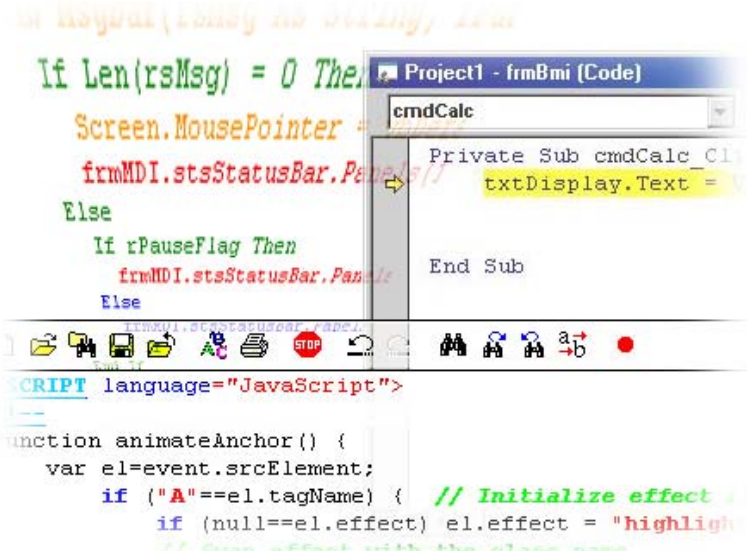
البرمجيات

برمجة قواعد بيانات

٢٦٢ حاب



مقدمة

الحمد لله وحدة، والصلاة والسلام على من لا نبي بعده، محمد وعلى آله وصحبه، وبعد:

تسعى المؤسسة العامة للتعليم الفني والتدريب المهني لتأهيل الكوادر الوطنية المدربة القادرة على شغل الوظائف التقنية والفنية والمهنية المتوفرة في سوق العمل، ويأتي هذا الاهتمام نتيجة للتوجهات السديدة من لدن قادة هذا الوطن التي تصب في مجملها نحو إيجاد وطن متكامل يعتمد ذاتياً على موارده وعلى قوة شبابه المسلح بالعلم والإيمان من أجل الاستمرار قدماً في دفع عجلة التقدم التنموي؛ لتصل بعون الله تعإلى لمصاف الدول المتقدمة صناعياً.

وقد خطت الإدارة العامة لتصميم وتطوير المناهج خطوة إيجابية تتفق مع التجارب الدولية المتقدمة في بناء البرامج التدريبية، وفق أسإلىب علمية حديثة تحاكي متطلبات سوق العمل بكافة تخصصاته لتلبي متطلباته ، وقد تمثلت هذه الخطوة في مشروع إعداد المعايير المهنية الوطنية الذي يمثل الركيزة الأساسية في بناء البرامج التدريبية، إذ تعتمد المعايير في بنائها على تشكيل لجان تخصصية تمثل سوق العمل والمؤسسة العامة للتعليم الفني والتدريب المهني بحيث تتوافق الرؤية العلمية مع الواقع العملي الذي تفرضه متطلبات سوق العمل، لتخرج هذه اللجان في النهاية بنظرة متكاملة لبرنامج تدريبي أكثر التصاقاً بسوق العمل، وأكثر واقعية في تحفيق متطلباته الأساسية.

وتتنأول هذه الحفيبة التدريبية " برمجة قواعد بيانات " لمتدربي قسم" البرمجيات " للكليات التقنية موضوعات حيوية تتنأول كيفية اكتساب المهارات اللازمة لهذا التخصص.

والإدارة العامة لتصميم وتطوير المناهج وهي تضع بين يديك هذه الحفيبة التدريبية تأمل من الله عز وجل أن تسهم بشكل مباشر في تأصيل المهارات الضرورية اللازمة، بأسلوب مبسط يخلو من التعفيد، وبالاستعانة بالتطبيقات والأشكال التي تدعم عملية اكتساب هذه المهارات.

والله نسأل أن يوفق القائمين على إعدادها والمستفيدين منها لما يحبه ويرضاه إنه سميع مجيب الدعاء.

الإدارة العامة لتصميم وتطوير المناهج

تمهيد

لقد تعلمت في مقررات قواعد البيانات السابقه أساسيات لغة PL/SQL وتعلمت كيفية تعريف المتغيرات وكيفية عمل الأبنيهةالهيكلية كما تعرفت على كيفية استخدام جمل معالجة البيانات DML داخل القطع البرمجية PL/SQL.

سنتابع في هذه الحقيبة دراستنا للغة PL/SQL وسنتعرف على مواضيع ومفاهيم مهمة في برمجة قواعد البيانات باستخدام لغة PL/SQL، حيث سنقوم بشرح المواضيع التإلية:

الوحدة الأولى: إدارة المستخدمين، تتحدث هذه الوحدة بشكل رئيس عن كيفية إنشاء المستخدمين ومنحهم امتيازات على مستوى النظام وعلى مستوى العناصر أو منع الامتيازات عنهم، كما سنتعرف على كيفية تغيير كلمة السر وكيفية التأكد من الامتيازات الممنوحة لمستخدم ما.

الوحدة الثانيه: المشيرات التصريحية، في هذه الوحدة سنتعرف على مفهوم المشيرات التصريحية وتحديد خصائص المشيرة، كما سنتعرف على كيفية التحكم بالمشيرات من حيث فتح المشيرة، جلب البيانات من المشيرة وإغلاق المشيرة. ستتعرف أيضا على كيفية استخدام المشيرة مع جملة التكرار for وكيفية استخدام السجلات مع المشيرات التصريحية.

الوحدة الثالثة: معالجة الاستثناءات، سنتعرف من خلال هذه الوحدة على مفهوم الاستثناءات: ما هي أنواعها وكيفية إطلاق الاستثناء كيف يمكننا الإيقاع أو الامساك بالاستثناء المنطلق وما هي الدوال المستخدمة للايقاع بالاستثناءات. وفي حالة أن الاستثناء حدث في قطعة برمجية لا تحتوي على معالج للاستثناء كيف ينتقل الاستثناء إلى القطع البرمجية الأخرى، وأين يتم الايقاع أو الإمساك به.

الوحدة الرابعة: القطع البرمجية، هذه الوحدة مقدمة تعريفية للوحدتين اللتين تليانها (وحدة الروتين ووحدة الدوال) ففي هذه الوحدة سنتعرف على أقسام القطع البرمجية المختلفة، وما هي الفوائد من استخدام كل نوع.

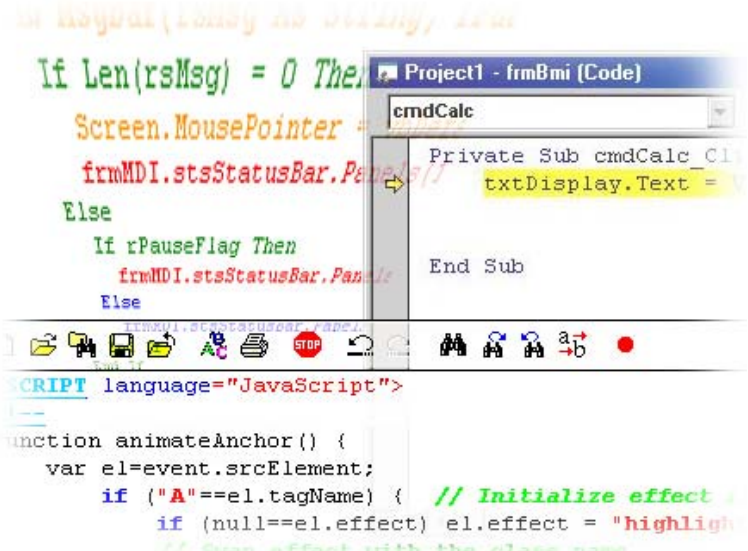
الوحدة الخامسة: الروتين، في هذه الوحدة سنتعرف على مفهوم الروتين، مراحل بناء الروتين وما هي حالات البارميترات الخصة بالروتين، كما سنتعرف على طرق تمرير البراميترات وكيفية مناداة الروتين من وحدة برمجية أخرى، وكيفية حذف الروتين من جهة خادم قواعد البيانات.

الوحدة السادسة: الدوال، ينتعرف هنا على مفهوم الدوال ومراحل بنائها وكيفية تنفيذ دالة معينة، كما سنقارن بين الروتين والداله أو الاقتران وما هي فوائد كل من الروتين والدالة.

الوحدة السابعة: زناد قواعد البيانات، سنتعرف في هذه الوحدة على مفهوم زناد قواعد البيانات، والإرشادات الواجب مراعاتها عند تصميم الزناد، وسنتعرف أيضا على أمر إنشاء الزناد على مستوى جملة أو سطر، وكيفية استخدام الكلمات الشرطية، وكيفية التمييز بين زناد قواعد البيانات والروتين وكيفية حذف الزناد من قواعد البيانات.

برمجة قواعد بيانات

إدارة المستخدمين



الجدارة:

أن يكون المتدرب قادراً على فهم الهدف من استخدام الامتيازات والصلاحيات في قاعدة البيانات، والتحكم بوصول قاعدة البيانات إلى الكائنات والعناصر المختلفة كما على المتدرب أن يكون قادراً على إنشاء مستخدمين بمستويات مختلفة من الامتيازات

الأهــداف:

بنهاية هذه الوحدة، عليك أن تكون قادرا على:

١. إنشاء المستخدمين
٢. إنشاء وظائف Rolls لتسهيل عملية صيانة ومتابعة مستوبات الحماية للمستخدمين
٣. استخدام أمري المنح Grant والمنع Revoke لمنح أو لمنع امتيازات العناصر لمستخدم معين

مستوى الأداء المطلوب:

ان يصل المتدرب إلى اتقان الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ٥ ساعات معتمدة

الوسائل المساعدة:

- وجود حاسب آلي
- وجود عارض شرائح Projector
- دفتر
- قلم

الوحدة الأولى: إدارة المستخدمين

التحكم بوصول المستخدمين لقاعدة البيانات

في بيئة متعددة المستخدمين Multi-user Environment أنت بحاجة لمتابعة حمايه قاعدة البيانات من حيث إمكانية الوصول إليها واستخدامها. ويمكن تقسيم الحماية في قاعدة البيانات إلى قسمين رئيسن:

١. حماية النظام System Security .

تتحكم حماية النظام في الوصول لقاعدة البيانات واستخدامها على مستوى النظام System Level مثل التحكم باسم المستخدم وكلمة المرور، حجم الديسك الممنوح للمستخدمين، وعمليات النظام المسموح للمستخدم الفيام بها.

٢. حماية البيانات Data Security

تتحكم حماية البيانات بعملية الوصول إلى عناصر قاعدة البيانات واستخدامها والعمليات الممكن للمستخدمين الفيام بها على هذه العناصر.

الامتيازات

الامتياز هو الحق في تنفيذ جملة SQLمعينة، يعتبر مشرف قاعدة البيانات Database administrator صاحب أعلى امتيازات والذي يستطيع أن يعطي المستخدمين الآخرين إمكانية الوصول لقاعدة البيانات وعناصرها. يحتاج المستخدمين إلى امتيازات النظام لتحفيق إمكانية الوصول لقاعدة البيانات وامتيازات على العناصر لتحفيق إمكانية معالجة محتويات العناصر داخل قاعدة البيانات. ويمكن إعطاء المستخدمين امتيازات تمكنهم من منح امتيازات معينه لمستخدمين آخرين أو لوظائف معينة (الوظيفة هي مجموعة من الامتيازات لها اسم واحد).

المخطط Schema: اسم يطلق على مجموعة واحدة من العناصر مثل الجدأول Tables، الجدأول الوهمية Views، والمتسلسلات Sequences. تكون Schema مملوكة من قبل مستخدم معين، ولها نفس اسم المستخدم.

امتيازات النظام System Privileges.

يوجد هناك أكثر من ٨٠ امتياز للنظام متوفرة للمستخدمين والوظائف Roles، ويتم إعطاء امتيازات النظام من قبل مشرف قاعدة البيانات Database Administrator. وإليك الآن عينة من أهم امتيازات النظام:

الامتياز	النشاطات الممكن أداؤها
CREATE USER	إنشاء مستخدم، وإسناده حصة من أي مساحة جدولية Tablespace.
DROP USER	إسقاط أو حذف مستخدم آخر.
ALTER USER	تعديل مستخدم آخر: تغيير كلمة المرور لمستخدم غير المستخدم الحالى، إسناد حصص في المساحة الجدولية.
CREATE ANY TABLE	إنشاء جدول في أي مخطط Schema
DROP ANY TABLE	حذف جدول من أي مخطط
CREATE SEQUENCE	إنشاء تتابع أو سلسلة في المخطط الخاص بالمستخدم
DROP SEGUENCE	حذف التتابع
CREATE TRIGGER	إنشاء زناد في المخطط الخاص بالمستخدم
DROP TRIGGER	حذف الزناد
ALTER TRIGGER	تعديل الزناد: تعطيل، تفعيل أو إعادة ترجمة الزناد
CREATE VIEW	إنشاء جدول وهمي (عرض)
DROP ANY VIEW	حذف عرض من أي مخطط
CREATE ROLE	إنشاء وظيفة
ALTER ANY ROLE	تعديل أي وظيفة في قاعدة البيانات
DROP ANY ROLE	حذف أي وظيفة من قاعدة البيانات
GRANT ANY ROLE	منح أي وظيفة في قاعدة البيانات
CREATE SYNONYM	إنشاء مرادف في قاعدة البيانات الخاصة بالمستخدم
DROP ANY SYSNONYM	حذف أي مرادف من قاعدة البيانات الخاصة بالمستخدم

إنشاء المستخدمين

يمكن لمشرف قاعدة البيانات إنشاء مستخدم وذلك بتنفيذ أمر CREATE USER، عند إنشاء مستخدم جديد بواسطة هذا الأمر فإنه لا يكون له أي امتيازات حتى هذه اللحظة. بعد ذلك يقوم مشرف قاعدة البيانات بمنح مجموعة من الامتيازات للمستخدم. هذه الصلاحيات تحدد ما الذي يستطيع المستخدم فعله على مستوى قاعدة البيانات.

```
CREATE USER user_name
IDENTIFIED BY password;
```

حيث أن :

User_name يمثل اسم المستخدم المراد إنشاؤه

Password تمثل كلمة المرور التي يجب على المستخدم استعمالها للدخول

مثال:

```
SQL> CREATE USER scott
  2  IDENTIFIED BY tiger;
User Created.
```

حالما يتم إنشاء المستخدم يقوم مشرف قاعدة البيانات DBA بمنحه مجموعة من الامتيازات باستخدام الأمر منح على النحو التإلى:

```
GRANT privilege [, privilege …]
TO user_name [, user_name, …];
```

حيث أن:

Privilege هو اسم الامتياز المراد منحه

User_name اسم المستخدم الممنوح له الامتياز

فمثلا من اهم الامتيازات الممكن منحها لمطور تطبيقات قواعد البيانات من الجدول أعلاه ما يلي:

- القدره على الاتصال بقاعدة البيانات CREATE SESSION
- إنشاء جدول CREATE TABLE
- إنشاء تتابع CREATE SEQUENCE
- إنشاء عرض CREATE VIEW
- إنشاء الروتين CREATE PROCEDURE

مثال:

```
SQL> GRANT create session, create table, create sequence, create view
  2  TO scott;
Grant succeeded.
```

Roleما هي الوظيفة

الوظيفة هي عدد من الامتيازات المجتمعة مع بعضها ولها اسم يمكن الرجوع إليها من خلاله، ويمكن منحها لمستخدم ما. هذه الطريقة تجعل من السهل علينا منح ومنع الامتيازات للمستخدمين. يمكن للمستخدم الواحد أن يمنح أكثر من وظيفة ويمكن للوظيفه أن تمنح لأكثر من مستخدم. يقوم مشرف قاعدة البيانات DBA بإنشاء الوظيفة أولا، ثم منح الامتيازات للوظيفة ومن ثم منح الوظيفة للمستخدمين.

```
CREATE ROLE role_name;
```

المثال التإلى يقوم بإنشاء وظيفة جديدة اسمها manager والسماح لها بإنشاء جدأولTables ، عروض views، ومن ثم منح كل من المستخدم BLAKE والمستخدم CLARK هذه الوظيفة. وبالتإلى فإن كلا المستخدمين يمكنهما الآن إنشاء الجدأول والعروض.

```
SQL> CREATE ROLE manager;
Role Created.
```

```
SQL> GRANT create table, create view
  2  TO manager;
Grant succeeded.
```

```
SQL> GRANT manger to CLARK;
Grant succeeded.
```

تغيير كلمة المرور

عندما يقوم مشرف قاعدة البيانات DBA بإنشاء مستخدم جديد، يقوم بإعطائه كلمة مرور مبدئية ليتمكن من الدخول. ويستطيع المستخدم تعديل كلمة المرور هذه بعد دخوله باسم المستخدم وكلمة المرور التي أعطاه اياها المشرف أول مرة، ويتم ذلك باستخدام أمر ALTER USER.

```
ALTER USER user_name IDENTIFIED BY password;
```

حيث أن :

User_name اسم المستخدم.

Password كلمة المرور الجديدة.

على الرغم من أن هذا الأمر يستخدم لتغيير كلمة المرور للمستخدم، إلا أن له استخدامات وخيارات أخرى عديده.

امتيازات العناصر Object Privileges

امتياز العنصر هو امتياز أو حق للفيام بعمل محدد على جدول Table ، عرض View ، تتابع Sequence أو روتين Procedure. لكل عنصر مجموعة من الامتيازات الممكن منحها. الجدول التإلى يوضح قائمة من الامتيازات للعديد من العناصر في قاعدة البيانات.

امتياز العنصر	جدول Table	عرض View	تتابع Sequence	روتين Procedure
ALTER	√		√	
DELETE	√	√		
EXECUTE				√
INDEX	√			
INSERT	√	√		
REFERENCE	√			
SELECT	√	√	√	
UPDATE	√	√		

منح امتيازات العناصر

إن العناصر المختلفة في مخطط ما ، لها امتيازات مختلفة. كما أن كل مستخدم يمتلك كل الامتيازات الخاصة بالعناصر الموجودة في المخطط الخاص به. ويمكن للمستخدم منح أي امتياز على عنصر معين لديه في مخططه لأي مستخدم آخر أو وظيفة معينة. وإذا احتوى أمر المنح على جملة with grant option فإن المستخدم الممنوح لهذا الامتياز يمكنه ايضا إعادة منحه لمستخدم آخر. وإذا لم تستخدم هذه الجملة فإن الشخص الممنوح للامتياز يمكنه استخدامه دون إعادة منحه لمستخدم آخر.

```
GRANT object_privilege ([columns])
ON      object
TO      {user | role | PUBLIC}
[WITH GRANT OPTION]
```

حيث أن:

Object_privilege	امتياز العنصر المراد منحه
Columns	أسماء الأعمدة من الجدول أو العرض المراد منح الامتيازات عليه
ON object	العنصر الذي تمنح عليه الامتيازات
TO	تحدد من الذي سيمنح هذا الامتياز
PUBLIC	لمنح الامتيازات لكل المستخدمين
WITH GRANT OPTION	تسمح للشخص الممنوحة له الامتيازات بمنحها إلى مستخدم آخر

مثال: امنح امتياز الاستعلام على جدول الموظفين EMP لكل من المستخدم sue و rich.

```
SQL> GRANT select
       ON emp
       TO sue, rich;
Grant succeeded.
```

إذا أراد المستخدم sue أو rich الاستعلام عن الجدول EMP فإن الصيغة كالآتي:

```
SQL> SELECT *
  2  FROM scott.emp;
```

وكبديل لذلك يستطيع كل من المستخدمين sue و rich إنشاء مرادف واستخدامه كالتإلى:

```
SQL> CREATE SYNONYM emp FOR scott.emp;
SQL> SELECT * FROM emp;
```

وإليك المثال التإلى الذي يستخدم جملة WITH GRANT OPTION والتي تعطي المستخدم الحريه في إعادة منح الامتياز إلى مستخدمين آخرين.

```
SQL> GRANT select, insert
  2  ON dept
  3  TO scott
  4  WITH GRANT OPTION;
Grant succeeded.
```

في هذا المثال يستطيع المستخدم scott الاستعلام والإدخال في جدول dept ، كما يستطيع أيضا منح هذه الامتيازات لأي مستخدم آخر.

كيفية التأكد من الامتيازات الممنوحه لمستخدم ما.

يمكنك التأكد من الامتيازات الممنوحه لك كمستخدم لقاعدة البيانات بالرجوع إلى مجموعة من الجدأول الموجودة في قاموس البيانات Data Dictionary ، والجدول التإلى يوضح أهم الجدأول ووصف كل جدول وكيفية استخدامه.

جدول قاموس البيانات Data Dictionary Table	الوصف
ROLE_SYS_PRIVS	امتيازات النظام الممنوحة للوظائف
ROLE_TAB_PRIVS	امتيازات الجدأول الممنوحة للوظائف
USER_ROLE_PRIVS	الوظائف الممكن استخدامها من قبل المستخدم
USER_TAB_PRIVS_MADE	امتيازات العناصر الممنوحة على عناصر المستخدم
USER_TAB_PRIVS_RECD	امتيازات العناصر الممنوحة للمستخدم
USER_COL_PRIVS_MADE	امتيازات العناصر الممنوحة على الأعمده لعناصر المستخدم
USER_COL_PRIVS_RECD	امتيازات العناصر الممنوحة للمستخدم عل أعمدة محددة

ملحوظة: في حال محأولة المستخدم إجراء عملية دون امتلاك الامتياز المناسب لإجرائها ، سيقوم خادم قواعد البيانات بمنع هذه العملية وتمرير رسالة خطأ للمستخدم.

منع امتيازات العناصر.

يمكنك منع امتيازات قمت بمنحها لمستخدم ما باستخدام الأمر REVOKE، عند استخدامك لهذا الأمر، فإنه يتم منع الامتيازات المحددة من المستخدمين الموضحين في الأمر ومن أي مستخدم آخر حصل عليها بواسطة الخيار WITH GRANT OPTION من أحد هؤلاء المستخدمين.

```
REVOKE {privilege [,privilege . . .] | ALL }
ON          Object
FROM        {user [, user . . .] | role | PUBLIC}
[CASCADE CONSTRAINTS];
```

حيث أن:

CASCADE CONSTRAINTS	تستخدم لحذف أي محدد مرجعي Referential Integrity Constraint معمول على العنصر بواسطة جملة References

مثال: امنع المستخدم Scott من إجراء عمليات الاستعلام على جدول EMP لديك.

```
SQL> REVOKE select
  2  ON emp
  3  FROM scott;
Revoke succeeded.
```

تمارين

س١) ما هو الامتياز الواجب إعطاؤه للمستخدم لعمل دخول على خادم قاعدة البيانات؟ هل هو امتياز نظام أم امتياز عنصر؟

_____

س٢) ما هو الامتياز الواجب إعطاؤه للمستخدم لإنشاء جدول؟

_____

س٣) إذا أنشأت جدولاً، من يستطيع أن يمرر الامتيازات على هذا الجدول للمستخدمين الآخرين؟

_____

س٤) إذا كنت مشرف قاعدة البيانات DBA، فإنك عادة تنشئ العديد من المستخدين الذين يحتاجون نفس امتيازات النظام، مإذا يمكنك أن تفعل للفيام بهذا العمل بشكل أسهل ؟

_____

س٥) ما الأمر الذي تستخدمه لتغيير كلمة المرور الخاصة بك؟

_____

س٦) أنت المستخدم SALEM، قم بمنح المستخدم ALI امتياز الاستعلام على جدول DEPT الخاص بك.

س٧) أنت المستخدم ALI اكتب جملة الاستعلام عن جدول DEPT الخاص بالمستخدم SALEM.

س٨) أنت المستخدم ALI اكتب جملة إنشاء مرادف للجدول DEPT الخاص بالمستخدم SALEM.

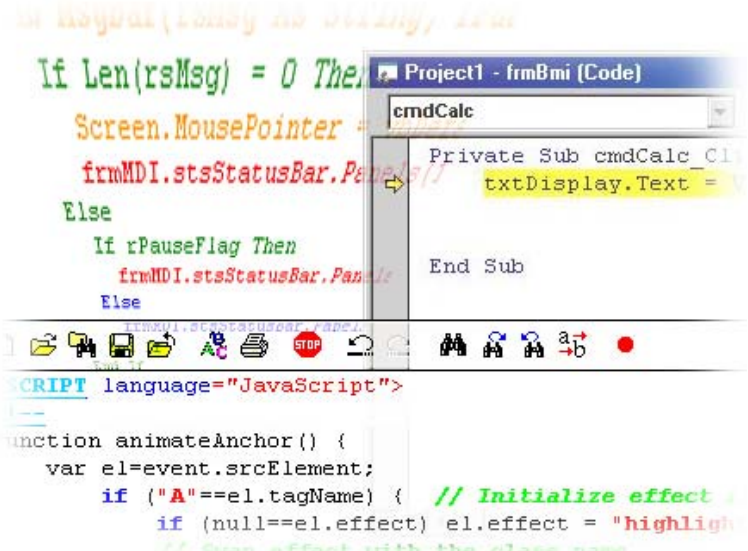
س٩) اكتب جملة الاستعلام عن الجدول DEPT باستخدام المرادف الذي أنشأته.

س١٠) استعلم عن جدول USER_TABLES من قاموس البيانات Data Dictionary لمعرفة أسماء الجدأول التي تملكها.

س١١) أنت المستخدم SALEM ، امنع امتياز الاستعلام عن جدول DEPT من المستخدم ALI.

برمجة قواعد بيانات

المشيرات التصريحية



الجدارة:

أن يكون المتدرب قادراً على التمييز بين المشيرات الضمنية والمشيرات التصريحية وأن يكون قادراً على كتابة تعريف المشيرة واستخدامها في القطع البرمجية

الأهـداف:

بنهاية هذه الوحدة، عليك أن تكون قادرا على:

١. تعريف مفهوم المشيرة التصريحية
٢. تحديد أهمية تعريف المشيرات التصريحية
٣. استخدام المتغيرات من نوع سجل
٤. استخدام المشيرات مع حلقات التكرار

مستوى الأداء المطلوب:

أن يصل المتدرب إلى اتقان الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ٤ ساعات معتمدة

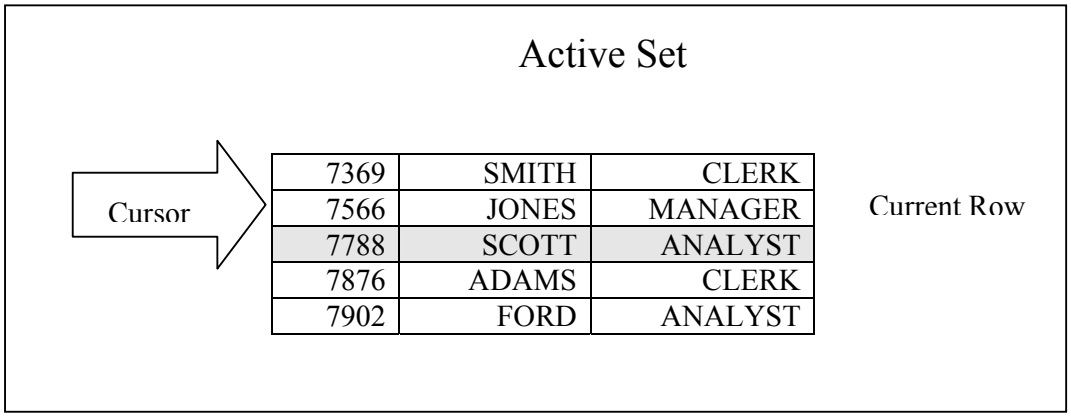
الوسائل المساعدة:

- وجود حاسب آلي
- وجود عارض شرائح Projector
- دفتر
- قلم

الوحدة الثانية

المشيرات التصريحية Explicit Cursors

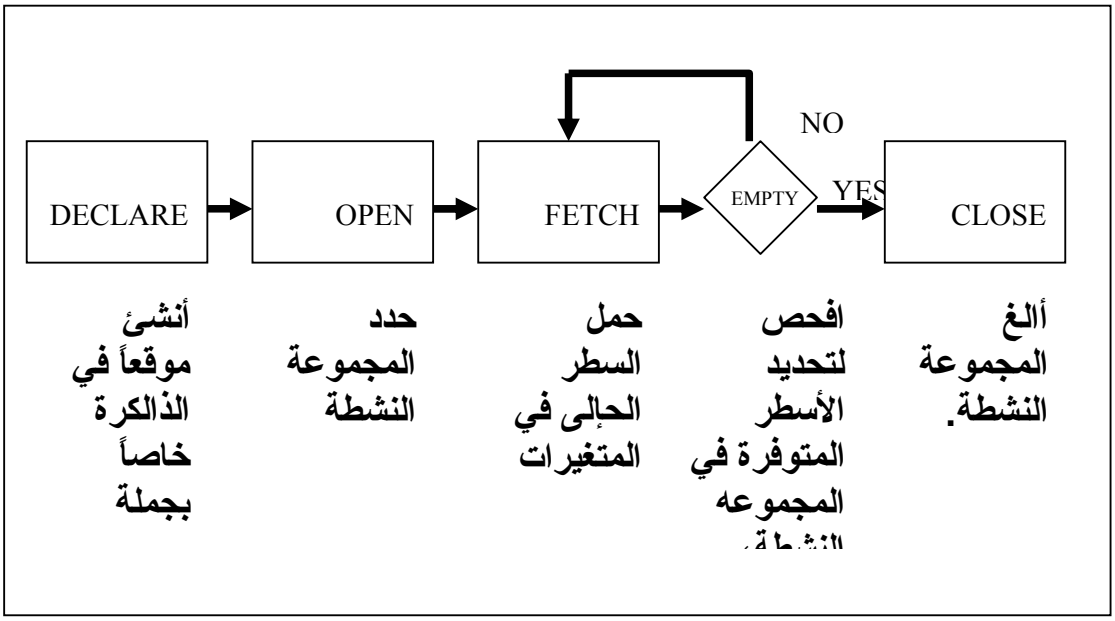
استخدم المشيرات التصريحية لمعالجة كل سطر مرتجع من جملة الاستعلام Select التي بها أكثر من سطر، بشكل مستقل عن باقي البيانات. تسمى مجموعة الأسطر المرتجعة من جملة الاستعلام بالمجموعة النشطة Active set ولها حجم مسأو لعدد الأسطر الذي يتوافق مع الشرط المحدد بجملة الاستعلام والشكل ادناه يوضح كيف أن المشيرة تؤشر دائما على السطر الحالى في المجموعة النشطة، وهذا يسمح لك بمعالجة الاسطر في برنامجك .واحداً تلو الآخر.



يقوم برنامج PL/SQL بفتح المشيرة، معالجة الأسطر المرتجعة من جملة الاستعلام ومن ثم إغلاق المشيرة. وتحدد المشيرة موقع السطر الواجب معالجته في المجموعه النشطة.

التحكم بالمشيرات التصريحية

الشكل التإلى يوضح مراحل عمل المشيرة التصريحية:



اعتمادا على الشكل أعلاه تلاحظ أن التحكم بالمشيرة التصريحية يتم من خلال أربع أوأمر رئيسة هي:

١. أمر التعريف Declare: ويتم هنا تعريف المشيرة بإعطائها اسماً وتعريف هيكل جملة الاستعلام التي ستنفذ من خلال المشبرة.
٢. أمر فتح المشيرة Open: يقوم هذا الأمر بتنفيذ جملة الاستعلام في المشيرة، ووضع مجموعة الاسطر الناتجه عنها في المجموعة النشطة في الذاكرة.
٣. أمر جلب البيانات من المشيرة Fetch: يقوم هذا الأمر بتحميل السطر الحالى من المجموعة النشطة للمشيرة إلى المتغيرات المحددة في البرنامج. إن كل عملية جلب للبيانات تتسبب في نقل المؤشر إلى السطر التإلى في المجموعة النشطة. عند كل عملية جلب للبيانات، تقوم المشيرة بفحص المجموعة النشطة للتأكد من أن هناك اسطر يمكن جلبها، إذا لم يكن هناك أسطر متبقية في المجموعة فإنه يتم غلق المشيرة.
٤. أمر إغلاق المشيرة Close: يقوم هذا الأمر بإخلاء المجموعة النشطة للمشيرة من الذاكرة، ويصبح بالإمكان إعادة فتح المشيرة من جديد.

وسنتعرف الآن على تفاصيل كتابة كل أمر من هذه الأوامر، وكيفية كتابة كود متكامل لاستخدامها.

تعريف المشيرة

نستخدم أمر Cursor لتعريف المشيرة بشكل تصريحي ويحتوي هذا التعريف بشكل رئيس على

- اسم المشيرة وهو عبارة عن معرف PL/SQL.
- جملة استعلام وهي جملة Select لكن لا تحتوي على جزء INTO حيث ستظهر لاحقا في أمر FETCH.

```
CURSOR cursor_name IS
        Select_statement
```

يمكنك استخدام الجزء ORDER BY في داخل جملة الاستعلام الموجودة في المشيرة.

مثال:

```
DECLARE
   CURSOR emp_cursor IS
         SELECT empno, ename
         FROM emp;

CURSOR dept_cursor  IS
      SELECT *
      FROM dept
      WHERE deptno = 10;
BEGIN
  ……
```

فتح المشيرة

يستخدم هذا الأمر لفتح المشيرة وتنفيذجملة الاستعلام ومن ثم تحديد المجموعة النشطة التي تحتوي على كل الاسطر التي تحقق شروط جملة الاستعلام، ويعرف مؤشر يشير إلى السطر الأول في المجموعة النشطة.

ملحوظة: في حال أن جملة الاستعلام لم تعط أي سطر عند تنفيذها فإن PL/SQL لا تحدث استثناء Exception، ويمكنك فحص حالة المشيرة بعد جملة Fetch.

```
OPEN Cursor_name;
```

جلب البيانات من المشيرة:

يقوم أمر Fetch بجلب البيانات من سطر من المجموعة النشطة واحداً تلو الآخر. بعد كل عملية جلب للبيانات يتم نقل المؤشر إلى السطر التإلى في المجموعة.

```
FETCH cursor_name INTO [variable1, variable2, …] | record_name];
```

عند استخدامك لأمر Fetch لا بد من مراعاة مايلي:

١. استخدم عدداً من المتغيرات في جملة INTO داخل أمر Fetch مساو لعدد الأعمدة المحددة في جملة الاستعلام Select وتأكد من أن أنواع البيانات متوافقة.
٢. يمكنك تعريف سجل Record للمشيرة واستخدامه في جملة FETCH INTO.
٣. افحص المشيرة للتاكد من وجود أسطر متبقية لجلبها، كما سيرد ذكره لاحقا.

عند تنفيذ الأمر Fetch يتم عمل ما يلي:

١. نقل المؤشر إلى السطر التإلى بالمجموعة النشطة
٢. قراءة البيانات من السطر الحإلى إلى المتغيرات المحددة بالجملة.
٣. الخروج من جملة تكرار المشيرة FOR loop إذا كان المؤشر يشير إلى نهاية المجموعة النشطة.

مثال:

```
FETCH emp_cursor  INTO v_empno, v_ename;
```

اغلاق المشيرة:

يستخدم أمر إغلاق المشيرة لتعطيل المشيرة وإخلاء المجموعة النشطة من الذاكرة، وعليك إغلاق المشيرة بعد الانتهاء من عمل المعالجة اللازمة للمجموعة النشطة. ويسمح إغلاق المشيرة بإعادة فتحها عند الحاجة. ويجب عليك عدم محأولة جلب البيانات من المشيرة بعد إغلاقها لأن ذلك سيتسبب بحوث استثناء من نوع INVALID_CURSOR.

على الرغم من إمكانية إنهاء قطعة PL/SQL التي تعرف مشيرة بداخلها دون إغلاق المشيرة ودون التسبب بأي أخطاء إلا أنه يفضل أن تعود نفسك على إغلاق كل مشيرة يتم تعريفها بعد الانتهاء من المعالجه وذلك أجل إخلاء المصادر (الذاكرة) حتى يتسنى استخدامها لاحقا لتنفيذ عمليات جديدة.

ملحوظة: تذكر أنه يوجد هناك حد أقصى للمشيرات الممكن فتحها في نفس الوقت لنفس المستخدم، ويحدد هذا العدد في باراميتر قاعدة البيانات المسمى OPEN_CURSORS حيث إن الفيمة المسبقة له هي ٥٠.

```
...
 FOR I IN 1—10 LOOP
    FETCH  emp_cursor INTO v_empno, v_ename;
     ...
 END LOOP
 CLOSE emp_cursor;
END;
```

خصائص المشيرة التصريحية

كما هو موجود في المشيرة الضمنية، هناك أربع خصائص للحصول على معلومات تخص حالة المشيرة. عند إضافتها لاسم المشيرة هذه الصفات تعطي معلومات مفيدة عن تنفيذ جملة معالجة البيانات.
ملحوظة: لا يمكنك الرجوع إلى خصائص المشيرة مباشرة في جملة الاستعلام.

الخاصية	نوع الفيمة العائدة	الوصف
%ISOPEN	True/False	تشير فيما إذا كانت المشيرة مفتوحة أم مغلقة.
%NOTFOUND	True/False	تشير فيما إذا لم يتم إيجاد السطر في أخر عملية جلب
%FOUND	True/False	تشير فيما إذا تم إيجاد السطر في أخر عملية جلب
%ROWCOUNT	Number	يحسب العدد الكلي للأسطر المرتجعه حتى الآن

خاصية %ISOPEN

يمكنك جلب الأسطر فقط عندما تكون المشيرة مفتوحة. يمكنك استخدام خاصية المشيرة %ISOPEN، عند الضروره، لتحديد فيما إذا كانت المشيرة مفتوحة. حيث إن الفيمة TRUE تعني أن المشيرة مفتوحة. وغالبا فإنه ليس من الضروري فحص ما إذا كانت المشيرة مفتوحة.

مثال:

```
IF NOT emp_cursor%ISOPEN THEN
        OPEN emp_cursor;
END IF;
LOOP
   FETCH emp_cursor; …
```

خاصية %NOTFOUND و %ROWCOUNT.

- استخدام الخاصية %ROWCOUNT لاسترجاع عدد الأسطر التي تم جلبها حتى الآن.
- استخدام الخاصية %NOTFOUND لتحديد متى الخروج من حلقة التكرار.

مثال:

```
DECLARE
  v_empno    emp.empno%TYPE;
   v_ename    emp.ename%TYPE;
   CURSOR emp_cursor IS
         SELECT empno, ename
          FROM emp;
    BEGIN
    OPEN emp_cursor;
    LOOP
         FETCH emp_cursor INTO v_empno, v_ename;
         EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                        Emp_cursor%NOTFOUND;
         …
    END LOOP;
    CLOSE emp_cursor;
END;
```

قبل أول عملية جلب للبيانات، %NOTFOUND تكون فيمتها NULL وبالتإلى فإن جملة الجلب لن تنفذ بنجاح وعليه فإن حلقة التكرار لن تنتهي وذلك لأن الشرط بجملة WHEN لن ينتج الفيمة TRUE ، ولتفادي هذه الحالة يمكن كتابة الشرط بالشكل التإلى:

```
EXIT WHEN emp_curor%NOTFOUND
            OR emp_cursor%NOTFOUND IS NULL;
```

المشيرات والسجلات Cursors and Records

من المعلوم أن السجل هو عبارة عن هيكلية تحتوي على مجموعة من الأعمدة في جدول ما. ويمكنك تعريف سجل اعتمادا على قائمة الأعمدة الموجودة في المشيرة. إن هذه الطريقة مناسبة جدا لمعالجة الأسطر في المجموعة النشطة، حيث تقوم بجلب البيانات إلى السجل بسهولة وتحميلها في الحقول المناسبة داخل السجل .

المثال الموضح في الصفحه التالية يوضح كيفية تعريف واستخدام المشيرة في قطعة PL/SQL البرمجية. المثال يقوم باسترجاع رقم الموظف واسمه وإدخالهما في جدول مؤقت في قاعدة البيانات .

```
DECLARE CURSOR emp_cursor IS
        SELECT empno, ename
        FROM emp;
  Emp_record  emp_cursor%ROWTYPE;
  BEGIN
     OPEN emp_cursor;
     LOOP
          FETCH emp_cursor INTO emp_record;
          EXIT WHEN emp_cursor%NOTFOUND;
          INSERT INTO temp_list(empid,empname)
          VALUES (emp_record.empno, emp_record.ename);
     END LOOP;
    COMMIT;
     CLOSE emp_cursor;
 END;
```

المشيرة في حلقات FOR.

إن استخدام المشيرة في حلقات التكرار FOR هي الطريقة الأسهل في استخدام المشيرات، حيث يتم فتح، جلب البيانات وإغلاق المشيرة بشكل إلى ضمن نطاق حلقة التكرار FOR. يتم تعريف المتغير الخاص بحلقة التكرار بشكل إلى كسجل من نفس نوع وتركيبة سجلات المشيرة لتخزين السجل المرتجع عند كل عملية جلب للبيانات، ولا يمكن الإشارة إليه خارج نطاق حلقة التكرار. وإليك المثال التإلى:

```
SET SERVEROUTPUT ON
DECLARE
    CURSOR emp_cursor IS
           SELECT ename, deptno
           FROM emp;
BEGIN
   FOR emp_record IN emp_cursor LOOP
                فتح وجلب للبيانات بشكل إلى (ضمني) --
     IF emp_record.deptno = 30 THEN
         DBMS_OUTPUT.PUT_LINE (‘Employee ‘ ||
```

```
                emp_record.ename || ' works in the Sales Dept. ');
      END IF;
   END LOOP; --إغلاق ضمني للمشيرة
END; /
```

المشيرة في حلقات التكرار FOR باستخدام الاستعلام الفرعي

يمكنك الاستعاضة عن المشيرة باستخدام الاستعلام الفرعي، حيث تتم كتاية جملة الاستعلام داخل جملة التكرار، وتؤدي هذه الطريقة نفس عمل المشيرة المعرفة في الجزء الخاص بالتعريف. وإليك المثال التإلى وهو نفس المثال السابق ولكن دون تعريف المشيرة، واستخدام أسلوب الاستعلام الفرعي:

```
SET SERVEROUTPUT ON
BEGIN
   FOR emp_record IN (SELECT ename, deptno
                      FROM emp) LOOP
            -- فتح وجلب للبيانات بشكل إلى (ضمني)
      IF emp_record.deptno = 30 THEN
         DBMS_OUTPUT.PUT_LINE ('Employee ' ||
               emp_record.ename || ' works in the Sales Dept. ');
      END IF;
   END LOOP; --إغلاق ضمني للمشيرة
END; /
```

تمارين:

س١) اكتب الجملة التاليه لإنشاء جدول جديد لتخزين الموظفين ورواتبهم.

```
SQL> CREATE TABLE top_dogs
       (name          VARCHAR2(25),
        salary        NUMBER(11,2) );
```

س٢) أنشئ قطعة PL/SQL تحدد أعلى موظفين حسب رواتبهم.

١. استقبل عدد n كمدخل من المستخدم بواسطة المتغيرات التعويضية substitution variables.
٢. في حلقة التكرار، خذ الاسم الأخير والراتب لأعلى n موظف حسب الرواتب من جدول EMP.
٣. خزن الأسماء والرواتب في جدول Top_dogs.
٤. افرض انه لا يوجد موظفان اثنان لهما نفس الراتب.

```
Please enter the number of  top money makers: 5
NAME                SALARY
KING                  5000
FORD                  3000
SCOTT                 3000
JONES                 2975
BLAKE                 2850
```

س٣) خذ الآن بعين الاعتبار بأنه قد يكون هناك موظفون لهم نفس الراتب، إذا ظهر واحد منهم فيجب إظهار البافين، مثلا:

١. إذا أدخل المستخدم العدد ٢ فإنه يجب إظهار King, Ford, Scott.
٢. إذا أدخل المستخدم العدد ٣ فإنه يجب إظهار King, Ford, Scott and Jones.
٣. أعد كتابة القطعة البرمجية في السؤال الثاني لأخذهذه المسألة بالحسبان.

```
Please enter the number of top money makers: 2
NAME                SALARY
KING                  5000
FORD                  3000
SCOTT                 3000
```

```
Please enter the number of top money makers: 3
NAME                SALARY
KING                  5000
FORD                  3000
SCOTT                 3000
JONES                 2975
```

برمجة قواعد بيانات

معالجة الاستثناءات

الجدارة:

أن يكون المتدرب قادراً على استيعاب مفهوم الاستثناءات، والتمييز بين الأنواع المختلفة منها. كما يجب على المتدرب إتقان مهارة كتابة الأجزاء الخاصة بمعالجة الاستثناءات.

الأهـداف:

بنهاية هذه الوحدة، عليك أن تكون قادرا على:

١. تعريف الاستثناءات في القطع البرمجية
٢. التعرف على الاستثناءات الممكن حدوثها
٣. الكشف عن الاستثناءات ومعالجتها
٤. وصف تأثير انتقال الاستثناءات في القطع البرمجية المتداخلة

مستوى الأداء المطلوب:

أن يصل المتدرب إلى إتقان الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ٤ ساعات معتمدة

الوسائل المساعدة:

- وجود حاسب آلي
- وجود عارض شرائح Projector
- دفتر
- قلم

الوحدة الثالثة

معالجة الاستثناءات

مقدمة:

الاستثناء هو معرف في PL/SQL، ينطلق الاستثناء خلال تنفيذ القطعة البرمجية عند وقوع حدث ما ويؤدي ذلك إلى إنهاء عمل الجزء الرئيس للبرنامج وينتقل إلى جزء معالجة الاستثناءات إذا كان معرفا، اذا عند حدوث استثناء في قطعة برمجية فإن ذلك يؤدي إلى إنهاء القطعة مباشرة.

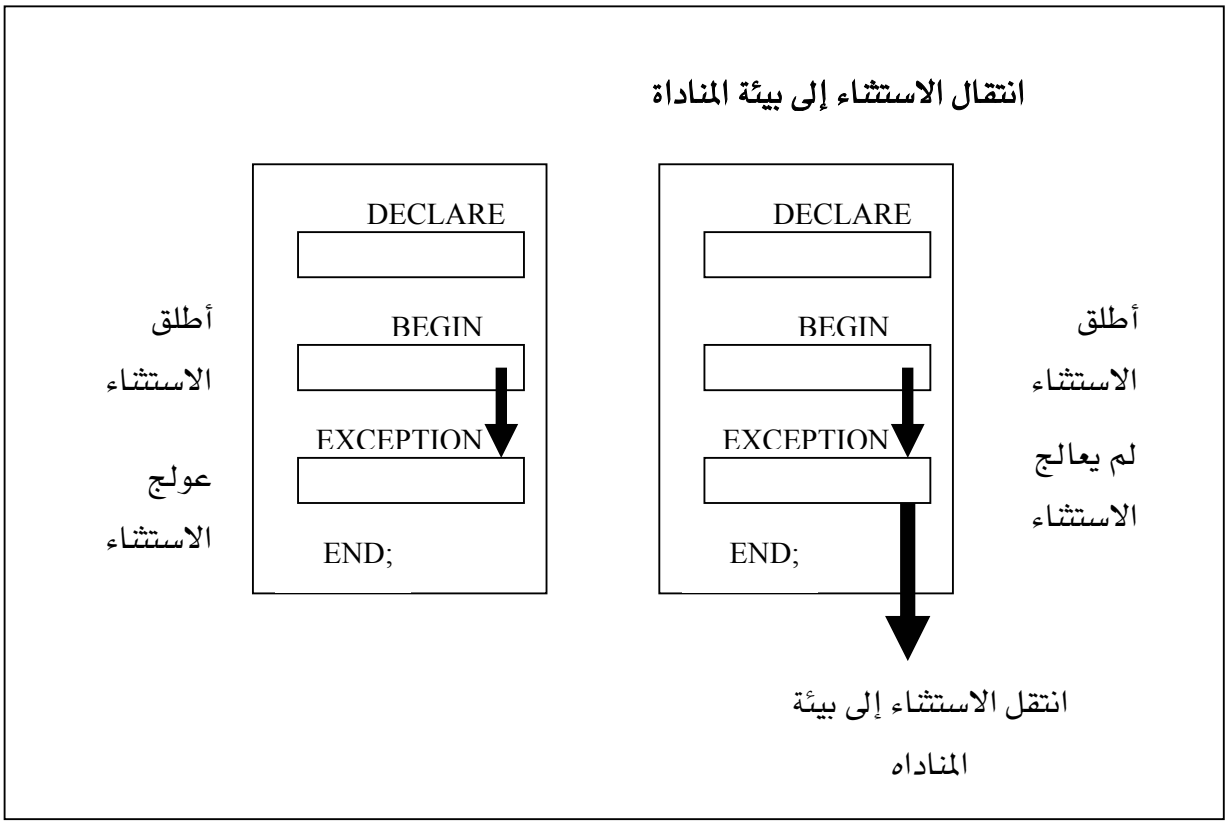
كيف يمكن للاستثناء أن ينطلق؟

يمكن أن ينطلق الاستثناء بإحدى طريقتين، هما:

١. عند حدوث خطأ في قاعدة البيانات، يتم إطلاق الاستثناء المسؤول عنه إلىا. مثلا الخطأ ORA-01403 يحدث عندما لا يكون هناك أي سطر مرتجع بعد تنفيذ جملة الاستعلام ففي هذه الحاله تقوم PL/SQL بإطلاق استثناء هو NO-DATA-FOUND.

٢. يمكن إطلاق الاستثناء بشكل تصريحي من قبل المبرمج باستخدام جملة RAISE داخل القطعه البرمجية، وقد يكون الاستثناء المنطلق معرفاً مسبقا في قاعدة البيانات أو معرفاً من قبل المستخدم.

عند حدوث استثناء في الجزء التنفيذي من القطعة البرمجية فإنه يتم انتقال المعالجة إلى الجزء الخاص بمعالجة الاستثناءات، وفي حال أن القطعة البرمجية نجحت بمعالجة الاستثناء فإنه لا يتم نقله إلى القطعة أو البلك المحتوي للقطعة المتسببة في الاستثناء(Enclosing Block)، ويتم إنهاء القطعة البرمجية بنجاح. أما في حالة أن الاستثناء لا يوجد له معالج مناسب في جزء معالجة الاستثناءات فإنه يتم إنهاء هذه القطعة بالفشل Failure ويتم نقل الاستثناء إلى البيئة أو القطعة البرمجية التي قامت بمناداة القطعة المسببة له.



انواع الاستثناءات

هناك ثلاثة انواع رئيسة من الاستثناءات هي:

١. الاستثناءات المعرفة مسبقا في قاعدة البيانات

هذه الاستثناءات لا تعرف وتقوم قاعدة البيانات بإطلاقها بشكل ضمني.

٢. الاستثناءات غير المعرفة في قاعدة البيانات

يتم تعريفها في الجزء الخاص بالتعريف في القطعة البرمجية وتنطلق بشكل ضمني.

٣. الاستثناءات المعرفه من قبل المستخدم

يتم تعريفها في الجزء الخاص بالتعريف في القطعة البرمجية وتنطلق بشكل تصريحي.

تتم عملية اصطياد الاستثناء ومعالجته في جزء معالجة الاستثناءات، والشكل العام لهذا الجزء هو:

```
EXCEPTION
   WHEN exception1 [ OR exception2 . . . ] THEN
         Statement1;
         Statement2;
  [WHEN exception3 [ OR exception4 . . . ] THEN
         Statement1;
         Statement2;
         .   .   . ]
  [WHEN OTHERS THEN
         Statement1;
         Statement2;
         .   .   . ]
```

لاحظ استخدام جملة WHEN OTHERS في نهاية جزء معالجة الاستثناءات، حيث إنها جملة اختيارية يمكن استخدامها للإيقاع بكافة الاستثناءات التي لم يتم تحديدها بالاسم في هذا الجزء.

الإيقاع بالاستثناءات

الإيقاع بالاستثناءات المعرفة مسبقا في قاعدة البيانات

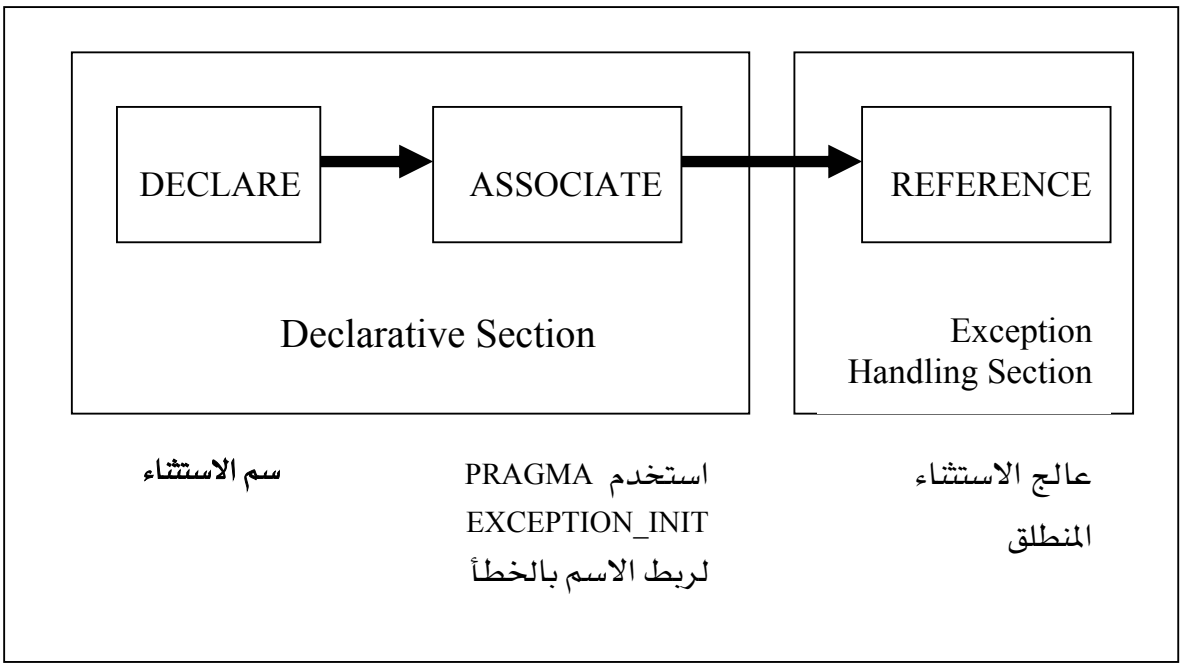
للإيقاع بالاستثناءات المعرفة في قاعدة البيانات نستخدم اسم الاستثناء في جزء المعالجة الخاص بذلك، ومن اهم هذه الاستثناءات ما يلي:

١. NO_DATA_FOUND
٢. TOO_MANY_ROWS
٣. INVALID_CURSOR
٤. ZERO_DIVIDE
٥. DUP_VAL_ON_INDEX

تقوم PL/SQL بتعريف الاستثناءات المعلومة مسبقا في قاعدة البيانات ضمن الحزمة الفياسية Standard Package، ويفضل أخذ الاستثناءين الأولين أعلاه NO_DATA_FOUND و TOO_MANY_ROWS بعين الاعتبار دائما حيث إنهما الأكثر شيوعا ومعالجة.

الإيقاع بالاستثناءات غير المعرفة مسبقا في قاعدة البيانات

يوجد العديد من الأخطاء الممكن حدوثها في قاعدة البيانات غير معرفة كاستثناءات، يمكنك تعريف الاستثناءات لهذه الأخطاء واعطائها أسماء حتى تتمكن من الكشف عنها والإيقاع بها لمعالجتها، أو يمكنك استخدام المعالج OTHER للإيقاع بكل الاستثناءات غير المعرفة. أما بالنسبه لإطلاق هذا النوع من الاستثناءات فإنه يتم تلقائيا مثله مثل الاستثناءات المعرفة في قاعة البيانات.



تستخدم الكلمة المحجوزة PRAGMA لربط اسم الاستثناء الجديد برقم خطأ معين، تعتبر هذه الكلمة موجهة للمترجم للفيام بتحويل كل استخدام لاسم الاستثناء في البرنامج إلى رقم الخطأ المحدد في جملة PRAGMA.

مثال:

```
DECLARE
   e_emps_remaining      EXCEPTION;
   PRAGMA_EXCEPTION_INIT (e_emps_remaining, -2292) ;
   v_deptno                  dept.deptno%TYPE := &p_deptno ;
BEGIN
    DELETE FROM dept
    WHERE deptno = v_deptno;
    COMMIT;
EXCEPTION
     WHEN e_emps_remaining THEN
         DBMS_OUTPUT.PUT_LINE (‘Cannot remove dept ‘||
                                  TO_CHAR (v_deptno) || ‘. Employees exist ‘);
END;
```

الإيقاع بالاستثناءات المعرفة من قبل المستخدم.

تمكن PL/SQL المستخدمين من تعريف الاستثناءات الخاصة بهم، ويجب في هذه الاستثناءات ما يلي:

١. تعريفها في الجزء الخاص بالتعريف

٢. إطلاقها بشكل تصريحي باستخدام جملة RAISE.

```
DECLARE
      e_invalid_product       EXCEPTION;
BEGIN
      UPDATE product
      SET descrip = ‘&product_description’
      WHERE prodid = &product_number;
      IF SQL%NOTFOUND THEN
            RAISE e-invalid_product;
     END IF;
     COMMIT;
EXCEPTION
     WHEN e_invalid_product THEN
           DBMS_OUTPUT.PUT_LINE (‘Invalid product number. ‘);
END;
```

دوال للإيقاع بالاستثناءات

عند حدوث استثناء، يمكنك تحديد رقم الخطأ الحاصل ورسالة الخطأ أيضا باستخدام دالتين هما:

١. SQLCODE وتعيد فيمة عددية تمثل رقم الخطأ

٢. SQLERRM وتعيد بيانات حرفية تحتوي رسالة .

```
DECLARE
      v_error_code        number;
      v_error_message  varchar2(255);
BEGIN
. . .
exception
WHEN OTHERS THEN
    ROLLBACK;
    v_error_code := SQLCODE;
    v_error_message := SQLERRM;
     INSERT INTO errors VALUES (v_error_code, v_error_message);
END;
```

انتقال الاستثناءات من قطعة برمجية إلى أخرى

عند حدوث الاستثناء في قطعة برمجية ما ومعالجته في نفس القطعة، ينتهي تنفيذ القطعة بشكل طبيعي وينتقل التنفيذ إلى البلك الذي قام بمناداة هذه القطعة. أما في حال أن القطعة التي تسببت بالاستثناء لا تحتوي على المعالج المناسب له فإنه يتم انتقاله إلى البلك الذي يحتوي على هذه القطعة البرمجية بداخله Enclosing Block وتستمر العملية هكذا.

```
DECLARE
. . .
e_no_rows exception;
e_integrity exception;
PRAGMA EXCEPTION_INIT (e_integrity, -2292);
BEGIN
FOR c_reocrd IN emp_cursor LOOP
    BEGIN
    SELECT …
    UPDATE …
    IF SQL%NOTFOUND THEN
    RAISE e_no_rows;
    END IF;
    EXCEPTION
    WHEN e_inventory THEN …
    WHEN e_no_rows THEN …
    END;
END LOOP;
```

في المثال أعلاه لو افترضنا أن جملة الاستعلام لم تعد أي سطر، سيتسبب ذلك بانطلاق الاستثناء NO_DATA_FOUND، لكن وكما تلاحظ فإن جزء معالجة الاستثناءات في البلك الداخلي لا يحتوي على معالج لهذا الاستثناء، في هذه الحالة سينتهي البلك الداخلي بالفشل وينتقل الاستثناء إلى البلك الذي يليه(بالترتيب من الداخل إلى الخارج) ليبحث عن المعالج المناسب لهذا الاستثناء وهكذا.

الروتين RAISE_APPLICATION_ERROR.

يمكنك استخدام الروتين RAISE_APPLICATION_ERROR لإرسال رقم الخطأ والنص المناسب للتطبيق الذي يستخدم هذه القطعة البرمجية.

```
…
EXCEPTION
   WHEN NO_DATA_FOUND THEN
          RAISE_APPLICATION_ERROR (-20201,
                                        'Manager is not a valid employee.');
END;
```

ويمكن استخدام هذا الروتين داخل الجزء التنفيذي من القطعة البرمجية، عالى الشكل التإلى:

```
…
DELETE FROM emp
WHERE  mgr = v_mgr;
IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR (-20202, 'This is not a valid manager');
END IF;
…
```

تمارين

س١) اكتب بلك PL/SQL لاسترجاع اسم الموظف صاحب راتب معين.

١. إذا الراتب المدخل تسبب في استرجاع أكثر من سطر، عالج الاستثناء الناتج بالمعالج المناسب وأدرج في جدول MESSAGES الرسالة "هنالك أكثر من موظف لهم راتب <Salary>
٢. إذا الراتب المدخل لم يسترجع أي سطر عالج الاستثناء الناتج بالمعالج المناسب وإدرج في جدول MESSAGES الرسالة "لا يوجد موظفون لهم رواتب <Salary>.
٣. إذا الراتب المدخل يسترجع سطراً واحداً فقط، أدرج في جدول MESSAGES اسم الموظف وراتبه.
٤. عالج أي استثناء آخر بالمعالج المناسب وأدرج في جدول MESSAGES الرسالة "حدث هناك خطأ ما".
٥. قم بتجريب الحالات المختلفة في البرنامج المكتوب وتفحص محتويات الجدول MESSAGES.

RESULTS
محمد – ٢٠٠٠
هناك أكثر من موظف له راتب ٥٠٠
لا يوجد موظفون لهم رواتب ٣٥٠٠

س٢) اكتب بلك PL/SQL يقوم بتعديل موقع دائرة معينة في جدول الدوائر DEPT بالطلب من المستخدم إدخال رقم الدائرة وموقع الدائرة الجديد باستخدام المتغيرات التعويضية Substitution variables.

١. اكتب معالج استثناء في حالة أن رقم الدائرة المدخل غير موجود يقوم بتمرير رسالة للمستخدم مفادها أن هذه الدائرة غير موجودة.
٢. نفذ بلك PL/SQL وأدخل رقم دائرة غير موجود فعلا.

Please enter department number: 50 Please enter department location: Huston PL/SQL procedure successfully Completed G_MESSAGE
الدائرة ٥٠ ليست موجودة في الجدول

س٣) اكتب بلك PL/SQL يقوم بطباعة عدد الموظفين الذين يكسبون نفس فيمة الراتب المدخل زائد أو ناقص ١٠٠$.

١. إذا لم يكن هناك أي موظف بالمدى المحدد ، اطبع رسالة لمستخدم تخبره بهذه الحاله.
٢. إذا كان هناك موظف أو أكثر بالمدى المحدد ، اطبع رسالة توضح عدد الموظفين بهذا الراتب.
٣. إذا حدث أي استثناء آخر عالجه بالمعالج المناسب واطبع رسالة توضح أن هناك خطأ ما.

Please enter the salary: 800
PL/SQL procedure successfully completed.
G_MESSAGE

يوجد هناك عدد موظفين (١)براتب بين ٧٠٠ و ٩٠٠

Please enter the salary: 3000
PL/SQL procedure successfully completed.

G_MESSAGE

يوجد هناك عدد موظفين(٣) براتب بين ٢٩٠٠ و ٣١٠٠

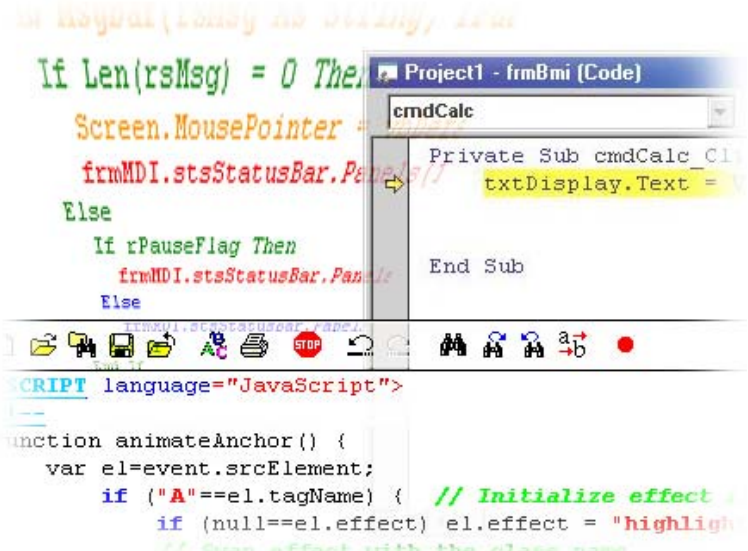
Please enter the salary: 6000
PL/SQL procedure successfully completed.

G_MESSAGE

لا يوجد مووفين براتب بين ٥٩٠٠ و ٦١٠٠

برمجة قواعد بيانات

القطع البرمجية



الجدارة:

أن يكون المتدرب قادراً على التمييز بين الأنواع المختلفة للقطع البرمجية في المعنى وفي طريقة الكتابة، وأن يكون قادرا على تعداد الفوائد لهذه القطع البرمجية.

الأهـداف:

بنهاية هذه الوحدة، عليك أن تكون قادرا على:

١. تعريف مفهوم القطع البرمجية
٢. تعداد الأنواع المختلفة للقطع البرمجية
٣. التمييز بين الأنواع المختلفة للقطع البرمجية
٤. معرفة أسس كتابة كل نوع من القطع البرمجية
٥. تحديد فؤائد استخدام القطع البرمجية

مستوى الأداء المطلوب:

أن يصل المتدرب إلى إتقان الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ٤ ساعات معتمدة

الوسائل المساعده:

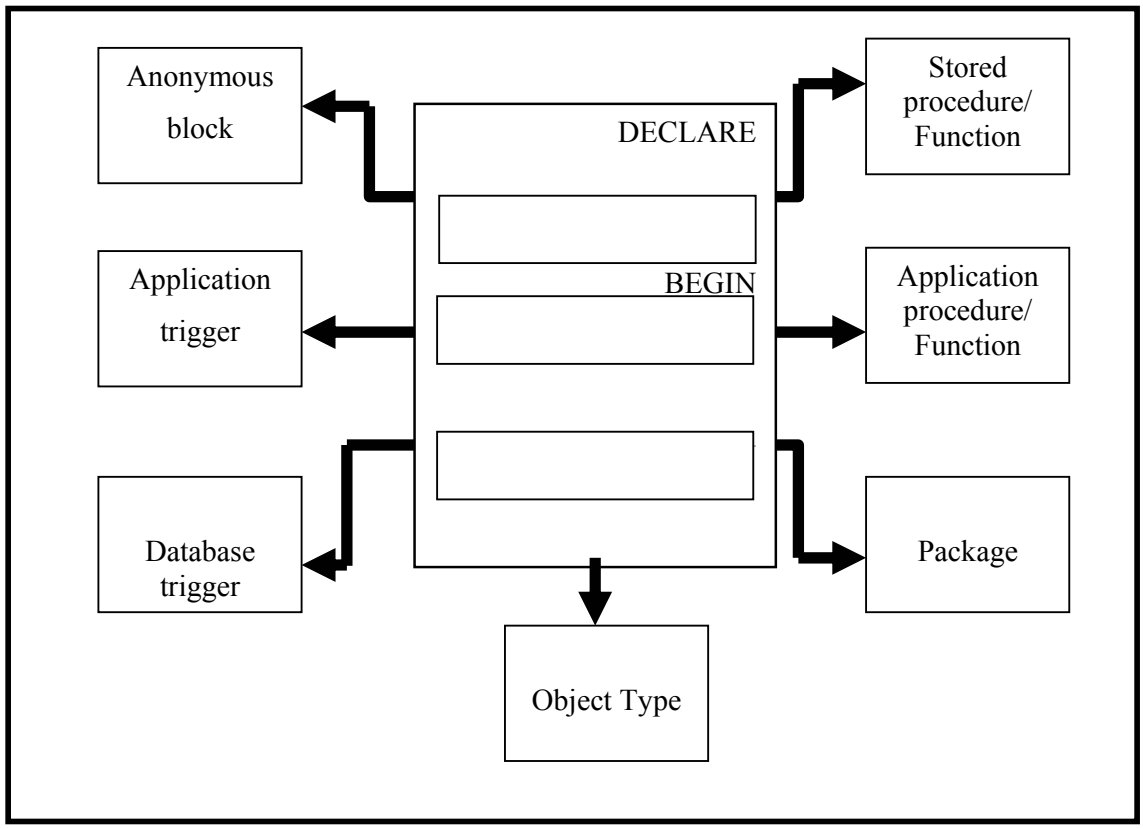
- وجود حاسب آلي
- وجود عارض شرائح Projector
- دفتر
- قلم

الوحدة الرابعة :

القطع البرمجية Program Constructs

مقدمة :

سنتعلم في هذا الدرس كيفية تعريف واستخدام الوحدات البرمجية Program Modules، إن القطع البرمجية طريقة لبناء البرنامج من مجموعات منفصلة من الوحدات البرمجية Modules، كل منها يقوم بعمل وظيفة أو مهمة محددة باتجاه الوصول إلى الهدف النهائي في البرنامج، وعندما يتم الانتهاء من كتابة الوحدات البرمجية وتخزينها في خادم قاعدة البيانات Database Server تصبح هذه الوحدات كائنات في قاعدة البيانات بحيث يمكن استخدامها من قبل أي وحدة برمجية في قاعدة البيانات هذه. ولتخزين الوحدات البرمجية في قاعدة البيانات يجب إرسال البرنامج المصدري Source Code إلى خادم قاعدة البيانات ليتم ترجمته Compile إلى لغة انتقاليه تسمى P-Code. ويوضح الرسم أدناه الأنواع المختلفة للقطع البرمجية المختلف':



أقسام الوحدات البرمجية الجزئية

وبشكل عام فإنه يمكننا القول ان الوحدات البرمجية يمكن تقسيمها إلى قسمين أساسيين هما:

١. وحدة برمجية مجهولة Anonymous Block: وهي الوحدات البرمجية التي ليس لها اسم محدد. ولا يمكن تخزينها في قاعدة البيانات ولكن يتم تحميلها في الذاكرة وتنفيذها عند الحاجة لها.

٢. وحدة برمجية معروفة Named Block: وتسمى أحيانا Subprograms، وهي الوحدات البرمجية التي لها اسم محدد عند تعريفها ويندرج ضمنها: Function, Procedure, Trigger, Package وهي كلها لها اسماء محدده.

ويمكن للقطعة ابرمجية أن تحتوي على وحدة برمجية واحده أو اكثر وبالتإلى يمكن ان توجد الوحدات البرمجية بداخل بعضها Nested Blocks.

وحدة برمجية مجهولة Anonymous PL/SQL Block

كما أسلفنا سابقا فإن هذه الوحدات لا تمتلك اسما، وتعرف عند النقطة أو المرحلة في البرنامج عندما نكون بحاجة لتنفيذها، ويتم تمريرها إلى محرك PL/SQL لتنفيذها في وقت التنفيذ Execution time، والهيكل العام لهذه الوحدة هو كالتإلى:

```
DECLARE        (Optional اختياري)
        تعريف الكائنات التي ستستخدم في هذه القطعة البرمجية
BEGIN          (Mandatory إجباري)
كتابة الجمل التنفيذية للفيام بعمل معين
EXCEPTION      (Optional اختياري)
تعريف ما سيتم فعله عند حدوث خطأ ما
END;           (Mandatory إجباري)
```

كما هو ملاحظ، فإن هذه الوحدة كأي وحدة برمجية تحتوي على ثلاثة أجزاء هي: -

أ. قسم التصريح Declaration section

ويبدأ هذا القسم بالكلمة المحجوزة Declare وينتهي عند بداية جزء التعليمات أي عند الوصول للكلمة المحجوزة Begin. ويعتبر هذا الجزء اختياري، حيث يتم تعريف المتغيرات، الثوابت، المؤشرات والاستثناءات المعرفة من قبل المستخدم في هذا الجزء.

ب. قسم التنفيذ Executable Section

وهو القسم الذي يتم فيه كتابة التعليمات التي ستنفذ للفيام بعمل معين ويمثل الجسم الرئيس للوحدة البرمجية، ويبدأ هذا القسم بالكلمة المحجوزة Begin وينتهي بالكلمة المحجوزة End وهذه الكلمات إجبارية وليست اختيارية.

ت. قسم الاستثناءات Exception Section

ويبدأ هذا القسم بالكلمة المحجوزة Exception وهي اختيارية ويستخدم هذا الجزء لمتابعة الأخطاء الممكن حدوثها أثناء تنفيذ الجزء التنفيذي حيث يتم الإيقاع بها ومعالجتها في هذا الجزء.

وحدات برمجية ذات اسم Named PL/SQL Block

تختلف هذه الوحدات البرمجية عن سابقتها بأن لها اسماً يميزها عن غيرها ويمكن أن تستقبل مدخلات باراميتر Parameters ويمكن استدعاؤها من خلال هذا الاسم، وتحتوي لغة PL/SQL على نوعين من هذه الوحدات هما الاقتران Function، الروتيProcedure .

```
Header
IS
   Declaration section
BEGIN
   Executable Section
Exception      (optional)
   Exception section
END;
```

كما هو ملاحظ فإن هذه الوحدات البرمجية الجزئية ذات الاسم، وتسمى أيضا Subprograms، تحتوي على أجزاء أربعه، فهي تتشابه مع الوحدات البرمجية المجهولة في كل الأجزاء باستثناء أن لها جزءاً جديداً هو جزء الترويسية أو Header، ويحتوي هذا الجزء على مايلي:

أ. نوع الوحدة البرمجية الجزئيه هل هي Procedure أو Function.

ب. اسم الوحدة البرمجية الجزئية

ت. قائمه بالباراميترات للوحدة، إن وجدت.

ث. جملة الاسترجاع Return والتي تنطبق فقط على الاقترانات Functions.

ويلي جزء الترويسية كلمة IS وهي كلمة محجوزة وتجب كتابتها عند تعريف الوحدة وما يلي كلمة IS هو نفس ما تم شرحه في موضوع الوحدات المجهولة Anonymous Blocks.

ما هو الفرق بين الوحدات البرمجية الجزئية المجهولة Anonymous Blocks والوحدات البرمجية الجزئية ذات الاسم Subprograms؟.

إنه وكما يمكن استنتاجه من اسم الوحدة المستخدمة فإن الوحدات المجهولة ليس لها اسم وبالتإلى لا يمكن مناداتها من وحدة أخرى ولا تخزن في قاعدة البيانات ككائن مستقل ويجب تحميلها من الملف وترجمتها قبل عملية تنفيذها.

فؤائد الوحدات البرمجية الجزئية ذات الاسم Advantages of Subprograms.

١. إمكانية إعادة الاستخدام Improve Reusability.

ان تقسم المسألة إلى مجموعة من الوحدات البرمجية التي تقوم كل منها بعمل محدد وربطها مع بعضها لتحفيق حل للمسألة الكاملة، يمكننا من إعادة استخدام هذه الوحدات المكتوبة مسبقا في حل مسائل اخرى.

٢. تحسين فعإلية الصيانة Improve maintenance.

إذا وجدنا على سبيل المثال أن إحدى الوحدات البرمجية المكتوبة يمكن اعادة كتابتها بطريقة تزيد من كفاءتها أو أن بها خطأ ما، فإنه ليس علينا سوى الذهاب لهذه الوحدة وإعادة كتابتها من جديد أو تصحيح الخطأ المنطفي بها من غير أن نغير أياً من الوحدات الأخرى أو أياً من التطبيقات التي تستخدم هذه الوحدة، وبالتإلى اصبحت عملية صيانة الوحدة البرمجية أسهل واحتمإلية حدوث أخطاء أقل.

٣. تحسين الكفاءة Improve Performance.

- ان أي وحدة برمجية مكتوبة ومترجمة في خادم قاعدة البيانات يمكن لأي مستخدم أن يستخدمها دون الحاجة إلى إعادة ترجمتها وبالتإلى تقليل وقت التنفيذ.
- إن استخدام الوحدات البرمجية يساعد على تقليل عدد مرات المناداة لخادم قاعدة البيانات وبالتإلى تقليل الضغط على الشبكة Network Traffic

أين يمكننا كتابة أو تعريف الوحدات البرمجية الجزئية المختلفة؟

يمكننا تعريف الوحدات البرمجية على خادم قاعدة البيانات Database Server وذلك باستخدام SQL*Plus. أو على مستدعي قاعدة البيانات Database Client ونستخدم هنا Procedure Builder وهي أداة يمكن استخدامها داخل تطبيق باني النماذج Forms Builder.

وسندرس الآن كلاً من الوحدات البرمجية Functions و Procedures كلا على حدة.

برمجة قواعد بيانات

الروتين

الجدارة:

أن يكون المتدرب قادراً على كتابة روتين محدد في قاعدة البيانات للفيام بمهمة محددة، وتحديد عدد ونوع البارميترات التي يحتاجها الروتين.

الأهــداف:

بنهاية هذه الوحدة، عليك أن تكون قادرا على:

١. تعريف مفهوم الروتين
٢. تحديد مراحل بناء الروتين
٣. التمييز بين الأنواع المختلفة لباراميترات الروتين
٤. التمييز بين الطرق المختلفة لتمرير الباراميترات
٥. كتابة روتين للفيام بمهمة محددة

مستوى الأداء المطلوب:

أن يصل المتدرب إلى إتقان الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ٥ ساعات معتمدة

الوسائل المساعدة:

- وجود حاسب آلي
- وجود عارض شرائح Projector
- دفتر
- فلم

الروتين Procedure

يعرف الروتين على أنه وحدة برمجية جزئية ذات اسم تقوم بعمل وظيفة معينة وليس بالضرورة أن يكون لها فيمة مرتجعة Return value وتتلخص صيغة تعريف Procedure كما يلي:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
    [ (Parameter1 [,Parameter2, …] ) ]
AS
    [ Local Declaration ]
Begin
      Executable Statements
[Exception
      exception handlers ]
END [procedure_name]
```

يمكن أن يكون للروتين Procedure صفراً أو أكثر من الباراميترات، وكما هو موضح أعلاه فإن الروتين له جزئان رئيسان هما:

١. الترويسية Header

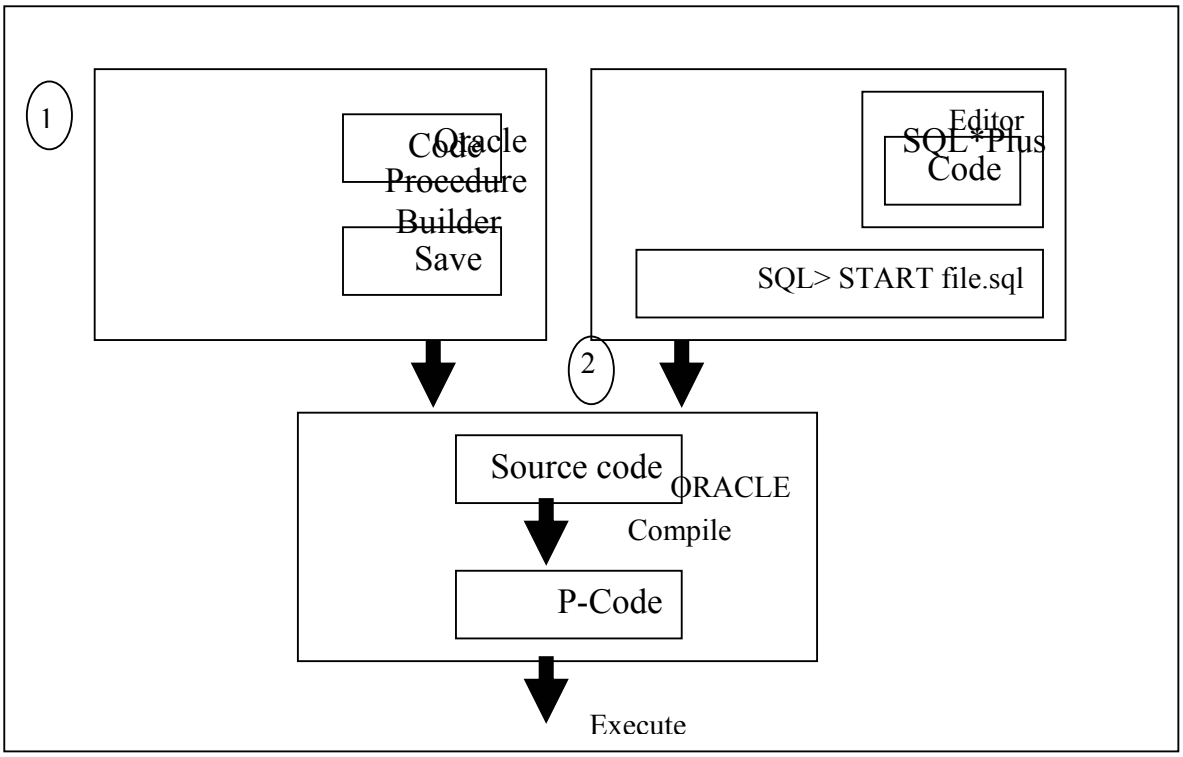
ويأتي هذا الجزء قبل كلمة AS أو IS ويعرف فيه الاسم والمدخلات للوحدة البرمجية

٢. الجسم Body

وهو كل شيء بعد كلمة IS ويعرف هذا الجزء العملية المراد تنفيذها في الوحدة البرمجية. إن كلمة REPLACE هي كلمة اختيارية يمكن عدم كتابتها، فعند استخدامها، إذا كانت الوحدة البرمجية معرفة مسبقا فإنه يتم حذفها واستبدالها بالنسخة الجديدة المعرفة بالوحدة الحالية.

مراحل بناء الروتين Procedure

ما هي المراحل أو الخطوات لبناء وحدة برمجية جزئية من نوع روتين Procedure؟.



١. اختر بيئة لكتابة الوحدة البرمجية الجزئية Procedure Builder أو SQL*Plus واكتب تعليمات الوحدة البرمجية.

٢. ترجم تعليمات الوحدة Compile the code

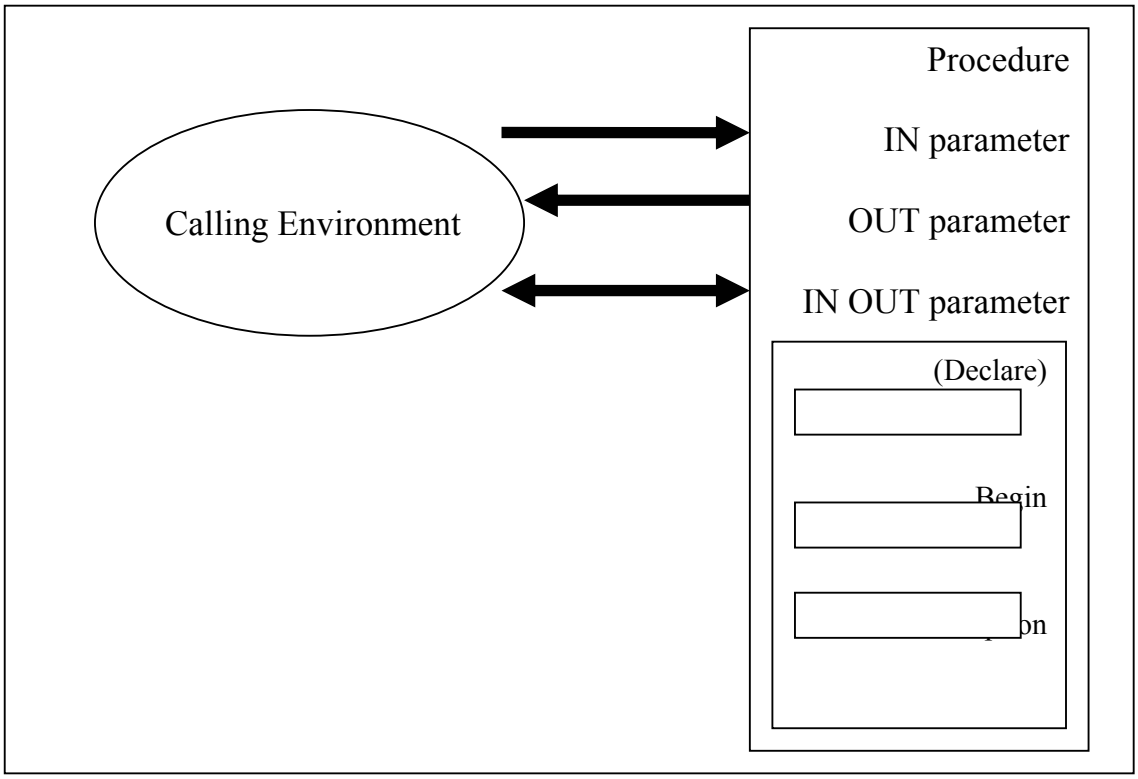
يتم تحويل التعليمات من لغة المصدر Source Code إلى لغه انتقالية p-code

٣. نفذ الوحدة البرمجية الجزئية

ويتم ذلك بتخزين الوحدة إذا كنا نستخدم Procedure Builder أو تنفيذ أمر Start للملف الذي يحتوي الوحدة البرمجية إذا كنا نستخدم SQL*Plus

حالات باراميترات الروتين Procedural Parameter Modes

إن الباراميترات هي الوسيلة المستخدمة لنقل الفيم من وإلى البيئة التي قامت بمناداة هذه الوحدة البرمجية الجزئية Calling Environment، هذه الفيم سيتم استخدامها في المعالجة داخل الجزء التنفيذي من الوحدة البرمجية وإعادة النتائج إلى بيئة المناداة، إذا كان هناك فيم مرتجعة، كل ذلك يتم من خلال باراميترات وهناك ثلاث حالات للباراميترات وهي: مدخل IN، مخرج OUT، مدخل ومخرج IN OUT.



ملحوظة: إن محأولة تغيير فيم المدخل IN سيتسبب بحدوث خطأ.

نوع المدخل	الوصف
IN (افتراضي)	يمرر فيمة ثابتة من بيئة المناداة إلى الوحدة البرمجية Procedure
OUT	يمرر فيمة من الوحد البرمجية Procedure إلى بيئة المناداة
IN OUT	يمرر فيمة من الوحدة البرمجية Procedure إلى بيئة المناداة وبالعكس باستخدام نفس المدخل Parameter

حالات الباراميتر للباراميترات الرسمة Parameter Modes for Formal Parameters

عندما تنشأ الوحدة البرمجية الجزئية روتين، تعرف المدخلات الرسمة التي ستستخدم في الجزء التنفيذي من الوحدة البرمجية، في حين أن المدخلات الحقيقية Actual Parameter هي التي تظهر في جملة مناداة الوحدة البرمجية من البيئة الرئيسة.

IN	OUT	IN OUT
الوضع الافتراضي	يجب أن يحدد	يجب أن يحدد
يمرر الفيمة إلى الوحدة البرمجية Subprogram	يمرر الفيمة إلى بيئة المناداة	يمرر الفيمة من وإلى الوحدة البرمجية
يستخدم المدخل كثابت في الوحدة البرمجية	متغير من غير فيمة أولية Un initialized variable	متغير ذو فيمة أولية Initialized Variable
المحلات الرسمة Actual Parameters يمكن ان تكون فيمة، تعبير حسابي، ثابت أو متغير ذت فيمة	يجب أن يكون متغير	يجب أن يكون متغيراً

مثال: باراميتر مدخل

الوحدة البرمجية ادناه تستخدم مدخل بحالة IN، وعند تنفيذ هذه الوحدة فإنه سيتم إنشاء الوحدة البرمجية ككائن في قاعدة البيانات، وعند مناداتها فإن الوحدة تأخذ مدخلاً يمثل رقم الموظف وتقوم برفع راتب الموظف بنسبة ١٠٪.

```
SQL> CREATE OR REPLACE PROCEDURE raise_salary
  2   (v_id   in   emp.empno%TYPE)
  3    IS
  4   BEGIN
  5        UPDATE emp
  6         SET sal = sal * 1.10
  7         WHERE empno = v_id;
  8    END raise_salary;
  9   /
Procedure created.
SQL> EXECUTE raise_salary(7369)
PL/SQL procedure successfully completed
```

إن المثال أعلاه يعرض روتينا يستخدم باراميترا مدخلا واحدا، وعند تنفيذ هذه الجملة في بيئة SQL*Plus فإنه سيتم إنشاء روتينا اسمه Raise_salary وعندما يتم استدعاؤه فإنه يأخذ باراميتر يمثل رقم الموظف ويعدل سجل الموظف في قاعدة البيانات بزيادة الراتب ١٠٪

لمنادات روتين ما من بيئة SQL*Plus، استخدم أمر EXECUTE

```
SQL> EXECUTE raise_salary (7369)
```

ولمنادات الروتين من داخل أي وحدة برمجية أخرى أو من بيئة Procedure Builder، استخدم اسم الروتين مباشرة مع الباراميترات

```
PL/SQL> raise_salary (7369)
```

لاحظ أن الباراميترات من نوع مدخل تمرر إلى الوحدة البرمجية الجزئيه كثوابت ومحأولة تغير فيمةا سيتسبب بحدوث خطأ.

مثال: باراميتر مخرج

```
SQL> CREAE OR REPLACE PROCEDURE query_emp
        (v_id          IN        emp.empno%TYPE,
         v_name        OUT       emp.ename%TYPE,
         v_salary      OUT       emp.sal%TYPE,
         v_comm        OUT       emp.empno%TYPE)
       IS
       BEGIN
            SELECT ename, sal, comm.
            INTO v_name, v_salary, V_comm
            FROM emp
            WHERE empno = v_id;
       END query_emp;
/
```

إن المثال أعلاه يحتوي على أربعة باراميترات، ثلاثة منها من نوع مخرج والتي ستعيد القيم إلى بيئة المناداة والآخر من نوع مدخل والذي سيدخل فيمة إلى الروتين.

حتى تتمكن من مناداة الروتين السابق من بيئة SQL*Plus وعرض فيم الباراميترات لا بد لك من تعريف المتغيرات اللازمة لاستقبال هذه الفيم، ويتم ذلك بالشكل التإلى وبالترتيب:

```
SQL> START emp_query.sql
Procedure created.
```

حيث إن emp_query.sql هو اسم الملف الذي يحتوي على نص الروتين.

```
SQL>  VARIABLE g_name          VARCHAR2(15)
 SQL> VARIABLE g_sal           NUMBER
 SQL> VARIABLE g_comm          NUMBER
```

```
SQL> EXECUTE query_emp(7654,:g_name, :g_sal, :g_comm)
PL/SQL procedure successfully completed.
```

لاحظ استخدامنا للنقطتين (:) للرجوع إلى المتغيرات المضيفة Host Variables

```
SQL> PRINT g_name
G_NAME
MARTIN
```

مثال: باراميتر مدخل مخرج

```
SQL> CREATE OR REPLACE PROCEDURE format_phone
       (v_phone_no  IN OUT VARCHAR2)
      IS
      BEGIN
      v_phone_no :=     '(' || SUBSTR(V_phone_no,1,3) ||
                        ')' || SUNSTR(V_phone_no,4,3) ||
                        '-' || SUBSTR(v_phone_no,7);
     END format_phone;
/
```

باستخدام بارميتر من نوع مدخل مخرج بامكانك إرسال واستقبال الفيم من وإلى الروتين. وعليه فإن هذا البارميتر يمكن النظر إلية كمتغير يحمل فيمة ابتدائية يمكن لها أن تتغير داخل الروتين أو تبفي دون تغير.

```
SQL> VARIABLR g_phone_no   VARCHAR2(15)

SQL> BEGIN :g_phone_no := '8006330575'; END;
       /
PL/SQL procedure successfully completed.
G_PHONE_NO
8006330575
```

```
SQL> EXECUTE format_phone (:g_phone_no)
PL/SQL procedure successfully completed

SQL> PRINT g_phone_no
G_PHONE_NO
(800)633-0575
```

طرق تمرير الباراميترات

يمكنك تحديد القيم المختلفة للباراميترات في الروتين الذي يحتوي على العديد منها بعدة طرق مختلفة حسب: الموقع، الاسم أو مزيج منهما.

الطريقة	الوصف
الموقع Position	اعرض القيم بنفس ترتيب الباراميترات
الاسمName	اعرض القيم بشكل عشوائي مع ربط كل قيمة بالباراميتر الخاص بها باستخدام الصيغة الخاصة (=>)
مزيج Combination	أرض القيم الأولى حسب الموقع والبقية باستخدام الصيغة الخاصة.

مثال:

```
BEGIN
add_dept;
add_dept ('TRAINING', 'NEW YORK');
add_dept (v_loc => 'DALLAS', v_name => 'EDUCATION');
add_dept (v_loc => 'BOSTON');
END;
/
```

```
SQL> SELECT * FROM dept;

DEPTNO      DNAME          LOC
 ....       .....          ....
  41        unknown        unknown
  42        TRAINING       NEW YORK
  43        EDUCATION      DALLAS
  44        unknown        BOSTON
```

استخدام الكلمة المحجوزة Default مع البارامیتر

يمكنك استخدام الكلمه المحجوزه Default مع أي بارامیتر لإعطائه فيمة ابتدائية يستخدم في حالة عدم ارسال فيمة لهذا البارامیتر من بيئة المناداة، وإليك الثال التإلى:

```
SQL> CREATE OR REPLACE PROCEDURE add_dept
       (v_name        IN   dept.dname%TYPE   DEFAULT 'unknown',
        v_loc         IN   dept.loc%TYPE     DEFAULT 'unknown')
      IS
      BEGIN
         INSERT INTO dept
         VALUES  (dept_deptno.NEXTVL, v_name, v_loc);
     END add_dept;
     /
```

مناداة الروتين من وحدة برمجية أخرى

لقد رأينا كيف تتم عملية منادات الروتين من بيئة SQL*Plus وسنرى الآن كيف تتم مناداة الروتين من داخل وحدة برمجية مثل وحدة برمجية باسم Named Subprogram أو بدون اسم Anonymous Block، حيث تتم مناداة الروتين من خلال كتابة اسمه والبارامیترات التي يحتاجها.

مثال:

```
SQL> CREATE OR REPLACE PROCEDURE process_emps
       IS
           CURSOR emp_cursor IS
              SELECT empno
              FROM emp;
       BEGIN
            FOR emp_rec IN emp_cursor
            LOOP
               raise_salary (emp_rec.empno);  -- invoke procedure
            END LOOP;
            COMMT;
     END;
/
```

حذف الروتين من جهة الخادم Removing Server-Side procedure

لحذف الروتين من جهة الخادم في بيئة SQL*Plus قم بتنفيذ أمرDROP PROCEDURE، كما يلي:

```
SQL> DROP PROCEDURE raise_salary;
Procedure dropped.
```

ملحوظة:

أثناء كتابتك للروتين إذا واجهتك أخطاء أثناء عملية الترجمة Compilation Errors في بيئة SQL*Plus استخدم الأمر SHOW_ERRORS لمعرفة الأخطاء.

تمارين

س١)

١. أنشئ روتينا سمه ADD_PROD لإدراج منتج جديد في جدول المنتجات PRODUCT استخدم رقم المنتج والوصف كباراميترات للروتين.
٢. نفذ الروتين المكتوب في الفرع الأول أعلاه، ثم استدعي الروتين مع إعطائه مدخلات معينة، ومن ثم استفسر عن جدول المنتجات PRODUCT لملاحظة النتائج.

س٢)

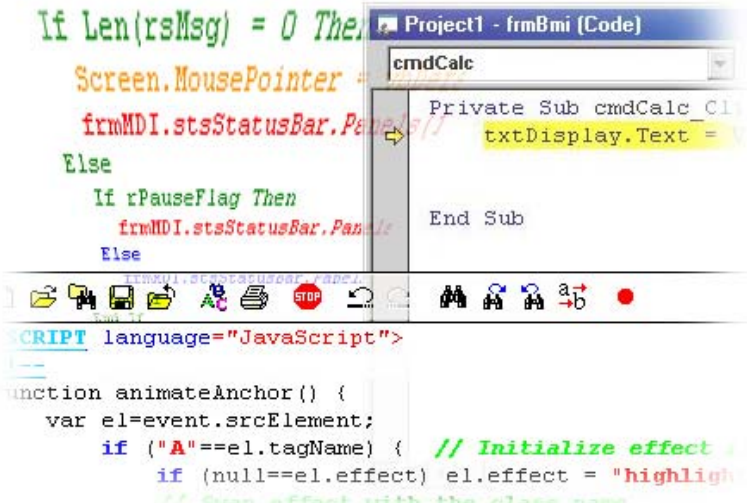
١. أنشئ روتينا سمهUPD_PROD لتعديل وصف منتج لمنتج ما في جدول PRODUCT، وزود هذا الروتين ببراميتر يمثل رقم المنتج وآخر يمثل الوصف الجديد، أضف جزءاً خاصاً بمعالجة استثناء عدم وجود منتج للتعديل.
٢. نفذ الروتين المكتوب في الفرع أعلاه، ثم استدع الروتين مع إعطائه مدخلات معينة، ومن ثم استفسر عن جدول المنتجات PRODUCT لملاحظة النتائج. قم بمناداة الروتين مع تزويده برقم منتج غير موجود في الجدول.

س٣)

١. أنشئ روتينا سمهDEL_PROD لحذف منتج ما في جدول PRODUCT، وزود هذا الروتين ببراميتر يمثل رقم المنتج المراد حذفه، اضف جزءاً خاصاً بمعالجة استثناء عدم وجود منتج للحذف.
٢. نفذ الروتين المكتوب في الفرع أعلاه، ثم استدعي الروتين مع إعطائه مدخلات معينه، ومن ثم استفسر عن جدول المنتجات PRODUCT لملاحظة النتائج. قم بمناداة الروتين مع تزويده برقم منتج غير موجود في الجدول.

برمجة قواعد بيانات

عرض البيانات من أكثر من جدول



الجدارة:

أن يكون المتدرب قادراً على كتابة اقتران محدد في قاعدة البيانات للفيام بمهمة محددة.

الأهـداف:

بنهاية هذه الوحدة، عليك أن تكون قادرا على:

٦. تعريف مفهوم الاقتران
٧. تحديد مراحل بناء الاقتران
٨. كتابة اقتران للفيام بمهمة محددة
٩. معرفة كيفية تنفيذ الاقتران
١٠. التفريق بين الروتين والاقتران

مستوى الأداء المطلوب:

أن يصل المتدرب إلى إتقان الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ٤ ساعات معتمدة

الوسائل المساعده:

- وجود حاسب إلى
- وجود عارض شرائح Projector
- دفتر
- قلم

الدوال أو الاقترانات Functions

يعرف الاقتران على أنه وحدة برمجية جزئيه ذات اسم تقوم بعمل وظيفة معينة وله فيمة مرتجعة Return value، كما ويمكن مناداة الاقتران كجزء من تعبير Expression. وتتلخص صيغة التعريف Function كما يلي:

```
CREATE [OR REPLACE] FUNCTION function_name
    [ (argument1 [mode1] datatype1 [,argument2 [mode2] datatype2, …] ) ]
RETURN datatype
IS|AS
    [ Local Declaration ]
Begin
      Executable Statements
[Exception
      exception handlers ]
END [function_name]
```

يمكن أن يكون للاقتران Function صفر أو أكثر من الباراميترات ويجب أن يكون هناك فيمة مرتجعه واحده، وكما هو موضح أعلاه فإن الاقتران له جزءان رئيسان هما:

١. الترويسيه Header

ويأتي هذا الجزء قبل كلمة AS أو IS ويعرف فيه الاسم والمدخلات والفيمة المرتجعة للاقتران، ويعني الخيار REPLACE أنه إذا كان الاقتران المراد إنشاؤه موجود سابقا فإنه سيتم حذفه وإنشاؤه من جديد

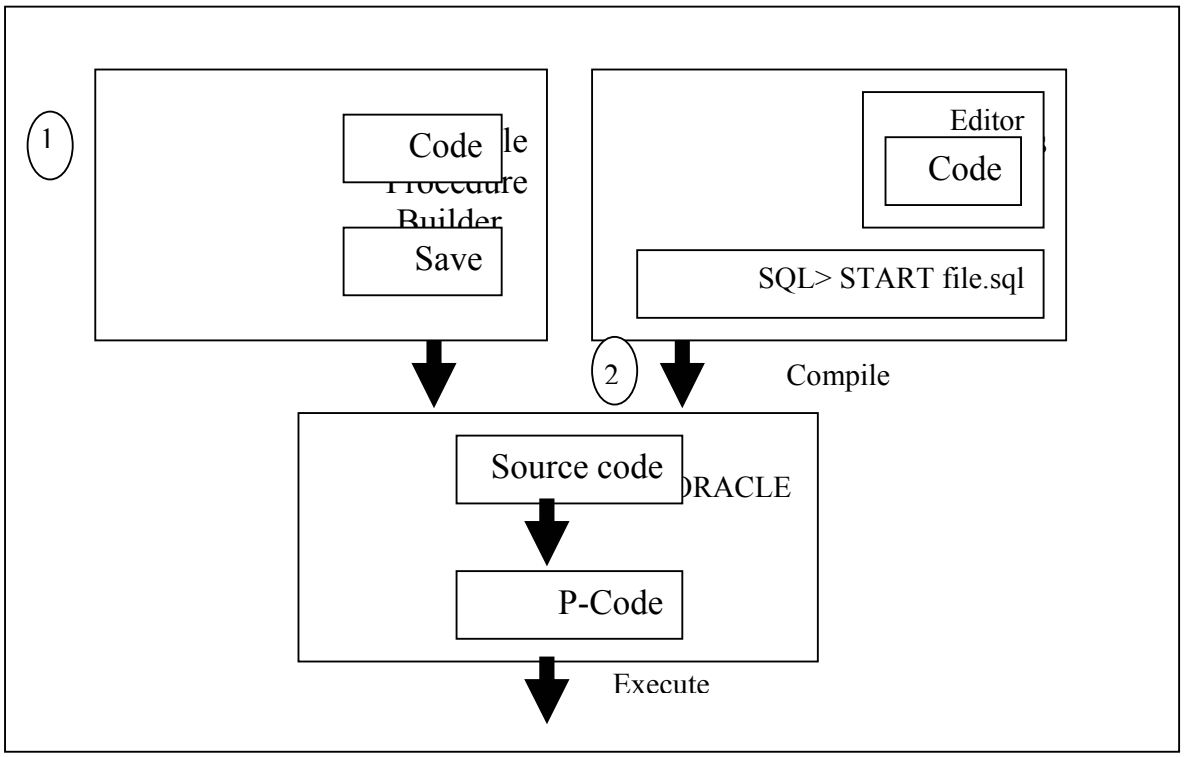
٢. الفيمة المرتجعه وتحدد بالكلمة المحجوزة RETURN ولا يجب أن تحتوي حجم الفيمة المرتجعه size ويكفي تحديد نوعها فقط.

٣. الجسم Body

وهو كل شي بعد كلمة IS ويعرف هذا الجزء العملية المراد تنفيذها في الوحدة البرمجية، كما يجب أن يحتوي على الأقل على جملة إرجاع واحدة RETURN (variable)

مراحل بناء الاقتران Function

ما هي المراحل أو الخطوات لبناء وحدة برمجية جزئيه من نوع اقتران Function؟.



١. اختر بيئة لكتابة الوحدة البرمجية الجزئيه Procedure Builder أو SQL*Plus واكتب تعليمات الوحدة البرمجية.

٢. ترجم تعليمات الوحدة Compile the code

يتم تحويل التعليمات من لغة المصدر Source Code إلى لغة انتقالية p-code

٣. نفذ الوحدة البرمجية الجزئيه

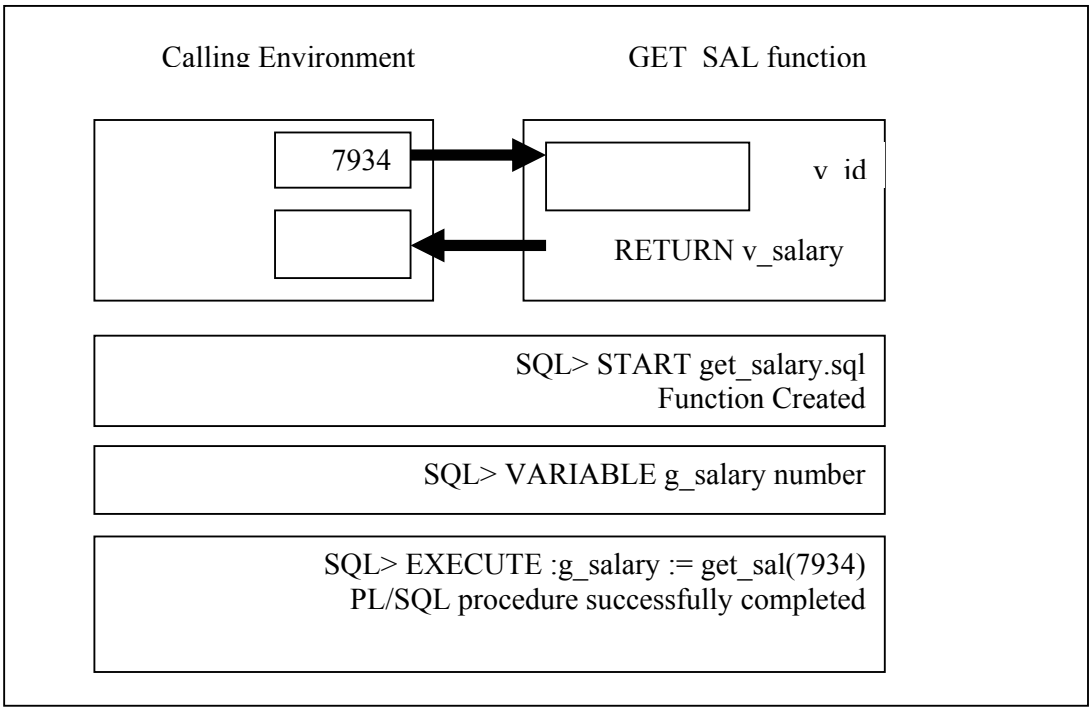
ويتم ذلك بتخزين الوحدة إذا كنا نستخدم Procedure Builder أو تنفيذ أمر Start للملف الذي يحتوي الوحدة البرمجية إذا كنا نستخدم SQL*Plus

إليك الإن المثال التإلى الذي يوضح كيفية إنشاء اقتران باستخدام SQL*Plus. هذا الاقتران له باراميتر مدخل واحد، وله فيمة مرتجعة من نوع عدد NUMBER.

```
SQL> CREATE OR REPLACE FUNCTION get_sal
          (v_id    IN  emp.empno%TYPe)
          RETURN NUMBER
       IS
          v_salary    emp.sal%TYPE := 0;
       BEGIN
           SELECT sal
           INTO v_salary
           FROM emp
           WHERE   empno = v_id;
            RETURN (v_salary);
       END;
/
```

كيف تنفذ الاقتران:

يمكن للاقتران أن يستقبل باراميتراً واحداً أو أكثر من نوع مدخل، ولكنه يرجع فيمة واحدة فقط. يمكنك مناداة الاقتران كجزء من تعبير PL/SQL أو باستخدام متغير تسند إليه الفيمة المرتجعة من تنفيذ الاقتران .



يمكننا مناداة الدالة المعرفة بواسطة المستخدم في أي مكان نستخدم فيه الدوال المبنية في لغة PL/SQL مثل:

١) قائمة الاختيارات في جملة SELECT
٢) جزء الشرط في جملة Where و Having
٣) جملة Connect By, Start with, Order by وجملة Group by
٤) جملة Values في أمر Insert
٥) جملة Set في أمر Update

لاحظ أن الدوال لا تستطيع تغيير البيانات في الجدأول حيث إنها لا يمكنها تنفيذ جمل INSERT, UPDATE, DELETE ولا يسمح للدوال بمناداة أي حدة برمجية مخالفة للقاعدة السابقة.

حذف الدوال من جهة الخادم Remove Server side functions

لحذف دالة مخزنة في جهة الخادم باستخدام SQL*Plus ننفذ أمر SQL الخاص بذلك وهو Drop على النحو التإلى:

```
Drop Function get_sal;
Function dropped.
```

ملحوظة: تذكر أنه ليس بمقدورك عمل تراجع Rollback عن أي عملية DDL مثل Drop function وبالتإلى عند حذفك لأي دالة لن يكون بمقدورك إعادة هذه الدالة مجددا.

مقارنه بين الروتين والاقتران

الروتين	الدوال
تنفذ كجملة PL/SQL	ينفذ كجزء من تعبير ما Expression
ليس له فيمة مرتجعة	يجب أن يحتوي على RETURN data type
يمكن ان يكون له فيمة مرتجعة واحدة أو أكثر	يجب أن يكون له فيمة مرتجعة واحدة فقط

فوائد استخدام الروتين والاقتران:

١. تحسين الكفاءة
٢. تحسين إمكانية صيانة ومتابعة البرنامج
٣. تحسين فرص حماية البيانات

تمارين

س١)

أ. اكتب دالة اسمها PROD_DESC تستقبل باراميتر مدخل يمثل رقم المنتج ولها مخرج واحد يمثل وصف المنتج من جدول المنتجات PRODUCTS

ب. ترجم الداله في الفرع أ .

ت. عرف متغيراً في بيئة SQL*Plus وسمه Desc.

ث. نفذ الدالة وخزن الناتج في المتغير Desc.

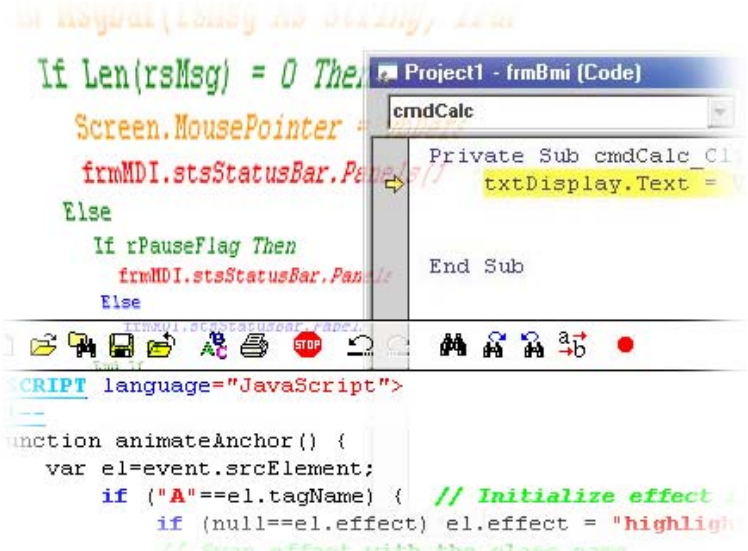
س٢)

أ. اكتب دالة اسمها Annual_Sal تستقبل باراميتراً يمثل رقم الموظف empno ولها فيمة مرتجعة واحده هي الدخل السنوي للموظف، علما أن الدخل السنوي للموظف يحسب بالعلاقه (sal*12)+comm، آخذا بعين الاعتبار أن فيمة comm يمكن أن تكون NULL .

ب. اكتب جملة استعلام لاسترجاع رقم الموظف، اسم الموظف والدخل السنوي للموظف، لجميع الموظفين، مستخدما الدالة المعرفة في الفرع السابق من السؤال.

برمجة قواعد بيانات

الزناد



الجدارة:

أن يكون المتدرب قادراً على كتابة زناد محدد في قاعدة البيانات للفيام بمهمة محددة.

الأهـداف:

بنهاية هذه الوحدة، عليك أن تكون قادرا على:

١. وصف أزندة قواعد البيانات

٢. إنشاء أزندة قواعد البيانات

٣. معرفة قواعد إطلاق الزناد

٤. حذف الزناد من قاعدة البيانات

مستوى الأداء المطلوب:

أن يصل المتدرب إلى إتقان الجداره بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ٨ ساعات معتمدة

الوسائل المساعدة:

- وجود حاسب آلي
- وجود عارض شرائح Projector
- دفتر
- قلم

الوحدة السابعة:

أزندة قواعد البيانات

زناد قواعد البيانات Database Triggerهو قطعة برمجية تنفذ تلقائيا عند إجراء عملية إدخال، تعديل أو حذف على الجدول المرفق معه هذا الزناد بغض النظر عن المستخدم أو التطبيق الذي قام بإجراء هذه العملية.

في حين يتم تنفيذ زناد التطبيقات Application Trigger تلقائيا عند حدوث حدث معين في التطبيق، مثال على هذا النوع من الزناد هو الزناد الذي تتم كتابته في باقي النماذج Forms Builder.

إن زناد قواعد البيانات يمكن تعريفه على الجدأول العادية والجدأول الوهمية Views، وفي حالة الرغبة بالفيام بعمل معين عند حدوث عملية معالجه DML للجدول الوهمي فإن الزناد المعرف على هذه الجدأول يكون من نوع "بدل عن" instead of.

إرشادات تصميم الزناد

عند تصميم الزناد Trigger لا بد لنا من مراعاة الأمور التالية:

١. استخدم الزناد لضمان أنه عند حدوث حدث ما فإن عملاً معيناً يجب أن ينفذ.
٢. استخدم زناد قواعد البيانات للعمليات المركزية العامة والتي ستنفذ بغض النظر عن تطبيق المستخدم الذي تسبب بهذا الحدث.
٣. لا تبن زناداً يكرر أو يستبدل عملية مكتوبة ومبرمجة مسبقا في قاعدة البيانات أوراكل، فمثلا لا تعد كتابة زناد متعلق بتطبيق فكرة المفتاح الرئيس Primary key أو أي محدد Constraint آخر معرف في قاعدة البيانات.
٤. استخدم الزناد عند الحاجة دون إفراط لأن كثرة تعريف الأزندة يتسبب في تعفيدات كثيرة عند متابعة التطبيق أو صيانته.

إنشاء الزناد

حتى تقوم بإنشاء الزناد بصورة صحيحة لا بد لك من تحديد العديد من الأمور قبل البدء بكتابة محتوى هذا الزناد، إذ عليك تحديد وقت الزناد، الحدث المسبب لإطلاق الزناد، نوع الزناد وما هو العمل المنوط بهذا الزناد ليقوم به. الجدول ادناه يبين تفاصيل هذه الأجزاء، وهي على النحو التإلى:

الجزء	الوصف	الفيم المحتملة
وقت الزناد Trigger Timing	متى سينفذ الزناد بالنسبة للحدث المسبب لإطلاقه.	قبل Before بعد After بدلا عن Instead of
حدث الزناد Trigger Event	أي عمليات معالجة البيانات DML على الجدول أو الجدول الوهمي view سيتسبب بإطلاق الزناد	الإدخال Insert تعديل Update حذف Delete
نوع الزناد Trigger Type	كم مرة سيتم تنفيذ جسم الزناد	على مستوى جملة Statement على مستوى سطر Row
جسم الزناد Trigger body	ما هو العمل الذي سيقوم به الزناد	قطعة PL/SQL

ان ترتيب الأزندة من نفس النوع على نفس الجدول يتم بصورة عشوائية، أما إذا أردت التأكد من أن هذه الأزندة تنفذ بترتيب معين فما عليك إلا أن تجمعها في زناد واحد يقوم بمناداة مجموعة من البرامج الجزئية (روتين أو دالة) وبالترتيب الذي تحدده.

وقت الزناد

يحدد هذا الجزء من تعريف الزناد متى سيتم تنفيذ الزناد ، وهناك وقتان لتنفيذ الزناد هما :

١. قبل Before : في هذه الحاله يتم تنفيذ جسم الزناد قبل حدوث عملية معالجة البيانات، ويستفاد من هذا النوع على النحو التإلى:

- عند الحاجة للتأكد من أن جملة معالجة البيانات المسببة لحدوث إطلاق للزناد Trigger Firing يجب أن تستكمل أم لا.بمعنى أن نتأكد قبل الشروع بعملية الإدخال أو التعديل أو الحذف بان شروط إجراء العملية مكتملة وذلك بتنفيذ جسم الزناد قبل تنفيذ جملة معالجة البيانات. وهذا يقلل من احتمإلية حدوث تراجع عن جملة معالجة البيانات Rollback.
- من أجل اشتقاق فيم بعض الأعمدة قبل تنفيذ حملة معالجة البيانات.

٢. بعد After: في هذه الحالة يتم تنفيذ جسم الزناد بعد حدوث عملية معالجة البيانات، ويستخدم هذا النوع بشكل كبير في الحالات التالية:

- عند الحاجة لتنفيذ جملة معالجة البيانات كليا قبل إجراء أي عملية أخرى من خلال الزناد
- في حالة وجود زناد "قبل" فإنه بإمكانك استخدام زناد "بعد" لتنفيذ عمل آخر على نفس جملة معالجة البيانات.

٣. بدلا عن Instead of: في هذه الحالة يتم تنفيذ جسم الزناد بدلا عن تنفيذ جملة معالجة البيانات التي تسببت في تنفيذه. وتستخدم هذه الطريقة عادة مع الجدأول الوهمية views غير القابلة للتعديل أو إجراء عملية معالجة البيانات عليها.

يتم تنفيذ الزناد من نوع "بدلا عن" والذي يتولى بدوره الفيام بالعمليات اللازمة، وبالتإلى فإنه بإمكانك كتابة جمل Insert, update, delete بصورة عادية إلا أنه داخليا يتم تنفيذ الزناد"بدلا عن" لإجراء اللازم.

الحدث المسبب لإطلاق الزناد

في هذا الجزء من تعريف الزناد تقوم بتحديد الحدث الذي سيؤدي إلى إطلاق الزناد وتنفيذه. إن العمليات التي تؤدي إلى تنفيذ الزناد هي جمل insert ,update or delete على جدول ما.

إذا كانت الجملة المسببه لإطلاق الزناد هي update فإنه بإمكانك تحديد العمود أو الأعمدة التي يجب أن تنفذ لإطلاق الزناد أو يمكنك عدم تحديدها وعندها أي عملية تعديل على السطر ستؤدي إلى إطلاق الزناد.

```
. . . . UPDATE OF sal . . . .
```

في هذه الحالة لن ينفذ الزناد المرتبط بعملية التعديل إلا إذا كان العمود المعدل هو عمود sal.

يمكن للحدث المسبب لإطلاق الزناد أن يكون عدة عمليات مثل:

```
. . . . INSERT or UPDATE or DELETE . . . .
```

```
. . . . INSERT or UPDATE OF jobs . . . .
```

نوع الزناد:

يستفاد من بيان نوع الزناد في تحديد عدد المرات التي سينفذ فيها جسم الزناد: مرة واحدة لكل سطر تأثير بجملة معالجة البيانات (مثلا أكثر من سطر تأثر بعملية UPDATE) أو مرة واحدة لكل جملة معالجة البيانات بغض النظر عن عدد الأسطر التي تأثرت بهذه الجملة.

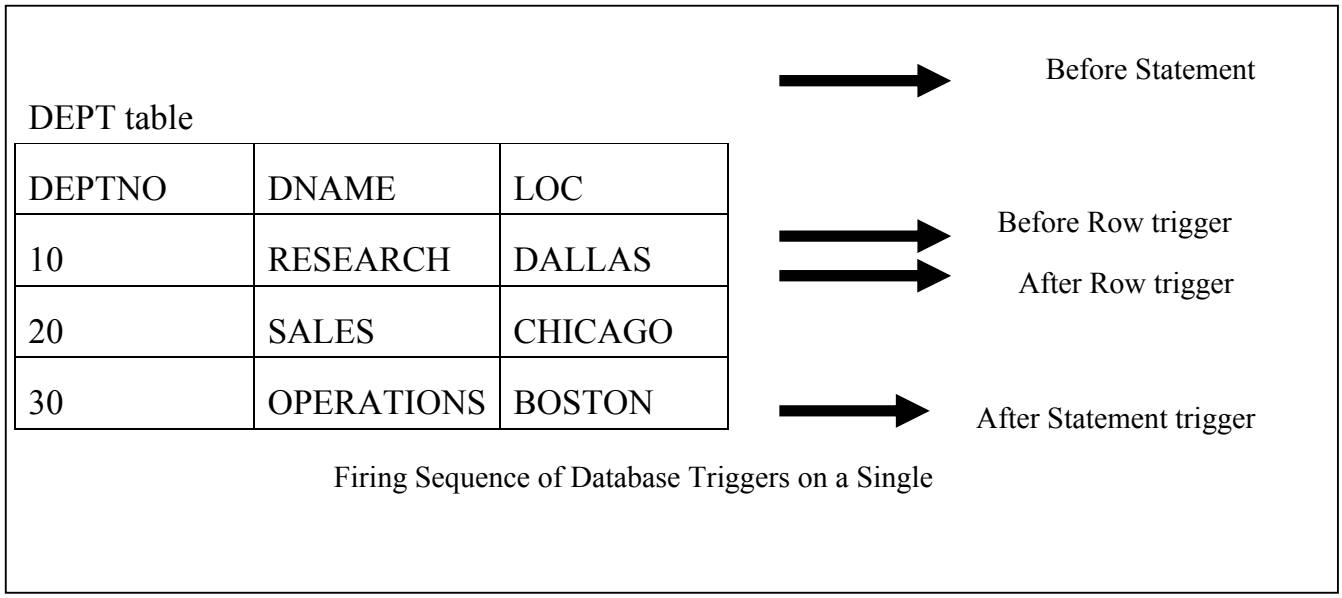
١. زناد على مستوى جملة: الزناد المعرف على مستوى جملة ينفذ مرة واحدة، حتى لو لم تؤثر جملة معالجة البيانات على أي سطر. ويستفاد من هذا النوع للحالات التي لا تعتمد على الأسطر المتأثرة بعملية معالجة البيانات أو عددها.

٢. زناد على مستوى سطر: الزناد المعرف على مستوى السطر ينفذ كل مرة يتأثر بها الجدول أي على كل سطر يتأثر بجملة معالجة البيانات، وفي حالة عدم تأثر أي سطر بجملة معالجة البيانات فإن الزناد المعرف على مستوى السطر لا ينفذ مطلقا. ويستفاد من هذا النوع في برمجة العمليات التي تعتمد على الأسطر المتأثرة بعملية معالجة البيانات.

مثال : المثال التإلى يقوم بإدراج سطر واحد على جدول DEPT .

```
SQL> INSERT INTO dept (deptno, dname, loc)
        VALUES (50, 'EDUCATION', 'NEW YORK');
```

الشكل أدناه يوضح تنفيذ سلسلة من أزندة قواعد البيانات على سطر واحد في الجدول. بعد تنفيذ جملة الإدخال أعلاه ينفذ الزناد "قبل" المعرف على مستوى جملة معالجة البيانات، يليه زناد "قبل" المعرف على مستوى السطر يليه زناد "بعد" المعرف على مستوى السطر ثم يليه زناد "بعد" المعرف على مستوى جملة معالجة البيانات. وبذلك تلاحظ أن جملة معالجة البيانات التي تؤثر على سطر واحد فقط يتسأوى فيها عدد مرات تنفيذ الأزندة المعرفة على مستوى جملة والأخرى المعرفة على مستوى سطرحيث ينفذ كلا منهأمرة واحدة فقط.

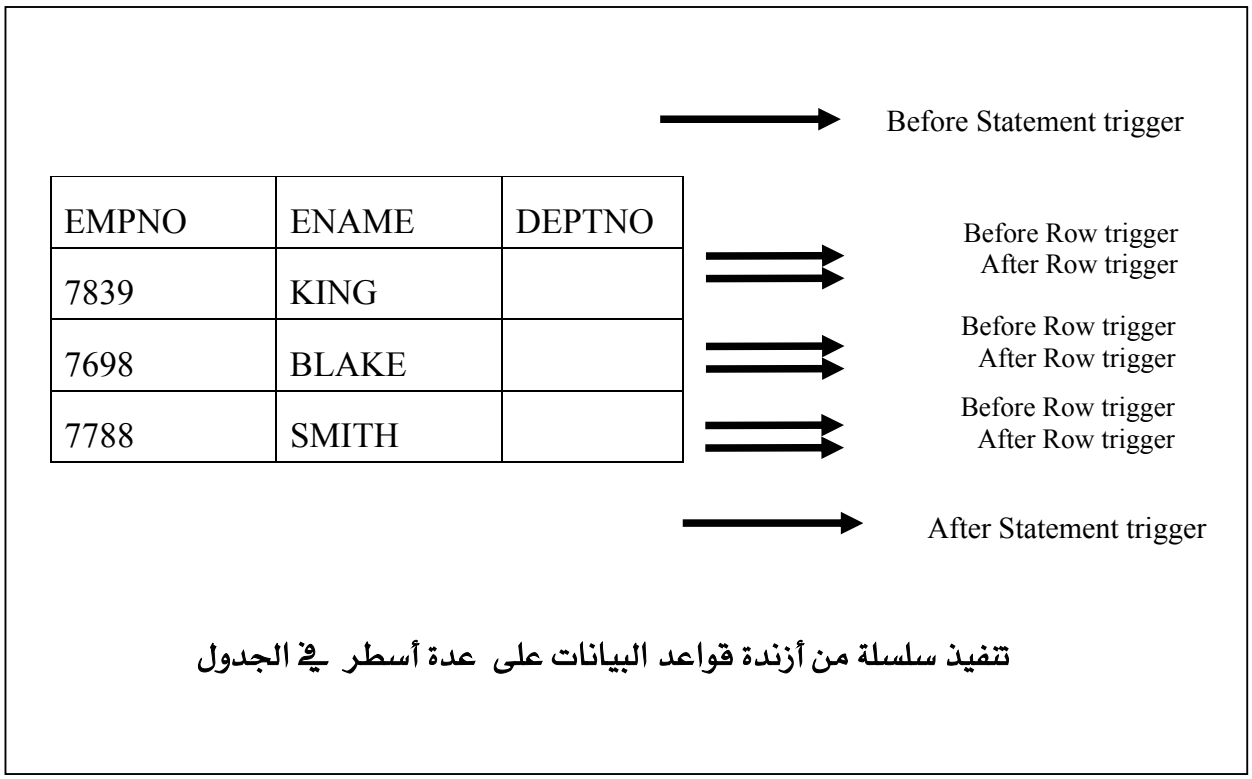


Firing Sequence of Database Triggers on a Single

مثال: المثال التإلى يوضح عملية تعديل رواتب كافة الموظفين الذين يعملون في الدائره رقم ٣٠.

```
SQL> UPDATE emp
     SET sal = sal * 1.1
     WHERE deptno = 30;
```

الشكل أدناه يوضح تنفيذ سلسلة من أزندة قواعد البيانات على عدة ألآسطر في الجدول وذلك بعد تنفيذ الجملة أعلاه، حيث ينفذ الزناد "قبل" المعرف على مستوى جملة معالجة البيانات أولا، يليه ينفذ زناد "قبل" المعرف على مستوى السطر يليه زناد "بعد" على مستوى السطر وذلك لكل سطر تأثر بجملة التعديل UPDATE أعلاه. ثم يليه تنفيذ زناد "بعد" المعرف على مستوى جملة معالجة البيانات.



تنفيذ سلسلة من أزندة قواعد البيانات على عدة أسطر في الجدول

جسم الزناد:

يحدد هذا الجزء العمل الذي سيتم تنفيذه عندما يتم إطلاق الزناد بناءً على تنفيذ جملة معالجة بيانات معينة، ويمكن لهذا الجزء أن يحتوي على جمل SQL و PL/SQL، ويمكن أن يحتوي على جمل لتعريف المتغيرات، المؤشرات، الاستثناءات، وغيرها.

إضافة لذلك فإن أي زناد معرف على مستوى سطر يستطيع الوصول إلى القيم الجديدة والقديمة للأعمدة الموجودة في السطر الذي يتم معالجته حالياً بواسطة الزناد وذلك باستخدام أسماء ارتباط Correlation names وسيرد ذكرها لاحقا. يتم تعريف جسم الزناد من خلال قطعة برمجية من غير اسم anonymous PL/SQL Block على النحو التإلى:

```
[DECLARE]
BEGIN
[EXCEPTION]
END;
```

صيغة تعريف الزناد على مستوى جملة:

```
CREATE [OR REPLACE] TRIGGER trigger_name
Timing event1 [OR timing event2 Or timing event3]
ON table_name
PL/SQL block;
```

اسم الزناد	يحدد هذا الجزء اسم الزناد
وقت الزناد	يحدد الوقت الذي سيتم فيه تنفيذ الزناد بالنسبة لحدث الزناد
حدث الزناد	يحدد عملية معالجة البيانات التي تسبب إطلاق الزناد
اسم الجدول	يحدد اسم الجدول المرتبط به تنفيذ الزناد
القطعة البرمجية	هي جسم الزناد التي تحدد العمل الواجب القيام به من قبل الزناد، ويبدأ هذا الجزء بالكلمة المحجوزة DECLARE أو BEGIN وينتهي بـ END

مثال: إنشاء زناد "قبل" على مستوى جملة.

```
SQL> CREATE OR REPLACE TRIGGER secure_emp
       BEFORE INSERT ON emp
       BEGIN
        IF (TO_CHAR (sysdate,'DY') IN ('THU','FRI')) OR
            (TO_CHAR (sysdate, 'HH24') NOT BETWEEN '08' AND '18')
        THEN RAISE_APPLAICATION_ERROR (-20500,
        'You may only insert into EMP during normal hours.');
        END IF;
       END;
       /
```

كما أسلفنا سابقا، فإنك تستطيع استخدام الزناد "قبل" على مستوى الجملة من أجل وقف تنفيذ جملة معالجة البيانات التي تسببت في إطلاق الزناد إذا تم مخالفة شرط ما.

والزناد أعلاه مثال على هذه النقطة، حيث تم تعريف زناد لإيقاف عملية إدخال بيانات موظفين جدد خارج أوقات الدوام الرسمي. فلو حأول أحدهم إدخال بيانات موظف جديد في يوم الخميس أو الجمعة أو في أي يوم قبل الساعة الثامنة صباحا أو بعد الساعة السادسة مساءً فإنه سيرى رساله تقول له يمكنك الإدخال في جدول الموظفين في أوقات الدوام الرسمي فقط. وستفشل جملة الإدخال من إكمال عملها وسيتم عمل تراجع عنها Rollback.

الجملة RAISE_APPLICATION_ERROR هي روتين معرف في جهة الخادم Server side built in عند استخدامه ستظهر رسالة للمستخدم ويتسبب بعدها بإفشال القطعة البرمجية التي تحتويه.

```
SQL> INSERT INTO emp (empno, ename, deptno)
     2  VALUES          (77, 'ALI' 40);
INSERT INTO emp (empno, ename, deptno)
                 *
ERROR at line 1:
ORA-20500: You may only insert into EMP during normal hours.
ORA-06512: at "SCOTT.SECURE_EMP", line 4
ORA04088: error during execution of trigger 'SCOTT.SECURE_EMP'
```

ملحوظة: عندما تفشل جملة معينه بالتفنيذ فإنه يتم التراجع عنها تلقائيا من قبل خادم ORACLE.

استخدام الكلمات الشرطيةUsing Conditional Predicates :

يمكنك دمج العديد من الأحداث المسببة لإطلاق الزناد ومعالجتها في حدث واحد وذلك بالاستفادة من الكلمات الشرطية INSERTING, DELETING, UPDATING واستخدامها داخل جسم الزناد.
مثال: أنشئ زناداً واحداً فقط يسمح بإجراء عمليات معالجة الأحداث المختلفة على جدول الموظفين EMP في ساعات العمل الرسمي.

```
SQL> CREATE OR REPLACE TRIGGER secure_emp
        BEFORE INSERT OR UPDATE OR DELETE ON emp
        BEGIN
        IF (TO_CHAR(sysdate, 'DY') IN ('THU','FRI')) OR
            (TO_CHAR (sysdate, 'HH24') NOT BETWEEN '08' and '18')
        THEN
           IF INSERTING THEN
              RAISE_APPLICATION_ERROR (-20500,
                    'You may only insert into EMP during normal hours');
           ELSIF DELETING
              RAISE_APPLICATION_ERROR (-20502,
                    'You may only delete from EMP during normal hours');
           ELSIF UPDATING('SAL')
              RAISE_APPLICATION_ERROR (-20503,
                    'You may only update SAL during normal hours');
          ELSE
              RAISE_APPLICATION_ERROR (-20504,
                    'You may only update EMP during normal hours');
          END IF;
       END IF;
  END;
/
```

يمكنك إنشاء زناد "بعد" من أجل مراقبة العملية التي تسببت في إطلاق الزناد أو عمل حسابات بعد الانتهاء من عملية معينة.

مثال: لنفرض أن لدينا جدول مراقبة يحدد عدد عمليات الإدخال، التعديل والحذف التي قام بها مستخدم ما على جدول ما، والحد الأقصى المسموح به لكل عملية على النحو التإلى:

USER _NAME	TABLENAME	COLUMN _NAME	INS	UPD	DEL	MAX _INS	MAX _UPD	MAX _DEL
SCOTT	EMP		1	1	1	5	5	5
SCOTT	EMP	SAL		1			5	
JONES	EMP		0	0	0	5	0	0

بناءً على الجدول أعلاه، اكتب زناداً ينفذ بعد تعديل الراتب SAL نحيث نتأكد بأن عدد مرات تعديل الراتب لا يتجأوز الحد الأعلى المسموح به.

```
SQL> CREATE OR REPLACE TRIGGER check_salary_count
        AFTER UPDATE OF sal ON EMP
        DECLARE
            V_salary_changes  NUMBER;
            V_max_changes    NUMBER;
        BEGIN
            SELECT upd, max_upd
            INTO  v_salary_changes, v_max_changes
            FROM audit_table
            WHERE user_name = user
                AND  tablename  = 'EMP'
                AND column_name = 'SAL';
```

```
        IF v_salary_changes > v_max_changes  THEN
            RAISE_APPLICATION_ERROR (-20501,
            ' You may only make a maximum of ' ||
            TO_CHAR (v_max_changes) || ' changes to the SAL column');
        END IF;
    END;
/
```

صيغة تعريف الزناد على مستوى سطر:

```
CREATE [OR REPLACE] TRIGGER trigger_name
Timing event1 [OR timing event2 Or timing event3]
ON table_name
  [ REFERENCING OLD AS OLD | NEW AS new]
FOR EACH ROW
  [WHEN condition]
PL/SQL block;
```

اسم الزناد	يحدد هذا الجزء اسم الزناد
وقت الزناد	يحدد الوقت الذي سيتم فيه تنفيذ الزناد بالنسبة لحدث الزناد
حدث الزناد	يحدد عملية معالجة البيانات التي تسبب إطلاق الزناد
اسم الجدول	يحدد اسم الجدول المرتبط به تنفيذ الزناد
المرجعية	يحدد أسماء الارتباط للفيم الجديدة والقديمة للسطر الحإلى

لكل سطر FOR EACH ROW	تحدد هذه الجملة أن الزناد ينفذ على مستوى السطر.

مثال: اكتب زناداً ينفذ بعد أي عملية معالجة للبيانات لمتابعة عدد مرات هذه العملية التي قام بها مستخدم معين، وتخزين ذلك على جدول المراقبة.

```
SQL> CREATE OR REPLACE TRIGGER audit_emp
       AFTER DELETE OR INSERT OR UPDATE ON emp
       FOR EACH ROW
       BEGIN
          IF DELETING THEN
             UPDATE audit_table SET del = del + 1
              WHERE  user_name = user AND tablename = 'EMP'
                 AND   col_name IS NULL;
          ELSIF INSERTING THEN
             UPDATE audit_table SET ins = ins + 1
              WHERE  user_name = user AND tablename = 'EMP'
                 AND   col_name IS NULL;
          ELSIF UPDATING('SAL') THEN
             UPDATE audit_table SET upd = upd + 1
              WHERE  user_name = user AND tablename = 'EMP'
                 AND   col_name = 'SAL;
          ELSE
             UPDATE audit_table SET upd = upd + 1
```

```
        WHERE  user_name = user AND tablename = 'EMP'
           AND   col_name IS NULL;
     END IF;
   END;   /
```

مثال: أنشئ زناداً على جدول الموظفين EMP لإضافة سطور على جدول AUDIT_EMP_TABLE لرصد التعديلات التي قام بها المستخدم على جدول EMP. حيث يقوم الزناد بتخزين الفيم القديمة والجديدة لمجموعة من الأعمدة.

```
SQL> CREATE OR REPLACE TRIGGER audit_emp_values
     AFTER DELETE OR INSERT OR UPDATE ON emp
     FOR EACH ROW
     BEGIN
          INSERT INTO audit_emp_table
                    (user_name, timestamp, id, old_last_name, new_last_name,
                     old_title, new_title, old_salary, new_salary)
          VALUES (USER, SYSDATE, :OLD.empno, :OLD.ename,
                     :NEW.ename, :OLD.job, :NEW.job, :OLD.sal,
                     :NEW.sal);
     END;
/
```

الجدول التإلى يوضح جزءاً من محتويات جدول Audit_Emp_Table

USER_NAME	TIMESTAMP	ID	OLD_LAST_NAME	NEW_LAST_NAME
SCOTT	12-NOV-97	NULL	NULL	HUSTON
ALI	10-DEC-97	7844	MAGE	TURNER

OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
NULL	ANALYST	NULL	3500
CLERK	SALESMAN	1100	1100

في الزناد المعرف على مستوى سطر، يمكنك الرجوع إلى فيمة عمود ما قبل وبعد إجراء التعديل على البيانات من خلال استخدام كلمتي OLD و NEW.

العملية	الفيمة القديمة	الفيمة الجديدة
الإدخال INSERT	NULL	الفيمة المدخلة
التعديل UPDATE	الفيمة قبل التعديل	الفيمة بعد التعديل
الحذف DELETE	الفيمة قبل الحذف	NULL

- الكلمتان OLD و New تستخدمان فقط في الزناد المعرف على مستوى سطر
- عند استخدام إحدى هاتين الكلمتين، يجب أن تسبق بـ (:) في جمل SQL و PL/SQL
- لا نستخدم (:) عند استخدام احدى الكلمتين في جملة الشرط WHEN .

إذا أردنا أن يقتصر تنفيذ جسم الزناد على مجموعة من الأسطر تحقق شرطاً معيناً، فإننا نستخدم جملة When clause.

مثال: أنشئ زناداً على جدول الموظفين EMP لحساب عمولة الموظف Employee's Commission عند إضافة موظف جديد أو تعديل راتب موظف موجود.

```
SQL> CREATE OR REPLACE TRIGGER derive_commmission_pct
       BEFORE INSERT OR UPDATE OF sal ON emp
       FOR EACH ROW
       WHEN (NEW.job = 'SALESMAN')
       BEGIN
           IF INSERTING THEN
              :NEW.comm := 0;
           ELSIF :OLD.comm IS NULL THEN
               :NEW.comm := 0;
           ELSE
               :NEW.comm := :OLD.comm * (:NEW.sal/:OLD.sal);
           END IF;
       END;
/
```

كيف تميز بين زناد قواعد البيانات والروتين؟

هناك العديد من الفروقات بين زناد قاعدة البيانات و الروتين أهمها مايلي:

الروتين	الزناد
١. استخدام CREATE PROCEDURE	١. استخدام CREATE TRIGGER
٢. يحتوي قاموس البيانات Data Dictionary على الكود المصدري Source والتنفيذي p_code	٢. يحتوي قاموس البيانات Data Dictionary على الكود المصدري Source والتنفيذي p_code
٣. يتم استدعاؤه تصريحيا (غير تلقائي)	٣. يتم استدعاؤه ضمنيا (تلقائيا)
٤. جمل معالجة الحركات Transaction Processing مثل Commit, Savepoint و Rollback مسموح باستخدامها في الزناد	٤. جمل معالجة الحركات Transaction Processing مثل Commit, Savepoint و Rollback غيرمسموح باستخدامها في الزناد.

الروتين	زناد قاعدة البيانات
١. يتم استدعاؤه تصريحيا (غير تلقائي)	١. يتم استدعاؤه ضمنيا (تلقائيا)
٢. جمل معالجة الحركات Transaction Processing مثل Commit, Savepoint و Rollback مسموح باستخدامها في الزناد	جمل معالجة الحركات Transaction Processing مثل Commit, Savepoint و Rollback غيرمسموح باستخدامها في الزناد.

عند تنفيذ جملة إنشاء الزناد فإنه تتم ترجمة الزناد إلى كود تنفيذي p_code ويخزن في قاموس البيانات Data Dictionary ، وبالتإلى فإن تنفيذ الزناد يتم مباشرة من خلال تشغيل الكود التنفيذي دون الحاجة إلى تحميل تعريف الزناد من جديد وتنفيذه. ويتم إنشاء الزناد حتى لو كان يحتوي على أخطاء.

إدارة أزندة قواعد البيانات

إن أي زناد معرف في قاعدة البيانات يكون في إحدى حالتين: أم أن يكون مفعلا Enabled أو معطلا Disabled. وعند إنشاء الزناد للمرة الأولى فإنه يكون في حالة تفعيل تلقائي، ويقوم خادم البيانات بالتأكد من أن كل زناد مفعل لا يتعارض تنفيذه مع محددات قواعد البيانات Data base integrity Constraints. يمكن تغيير حالة الزناد بين التفعيل والتعطيل باستخدام أمر ALTER TRIGGER كما هو موضح أدناه:

لتفعيل أو تعطيل زناد ما نستخدم الأمر:

```
ALTER TRIGGER trigger_name DISABLE | ENABLED
```

لتفعيل أو تعطيل كل الأزندة المعرفة على جدول ما نستخدم الأمر:

```
ALTER TABLE table_name DISABLE | ENABLE ALL TRIGGERS
```

ولإعادة ترجمة زناد ما من جديد نستخدم الأمر التإلى:

```
ALTER TRIGGER trigger_name COMPILE
```

لاحظ هنا أنه تتم ترجمة الزناد مرة أخرى بغض النظر إذا كان صحيحاً أو به أخطاء.

حذف زناد

عندما لا يصبح هناك حاجة للزناد في قاعدة البيانات فإنه من الأفضل حذف الزناد كليا من قاعدة البيانات، ويمكن الفيام بذلك باستخدام الأمر:

```
DROP TRIGGER trigger_name
```

مثال:

```
SQL> DROP TRIGGER secure_emp;
```

قواعد خاصه بالأزندة:

إن قراءة وكتابة البيانات باستخدام زناد قواعد البيانات يجب أن يحقق الشروط التالية:

١. إن لا يغير بيانات أعمدة المفتاح الرئيس Primary Key، المفتاح الأجنبي Foreign Key أو المفتاح النادر Unique Key في الجدول الذي تم تعريف هذه المحددات عليه.

٢. ان لا يقرأ البيانات من جدول Mutating، ويعرف الجدول على أنه Mutating في الحالات التالية:

- إذا كان الجدول هو نفسه الذي تنفذ عليه جملة الإدخال، التعديل أو الحذف. في هذه الحالة لا تستطيع كتابة جملة استعلام في داخل الزناد المعرف على الجدول لقراءة البيانات من نفس الجدول.
- إذا كان الجدول المعرف عليه الزناد مرتبط بجدأول أخرى من خلال مفتاح أجنبي Foreign Key.

مثال:

اكتب زناداً ينفذ عند إدراج موظف جديد، أو تعديل وظيفة أو راتب موظف موجود مسبقا في جدول الموظفين ويقوم بالتأكد من أن راتب الموظف يقع ضمن المدى المحدد للوظيفة.

```
SQL> CREATE OR REPLACE TRIGGER check_salary
        BEFORE INSERT Or UPDATE OF sal, job ON emp
        FOR EACH ROW
        WHEN (:new.job <> 'PRESIDENT')
        DECLARE
           v_minsalary emp.sal%type;
          v_maxsalary  emp.sal%type;
        BEGIN
           SELECT MIN(sal), MAX(sal)
           INTO v_minsalary, v_maxsalary
           FROM emp
           WHERE job = :new.job;
            IF :new.sal < v_minsalary OR :new.sal > v_maxsalary THEN
                RAISE_APPLICATION_ERROR (-20505, 'out of range');
            END IF;
        END;
/
```

بعد كتابة الزناد أعلاه قم بتنفيذه، ستلاحظ أنه سيتم إنشاء الزناد دون وجود أي خطأ قواعدي فيه.

والآن حأول إجراء عملية إدخال أو تعديل على راتب موظف ما، والحظ الناتج. كما يلي:

```
SQL> UPDATE emp
        SET sal = 1500
       WHERE ename = 'SMITH';
ERROR at line 2
ORA-04091: table EMP is mutating, trigger/function may not see it
ORA-06512: at "CHECK_SALARY", line 5
ORA-04088: error during execution of trigger 'CHECK_SALARY'
```

تلاحظ الآن ان هناك خطأ أثناء تنفيذ الزناد ، وهو أن الزناد يحأول استرجاع بيانات من جدول mutating هو جدول EMP. مما تسبب في فشل الجملة وعدم تنفيذها.

تمارين:

س١)

من المعلوم أن عمليات معالجة البيانات مسموحة خلال أوقات الدوام الرسمي، من الساعه ٨:٠٠ صباحا وحتى الساعه ٦:٠٠ مساء من السبت وحتى الأربعاء. قم بإجراء ما يلي:

١. أنشئ روتينا مخزنا في قاعدة البيانات سمه SECURE_DML والذي سيئودي إلى فشل عملية المعالجه خارج أوقات الدوام الرسمي، ويطبع رسالة:
"لا يسمح لك إجراء معالجة للبيانات في غير أوقات الدوام الرسمي"
٢. أنشئ زناداً على مستوى جملة على جدول PRODUCT والذي يستدعي الروتين في الفرع الأول أعلاه

س٢)

إذا علمت أن رجل المبيعات SALESMAN تتغير عمولته COMM لأي طلبية جديدة أو حتى للطلبيات الموجودة مسبقا. عمولة رجل المبيعات مخزنة في جدول الموظفين EMP ويتم تحديد رجل المبيعات لزبون معين في جدول الزبائن CUSTOMER.

١. اكتب روتينا يقوم بتعديل عمولة رجل المبيعات. استخدم باراميترات لاستقبال رقم الزبون customer ID، فيمة الطلبية القديمة، فيمة الطلبية الجديدة والتي سيتم إرسالها من الزناد الذي يستدعي الروتين. ثم يقوم الروتين بالبحث عن الموظف المسؤول عن هذا الزبون من جدول الزبائن والتعديل على بيانات الموظف في جدول الموظفين EMP بزيادة العمولة الجديدة على عمولة الموظف، افرض أن نسبة العمولة هي ٥٪.

أنشئ زناداً على مستوى سطر على جدول الطلبيات ORD والذي يستدعي الروتين في الفرع الأول من السؤال وتمرير الباراميترات اللازمة.

```
SQL> DESCRIBE emp;
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2 (10)
JOB		VARCHAR2 (9)
MGR		NUMBER (4)
HIREDATE		DATE
SAL		NUMBER (7,2)
COMM		NUMBER (7,2)
DEPTNO	NOT NULL	NUMBER (2)

```
SQL> DESCRIBE dept;
```

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2 (14)
LOC		VARCHAR2 (13)

```
SQL> DESCRIBE salgrade;
```

Name	Null?	Type
GRADE		NUMBER(4)
LOSAL		VARCHAR2 (10)
HISAL		VARCHAR2 (9)

```
SQL> DESCRIBE ord;
```

Name	Null?	Type
ORDID	NOT NULL	NUMBER(4)
ORDERDATE		DATE
COMMPLAN		VARCHAR2 (1)
CUSTID	NOT NULL	NUMBER (6)
SHIPDATE		DATE
TOTAL		NUMBER (8,2)

```
SQL> DESCRIBE product;
```

Name	Null?	Type
PRODID	NOT NULL	NUMBER(6)
DESCRIP		VARCHAR2 (30)

```
SQL> DESCRIBE item;
```

Name	Null?	Type
ORDID	NOT NULL	NUMBER(4)
ITEMID	NOT NULL	NUMBER(4)
PRODID		NUMBER(6)
ACTUALPRICE		NUMBER(8,2)
QTY		NUMBER(8)
ITEMTOT		NUMBER(8,2)

الملاحق

SQL> DESCRIBE customer;

Name	Null?	Type
CUSTID	NOT NULL	NUMBER(6)
NAME	NOT NULL	VARCHAR2 (45)
ADDRESS		VARCHAR2 (40)
CITY		VARCHAR2 (30)
STATE		VARCHAR2 (2)
ZIP		VARCHAR2 (9)
AREA		NUMBER (3)
PHONE		VARCHAR2 (9)
PERID	NOT NULL	NUMBER (4)
CREDITLIMIT		NUMBER (9,2)
COMMENTS		LONG

SQL> DESCRIBE price;

Name	Null?	Type
PRODID	NOT NULL	NUMBER(6)
STDPRICE		NUMER (8,2)
MINPRICE		NUMBER (8,2)
STARTDATE		DATE
ENDDATE		DATE

المراجع

قائمة المراجع العربية

- إصدار خاص في تطبيق أوراكل 8i/8 ، دار خالد سعيد باشماخ، الإصدار الأول – الطبعة الأولى ١٤٢٢ هـ.

قائمة المراجع الأجنبية

- Introduction to SQL and PL/SQL, Oracle University book, volume 2
- Develop PL/SQL Program Units, Oracle University book

المحتويات

تقدر المؤسسة العامة للتعليم الفني والتدريب المهني الدعم

المإلى المقدم من شركة بي آيه إي سيستمز (العمليات) المحدودة

GOTEVOT appreciates the financial support provided by BAE SYSTEMS