقراً مزدوجاً يظهر الإستعلام والمرتبط بمجموعة من الحقول متواجدة على شكل مجموعة **Group** .





ولنلاحظ أن محتويات نموذج البيانات Data Model كالتالي :

a. معاملات النظام System Parameters :

حيث يتم التعامل معها عن طريق الخصائص الخاصة لكل معامل نظام (نحدد معامل النظام ومن ثم نضغط على F4) وخاصة خاصية القيمة الإبتدائية **Initial Value** ومعاملات النظام هي :

1. الخلفية Background :

(إفتراضياً قيمة خاصية القيمة الإبتدائية No) وعندما تكون Yes فإننا نجعل التقرير يطبع في الخلفية بينما نحن نواصل العمل في برامج أخرى ،

2. عدد النسخ Copies :

وتعني تحديد عدد نسخ التقرير المطبوع (إفتراضياً قيمة خاصية القيمة الإبتدائية 1) ،

3. العملة Currency :

وتعني ما هو الرمز الذي تريد إضافته بجانب الأرقام أثناء طباعة التقرير (مثلاً نكتب **Y.R** أي ريال يمني) ،

4. الفواصل العشرية Decimal :

وتعني تحديد الرمز المستخدم مع الفواصل العشرية (مثلاً نكتب الرمز **,** أو الرمز ، أو الرمز **.**) ،

5. تنسيق الهدف DesFormat :

وتعني تحديد [illegible] أو ملف Pdf أو ملف rtf

6. إسم الهدف DesName :
وتعني تحديد إسم الملف الذي سيرسل إليه التقرير (مفيدة جداً إذا لم تكن هناك طابعة مرتبطة بجهاز الكمبيوتر) ،

7. نوع الهدف DesType :
وتعني تحديد وجهة الطباعة هل هي إلى الشاشة Screen أم إلى الطابعة Printer ...إلخ (بالطبع الطباعة إلى الشاشة أسرع من الطباعة على الطابعة بسبب أن ظهور الصفحة الأولى من التقرير على الشاشة لايستغرق وقتاً طويلاً بينما ظهور الصفحة الأولى في ورق الطابعة يتطلب أولاً تنسيق جميع الصفحات قبل البدء بعملية الطباعة) ،

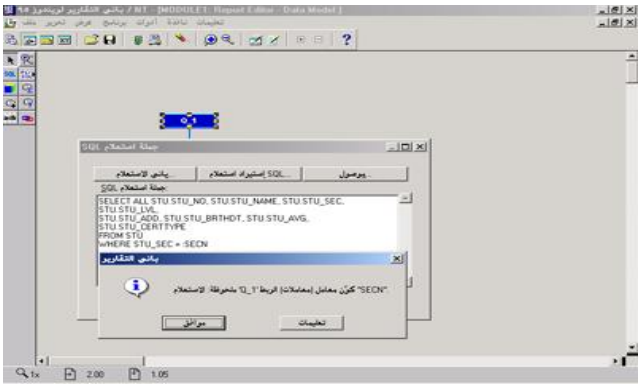
8. النمط Mode :
وتعني تحديد ما إذا كنت تريد أن يطبع تقريرك بشكل صورة Bitmap أم أحرف Character ،

9. الإتجاه Orientation :
وتعني تحديد إتجاه طباعة التقرير هل هي أفقية Landscape أم عمودية Portrait ،

10. وظيفة الطابعة PrintJob :
وتعني تحديد هل نريد إظهار شاشة خصائص الطابعة قبل التقرير أم لا (إفتراضياً قيمة خاصية القيمة الإبتدائية Yes) ،

11. الآلاف Thousands :
وتعني تحديد تنسيق الآلاف (مثلاً 99.999) .

b. معاملات المستخدم User Parameters :
وطريقة إنشاء معامل مستخدم يعتمد إعتماداً كلياً على جملة الإستعلام **SQL Query Statement** ، والتي نصل إليها بتحديد الـ(Data Model) الموجود في شاشة الـ(Object Navigator) ثم :



- نضغط F2 لتظهر نافذة فيها الإستعلامات الموجودة للتقرير (حالياً **Q_1**) فننقر عليها نقراً مزدوجاً ليظهر الإستعلام ،
- أو نحدد الإستعلامات Queries ثم نضغط F2 وبنفس الطريقة .

لنقم بإضافة معامل مستخدم وذلك بتعديل جملة الإستعلام (حالياً **Q_1**) ونكتب فيها

```
Where STU_SEC = :SECN
```

وليصبح لدينا الآن معامل مستخدم جديد هو **SECN** ، وذلك بعد ظهور رسالة التأكيد التي تخبرنا بأنه قد تم إنشاء معامل [illegible]

نقوم الآن بتحديد معامل المستخدم الجديد **SECN** ونمنحه الخصائص التالية :

	Property	Value
1	Data Type	Number

حيث نعطي نوع البيانات Data Type لمعامل المستخدم **SECN** القيمة Number ونهيئه بقيمة إبتدائية Initial Value هي 4 وكذلك نحدد قيمة خاصية القائمة المركبة **List of Value** من خلال الشاشة التي نحدد بواسطتها أن القائمة المركبة تجلب قيمها من قاعدة البيانات أي ديناميكية (**جملة Select**) ومن ثم نقوم بجلب جملة الإستعلام عن طريق زر برنامج باني الإستعلام وبالطريقة الموضحة لدينا مسبقاً أو كتابتها مباشرة في محرر جمل الإستعلام ; `Select * From Section` ،

والآن لاحظ عند تنفيذ التقرير كيف تظهر شاشة إعدادات الطباعة وهي محتوية على معامل المستخدم الخاص بالأقسام Section والمسمى **SECN** ، ولنختار القيمة التي نريد ثم نضغط **Enter** ، ولاحظ نموذج المعاملات Parameter Form عندما نقوم بتحديده ثم نضغط F2 لنشاهد الشاشة التي تظهر معاملات التقرير كاملة Form .

وبنفس الطريقة إذا أردنا إضافة معامل مستخدم خاص **بالمستوى** تكون الخطوات كالتالي :

1. إضافة المعامل عن طريق تعديل جملة الإستعلام فنكتب فيها `Where LVL_SEC = :LVLN` وليصبح لدينا الآن معامل مستخدم جديد هو **LVLN** ،

2. تعديل خصائص المعامل (نوع البيانات – القيمة الإبتدائية – القائمة المركبة والتي نجعلها إستاتيكية [قيم ثابتة] عن طريق كتابة القيمة ثم الضغط على الزر إضافة وهكذا بالنسبة لبقية القيم) .

<u>ومن أجل ظهور معاملات النظام System Parameters في شاشة إعداد التقرير نتبع الخطوات التالية :</u>

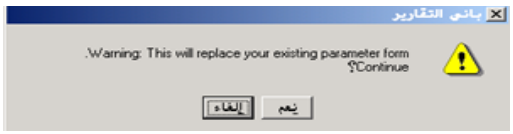
من قائمة **أدوات** نختار الأمر **باني نموذج معامل ...**

Tools → Parameter Form Builder... لنشاهد وجود جميع معاملات النظام **System Parameters** ومعاملات المستخدم المنشئة **User Parameters** ، فنقوم بتحديد المعامل الذي نريد ظهوره (**يصبح مظللاً أسود اللون**) ونكتب أمام كل معامل العنوان الذي نرغب في إظهاره فمثلاً بدلاً من ظهور معامل المستخدم (القسم) بالعنوان SECN نجعله رقم القسم ، ولتظهر رسالة تأكيد إختيار المعاملات ولنشاهد في نموذج المعاملات **Parameter Form** كيف تنضاف إليه بقية المعاملات التي قمنا بإنتقائها .

==========

تلميح :

إذا أردنا أن ننشئ تقرير يعمل على إيجاد بيانات من رقم إلى رقم معين أو من تاريخ إلى تاريخ معين فإنه يجب علينا أن نقوم بتعريف معاملي مستخدم بواسطة تعديل جملة الإستعلام ومن ثم تعديل خصائص المعاملين (نوع البيانات – القيمة الإبتدائية – القائمة المركبة) .



مثال ذلك :

بالإضافة إلى تحديد المستخدم للقسم أثناء إيجاد تقرير بيانات الطلاب فإننا نريد أيضاً جعل المستخدم يحدد المستويات بحيث تكون عملية التحديد من مستوى إلى مستوى آخر ؟

ولعمل ذلك فإننا سنقوم بما يلي :

a. إضافة معاملي مستخدم عن طريق تعديل جملة الإستعلام **SQL Query Statement**
وذلك بكتابة `And STU_Lvl Between :No1 and :No2`

وليصبح لدينا معاملي مستخدم جديدين هما **No1 و No2** ، وذلك بعد ظهور رسالة التأكيد التي تخبرنا بأنه قد تم إنشاء المعاملين من قِبل المستخدم ؟!!..

b. نعطي المعامل الأول والمعامل الثاني الخصائص التالية :

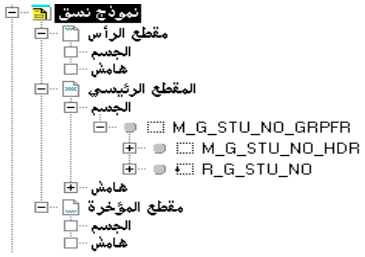
	Property	Value
1	Data Type	Number

	Property	Value
1	Data Type	Number

ولنظهر عبارة (**من مستوى رقم**) أمام المعامل **No1** وعبارة (**إلى مستوى رقم**) أما المعامل **No2** وذلك من خلال قائمة **أدوات** نختار الأمر **باني نموذج معامل ...** **Tools → Parameter Form Builder...**

لنشاهد وجود جميع معاملات النظام **System Parameters** ومعاملات المستخدم المنشئة **User Parameters** ، فنكتب أمام المعامل **No1** العنوان (**من مستوى رقم**) وأمام المعامل **No2** العنوان (**إلى مستوى رقم**) .

حيث نقوم بتحديد الـ(Layout Model) الموجود في شاشة الـ(Object Navigator) ونضغط F2 لتظهر الشاشة المحتوية على هيكلية العرض (**الهيكل الرئيسي للتقرير**) أي المحتوي على الحقول Fields والعناوين Label ، ويمكن أن نتتبع مكونات نموذج النسق Layout Model من خلال شاشة الـ(Object Navigator) أو شريط الأدوات الأفقي الخاص بشاشة الـ(Layout Model) حيث
توجد الأربع الأيقونات التالية :



1. **مقطع الرأس Header :**
تظهر أول صفحة فقط وتتكون من جسم وهامش ،

2. **المقطع الرئيسي Body :**
تظهر صفحات التقرير وتتكون من جسم وهامش ، حيث يتكون مقطع الجسم من مجموعات **Groups** كل مجموعة لها بيانات غير تكرارية Frame كالعناوين وبيانات تكرارية Repeating Frame كالحقول هذه المجموعات عددها مرتبط بنوع التقرير فمثلاً (**الجدولي يمتلك مجموعة واحدة فقط Group**) في الدرس القادم بإذن الله سنتحدث عن هذه النقطة الهامة بشكل تفصيلي أكثر .

3. **مقطع المؤخرة Footer :**
تظهر آخر صفحة فقط وتتكون من جسم وهامش ،

4. **تحرير الهامش Margin :**
تظهر هوامش التقرير .

==========

<u>ملاحظات هامة :</u>

- إذا أردت إظهار العناوين على كل صفحة في التقرير (رقم الطالب – إسم الطالب - ... إلخ) فيجب إعطاء القيمة **All Pages** للخاصية **Print Job On** وذلك في خصائص البيانات الغير متكررة .

- <u>تمتلك الخاصية **Print Job On** القيم التالية :</u>
1. **All Pages :** يسمح بالظهور في جميع الصفحات ،
2. **All But First Page :** يسمح بالظهور في جميع الصفحات عدا الصفحة الأولى ،
3. **All But Last Page :** يسمح بالظهور في جميع الصفحات عدا الصفحة الأخيرة ،
4. **Default :** يأخذ القيمة المحددة في محرر التسجيل Registry ،
5. **First Page :** يسمح بالظهور في الصفحة الأولى فقط ،
6. **Last Page :** يسمح بالظهور في الصفحة الأخيرة فقط .

- <u>بإمكانك كتابة ما تريد من عبارات توضيحية أو ترحيبية من خلال مقطع الرأس مثلاً وكالتالي :</u>
نحدد (**نموذج النسق**) الموجود في شاشة الـ(Object Navigator) ومنه **مقطع الرأس** ومنه **الجسم** ثم نضغط على F2 ليظهر النموذج الخاص بالجسم Form فنقوم بإختيار العنصر **Display Item** من شريط الأدوات ولنكتب (جامعة العلوم والتكنولوجيا فرع تعز) ونغير ما يلزم من إعدادات مثل نوع الخط وحجمه ولون الأمامية ولون الخلفية ... إلخ ، ولاحظ بعدها ظهور هذه الصفحة عند بداية تنفيذ التقرير لأنها في جزء الجسم من مقطع الرأس .

a_alyahawi@hotmail.com

المحاضرة العاشرة

❖ المجاميع Groups :

قبل الخوض بمفاهيم المجاميع يجب أن نعرف ماذا تعني الإطارات في التقاريروهي كالتالي :

1. الإطار الغير تكراري Frame :

حيث تظهر فيه البيانات الغير تكرارية كالعناوين والتي يتم جلب بياناتها مرة واحدة أثناء عملية تشـ التقرير

،

ويرمز له برمز الإطار المنقط ،

2. الإطار التكراري Repeating Frame :

حيث تظهر فيه البيانات التكرارية كالحقول والتي يتم جلب بياناتها أكثر من مرة أثناء عملية تش التقرير ،

ويرمز له برمز الإطار المنقط المحتوي على سهم وذلك للدلالة على التكرار ، (عدد التكرار يكون بعدد السجلات) .

مثال مبسط :

١	جمال مهيوب	١	٤	الروضة	١٤/٠٢/٨٣	٩٤	١
٢	عبدالرحمن عبدالولي	١	٤	بير باشا	١٢/٠٧/٨٢	٩٦	١
٣	عبدالعزيز شائف	١	٤	القصر	١٠/٠٥/٨٢	٩٢	١
٤	أحمد منصور	١	٤	الهريش	٠٢/٠٢/٨٣	٨٥	١
٥	بسام عبداللطيف	١	٤	الضربة	١٤/٠٣/٨٣	٧٩	٢
٦	أنور إسماعيل	١	٤	باب موسى	١٤/٠٦/٨٤	٨٠	١
٧	أميرة جمال	٤	٣	الروضة	٣١/٠٨/٠١	٩٦	١
٨	جواهر جمال	٢	١	الروضة	٢٦/٠١/٠٤	٩٦	١
٩	عرفات علي	١	٤	المرور	٠٧/١١/٨٣	٨٣	١
١٠	عماد طاهر	١	٤	وادي القاضي	٢٦/٠٤/٨٢	٨٠	١
١١	محمد علي	٢	١	الأجيلات	٠١/٠١/٨٢	٨٥	٢
١٢	حسام سيف	٣	٢	الضباب	١٥/٠٦/٨٢	٨٦	٢

إذن صفحات التقرير تتكون من رأس وجسم ومؤخرة ولكل منها جسم وهامش ، وفي جسم التقرير يتكون مقطع الجسم من مجموعات **Groups** كل مجموعة لها بيانات غير تكرارية Frame كالعناوين وبيانات تكرارية Repeating Frame كالحقول ، ويجب أن نفهم القاعدة التالية أثناء تصميم تقرير ما :

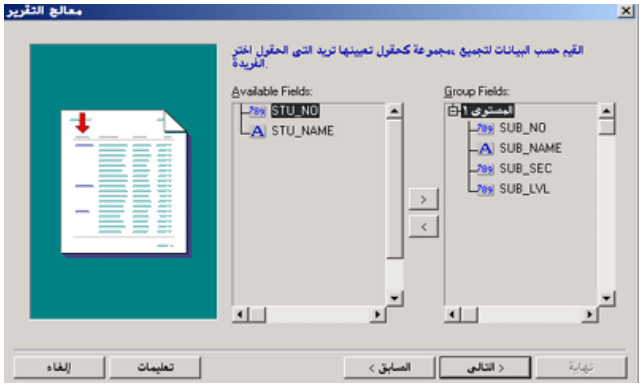
كل مجموعة Group لها إطار تكراري Repeating Frame خاص بها أي أن

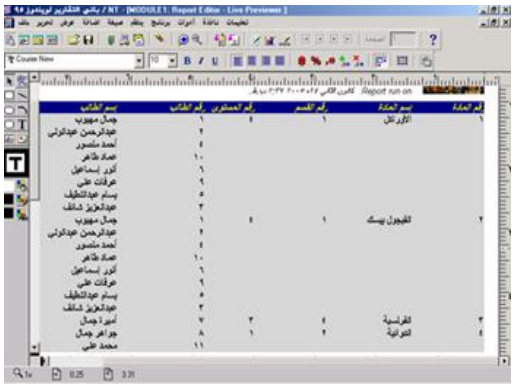
مثال ذلك :

لننشئ تقرير يوضح العلاقة **Relation** بين المادة والطلاب الدارسين لها بحيث نجلب الحقول (رقم المادة – إسم المادة – رقم القسم – إسم القسم) من جدول **Subject** والحقول (رقم الطالب – إسم الطالب) من جدول **Stu** ، وبواسطة معالج التقارير نجعل نوع التقرير **(تجميع لليسار)** ولتكن جملة الإستعلام كالتالي :

```
Select Sub_No , Sub_Name , Sub_Sec , Sub_Lvl , Stu_No , Stu_Name From
        Subject , Stu Where Sub_Sec = Stu_Sec And Sub_Lvl = Stu_Lvl ;
```

ولنلاحظ بعدها ظهور شاشة جديدة تطالب بتحديد المجموعات التي نريد إظهار البيانات فيها وذلك بسبب إختيارنا لنوع تقرير غير **النوع الجدولي والذي يتكون من مجموعة واحدة فقط** ، أما حالياً وبإختيارنا لنوع التجميع لليسار فسنقوم بنقل الحقول (رقم المادة – إسم المادة – رقم القسم – إسم القسم) كمجموعة أولى أما بقية الحقول فستندرج مباشرة تحت مظلة المجموعة الأخيرة وهي هنا الثانية ، ثم نكمل بقية متطلبات معالج التقرير ولنشاهد بعدها طريقة عرض هذا التقرير .

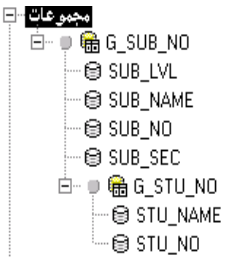




تلميح :

إذا أردت إنشاء مجموعة أخرى [illegible] أضف بقية الحقول لتراها [illegible] حقولاً تتبع مباشرة [illegible]

وللتأكد من عدد المجموعات التي تكونت نحدد (**نموذج البيانات**) الموجود في شاشة الـ(Object Navigator) ومنه **مجموعات** ولنشاهد أن المجموعات التي تكونت لدينا حالياً هي مجموعتين :



المجموعة الأولى : (رقم المادة – إسم المادة – رقم القسم – إسم القسم) ،

المجموعة الثانية : (رقم الطالب – إسم الطالب) .

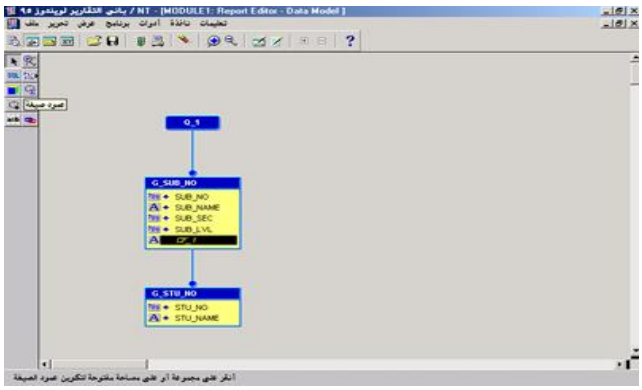
ولكننا نلاحظ أيضاً أن :

المجموعة الأولى **G_SUB_NO** هي أب للمجموعة الثانية **G_STU_NO** ، وبالتالي يجب علينا تحديد المصدر الرئيسي (المجموعة المباشرة) لكل إطار تكراري من خلال الخاصية **Source** وذلك لتوضع حقول البيانات عليه لأن ذلك مفيد جداً خاصة عند عمليات إضافة حقول جديدة على التقرير يدوياً Manually ، لأن الإطار التكراري بالنسبة لحقول البيانات المتوضعة عليه بمثابة **القماشية Canvas** فلا يمكن أن ينفذ التقرير عندما تتوضع الحقول خارجه .

==========

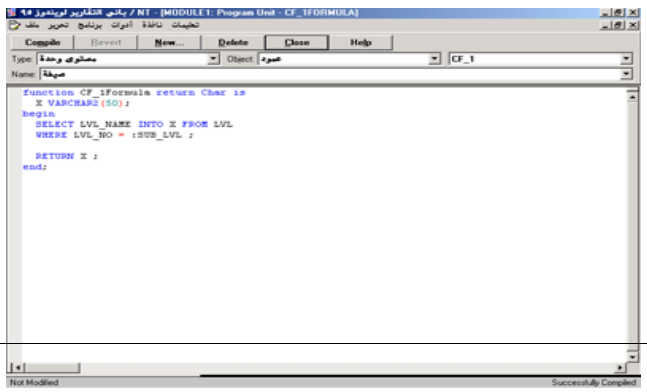
❖ الإضافة بإستخدام حقل صيغة Formula :

إذا أردنا في التقرير السابق أن نظهر **إسم المستوى** بجوار رقم المستوى سنلاحظ أنه غير متوفر في جدول **Subject** ولا في جدول **Stu** وأنه موجود في جدول آخر لم نقم بجلبه وهو جدول **Lvl** لذلك سنضيف إسم المستوى بإستخدام حقل صيغة Formula وكالتالي :



أولاً) نقوم بتحديد (**نموذج البيانات**) الموجود في شاشة الـ(Object Navigator) ونضغط F2 لتظهر نافذة فيها الإستعلامات الموجودة للتقرير (حالياً **Q_1**) ومن شريط الأدوات العمودي نضيف **عمود صيغة** إلى داخل المجموعة الأولى **G_SUB_NO** ولنرى أنه قد تكون شريط تمرير عند إضافته فنقوم بتوسعة عرض عناصر المجموعة (نلاحظ أن إسم عمود الصيغة الإفتراضي هو **CF_1**) ،

ثانياً) نقوم بتحديد عنصر عمود الصيغة CF_1 ونضغط F4 لنعدل خصائصه كالتالي :

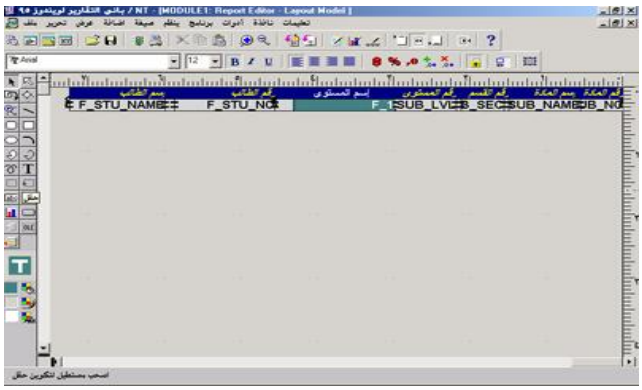


1	Data Type

ليكن نوع البيانات Data Type من النوع المحرفي Char ولنكتب في خاصية PL/SQL Formula التالي :

```
FUNCTION CF_1FORMULA RETURN CHAR IS
  X Varchar2(50);
BEGIN
   Select Lvl_Name Into X From Lvl
   Where Lvl_No = :Sub_Lvl ;

   Return X ;
END;
```

ثالثاً) ننشئ من شريط الأدوات العمودي التابع لـ(**نموذج النسق Layout Model**) ما يلي :



- **عنصر حقل Field :**

لنعرض البيانات داخله ثم نجعله في الإطار التكراري الأب **G_SUB_NO** بعد فتح قفل طور الإحتواء من شريط الأدوات الأفقي للسماح بإضافة الحقل ، ولا ننسى ربطه مع عمود صيغة من خلال الخاصية **Source** وذلك بعد تحديده وضغط F4 ،

Property	Value
Source	CF_1

- **عنصر نص Display :**

ليوضح عنوان عمود الصيغة وليكن (إسم المستوى) .

❖ الإضـــافـــة بـــإســـتـــخـــدام (الإســـتـــعلام SQL Query) :

إذا أردنا في التقرير السابق أن نظهر **إسم القسم** بجوار رقم القسم سنلاحظ أنه غير متوفر في جدول **Subject** ولا في جدول **Stu** وأنه موجود في جدول آخر لم نقم بجلبه وهو جدول **Section** لذلك سنضيف إسم القسم بإستخدام الإستعلام SQL Query وكالتالي :

أولاً) نقوم بتحديد (**نموذج البيانات**) الموجود في شاشة الـ(Object Navigator) ونضغط F2 لتظهر نافذة فيها الإستعلامات الموجودة للتقرير (حالياً **Q_1**) ومن شريط الأدوات العمودي نضيف **إستعلام SQL** لينشئ إستعلام جديد هو (**Q_2**) ولتظهر قبل ذلك شاشة والتي يمكنك من خلالها تحرير جملة الإستعلام مباشرة في SQL Query Statement أو بناءها من جديد من خلال برنامج باني الإستعلام ...Build SQL Query أو إستيرادها من ملف ...Import SQL Query وهناك أيضاً زر الإتصال إن لم تكن متصلاً بالقاعدة ...Connect فنكتب :

```
Select * From Section ;
```

ثانياً) ننشئ من شريط الأدوات العمودي التابع لـ(**نموذج النسق Layout Model**) ما يلي :

- **عنصر حقل Field :**

لنعرض البيانات داخله ثم نجعله في إطاره التكراري ولكن أين هو إطاره التكراري فلدينا الآن ثلاث مجموعات وهي :

المجموعة الأولى **G_SUB_NO** هي أب للمجموعة الثانية **G_STU_NO** (من **Q_1**) ،

والمجموعة الثالثة **G_SEC_NO** (من **Q_2**) ، وبحسب القاعدة التي تقول أن :

عدد المجموعات = عدد الإطارات التكرارية

فيجب علينا من أن ننشئ قبل كل شيء ما يسمى بـ**(الإطار التكراري Repeating Frame)** وذلك من شريط الأدوات العمودي التابع لـ(**نموذج النسق Layout Model**) ، ثم نوضع عنصر حقل Filed داخله ولينتمي إليه ، ولا ننسى ربطه مع إسم القسم من خلال الخاصية **Source** وذلك بعد تحديده وضغط F4 ،

Property	Value
Source	SEC_NAME

- **عنصر نص Display :**

ليوضح عنوان عمود الصيغة وليكن (إسم القسم) .

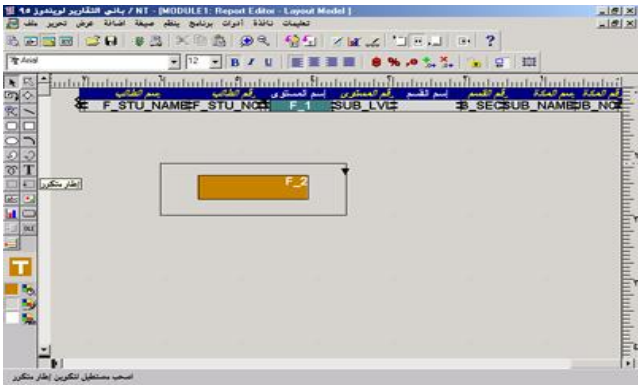
ثالثاً) نقوم بتحديد (**نموذج البيانات**) الموجود في شاشة الـ(Object Navigator) ونضغط F2 لتظهر نافذة فيها الإستعلامات الموجودة للتقرير (حالياً **Q_1 & Q_2**) ومن شريط الأدوات العمودي نضيف **ربط البيانات Data Link** بين رقم القسم من المجموعة الثالثة (المنتمية لإستعلام Q_2) رقم القسم من المجموعة الأولى (المنتمية لإستعلام Q_1) والذي يجب أن يكون مطابق لأحد المجموعتين والذي يجب أن يكون (**SEC_NO**) وذلك من أجل أن يتم الإستعلام

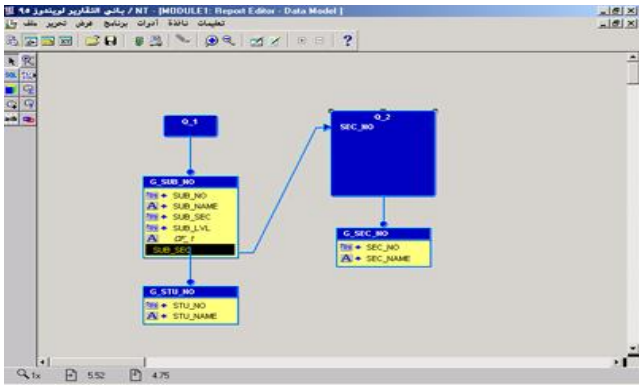
ولاحظ في **نموذج البيانات** أنه قد أضيف مجموعة إستعلام جديدة في جزء (الإستعلامات) ومجموعة جديدة في جزء (المجموعات) ورابط بيانات جديد في جزء (روابط البيانات) .

==========

ملاحظة هامة :

- تفضل الإضافة بإستخدام حقل صيغة Formula لأنها أسرع بالتنفيذ وأسهل من أجل أنها لم تتطلب إنشاء :
  1. **إستعلام SQL ،**
  2. **إطار تكراري Repeating Frame ،**
  3. **ربط البيانات Data Link .**





❖ حقول التجميع Summary :

في كثير من الأحيان نحتاج إلى إضافة توابع تجميعية للتقرير (**المجموع – المتوسط – الحد أدنى – الحد أقصى – العدد – أول – أخير – من الإجمالي % - الإنحراف المعياري – التفاوت**) وذلك بغرض عمل تلخيص توضيحي لبيانات السجلات خاصة عندما يكون حجم البيانات في التقرير كبير جداً ، وكمثال على ذلك لو أردنا في التقرير السابق أن نظهر **عدد المواد التي تم تدريسها** فستكون الخطوات كما يلي :

أولاً نقوم بتحديد (**نموذج البيانات**) الموجود في شاشة الـ(Object Navigator) ونضغط F2 لتظهر نافذة فيها الإستعلامات الموجودة للتقرير (حالياً **Q_1 & Q_2**) ومن شريط الأدوات العمودي نضيف **عمود ملخص** إلى داخل المجموعة الأولى **G_SUB_NO** ولنرى أنه قد تكون شريط تمرير عند إضافته فنقوم بتوسعة عرض عناصر المجموعة (نلاحظ أن إسم عمود ملخص الإفتراضي هو **CS_1**) ،

ثانياً) نقوم بتحديد عنصر عمود ملخص CS_1 ونضغط F4 لنعدل خصائصه كالتالي :

	Property	Value
1	Function	العدد
2	Source	Sub_No
3	Reset act	Page

ليكن التابع التجميعي هو العدد Counter وليكن مصدره رقم المادة Sub_No ولنجعل إعادة تعيين عمود ملخص هذا في كل صفحة Page وليس كل تقرير Report (**أي في كل صفحة يتم التجميع للعداد ثم في بداية الصفحة الجديدة يتم تصفير العداد وليبدأ العداد بعملية التجميع من جديد كل صفحة على حده**) ،

ثالثاً) ننشئ من شريط الأدوات العمودي التابع لـ(**نموذج النسق Layout Model**) ما يلي :

- **عنصر حقل Field :**

لنعرض البيانات داخله ثم نجعله في الإطار التكراري الأب **G_SUB_NO** بعد فتح قفل طور الإحتواء من شريط الأدوات الأفقي للسماح بإضافة الحقل ، ولا ننسى ربطه مع عمود ملخص من خلال الخاصية **Source** وذلك بعد تحديده وضغط F4 ،

Property	Value
Source	CS_1

- **عنصر نص Display :**

ليوضح عنوان عمود الصيغة وليكن (عدد المواد) .

==========

تلميح :

من الممكن أيضاً إضافة عنصر يوضح عدد الطلاب الدارسين لكل مادة وبنفس الطريقة .

المحاضرة الحادية عشر

❖ القـــوائـــم Menus :

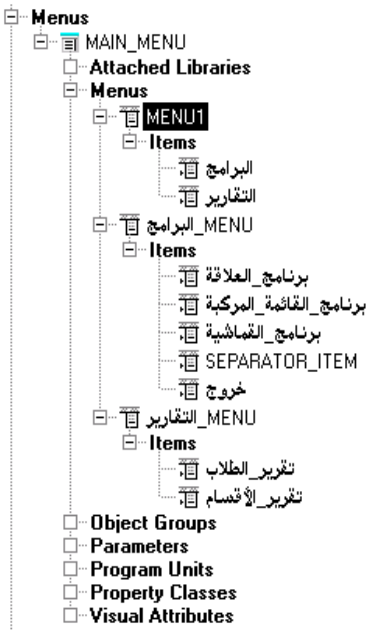
وهي القوائم المنسدلة في بداية تشغيل البرنامج ومن المعلوم أن القوائم في Oracle تأخذ ملفاً مستقلاً بحيث نستطيع إستدعاء ملفها التنفيذي من أي برنامج وقد ذكرنا سابقاً أنه عند عملية :

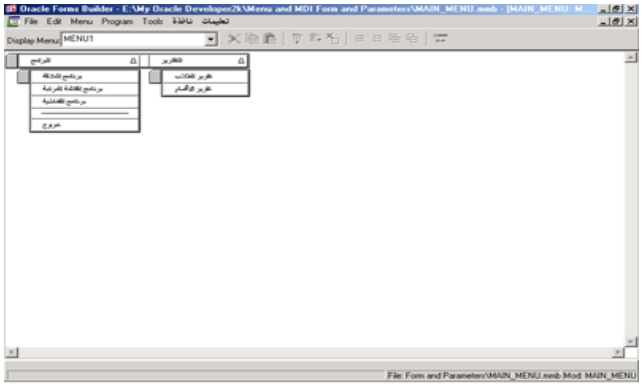
- حفظ القوائم Save يتولد ملف مصدري Source ذو إمتداد *.MMB ويشغله برنامج Form Builder
- الترجمة Compile يتولد ملف تنفيذي Execute ذو إمتداد *.MMX ويشغله برنامج Forms Runtime ،

ليس ذلك فحسب بل يجب أن تكون النسخة التنفيذية منتمية إلىMODULE معين لإتمام عملية التنفيذ

مثال ذلك :

نقوم بتحديد الـ(Menus) الموجود في شاشة الـ(Object Navigator) ثم Create لينشئ لنا إسماً إفتراضياً للقائمة المنشئة فنحددها ثم ننقر نقراً مزدوجاً لتصميم القائمة
وكالتالي :





لنكتب في محرر القوائم عنوان القائمة الأولى (**البرامج**) ثم ننشئ العناصر التابعة لهذه القائمة من خلال إختيار الأمر **Create Down** من القائمة Menu الموجودة في نافذة محرر القوائم أو من خلال الأيقونة الموجودة في شريط الأدوات الأفقي في نافذة محرر القوائم ونكتب (**برنامج العلاقة**) وبنفس الطريقة ننشئ العنصر الثاني ونكتب (**برنامج القائمة المركبة**) فالعنصر الثالث بكتابة (**برنامج القماشية**) فالعنصر الأخير (**خروج**) ، نقوم الآن بالرجوع إلى العنصر الرئيس للقائمة الأولى (البرامج) لنحددها ثم ننشئ القائمة المجاورة لهذه القائمة من خلال إختيار الأمر **Create Right** من القائمة Menu الموجودة في نافذة محرر القوائم أو من خلال الأيقونة الموجودة في شريط الأدوات الأفقي في نافذة محرر القوائم ونكتب (**التقارير**) ثم ننشئ العناصر التابعة لهذه القائمة من خلال إختيار الأمر **Create Down** من القائمة Menu الموجودة في نافذة محرر القوائم أو من خلال الزر الموجود في شريط الأدوات الأفقي في نافذة محرر القوائم ونكتب (**تقرير الطلاب**) وبنفس الطريقة ننشئ العنصر الثاني ونكتب (**تقرير الأقسام**) ، وبذلك نكون قد إنتهينا فقط من مرحلة تصميم القائمة لنلاحظ في الـ (Menus) تكون قائمتي البرامج والتقارير بالإضافة إلى قائمة حاوية للقائمتين المنشئتين .

حتى الآن لا يمكن أن نولد الملف التنفيذي لعدم إحتواء القائمة على أية شفرة مصدرية Code

==========

❖ خصائص عناصر القائمة Menu Items :

	Property	Value
1	Enabled	Yes
2	Menu Item Type	Plain
3	Magic Item	None
4	Menu Item Code	**More...**
5	Visible in Menu	Yes
6	Visible in Horizontal Menu Toolbar	No
7	Visible in Vertical Menu Toolbar	No
8	Icon in Menu	No
9	Icon Filename	

✍ شرح مبسط :

- تمتلك خاصية **Menu Item Type** خمسة قيم كل قيمة لها وظيفة معينة وكالتالي :
  a. **Plain :** وهي الخاصية الإفتراضية (تنفيذ شفرة مصدرية Code فقط) ،
  b. **Check :** تظهر الإشارة (✓) لتعبر عن الإختيار المتعدد وذلك بجانب تنفيذ الشفرة المصدرية Code ،
  c. **Radio :** تظهر الإشارة (•) لتعبر عن الإختيار المفرد وذلك بجانب تنفيذ الشفرة المصدرية Code ،
  d. **Separator :** يولد خط فاصل فقط وبدون كتابة أي شفرة مصدرية Code ،
  e. **Magic :** وتأتي مرتبطة مع الخاصية **Magic Item** ولتقوم بتنفيذ إستدعاءات معرفة في اللغة مسبقاً (Cut , Copy , Paste , Clear , Undo , Help , About , Quit , Window) وهي

- بالعودة للقائمة الأولى (**البرامج**) نضيف في محرر القوائم عنصر فاصل SEPARATOR_ITEM قبل العنصر الأخير (**خروج**) بحيث نجعل هذا العنصر يمتلك القيمة**Separator** التابعة للخاصية **Menu Item Type** .

- من أهم الخصائص خاصية **Menu Item Code** وهي بمثابة الـTrigger الذي سيكتب من خلاله الشفرة المصدرية Code و بدونها لا يمكن أن يتولد أي ملف تنفيذي للقوائم ، قم حالياً بكتابة التعليمة `Execute_Query;` كشفرة لكل عناصر القائمة لأننا لم ندرس حتى الآن فقرة الربط مع النماذج الأخرى Forms .

- عند إعطاء القيمة No للخاصية **Visible in Menu** فإن العنصر المحدد من القائمة لا يظهر وهذه الطريقة مفيدة في منح الصلاحيات لمستخدمين معينين .

- في كثير من الأحيان نحتاج إلى إظهار الأيقونات على شريط الأدوات الأفقي أو العمودي لتسريع عمل البرنامج ولتوقع الإستخدام بكثرة (**أشرطة الأدوات التابعة للقائمة وليس للـ Canvas**)
  a. الخاصية **Visible in Horizontal Menu Toolbar**
  b. الخاصية **Visible in Vertical Menu Toolbar**

عند إعطائهما القيمة Yes يعملان على إظهار الأيقونة بالطبع بعد إعطاء القيمة Yes للخاصية **Icon in Menu** .

- بالإمكان تصميم أيقونة خاصة بالعنصر الذي يتبع القائمة من خلال الخاصية **Icon Filename** وذلك بعد إعطاء القيمة Yes للخاصية **Icon in Menu** .

- عند تصميم برنامج Form Builder وأردنا إعطاءه القائمة المنشئة فيكون ذلك من خلال منح الملف التنفيذي للقائمة كقيمة للخاصية **Menu Module** التابعة للـ MODULE وكالتالي :

Property	Value
Menu Module	C:\MAIN_MENU.MMX

- لاحظ في محرر القوائم زر **Switch Orientation** والذي يعمل على تحويل المنشئ بزر الـ Create Down إلى Create Right وكذلك تحويل المنشئ بزر الـ Create Right إلى Create Down .

- إذا قمت بتعديل ما على القائمة فيجب أن تحفظ ذلك**File → Save** وتترجمه من أجل أن يتولد الملف التنفيذي الجديد **File → Administration → Compile File** .

- نلاحظ عند التنفيذ وجود قائمة تسمى (**نافذة Window**) وهي تأتي إفتراضياً مع القوائم المنشئة .

- يمكن التحكم بإغلاق النموذج من الذاكرة من خلال تعليمة `Close_Form('') ;` ، حيث نكتب في المعامل المرسل argument إسم البرنامج مباشرة بدون ذكر المسار لأنه موجود بالذاكرة .

❖ **القـــوائـــم**

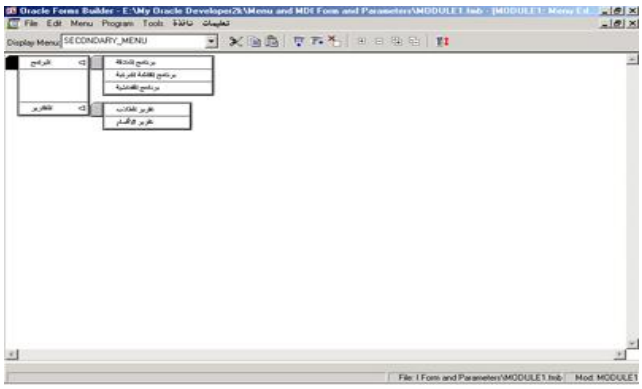
المـــنـــبـــثـــقـــة بـــزر الـــفـــأرة الأيـــمـــن Popup Menus :

وهي القوائم التي تنبثق عندما نقوم بالنقر على زر الفأرة الأيمن وذلك على الكائن الذي صممت القائمة المنبثقة له وتختلف إختلافاً جوهرياً عن القوائم Menus في أنها :

a. لا تحتاج إلى ملف مستقل لإنشائها وبالتالي تصمم داخل برنامج الـ Form Builder ،
b. التصميم فيها معكوس فالزر **Create Down** لا يولد عنصراً مرؤوساً وإنما يولد عنصراً زميلاً والزر **Create Right** لا يولد عنصراً زميلاً وإنما يولد عنصراً مرؤوساً ، والسبب في ذلك يعود إلى طبيعة عمل وشكل القوائم المنبثقة فهي جانبية وليست عمودية .
c. تُنفذ القائمة المنبثقة من خلال برنامج الـ Form Builder .

مثال ذلك :

نقوم بتحديد الـ(Popup Menus) الموجود في شاشة الـ(Object Navigator) ثم Create لينشئ لنا إسماً إفتراضياً للقائمة المنشئة فنحددها ثم ننقر نقراً مزدوجاً لتصميم القائمة وكالتالي :



لنكتب في محرر القوائم عنوان القائمة الأولى (**البرامج**) ثم ننشئ العناصر التابعة لهذه القائمة من خلال إختيار الأمر **Create Right** من القوائم أو من خلال [illegible] (**برنامج العلاقة**) وبنفس [illegible] (**برنامج القماشية**)

فالعنصر الأخير (**خروج**) ، نقوم الآن بالرجوع إلى العنصر الرئيس للقائمة الأولى (البرامج) لنحددها ثم ننشئ القائمة المجاورة لهذه القائمة من خلال إختيار الأمر **Create Down** من القائمة Menu الموجودة في نافذة محرر القوائم أو من خلال الأيقونة الموجودة في شريط الأدوات الأفقي في نافذة محرر القوائم ونكتب (**التقارير**) ثم ننشئ العناصر التابعة لهذه القائمة من خلال إختيار الأمر **Create Right** من القائمة Menu الموجودة في نافذة محرر القوائم أو من خلال الزر الأيقونة الموجودة في شريط الأدوات الأفقي في نافذة محرر القوائم ونكتب (**تقرير الطلاب**) وبنفس الطريقة ننشئ العنصر الثاني ونكتب (**تقرير الأقسام**) ، وبذلك نكون قد إنتهينا فقط من مرحلة تصميم القائمة لنلاحظ في الـ (Menus) تكون قائمتي البرامج والتقارير بالإضافة إلى قائمة حاوية للقائمتين المنشئتين .

الآن يمكن أن ننفذ البرنامج مع عدم إحتواء القائمة على أية شفرة مصدريةCode

==========

❖ خصائص عناصر القائمة المنبثقة Popup Menu Items :

نفس الخصائص المشروحة في القوائم Menu ، **عدا :**

a. ظهور القائمة المنبثقة يكون من خلال منحها لخاصية Popup Menu المضمنة في خصائص (العناصر **Items** أو القماشية **Canvas**) فمثلاً لإظهار القائمة الحاوية لقائمتي البرامج والتقارير في عنصر Display Item فإننا نجعل خصائصه كالتالي :

	Property	Value
1	Popup Menu	SECONDARY_MENU
2	Database Item	No

b. لا توجد أيقوناتها على أشرطة الأدوات الأفقية أو العمودية أي لا توجد بها خصائص **Visible in Horizontal Menu Toolbar** و **Visible in Vertical Menu Toolbar** ،

c. بالطبع لا وجود للملف التنفيذي المستقل أو قائمة تسمى (**نافذة Window**) تأتي إفتراضياً أثناء التنفيذ .

والآن إذا أردنا إنشاء عنصرين من النوع Display Item بحيث نجعل أحدهما يظهر قائمة البرامج المنبثقة والآخر يظهر قائمة التقارير المنبثقة فما علينا سوى أن نجعل خصائصهما كالتالي :

	Property	Value
1	Popup Menu	البرامج_MENU
2	Database Item	No

	Property	Value
1	Popup Menu	التقارير_MENU
2	Database Item	No

❖ طـرق الـربـط بـيـن أكـثـر مـن (Forms) أو بـيـن الـ(Form) والـ(Report) :

عند التحدث عن طرق ربط أكثر من نموذج يتبادر إلى الذهن مباشرة موضوع النموذج الأب **Parent** والنموذج الإبن **Child** فالأب يستدعي الإبن Calling ، تحت مظلة ما يدعى بواجهة المستندات المتعددة Multi)**MDI** Documents Interface) وبالتالي فإن طريقة إظهار نموذج آخر من نموذج حالي تختلف من برنامج إلى آخر حسب طبيعة عمل البرنامج وهذه الطرق مهما إختلفت فهي نتيجة للضغط على زر أو قائمة رئيسية أو قائمة منبثقة ، وطرق الربط كما يلي :

```
1. OPEN_FORM(Form_Name , Active_Mode , Session_Mode , Data_Mode ,
   Paramlist_Name Or Paramlist_Id) ;

2. CALL_FORM(FormModule_Name , Display , Switch_Menu , Query_Mode ,
   Data_Mode , Paramlist_Name Or Paramlist_Id) ;

3. RUN_PRODUCT(Product , Module , Comm_Mode , Exec_Mode , Location ,
   Paramlist_Name Or Paramlist_Id , Display ) ;
```

✍ ملاحظة هامة :

- لا يمكن الربط بين الـ(Form) والـ(Report) إلا عن بطريقة الربط**Run_Product** .

==========

۞ طـريـقـة الـربـط الأولـى (فـتـح الـنـمـوذج Open_Form) ۞

وتمتلك الـ arguments التالية :

1. **إسم النموذج Form_name :**

حيث نكتب هنا مسار البرنامج الإبن المراد فتحه ،

2. **نمط التنشيط Active_Mode :**

له قيمتين إما **Activate** أي أنه بعد فتح النموذج الإبن سينتقل المؤشر مباشرة إلى كُتل النموذج الإبن (وهو الإفتراضي) أو **Non_ Activate** أي أنه بعد فتح النموذج الإبن سيبقى المؤشر في النموذج الأب ،

3. **نمط الجلسة Session_Mode :**

لها قيمتين إما **NO_Session** وتعني أن النموذج الإبن والنموذج الأب ليسوا في جلسة واحدة وبالتالي إذا تم عمل تثبيت أو تراجع فإنه سيتم حفظ التغييرات التي حدثت على النموذج الإبن فقط (وهو الإفتراضي) ،

أو **Session** وتعني أن النموذج الإبن والنموذج الأب في جلسة واحدة وبالتالي إذا تم عمل تثبيت أو تراجع فإنه سيتم حفظ التغييرات التي حدثت على النموذج الإبن والنموذج الأب معاً .

4. **نمط البيانات Data_Mode :**

لها قيمتين إما **No_Share_Library_Data** وتعني عدم مشاركة البيانات لنموذج الأب والإبن في مكتبة بيانات واحدة (وهو الإفتراضي) أو **Share_Library_Data** وتعني مشاركة البيانات لنموذج الأب والإبن في مكتبة بيانات واحدة

5. [illegible]

Examples :

```
Open_Form('C:\Prog4.Fmx');

Open_Form('C:\Prog4.Fmx' , , , , PL_ID);
```

==========

۞ طـريـقـة الـربـط الـثـانـيـة (إسـتـدعـاء الـنـمـوذج Call_Form) ۞

وتمتلك الـ arguments التالية :

1. **إسم النموذج FormModule_name :**
كما شرح سابقاً .

2. **العرض Display :**
له قيمتين إما **Hide** أي أنه يستدعي النموذج الإبن ويخفي النموذج الأب (وهو الإفتراضي) أو **No_Hide** أي أنه يستدعي النموذج الإبن ويجعل النموذج الأب موجوداً خلف النموذج الإبن ولكنه غير محفز (غير منشط) ،

3. **قائمة التحويل Switch_Menu :**
لها قيمتين إما **No_Replace** وتعني عدم تبديل قائمة النموذج الأب وإنقلها للنموذج الإبن عند الإستدعاء (وهو الإفتراضي) أو **Do_Replace** وتعني تبديل قائمة النموذج الإبن بما لديه من قوائم .

4. **نمط الإستعلام Query_Mode :**
لها قيمتين إما **NO_Query_Only** وتعني فتح النموذج الإبن بغرض تطبيق العمليات المختلفة من إضافة وتعديل وحذف وإستفسار ... إلخ (وهو الإفتراضي) أو **Query_Only** وتعني فتح النموذج الإبن بغرض الإستفسار فقط .

5. **نمط البيانات Data_Mode :**
كما شرحت سابقاً .

6. **إسم المعامل أو رقم المعامل Paramlist_Name Or Paramlist_Id :**
كما شرحت سابقاً .

Examples :

```
Call_Form('C:\Prog4.Fmx');

Call_Form('C:\Prog4.Fmx' , Hide , No_Replace , No_Query_Only);
```

۞ طـريـقـة الـربـط الـثـالـثـة (تـنـفـيـذ الـمـنـتـج Run_Product) ۞

وتمتلك الـ arguments التالية :

1. **إسم المنتج Product :**
له ثلاث قيم إما **Forms** أو **Reports** أو **Graphics** أو بالامكان كتابة القيمة [**1** أو **2** أو **3**] وبنفس ال[illegible]

2. Module :
حيث نكتب هنا مسار البرنامج الإبن المراد فتحه ،

3. نمط الإتصال Comm_Mode :
لها قيمتين إما **Synchronous** وتعني السماحية بالعمل على النموذج الأب بجوار العمل مع النموذج الإبن

أو **Asynchronous** وتعني العمل مع النموذج الإبن فقط .

4. نمط التنفيذ Exec_Mode :
له قيمتين إما **Runtime** (وتستخدم مع Form Builder & Report Builder & Graphics Builder) أو **Batch** (وتستخدم مع Builder & Graphics Builder) فقط .

5. الموقع Location :
له قيمتين إما أن يكون ملف نظام **File System** أو ملف مخزن على قاعدة البيانات **Database** .

6. إسم المعامل أو رقم المعامل Paramlist_Name Or Paramlist_Id :
كما شرحت سابقاً .

7. العرض Display :
تستخدم مع الرسم البياني وذلك لإختيار نوع المخطط Graphics .

Examples :

```
Run_Product(1 , 'C:\Prog4.Fmx' , Synchronous , Runtime , Filesystem ,
                                                        '' , '');

                                    Run_Product(2 , 'C:\Rep4.Rep');
```

==========

ملاحظة هامة :

كنا قد أشرنا في فقرة القوائم إلى فقرة الربط مع النماذج الأخرىForms وأجلنا آنذاك كتابة الشفرة المصدرية Code لعناصر القائمة وها نحن الآن بعد إتمام هذه الفقرة نعاود كتابة الشفرة لتلك العناصر بعد تحديد العنصر وضغط F4 ومنها خاصية **Menu item Code** ولنكتب كما هو موضح ما يلي :

قائمة البرامج

(برنامج العلاقة)

```
Open_Form('C:\Prog1.Fmx');
```

(برنامج القائمة المركبة)

```
Call_Form('C:\Prog2.Fmx' , Hide , No_Replace , Query_Only);
```

(برنامج القماشية)

```
Call_Form('C:\Prog3.Fmx' , Hide , Do_Replace , No_Query_Only);
```

قائمة التقارير

(تقارير [illegible])

```
Run_Produ[illegible]2 , [illegible]
, '');
```

(تقارير الأقسام)

```
Run_Product(2 , 'C:\Rep2.Rep' , Synchronous , Runtime , Filesystem , ''
, '');
```

==========

❖ الـــمعـــامـــلات Parameters :

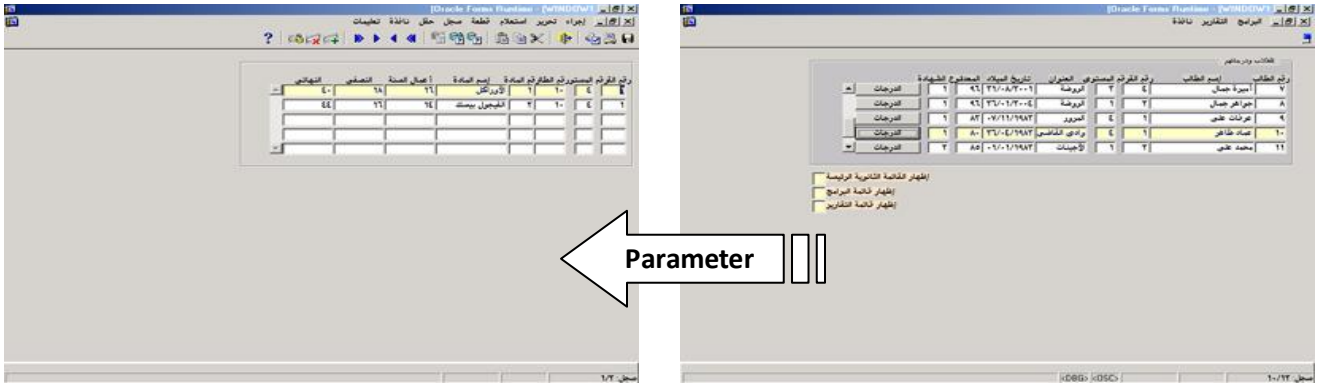
أحياناً وقبل فتح النموذج الإبن نريد أن نرسل له وسائط معينة (متحولات) لكي نتحكم بطريقة عرضه فمثلاً قد نحتاج لذلك في برامج : (الطلاب ودرجاتهم ، الدوائر وموظفيها ، ... إلخ) أشبه ببرامج الـ **Relations** ، ولعمل ذلك لا بد لنا أولاً من معرفة الخطوات التالية :

1. **في النموذج الأب :**
   - ✓ نعرف الـ Parameter ،
   - ✓ نعمل على تحميله بالقيمة ،
   - ✓ نرسل القيمة .
2. **في النموذج الإبن :**
   - ✓ ننشئ Parameter متوافق مع الـ Parameter المرسل ،
   - ✓ نعمل على إستقبال القيمة التي يحملها الـ Parameter المرسل ،
   - ✓ نستخدم القيمة .

مثال ذلك :

يوجد لدينا Form يحتوي على بيانات الطلاب STU و Form آخر يحوي الدرجات MARK ،

المطلوب : ربط النموذجين Forms بواسطة زر Push_Button بإستخدام Parameters ؟



✍ لعمل ذلك نقوم بالخطوات التالية :

1. في النموذج الأب نعرف الـ Parameter و نعمل على تحميله بالقيمة و إرسالها وذلك بواسطة حدث (**WHEN_BUTTON_PRESSED**) التابع لزر الدرجات وكالتالي :

```
Declare
   PL_ID Paramlist;
تعريف متحول من نوع البارامتر
Begin
   PL_ID := Get_Parameter_List('Student_Number');
إحضار البارامتر إن وجد إلى داخل المتحول
   If Id_Null(PL_ID) Then
فحص المتحول هل البارامتر منشئ أم لا
      PL_ID := Create_Parameter_List('Student_Number');
إنشاء بارامتر جديد إن لم يكن قد أنشئ من قبل
   Else
      Delete_Parameter(PL_ID,'Student_Number');
حذف القيمة التي كان البارامتر محملاً إياها
   End If ;

   Add_Parameter(PL_ID,'Student_Number',Text_Parameter,:Stu.Stu_No);
تحميل البارامتر النصي بالقيمة (رقم الطالب)

Run_Product(1,'C:\Mark\Module1.Fmx',Synchronous,Runtime,Filesystem,PL_I
D);
فتح النموذج الإبن مع إرسال قيمة البارامتر معه
End ;
```

2. في النموذج الإبن نقوم بتحديد الـ(Parameters) الموجود في شاشة الـ(Object Navigator) ثم Create لينشئ لنا إسماً إفتراضياً للمعاملات فنحدده ثم نضغط F4 ونغير قيمة خاصية الإسم البرمجي كما يلي :

Property	Value
Name	Student_Number

وفي حدث تحميل النموذج (**WHEN_NEW_FORM_INSTANCE**) نكتب التالي :

```
Set_Block_Property('MARK',Default_Where,'Mark_Stu:=Parameter.Student_Number');

إنتقاء سجلات جدول الدرجات عندما يكون رقم الطالب مساوياً للبارامتر

الذي تم إستقبال القيمة المرسلة به

Execute_Query;

إستخدام القيمة بالإستعلام
```

==========

✍ ملاحظات هامة :

- إذا أردت تحميل القيم (رقم القسم – رقم المستوى – رقم الطالب) فما عليك سوى كتابة تعليمة التحميل ثلاث مرات ويقيم مختلفة مرة برقم القسم ومرة برقم المستوى ومرة برقم الطالب Add_Parameter .
- يوجد نوع آخر من الـ Parameters غير الـ (Text Parameters) وهو الـ **(Data Parameters)** ويستخدم لإرسال مجموعة سجل Record Group .
- عند إرسال Parameter من برنامج مصمم بالـ Form Builder وتريد إستقباله في برنامج مصمم بالـ Report Builder فإنك ستستخدم **User Parameters** لإستقبال القيمة .

المحاضرة الثانية عشر

❖ التعامل مع ملفات الصورة :

يمكن التعامل مع ملفات الصورة بإحدى طريقتين :

1. تكون الصور موجودة على ملفات نظام System File وهذه الطريقة لا تعتمد على قاعدة البيانات Non_DB ،
2. تكون الصور مخزنة داخل قاعدة البيانات Database وهذه الطريقة تعتمد على قاعدة البيانات DB .

(والطريقة الثانية هي الأفضل من أجل عمليات الإستيراد والتصدير والأمنية)

أولاً : بواسطة ملفات النظام System File :

نضيف من شاشة الـ Layout Editor عنصر صورة Image Item ولتكن لديه الخصائص التالية :

	Property	Value
1	Name	TEST_IMAGE
2	Image Format	BMP
3	Image Depth	Original
4	Compression Quality	Minimum
5	Display Quality	High
6	Popup Menu	
7	Show Palette	Yes
8	Sizing Style	Adjust
9	Popup Menu	
10	Database Item	No

حيث ننتقي تنسيق الصورة Image Format صورة نقطية BMP ، وعمق الصورة Image Depth أصلي Original بينما نجد جودة الضغط Compression Quality بأدنى قيمة Minimum ، وجودة العرض Display Quality عالية High ، ونضع قيمة Yes لخاصة إظهار لوحة الألوان Show Palette ، ونسمح بتحريك الصورة (بجميع الاتجاه) من خلال مقبض Sizing Style للتحكم بحجم الصورة ، ونترك القيمة الافتراضية لقائمة الإقتطاع Crop

بحيث إذا كان حجم الصورة كبير يقوم بإضافة شريط تمرير لرؤية الصورة أم نريده موائمة Adjust بحيث يوائم الصورة على حجم الإطار الذي يحتويها .

بعد ذلك نحدد الحدث المناسب لقراءة الصورة وليكن عند نقر عنصر الصورة ولنكتب الشفرة المصدرية التالية :

(Trigger : WHEN_IMAGE_PRESSED)

```
Declare
  FileName Varchar2(50);
تعريف متحول من نوع محرفي ليخزن مسار الصورة
Begin
  FileName := 'C:\Img_Stu\Amerah.Bmp' ;
إسناد ملف صورة نقطية إلى المتحول
  Read_Image_File(FileName,'Bmp','Test_Image') ;
قراءة ملف الصورة وذلك من المتحول بإمتداد نقطي إلى داخل العنصر الذي تعرض به الصورة
End;
```

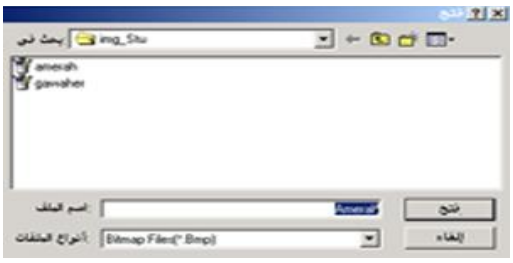
ولكننا نلاحظ أن الصورة أصبحت **صورة ثابتة** لجميع الطلاب (مثال تطبيقي) لذلك نقوم بتعديل قيمة المتحول وكالتالي :

```
FileName := Get_File_Name ;
ناتج إستدعاء دالة جلب الملف (معالج الفتح) يخزن إلى المتحول

FileName := Get_File_Name('C:\Img_Stu\' , 'Amerah.Bmp' ,'Bitmap
Files(*.Bmp)|*.Bmp|');
إستدعاء دالة جلب الملف مع منحها مسار إفتراضي وقيمة إبتدائية ونوع ملفات صورة نقطية

FileName := Get_File_Name('C:\Img_Stu\' , 'Amerah.Bmp' ,'Bitmap
Files(*.Bmp)|*.Bmp|', 'Jpg Files(*.Jpg)|*.Jpg|','Gif
Files(*.Gif)|*.Gif|','All Files(*.*)|*.*|');
```

فكلمة Bitmap Files هي عبارة توضيحية من إختيار المبرمج ،

والـ(Bmp.*) هي القيمة التي ستعرض داخل List Item الخاص بأنواع الملفات ،

والـ(Bmp.*|) هي القيمة الإبتدائية للملف ، وقس بقية التعليمات على غرارها .



📖 **التعليمة Get_File_Name تمتلك الـ arguments التالية**

6. إسم المجلد Directory_File ،
7. إسم الملف الإبتدائي File_Name ،
8. تعليمة فلترة الملفات المعروضة Filter ،
9. رسالة توضيحية Message ،
10. نوع المعالج هل هو OPEN أم SAVE AS .

ليكن لدينا الآن زر نقوم من خلاله بإستعراض ملفات الصور ومن ثم حفظها لدينا في البرنامج وكالتالي :

(Trigger : WHEN_BUTTON_PRESSED)

```
Declare

  FileName Varchar2(50);

Begin

 FileName := Get_File_Name('C:\Img_Stu\' , 'Amerah.Bmp' ,'Bitmap
Files(*.Bmp)|*.Bmp|', Save_File);

ناتج إستدعاء دالة جلب الملف يخزن إلى المتحول بعد فتح معالج الحفظ بإسم

   Write_Image_File(FileName,'Bmp','Test_Image') ;

كتابة ملف الصورة وذلك من المتحول بإمتداد نقطي من داخل العنصر الذي تعرض
به الصورة

End;
```

ولكن الأفضل أن لا نعطي للمستخدم فرصة الحفظ في مجلدات أخرى لذلك نقوم بتعديل قيمة المتحول
ليكون من المفترض القيام بسحب الصور إلى مجلد نقوم بإنشاءه ليتعامل معه برنامجنا ونسمي الصور بإسم
فرضاً أن تكون

```
FileName := Get_File_Name('C:\Img\' ||:Stu_Sec ||:Stu_Lvl ||:Stu_No
||'.Bmp');
```

والأفضل من ذا وذاك أن نلغي حدث نقر الصورة لنجعله عند الدخول على سجل جديد وكالتالي :

(Trigger : WHEN_NEW_RECORD_INSTANCE)

```
Declare

  FileName Varchar2(50);

Begin

 FileName := Get_File_Name('C:\Img\' ||:Stu_Sec ||:Stu_Lvl ||:Stu_No
||'.Bmp') ;

 Read_Image_File(FileName,'Bmp','Test_Image') ;

End ;
```

وهذه هي الشفرة المصدرية المعتمدة عند التعامل مع الصور بواسطة ملفات النظام System File .

ثانياً : بواسطة قاعدة البيانات Database :

1. ولعمل ذلك لا بد من إضافة هذا الحقل بأمر SQL وكالتالي :
```
SQL> alter Table Stu Add(Stu_Img Long Row);
```

2. (إذا قمت بإنشاء الحقل بعد جلب البيانات إلى الـForm فقم بتحديد الـ **Data Block (STU)** ثم إنقر على زر الفأرة الأيمن لتختار **Data Block Wizard** ومنه التبويب **Table** ثم إضغط الزر **Refresh** وبعدها قم بجلب العنصر STU_IMG ثم إضغط **نهاية**)

3. إمنح الخصائص التي تم ذكرها سابقاً لعنصر الصورة STU_IMG ولكن لاحظ أنها هنا معتمدة على قاعدة البيانات ،

Property	Value
Database Item	Yes

4. نقوم بكتابة الشفرة المصدرية في حدث (الدخول على عنصر جديد) التابع للحقل STU_IMG وكالتالي :

(Trigger : WHEN_NEW_ITEM_INATANCE)

```
Declare

  FileName Varch
```

```
Begin

 FileName := Get_File_Name('C:\Img\' ||:Stu_Sec ||:Stu_Lvl ||:Stu_No
||'.Bmp') ;

 Read_Image_File(FileName,'Bmp','Stu.Stu_Img') ;

End ;
```

❖ التعامل مع ملفات الصوت :

يمكن التعامل مع ملفات الصوت بإحدى طريقتين :

1. يكون الصوت موجود على ملفات نظامSystem File وهذه الطريقة لا تعتمد على قاعدة البياناتNon_DB
2. تكون الصوت مخزن داخل قاعدة البيانات Database وهذه الطريقة تعتمد على قاعدة البيانات DB .

(والطريقة الثانية هي الأفضل من أجل عمليات الإستيراد والتصدير والأمنية)

أولاً : بواسطة ملفات النظامSystem File :

نضيف من شاشة الـ Layout Editor عنصر صوت Sound Item ولتكن لديه الخصائص التالية :

	Property	Value
1	Sound Format	WAV
2	Audio Channels	Automatic
3	Compress	Automatic
4	Sound Quality	Automatic
5	Popup Menu	
6	Database Item	No
7	Show Play Button	Yes
8	Show Record Button	No
9	Show Rewind Button	No
10	Show Fast Forward	No
11	Show Volume Control	Yes
12	[illegible]	[illegible]

13	Show Slider	Yes

حيث ننتقي تنسيق الصوت Sound Format بـ WAV ، وقنوات الصوت Audio Channels بالأتوماتيكية Automatic ، بينما نجعل الضغط Compress أتوماتيكي Automatic وجودة الصوت Sound Quality أتوماتيكية Automatic ، ويفضل أن نظهر على عنصر الصوت أزرار (بداية تشغيل التسجيل والتحكم بحجم الصوت ... إلخ) من خلال منح القيمة Yes للخصائص (Show Play Button ، Show Volume Control ، Show Time Indicator ، Show Slider) .

ثم نحدد الحدث المناسب لقراءة الصوت وليكن عند الدخول على سجل جديد ولنكتب الشفرة المصدرية التالية :

(Trigger : WHEN_NEW_RECORD_INSTANCE)

```
Declare

  FileName Varchar2(50);

Begin

 FileName := Get_File_Name('C:\Snd\' ||:Stu_Sec ||:Stu_Lvl ||:Stu_No
||'.Wav') ;

 Read_Sound_File(FileName,'Wav','Test_Sound') ;

End ;
```

ثانياً : بواسطة قاعدة البيانات Database :

1. ولعمل ذلك لا بد من إضافة هذا الحقل بأمر SQL وكالتالي :
```
SQL> alter Table Stu Add(Stu_Snd Long Row);
```

2. (إذا قمت بإنشاء الحقل بعد جلب البيانات إلى الـForm فقم بتحديد الـ **Data Block (STU)** ثم إنقر على زر الفأرة الأيمن لتختار **Data Block Wizard** ومنه التبويب **Table** ثم إضغط الزر **Refresh** وبعدها قم بجلب العنصر STU_SND ثم إضغط **نهاية**)

3. إمنح الخصائص التي تم ذكرها سابقاً لعنصر الصوت STU_SND ولكن لاحظ أنها هنا معتمدة على قاعدة البيانات ،

Property	Value

4. نقوم بكتابة الشفرة المصدرية في حدث (الدخول على عنصر جديد) التابع للحقل STU_SND وكالتالي :
(Trigger : WHEN_NEW_ITEM_INATANCE)

```
Declare

  FileName Varchar2(50);

Begin

 FileName := Get_File_Name('C:\Snd\' ||:Stu_Sec ||:Stu_Lvl ||:Stu_No
||'.Wav') ;

 Read_Sound_File(FileName,'Wav','Stu.Stu_Snd') ;

End ;
```

❖ وامـــر تـــحـــجـــيـــم الـــصـــورة :

لنقم بتحديد الـ(Popup Menus) الموجود في شاشة الـ(Object Navigator) ثم Create لننشئ قائمة منبثقة بزر الفأرة الأيمن حيث نكتب في محرر القوائم عنوان القائمة (**تحجيم الصورة**) ثم ننشئ العناصر التابعة لهذه القائمة من خلال إختيار الأمر **Create Right** من القائمة Menu الموجودة في نافذة محرر القوائم أو من خلال الأيقونة الموجودة في شريط الأدوات الأفقي في نافذة محرر القوائم ونكتب (**تكبير**) وبنفس الطريقة ننشئ العنصر الثاني ونكتب (**تصغير**) فالعنصر الثالث بكتابة (**موائمة**) ، ومن خلال خاصية **Menu Item Code** التابعة لكل عنصر في القائمة نكتب الشفرة المصدرية وكالتالي :

(**Item : تكبير**)

```
Image_Zoom('Stu_Img' , Zoom_In_Factor , 3) ;
```

تـكـبـيـر بـمـقـدار ثـلاث مـرات

(**Item : تصغير**)

```
Image_Zoom('Stu_Img' , Zoom_Out_Factor , 2) ;
```

تـصـغـيـر بـمـقـدار مـرتـين

(**Item : موائمة**)

```
Image_Zoom('Stu_Img' , Adjust_To_Ft) ;
```

ولنحدد ظهور هذه القائمة المنبثقة على عنصر صورة الطالب Stu_Img وكالتالي :

Property	Value
Popup Menu	تحجيم الصورة_MENU

ملاحظات هامة :

- لا يمكن أن يحتوي الجدول على أكثر من حقل من النوع Long Row ،
- الفرق الجوهري بين التعامل (بواسطة ملفات النظام أو بواسطة قاعدة البيانات)هو أن :

ملفات النظام عند الخروج من البرنامج (إنهاء البرنامج) تطالب المستخدم بالحفظ لتؤثر على قاعدة البيانات Database .

غالباً ما نقوم بالإستيراد والتصدير من خلال أي زر أو عنصر قائمة منبثقة ولنكتب فيها الشفرة المصدرية التالية :

(التصدير)

```
Host('Exp UserId = Ust/Taiz File = C:\Expdat.Dmp') ;
```

تصدير قاعدة بيانات المستخدم إلى ملف محدد

```
Host('Exp UserId = Ust/Taiz File = C:\Expdat.Dmp Full = Y') ;
```

تصدير قاعدة بيانات كاملة إلى ملف محدد

(الإستيراد)

```
Host('Imp UserId = Ust/Taiz File = C:\Expdat.Dmp') ;
```

إستيراد قاعدة بيانات المستخدم من ملف محدد

```
Host('Imp UserId = Ust/Taiz File = C:\Expdat.Dmp Full = Y') ;
```

إستيراد قاعدة بيانات كاملة من ملف محدد

==========

تلميح :

- التعليمة **Host** تقوم بإستدعاء أمر نظام التشغيل **Start → Run** ثم تستدعي أحد البرنامجين **EXP** أو **IMP** (حسب الطلب) ومن ثم تعطي معاملاتها كاملة للبرنامج ولاحظ عند كتابتها إمتلاكها لفاصلتين علويتين إحداها بالبداية والأخرى بالنهاية .
- هناك أيضاً تصدير أو إستيراد لجدول معين من مستخدم محدد من قاعدة البيانات .

ملحق الصيغ العامة لجمل SQL

((DML & DDL & DCL))

select…from…;

select…from…where…;

select…from…order by…;

select…from…group by…;

select…from…having…group by…;

select…from…where…[in or =] select…from…where…;

---

insert into...values...;

insert into...select...;

---

update...set...;

update...set...where...;

---

delete from...;

delete from...where...;

---

create...;

create...as select...;

---

alter...add...;

alter...modif

alter...drop...;

---

drop...;

drop...cascade;

---

grant…to...;

grant…on…to...;

---

revoke...from...;

revoke...on...from...;

- **Fields of Emp Table**

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	*Deptno*

- **Fields of Dept Table**

Deptno	Dname	Loc

- **Fields of Bonus Table**

Ename	Job	Sal	Comm

- **Fields of Salgrade Table**

Grade	LoSal	HiSal

- **select** * **from** Tables_name **;**

- **select** Fileds_name **from** Tables_name **;**

- **select** Fileds_name **from** Tables_name **where** Conditions **;**

- **select** Fileds_name **from** Tables_name **order by** Filed_name Sort type [Asc or Desc] **;**

- **select** Fileds_name **from** Tables_name **order by** Filed_name i Sort type **,** … ,

Filed_name n Sort type **;**

- **select** Fileds_name **from** Tables_name **group by** Fileds_name **;**

- **select** Fileds_name **from** Tables_name **group by** Fileds_name **having** Conditions **;**

- **select** Fileds_name **from** Tables_name

where Conditions [**in** or **=**] **select** Fileds_name **from** Tables_name **where** Conditions **;**

- insert into Table_name **(**Fileds_name**) values (**Values by Fields Sort [Serial] **) ;**

- **insert into** Table_name **values (**Values by Table Description**) ;**

- **insert into** Table_name **(**Fileds_name**) values (&** message **,'&** message (string)**' ,** … **) ;**

- **insert into** Table_name **select** Fileds_name **from** Tables_name **where** Conditions **;**

- **insert into** Table_name **(**Fileds_name**) select** Fileds_name **from** Tables_name

where Condition **;**

- update Table_name **set** Filed_name = New value

- update Table_name **set** Filed_name = New value **where** Conditions **;**

- **delete from** Table_name **;**

- **delete from** Table_name **where** Conditions **;**

- **create user** User_name **identified by** Password **;**

- **create role** Role_name **;**

- **create role** Role_name **identified by** Password **;**

- **create table** Table_name **(**Filed_name i Filed_type i **constraint** Constraint_type **,** … ,

Filed_name n Filed_type n **constraint** Constraint_type**) ;**

- **create table** Table_name **(**Filed_name i Filed_type i **constraint** Constraint_type **references** Table_name(Filed_name i), … ,

Filed_name n Filed_type n **constraint** Constraint_type **references** Table_name(Filed_name n)**) ;**

- **create view** View_name **as select** Fileds_name **from** Tables_name **where** Conditions **;**

- **create synonym** Synonym_name **for** User_name**.**Table_name **;**

- **create index** Index_name **on** Table_name **(**Filed_name i **,** … ,Filed_name n**) ;**

- **create unique index** Index_name **on** Table_name **(**Fileds_name**) ;**

- **alter table** Table_name **add** Filed_name Filed_type

- **alter table** Table_name **modify (**Filed_name Filed_type**) ;**

- **alter table** Table_name **drop (**Filed_name Filed_type**) ;**

- **alter user** User_name **identified by** New Password **;**

- **drop user** user_name **cascade ;**

- **drop role** role_name **;**

- **drop table** table_name **;**

- **drop view** view_name **;**

- **drop synonym** synonym_name **;**

- **drop index** index_name **;**

- **drop unique index** index_name **;**

- **truncate table** Table_name ;

- **grant** Role_name **to** User_name **;**

- **grant** Role_name **to** Role_name **;**

- **grant** privileges **on** Table_name **to** User_name **;**

- **grant** privileges **to** Role_name **;**

- **revoke** Role_name **from** User_name **;**

- **revoke** Role_name **from** Role_name **;**

- **revoke** privileges **on** Table_name **from** User_name **;**

- **revoke** privileges **from** Role_name **;**

- **revoke** privileges **on** Table_name **from public ;**

- **revoke all on** Table_name **from public ;**

a_alyahawi@hotmail.com