الأوراكل

ORACLE®

نتعرض في هذا الفصل الى مقدمة في ادارة قواعد البيانات Database وخاصة قواعد بيانات الأوراكل والذي يعتبر أحد النظم القوية في قواعد البيانات Data Base Management System (DBMS) لما له من مميزات مقارنة بقواعد البيانات الاخرى .

مفاهيم قواعد البيانات

قاعدة البيانات Database

هي مجموعه من الملفات المنظمة بحيث يسهل الوصول إليها عندمت نريد استردادها

قاموس البيانات Data Dictionary

هو مجموعة من المعلومات عن جداول وفهارس البيانات تحفظ داخل هذا القاموس ، يستخدمها نظام إدارة قواعد البيانات

الاستعلام Query

هو نظام استفساري للاستعلام عن بيانات معينة وغاليا لا يغير الاستعلام في قاعدة البيانات حيث إن غالبية نظم الاستعلام عبارة عن قراءة فقط للبيانات

الدوال Function

عبارة عن مجموعة من تعليما ت أو أوامر تستخدم ضمن مسمى وظيفي لاداء عملية محددة وغالبا تعيد الدالة قيمة معينه بعد تنفيذها

الاجراء Procedure

مجموعة من التعليمات مثل تعليمات الدالة بهدف تنفيذ مهمة محددة لكن الاجراء لا يعيد قيمة مثل الدالة

مخطط Schema

عبارة عن مجموعة من الكائنات Objects مرتبطة بقواعد البيانات ويتألف مخطط Schema من كائنات مثل الجداول Tables والاجراءات Procedure والعروض Views والفهارس Index

مدير قواعد البيانات Database Administrator (DBA)

هو الشخص المسؤول عن عمليات إدارة قواعد البيانات ونظم أداء هذه القواعد وكيف يتم تكوينها وهو مسوؤل أيضا عن مراقبة أداء هذه القواعد ، وكذلك إجراء عمليات النسخ الاحتياطي وتثبيت البرامج ، والمحافظة على أمن المعلومات ، وإضافة مستخدمين جدد أو إلغاء صلاحيات مستخدمين

ويمكن ان يقوم مدير قواعد البيانات بالتخطيط لتطوير وتنمية النظام المطبق ، وتحديد الحاجه لهذه التطورات المستقبلية .

ويسعى فريق العمل الذي يتكون من مديري قواعد البيانات DBA للحفاظ على سير العمل داخل الشركة بشكل متجانس . وتتم تجزئة المهام بين هؤلاء المديرين .

مهام مدير قواعد البيانات (DBA)

تتباين مهام مدير قاعدة البيانات تبعا لحجم الشركة أو المؤسسة وتبعا لفريق العمل المساعد وتشمل هذه المهام النقاط التالية :

١- تثبيت البرمجيات الجديدة
٢- إدارة الحماية لنظام قواعد البيانات
٣- النسخ الاحتياطي والدوري للبيانات ونظام قواعد البيانات
٤- استكشاف الأخطاء ومعالجتها
٥- تسوية وإصلاح إخفاقات المستخدمين للوصول إلى بياناتهم
٦- متابعة ضبط أداء العمل
٧- تقسيم الأجهزة والنظم الجديدة
٨- العمل على تطوير النظام بالشركة

نظام إدارة قاعد البيانات Data Base Management System (DBMS)

هو عبارة عن مجموعة الأدوات البرمجيه (البرامج) التي تدير وتنظم قاعدة البيانات وتوجد علاقة ارتباط بين هذه البيانات تسمى علاقة (Relation) لذا احيانا يطلق عليها RDBMS أي نظام إدارة قواعد البيانات العلائقية (الارتباطية)

مميزات نظام إدارة قاعدة البيانات أوراكل

يتميز نظام قاعدة البيانات أوراكل عن غيره من نظم إدارة قواعد البيانات الأخرى بالآتي:

١- القدرة الفائقة على استيعاب كميات كبيرة من البيانات قد يصل عدد السجلات إلى الملايين مع الحفاظ على المستوى العالي في الأداء والسرعة عند استرجاع والتخزين والحذف
٢- السرية التامة والأمن لا حتوائه على نظام الصلاحيات والحقوق الذي يضمن تطبيق الشروط القياسية والأمنية للحفاظ على قاعدة البيانات
٣- فعالية التحكم المركزي بالبيانات الذي يضمن :
- تقليل التكررات غير اللازمة في البيانات المدخلة (No Repetition)
- تجنب التناقض بين البيانات (No Contradiction)
- إمكانية التشارك في البيانات (Data Sharing)
- الحفاظ على تكامل البيانات فيما بينها (Data Integrity)

٤- السيطرة التامة على عملية النسخ الاحتياطي لقاعدة البيانات وحمايتها من الفقدان أو التلف مع امكانية استرجاعها في أي لحظة

عمليات نظام إدارة قاعدة البيانات

يشتمل نظام إدارة قاعدة البيانات اوراكل على العمليات التالية

١- أوامر لغة تعريف البيانات Data Definition Language (DDL)

تستخدم هذه اللغة في تعريف وإنشاء الكائن Object ، ويمكن أن يكون الكائن ملفات وجداول بيانات ، فيمكننا إنشاء وتعديل وحذف الكائن ويمكننا إنشاء امتياز لمستخدم معين ، أو انشاء كائن خيارات لفحص وإضافة تعليقات إلى قاموس البيانات ومن هذه الأوامر : CREAT , DROP and ALTER

٢- أوامر لغة معاملة البيانات Data Manipulation Language (DML)

تتيح هذه الأوامر التعامل مع البيانات وتعديلها ضمن الكائن الموجود Object ومن هذه الأوامر : SELECT, DELETE,UPDATE and INSERT

٣- أوامر لغة التحكم في البيانات Data Control Language (DCL)

تتيح هذه الأوامر التحكم في قاعدة البيانات وأدائها كالصلاحيات والمستخدمين والحقوق وغالبا ماتكون هذه الأوامر مخصصة للاستخدام من قبل مدير قاعدة البيانات (DBA) ومن هذه الاوامر : GRANT and REVOKE

التركيب الداخلي لنظام أوراكل

أهداف الفصل

يتناول هذا الفصل التركيب الداخلي لنظام قاعدة البيانات أوراكل وفي نهاية هذا الفصل ستكون قادرا بمشيئة الله على :

١- فهم العلاقات بين الجداول
٢- تعريف نمذجة العلاقة
٣- فهم مكونات قاعدة البيانات العلائقية
٤- فهم العلاقة بين الخادم والمستفبد
٥- قهم الخادم SERVER
٦- فهم المستفيد Client

مقدمة في قواعد البيانات العلائقية

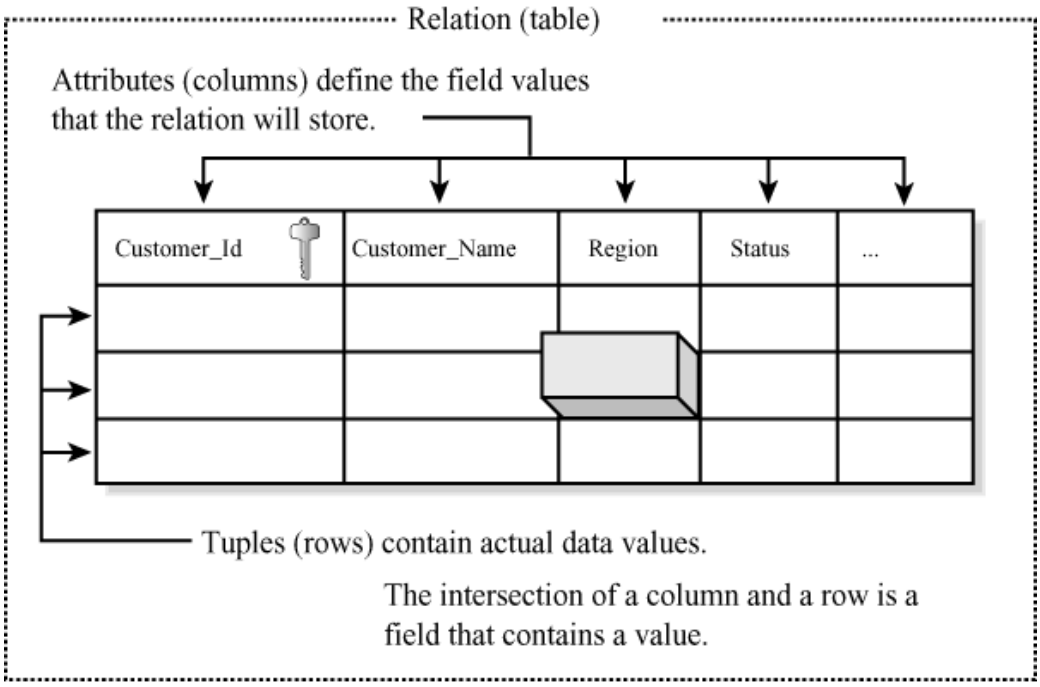
يقدم هذا الفصل التركيب والبناء الداخلي لنظام إدارة قواعد البيانات أوراكل ، فيحتوي على وصف سريع لمكونات أوراكل وتعتمد غالبية تطبيقات نظم قواعد البيانات في بنائها على أحد نماذج قاعد البيانات التالية:

١- نموذج هيكلي (هرمي) Hierarchical Model
٢- نموذج شبكي Network Model
٣- نموذج علائقي Relational Model
٤- نموذج شيئي Object Model

في الماضي كان النموذج الأول (الهرمي) الأكثر انتشارا مع أجهزة الكمبيوتر الكبيرة Main Frame ثم بدا النظام الثاني (الشبكي) في الانتشار وخاصة مع التوسع في بناء وتركيب شبكات الحاسب وكانت هناك صعوبات في استخدام النظامين الاول والثاني (الهرمي والشبكي) نتيجة لاستخدام مؤشرات البرمجة (Pointers) لربط سجلات البيانات بعضها مع بعض ، لذا نجد أن إضافة أو تعديل أو حذف السجلات يحتاج إلى المزيد من قهم طبيعة وعمل المؤشرات وفي هذه الفترة الماضية طرق برمجة المؤشرات تكتب بلغة الكوبول (COBOL).
أما النموذج العلائقي (Relational) فهو الأكثر سهولة في الاستخدام وفي برمجة تطبيقات نظم قواعد البيانات ومن الناحية النظرية الأكاديمية فيلزمك دراسة المفاهيم الأساسية والضرورية لهذا النموذج العلائقي لهذا سنركز هنا على

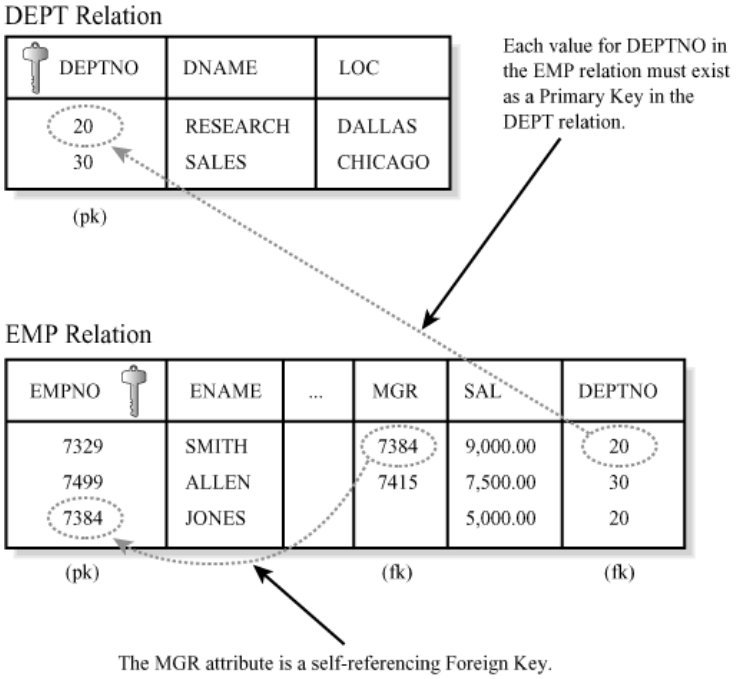
١- هيكل البيانات العلائقية Relational Data Structure
٢- الضوابط الحاكمة للبيانات العلائقية Constraint that govern organization of data structure
٣- العمليات التي تجرى على هذه البيانات Operations that are performance data structure

ويعتمد نظام هياكل البيانات العلائقبة (Relational Database) على هيكل منطقي ويطلق عليه علاقة (Relational)وعلى شكل ثنائي الأبعاد (Tow dimensional data structure) يتكون من صفوف وأعمدة ويسمى جدول (Table) هذا بالإضافة إلى عناصر البيانات (Data elements) تسمى في هذه الحالة attributes علاوة على هذا يتم تنظيم هذه البيانات الفعلية في وحدة أو وحدات تسمى Tuples تقسم إلى صفوف Rows أو (سجلات Records) وأعمدة Columns (حقول Fields) والشكل التالي يبين العناصر الأساسية لجدول الموظفين Employees



العلاقة بين الجداول

لربط علاقة بأخرى تتحتاج إلى طريقة للارتباط . وهذه الطريقة تستخدم حقلا Field يكون موجودا في الجدولين وحقل الارتباط يسمى في الجدول الأول بالمفتاح الرئيسي (Primary Key) للجدول الأول ويسمى بالمفتاح ال (Foreign Key) في الجدول الثاني



نمذجة العلاقة Relational Model Algebra

تعرف بانها العمليات التي يتم اجراؤها على جدول أو مجموعة من الجداول تبعا لعلاقة محددة ويوجد معاملان Operators هما Unary والثانيBinary والجدول التالي يعدد سبعة انواع لهذه العمليات

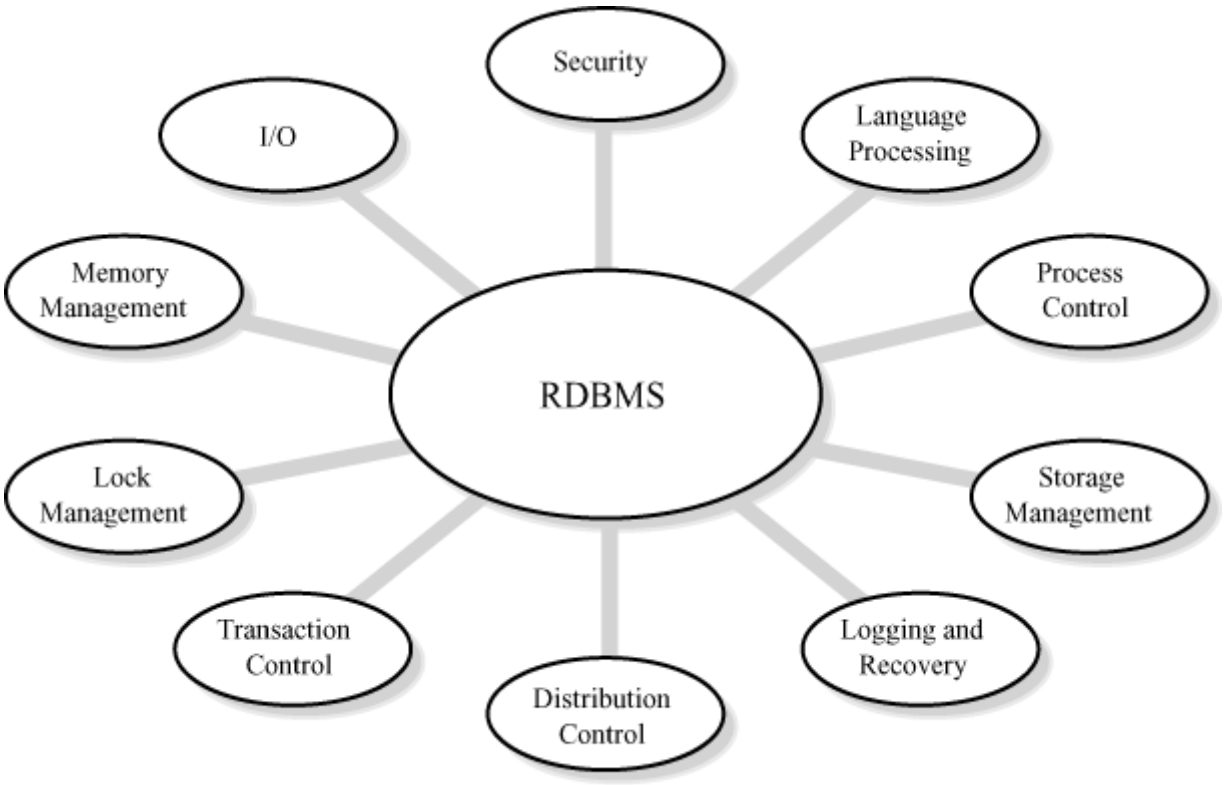
وصف العملية	نوع العملية Type	العمليةOperation
تجميع الصفوف السجلات من جدولين مع عدم السماح بتكرار سجلات	Binary	Union
تحديد السجلات (الصفوف) المشتركة بين جدولين	Binary	Intersection
اظهار السجلات الموجودة في الجدول الاول ولا توجد في الجدول الثاني	Binary	Difference
إظهار السجلات مع بعض الأعمدة (مصدر البيانات)	Unary	Projection
إظهار السجلات من جدول مصدر البيانات تبعا لمعيار البحث Criteria	Unary	Selection
وصل كل سجل من جدول البيانات الاول مع كل سجل في الجدول الثاني	Unary	Product
وصل وتمديد السجلات من الجدول الأول مع مايقابله من سجلات في الجدول الثاني	Unary	Join

مكونات قاعدة البيانات العلائقية RDBMS Components

تتكون قاعدة البيانات العلائقية من :

١- نظام تشغيل ثاعدة البيانات ويطلق عليه Kernel
٢- قاموس البيانات Data Dictionary

ويتكون نظام تشغيل وتنظيم قاعدة البيانات (Kernel) من مجموعة من الوحدات البرمجية Software والذي صمم ليحكم وينظم وينعامل مع البيانات مابين حفظ واسترجاع وطباعة وكذلك تحديد المسؤليات وعمل نظم الأمان وحماية البيانات
وعادة يحتفظ نظام قاعدة البيانات بقائمة من المستخدمين الذين لهم صلاحية للتعامل مع تطبيق قاعدة البيانات والشكل التالي يوضح بعضا من مكونات الـ Kernel في نظام أوراكل

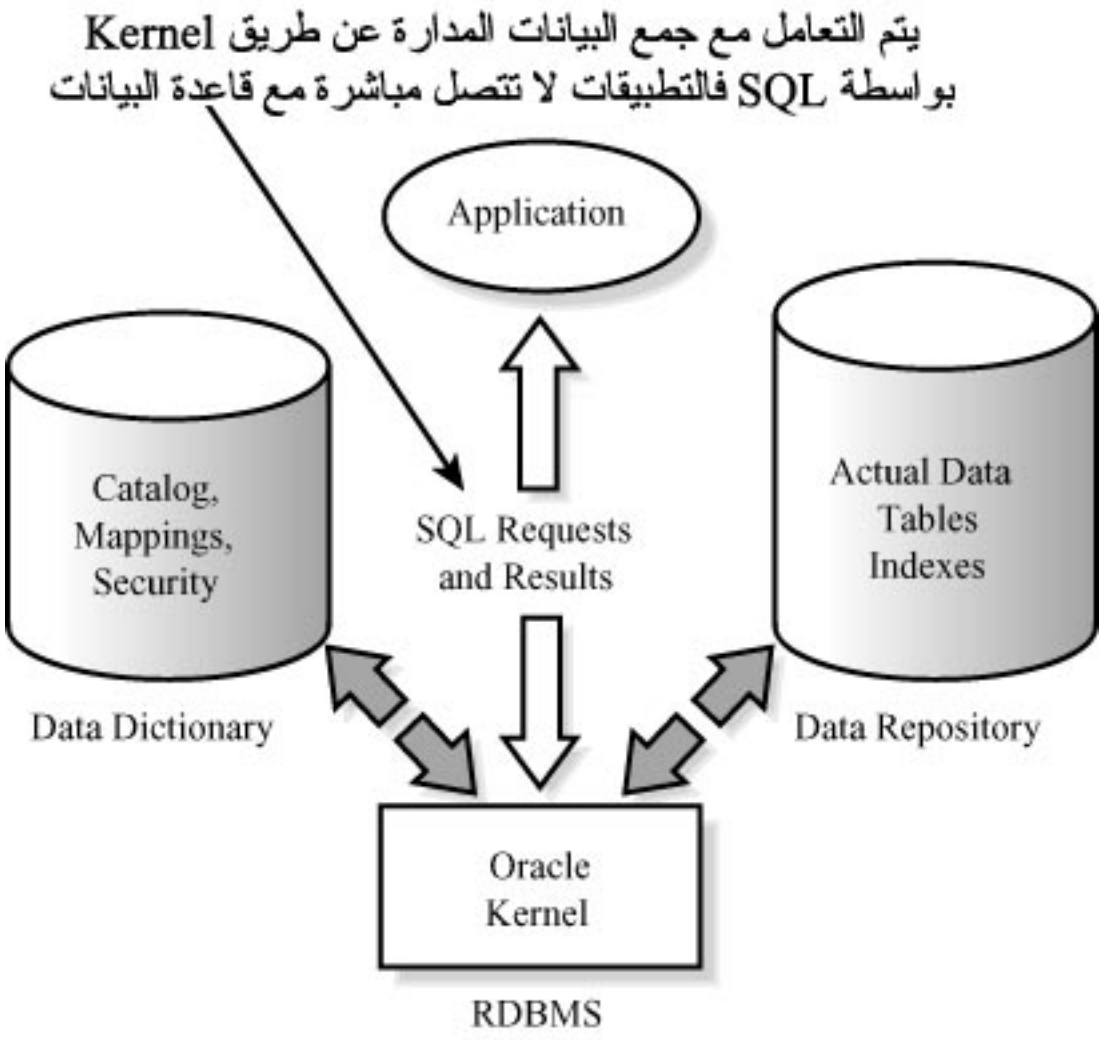


مكونات الـ Kernel في نظم قاعدة البيانات

ويحوي قاموس البيانات Data Dictionary وصفا لشكل البيانات ويطلق على هذا الوصف Meta Description أو Meta Data وهذا الوصف يرتبط بكافة الكائنا ت Objects ضمن قاعدة البيانات

وقاموس بيانات أوراكل يحتوي على مجموعة جداول تضم البيانات المخزنة عن طريق الـ Kernel وأيضا على كائنات الفهارس Indexesكائنات العرض Views كائنات الاستعلام والاستقسار SQL and Query كائنات الإجراء Procedure والدوال Function ...إلخ

الشكل التالي يوضح كيفية قراءة أو تعديل قاعدة بيانات باستخدام أوراكل



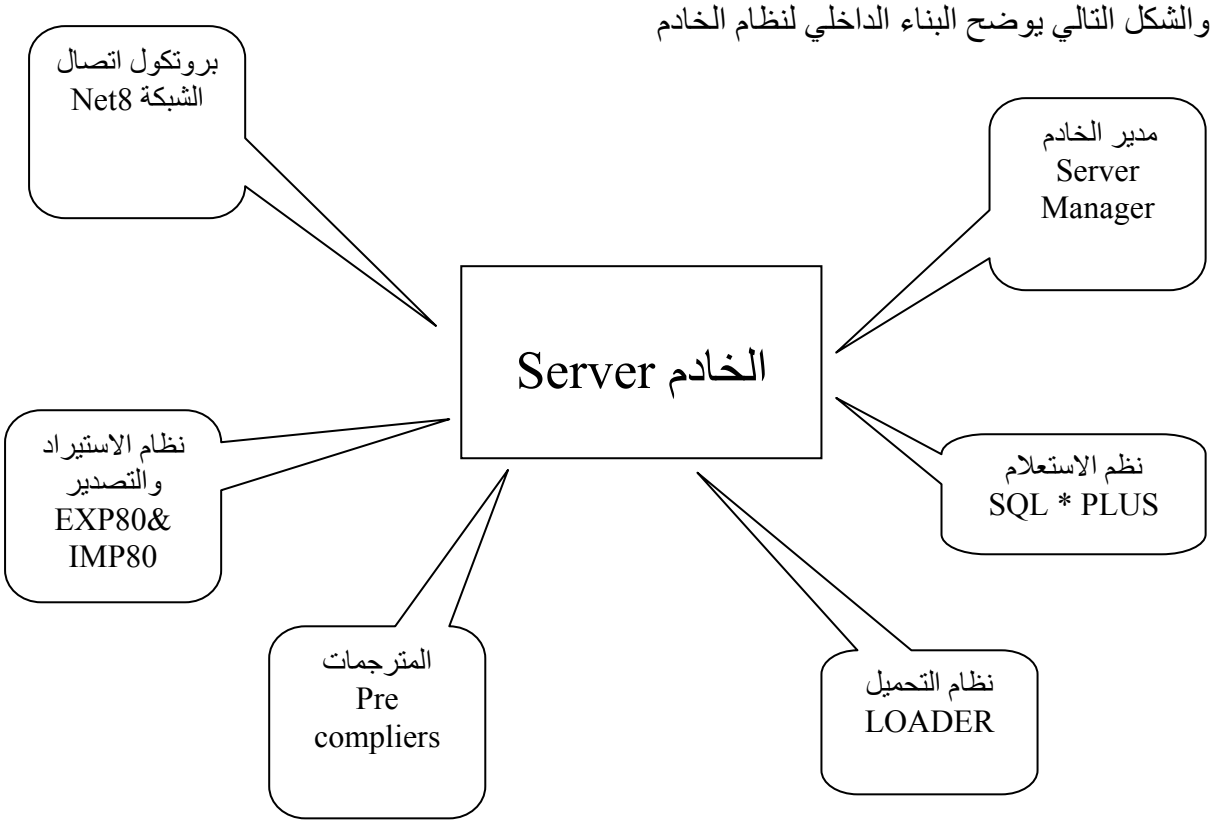
شكل العلاقة بين Kernel و SQL و قاموس البيانات

العلاقة بين الخادم والمستفيد(العميل) Client/Server

يتكون نظام أوراكل من مكونين
١- الخادم Server
٢- المستفيد (العميل – المزود) Client

الخادم Server

يحوي الخادم قاعدة البيانات المركزية ووظائفها ، وكذلك كافة العمليات الخلفية لحفظ هذه القاعدة
ويحوي ايضا أدوات تشغيل وإيقاف قاعدة البيانات
والشكل التالي يوضح البناء الداخلي لنظام الخادم



المستفيد Client

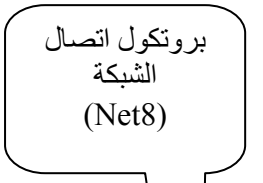
أما المستفيد Client فيحتوي على برامج الخدمات والبرامج المساعده والتي بمكننا تشغيلها عن بعد باستخدام الشبكة كما يتضمن نظام المستفيد الوسيط الرسومي Graphical Interface الذي يسهل علينا استخدامه وتوجد أداة الاتصال بين الخادم والمستقيد وهيNet 8 لتقوم بعمليات الربط والاتصال عبر الشبكة

و تشمل حزمة البرامج لنظام المستفيد على
١- مدير المشروع Enterprise Manager
ويحوي داخله المكونات التالية

- متحكم مدير المشروع Enterprise Manager
- مدير مخطط قاعدة البيانات Schema Manager
- مدير السرية Security Manager
- مدير الطبعة Instance Manager
- مدير التخزين Storage Manager
- مدير البيانات Data Manager
- مدير النسخ الاحتياطي للبيانات Backup and Recovery Manager
- ورقة عمل الاستعلام SQL Worksheet
- شريط أدوات المسؤول Administrator Toolbar

٢- المساعدون Assistants
٣- نظام الاستعلام SQL*PLUS

والشكل التالي يوضح مكونات حزمة البرامج للمستفيد Client



نظام الاستعلام SQL*PLUS		
المساعدون Assistants		
مدير المشروع Enterprise Manager		
متحكم مدير المشروع Enterprise Manager	مدير مخطط قاعدة البيانات Schema Manager	ورقة عمل الاستعلام SQL Worksheet
مدير السرية Security Manager	مدير النسخ الاحتياطي للبيانات Backup and Recovery Manager	مدير الطبعة Instance Manager
مدير البيانات Data Manager	مدير التخزين Storage Manager	شريط أدوات المسؤول Administrator Toolbar

ملخص الوحدة

تناولنا في هذا الفصل تعريف التركيب الداخلي لنظام أوراكل وأهم مكوناته والعلاقة بين الجداول ، وكذلك عملية نمذجة العلاقة على الجداول وتم تعريف الخادم والمستفيد والعلاقة بينهما .

جمل SQL البسيطة

أهداف الفصل

نتااول عملية استرجاع البيانات في حالات مختلفة مستخدمين في ذلك جملة SELECT وفي نهاية هذا الفصل ستكون إن شاء الله قادر على :

- استرجاع البيانات بواسطة جملة SELECT البسيطة
- فهم متطلبات وإرشادات كتابة جمل SQL
- استخدام العمليات الحسابية وأولويات تنفيذها مع جمل SQL
- استخدام الجمل الإلحاقية على البيانات
- استخدام عبارة DISTINCT لمنع تكرار عرض البيانات
- عرض مواصفات الجدول

أساسيات جملة SELECT

جملة SELECT

تستخدم جملة SELECT لاسترجاع البيانات المخزنة في جدول أو عدة جداول حيث أن عملية الاسترجاع لا تعدل في هذه البيانات ويمكننا من خلال جملة SELECT أن نقوم بالتالي:

١- اختيار وعرض مجموعة معينة من السجلات المخزنة في الجدول
٢- استرجاع بيانات بعض حقول الجدول
٣- استرجاع بيانات مخزنة في جداول مختلفة

ملاحظة
يمكن أن نستخدم كل أو بعض هذه المزايا في جملة استفسار واحدة

ملاحظة
أوامر محررSQL*PLUs هنالك بعض الاوامر البسيطة التي ستساعدك في كتابة وتحرير وتنفيذ الأوامر على محرر SQL ومنها
١- الامر EDIT ويمكن كتابته ED : يستخدم هذا الأمر لتحرير آخر امر تم كتابته على محرر SQL وعند تنفيذ هذا الأمر ستفتح لك شاشة المحرر (المفكرة) التي يمكنك خلالها اعادة تحرير الأمر وعند الانتهاء من ذلك احفظ الامر ثم اغلق شاشة المفكرة ولكن لاحظ لا تكتب الفاصلة المنقوطة (;) بعد نهاية الأمر في هذه الحالة فقط ٢- الأمر R وطريقة كتابته (/) ويستخدم لاعادة تنفيذ اخر امر SQLمحفوظ

الشكل العام لجملة SELECT

```
SELECT { * , COLUMN [alies], …}
FROM table_name ;
```

حيث أن
COLUMN اسم الحقل
table_name اسم الجدول
alies تسمية العمود

يمكن أن نضع الرمز * للدلالة على استرجاع جميع حقول الجدول

كتابة جمل SQL

نوضح فيما يلي بعض القواعد الإرشادية التي يجب أن توضع بعين الاعتبار عند كتابة جمل SQL

١- يمكن كتابة جمل SQL بالحروف الكبيرة أو الصغيرة
٢- يمكن كتابة جمل SQL في عدة أسطر

٣- لا يمكن فصل الكلمات المحجوزة عبر السطور مثل FROM
٤- اترك مساقات بين مكونات الجملة لتسهيل عملية القراءة
٥- في برنامج SQL* PLUS تكتب الأوامر مع مؤشر SQL ويتم تخزين هذا الامر مباشرة في الذاكرة

مثال

```
SELECT * FROM s_dept;
```

```
        ID  NAME                    REGION_ID
  --------  -------------          ----------

        10  Finance                         1
        31  Sales                           1
        32  Sales                           2
        33  Sales                           3
        34  Sales                           4
        35  Sales                           5
        41  Operations                      1
        42  Operations                      2
        43  Operations                      3
        44  Operations                      4
        45  Operations                      5
        50  Administration                  1
12 rows selected.
```

لتنفيذ جمل SQL

١- نضع فاصلة منقوطة في نهاية الجملة في مؤشر SQL
٢- نضع علامة / في نهاية الجملة في مؤشر SQL لاعادة اخر امرموجود في ال(Buffer)
٣- نضع علامة / في نهاية الجملة في الأمر في الذاكرة (BUFFER)
٤- كتابة عبارة RUN او R في مؤشر SQL لاعادة اخر امرموجود في ال(Buffer)

العمليات الحسابية

يمكن أن نستخدم العمليات الحسابية عند عرض البيانات دون أن تؤثر هذه المعالجة على البياانات المخزنة في الجدول ، ويمكن أن يحتوي التعبير الحسابي على اسم الحقل قيمة ثابتة و العملية الحسابية

العمليات الحسابية

العمليات التي يمكن أن تستخدم هي:

١- الجمع +
٢- الطرح -
٣- القسمة ÷
٤- الضرب *

ويمكن استخدام هذه العمليات في جميع أجزاء جملةSELECT عدا الجزء الخاص بـFROM

أولويات العمليات

١- الافواس الداخلية ثم الخارجية
٢- الضرب والقسمة
٣- الجمع والطرح

ملاحظة
العمليات التي لها نفس الاولية كالجمع والطرح تنفذ من اليسار الى اليمين

العملية الإلحاقية

يمكن أن نجري عملية إلحاق عدد من الأعمدة للجدول أو مجموعة حرفية نصية إلى حقول أخرى عند العرض باستخدام العملية الإلحاقية ||

مثال

```
SELECT   first_name||last_name
FROM     s_emp;

Employees
-------------------------------------------------
CarmenVelasquez
LaDorisNgao
MidoriNagayama
MarkQuick-To-See
```

```
AudryRopeburn
MollyUrguhart
...
```

نلاحظ ان الاسم الاول اتصل بالاسم الاخير بدون وجود مسافات فظهر كانه اسم واحد ولكي توجد مسافات نكتب

```
SELECT first_name ||' '|| last_name FROM s_emp;
```

استخدام عبارة DISTINCT

تستخدم عبارة DISTINCT لمنع تكرار ظهور بيانات السجل المسترجع فمثلا إذا أردنا أن نعرف على الوائف التي ينتسب إليها الموظفون والمخزنة في جدول الموظفين نجد أن هنالك وظائف تتكرر حسب عدد الموظفين الذين ينتمون إليها لمنع تكرار نستخدم عبارة DISTINCT

ويأخذ الأمر SELECT الشكل التالي:

SELECT [DISTINCT] { * , COLUMN [alies], …}
FROM table_name ;

حيث أن
COLUMN اسم الحقل
table_name اسم الجدول
alies تسمية العمود
DISTINCT لمنع التكرار

ملخص الفصل

تناولنا في هذا الفصل كتابة جمل SQL البسيطة (جملة SELECT) التي تستخدم في استرجاع البيانات وكذلك بعض القواعد الإرشادية التي يجب اتباعها عند كتابة وتنفيذ جمل الـSQL كما تناولنا استخدام العمليات الحسابية مع جملة SELECT والعملية الإلحاقية وكذلك عبارة DISTINCT التي تستخدم لمنع تكرار ظهور السجلات

استرجاع البيانات بشروط ترتيبها

اهداف الفصل

١- المعرفة التامة لأسماء الجداول وأسماء الحقول وخصائصها وانواع البيانات
٢- التعامل مع جملة SQL البسيطة (SELECT Statement)
٣- استخدام أوامر محرر SQL

استخدام جملة الشرط WHERE CLAUSE

مقدمة

لقد قمنا في الوحدة السابقة بالتعرف الى جملة الاستعلام البسيط (SELECT Statement) التي من خلالها تم استرجاع البيانات من الجداول . وفي هذا الفصل سنتابع الحديث عن هذه الجملة يشكل أوسع ، حيث سنسترجع البيانات من الجداول بناء على شروط معينة ، أو مرتبة حسب بيانات حقول معينة ، أي سندرس الامور المتعلقة بجملة الشرط في جملة الاسترجاع ، من حيث المعاملات الشرطية وطريقة التعبير عن الشرط والترتيب التصاعدي والترتيب التنازلي للبيانات

الصيغة العامة لجملة الشرط

```
SELECT {Field1, Field2, ...,...|*}
FROM Table1
WHERE Condition;
```

حيث أن

Field1.Field2	أسماء الحقول التي سيام استرجاع بياناتها
Table1	أسم الجدول الذي سيتم استرجاع البيانات منه
Condition	جملة شرطية أو أكثر تكون نتيجتها إما True او False وتتكون من أسماء حقول وتعبيرات ومعاملات

الشرح

تستخدم جملة WHERE لحصر البيانات التي سيتم استرجاعها من الجداول وهي تحتوي على شرط وتقع مباشرة بعد المقطع FROM وعند تنفيذ الجملة سيتم استرجاعها بيانات الحقول المذكورة التي تحقق الشرط المذكور في جملة WHERE ويمكن أن تقارن الجملة الشرطية بين قيم أو حقول في الجدول أو تعبيرات حسابية أي أن جملة WHERE تحتوي على مايلي:

- أسماء الحقول
- معاملات المقارنة
- قيم ثابتة
- متغيرات وتغبيرات حسابية

الأمور التي يجب أخذها بعين الاعتبار

١- عند استخدام حقول النص وحقول التاريخ في جملة الشرط يجب وضع القيم بين علامتي تنصيص مفردتين (' ')
٢- في حالة الحقول النصية يجب مراعاة حالة الأحرف كبيرة أو صغيرة
٣- في حالة حقول التاريخ يجب مراعاة صيغة التاريخ (FORMAT) والصيغة الأسا سية للتاريخ هي كما يلي DD-MON-YY

معاملات المقارنة COMPARISON OPERATORS

تشمل معاملات المقارنة الرموز التالية:

المعامل	المعنى
=	يساوي
>	أكبر من
<	أقل من
>=	أكبر من أو يساوي
<=	اقل من أو يساوي
<> !=	لا يساوي لا يساوي

تستخدم معاملات المقارنة في جملة الشرط لمقارنة تعبير بآخر في جملة WHERE كما في الصيغة التالية :

WHERE تعبير OPERATOR تعبير

مثال

WHERE hiredate='01-SEP-96'
WHERE sal>=1500
WHERE name='Ahmed'
WHERE sal<> other

معاملات مقارنة أخرى

المعامل	نفى المعامل	المعنى
BETWEEN	NOT BETWEEN	بين قيمتين
IN	NOT IN	ضمن قائمة من القيم
LIKE	NOT LIKE	مطابقة نمط النص
IS NULL	IS NOT NULL	هل هي قسمة فارغة

١- المعامل BETWEEN

يستخدم هذا المعامل لاسترجاع بيانات الصفوف التي تعتمد على مدى من القيم أي البيانات التي تقع بين قيمتين

مثال

```
SELECT * FROM S_EMP
WHERE sal BETWEEN 1500 AND 2500;
```

أي أنه سيتم استرجاع جميع بيانات الموظفين الذين اقع رواتبهم بين ١٥٠٠و ٢٥٠٠

ملاحظة
ستم استرجاع بيانات الموظف الذي يبلغ راتبه تماما ١٥٠٠ أو ٢٥٠٠ أي أن جملة **BETWEEN** تتضمن الحد الأعلى والحد الأدنى للقيم التي سيتم استرجاعها

٢- المعامل IN

يستخدم هذا المعامل للبحث عن قيمة داخل قائمة من القيم فتحتوي هذه القائمة قيما ثابتة أو قد تكون هذه القائمة عبارة عن جملة استعلام فرعي

مثال

```
SELECT name, sal , deptno FROM S_EMP
WHERE  detno IN (10,30);
```

أي أنه سيتم استرجاع اسم الموظف وراتبه ورقم القسم الذي يعمل به للموظفين الذين يعملون في قسم رقم ١٠ **أو** قسم رقم ٣٠

٣- المعامل LIKE.

يستخدم هذا المعامل للبحث عن نص معين داخل حقل نصي ، حيث سيتم مطابقة حروف النص المذكورة في جملة الشرط

مثال

```
SELECT * FROM S_EMP
WHERE name LIKE 'S%';
```

أي أنه سيتم استرجاع جميع بيانات الموظفين الذين تبدأ أسماؤهم بحرف S

ملاحظة
تستخدم الاشارة % عوضا عن أي قيمة نصية ، قد تكون لا شيء أو أي عدد من الحروف للبحث مثلا عن اسم معين ينتهي بحرف A نكتب '%A' وللبحث عن نص يحتوي حرف A نكتب '%A%' وهكذا بإمكاننا أن نبحث عن أي مقطع داخل حقل نصي معين

مثال

```
SELECT * FROM S_EMP
WHERE name LIKE '_A%';
```

أي أنه سيتم استرجاع جمبع بيانات الموظفين الذين يكون الحرف الثاني في أسمائهم هو A

ملاحظة
تستخدم الإشارة _ عوضا عن حرف واحد فقط

٤- المعامل IS NULL

يستخدم هذا المعامل لفحص القيمة (لا شيء) أي قيم القول التي لا تحتوي على بيانات

مثال

```
SELECT name, sal FROM S_EMP
WHERE comm. IS NULL;
```

أي أنه سيتم استرجاع اسم الموظف وراتبه للموظفين الذين ليس لهم عمولة أو لم يتم ادخال عمولتهم

المعاملات المنطقية LOGICAL OPERATORS

المعامل	المعنى
AND	يرجع TRUE اذا كانت كلتا القيمتين TRUE
OR	يرجع TRUE اذا كانت احدى القيمتين TRUE
NOT	يرجع TRUE إذا كانت القيمة FALSE

تستخدم المعاملات المنطقية لترجع قيمة واحدة TRUE أو FALSE كنتيجة بين قيمتين منطقيتين او لعكس قيمة منطقية من TRUE الى FALSE والعكس
من خلال المعاملات المنطقية يمكننا تكوين أكثر من شرط داخل جملة WHERE باستخدام المعاملAND او المعامل OR

١- المعامل AND

يتطلب هذا المعامل أن تكون كلا القيمتين TRUE

مثال

```
SELECT name, sal FROM S_EMP
WHERE sal>=2600 AND comm<200;
```

أي أنه سيتم استرجاع اسم الموظف وراتبه للموظفين الذين يزيد راتبهم عن ٢٦٠٠ وتقل عمولتهم في نفس الوقت عن ٢٠٠ أي انه يجب يتحقق الشرطين لاسترجاع البيانات

٢- المعامل OR

يتطلب هذا المعامل أن تكون أي من القيمتين TRUE

مثال

```
SELECT name, sal FROM S_EMP
WHERE sal <200٠ OR deptno=30;
```

أي أنه سيتم استرجاع اسم الموظف وراتبه للموفين الذين يزيد راتبهم عن ٢٠٠٠ أو يعملون في قسم رقم ٣٠ أي أنه يجب يتحقق أي من الشرطين لاسترجاع البيانات

٣- المعامل NOT

يقوم هذا المعامل بعكس قيمة منطقية من TRUE إلى FALSE والعكس

مثال

```
SELECT name, sal FROM S_EMP
WHERE deptno NOT IN (20,40);
```

أي أنه سيتم استرجاع اسم الموظف وراتبه للموظفين الذين لا يعملون في القسمين رقم ٢٠ و رقم ٤٠ وفي هذه الحالة تم عكس نتيجة المعامل IN

أولويات المعاملات المنطقية ومعاملات المقارنة

إذا ورد أكثر من معامل منطقي في نفس جملة الشرط فإن أولوية تنفيذ هذه المعاملات من الأعلى إلى الأدنى غي كالتالي :

١- الأقواس
٢- معاملات المقارنة (< ، > ، = ، <>)
٣- NOT
٤- AND
٥- OR

استخدام جملة الترتيب ORDER BY

الصيغة العامة

```
SELECT {Field1, Field2,…,…|*}
FROM Table1
WHERE Condition
ORDER BY Field3 [ASC|DESC];
```

حيث أن

Table1	اسم الجدول الذي سيتم استرجاع البيانات منه
Field1, Field2	اسماء الحقول التي سبتم استرجاع بياناتها
Condition	جملة شرطية أو اكثر تكون نتيجتها إما True أو False وتتكون من اسماء حقول وتعبيرات ومعاملات منطقية
Field3	اسم الحقل الذي سيتم الترتيب بناء عليه
ASC	استرجاع البيانات مرتبة تصاعديا ويجوز عدم ذكرها
DESC	استرجاع البيانات مرتبة تنازليا

الشرح

عند استرجاع البيانات تستخدم جملة ORDER BY لترتيب البيانات حسب حقل معين أو عدة حقول وتأتي هذه الجملة في نهاية جملة الاسترجاع SELECT وقد يكون هذا الترتيب تصاعديا (من القيمة الصغيرة إلى القيمة الكبيرة) عند استخدام المقطع ASC وقد يكون تنازليا (من الفيمة الكبيرة إلى القيمة الصغيرة) عند استخدتم المقطع DESC

مثال

```
SELECT * FROM S_EMP
WHERE deptno NOT IN (20,40)
ORDER BY empno ASC;
```

أي أنه سيتم استرجاع جميع بيانات الموظفين الذين لا يعملون في القسمين رقم ٢٠ ورقم ٤٠وستكون البيانات مرتبة **تصاعديا** حسب رقم الموظف

ويجوز كتابة الجملة السابقة دون المقطع ASC كمايلي:

```
SELECT * FROM S_EMP
WHERE deptno NOT IN (20,40)
ORDER BY empno;
```

مثال

```
SELECT * FROM S_EMP
WHERE deptno NOT IN (20,40)
ORDER BY empno DESC;
```

أي أنه سيتم استرجاع جميع بيانات الموظفين الذين لا يعملون في القسمين رقم ٢٠ ورقم ٤٠وستكون البيانات مرتبة **تنازليا** حسب رقم الموظف

ملخص الفصل

تناولنا من خلال هذا الفصل موضوع استرجاع البيانات بشروط من الجداول وموضوع ترتيب البيانات تصاعديا وتنازليا كما وتطرقنا من خلال دراسة جملة الشرط إلى معاملات المقارنة والمعاملات المنطقية ومعاملات أخرى مثل BETWEENو INو IS NULL ومايتعلق بها من حيث طريقة الاستخدام وتركيب جملة الشرط وأولويات معاملات المقارنة والمعاملات المنطقية.

الدوال الحرفية

اهداف الفصل

في نهاية هذا الفصل إن شاء الله ستكون قادر على :

١- تحويل الحروف من صغيرة إلى كبيرة وبالعكس من خلال SQL
٢- ضبط ومحاذاة الحروف داخل النص الموجود فيSQL
٣- القيام بحذف البيانات من الجداول
٤- التعامل مع اللفظ الصوتي للبيانات
٥- القيام بتعديل بيانات الجدول

الدال الحرفية والضبط والمحاذاة

دالة تحويل البيانات إلى حروف صغيرة LOWER Function

تستخدم هذه الدالة لتحويل البيانات من حروف كبيرة إلى حروف صغيرة ويتم استخدام هذه الدالة غاليا مع جمل أخرى

الشكل العام انظر المثال التالي

مثال .

```
SELECT LOWER (name), LOWER (job)
FROM S_EMP;
```

سيتم عرض اسم الموظف ووظيفته ولكن بحروف صغيرة

LOWER name	LOWER job
ali	salesman
ahmed	analyst
sami	manager
khaled	manager

دالة تحويل البيانات إلى حروف كبيرة UPPER Function

تستخدم هذه الدالة لتحويل البيانات من حروف صغيرة إلى حروف كبيرة ويتم استخدام هذه الدالة غاليا مع جمل أخرى

مثال

```
SELECT UPPER (name), UPPER (job)
FROM S_EMP;
```

دالة تحويل الحرف الأول من البيانات إلى حرف كبير INITCAP

تستخدم هذه الدالة لتحويل أول حرف من بيانات الحقل المحدد إلى حرف كبير ويتم استخدام هذه الدالة غالبا مع الجمل أخرى

مثال

```
SELECT INITCAP (name), INITCAP (job)
FROM S_EMP;
```

سيتم عرض اسم الموظف ووظيفته ولكن اول حرف في كلاهما سيكون كبير

INITCAP name	INITCAP job
Ali	Salesman
Ahmed	Analyst
Sami	Manager
Khaled	Manager

دالة اقتطاع (قص) جزء من بيانات الحقل SUBSTR

تستخدم لعرض أو قص جزء معين من بيانات العمود او الحقل

الصيغة العامة

```
SUBSTR(Field1,N,M)
```

حيث أن

String	الحقل المراد الاقتطاع منه
N	اول حرف يبدأ عنده الاقتطاع
M	عدد الحروف المراد اقتطاعها

مثال

```
SUBSTR('Ahmed',1,3)
```

قيكون الناتج
Ahm

مثال

```
SELECT SUBSTR (name, 1,2)
 FROM S_EMP;
```

سيتم اقتطاع الحرف الأول والثاني من اسم الموظف وعرضها

دالة تحديد موقع حرف في بيانات حقل INSTR Function

تستخدم هذه الدالة لتحديد مكان أو موقع حرف في بيانات العمود أو الحقل

الصيغة العامة

INSTR (Field, 'C')

حيث أن

Field	الحقل الذي سيتم التحديد منه
C	الحرف المراد استخراج موقعه

مثال

```
SELECT INSTR (name, 'l')
 FROM S_EMP;
```

سيتم استخراج موقع الحرف l من حقل اسم الموظف

```
INSTR name
--------------------
2
0
0
4
```

دالة ضبط ومحاذاة ناحية اليمين للبيانات RPAD Function

تستخدم هذه الدالة لمحاذاة البيانات ناحية اليمين حيث يتم ملء حرف معين (أو حروف) يمين البيانات

الشكل العام

```
RPAD (COL | VALUE, N, ' String')
```

حيث أن

COL	اسم الحقل المطلوب محاذاة بياناته ، ووضع حرف (حروف) يمينه
VALUE	يمكن وضع قيمة أو متغير حرفي بين علامتي تنصيص (" ")
N	عدد مرات ظهور الحرف الجديد

String الحرف أو المتغير الذي سيتم ملء الفراغات به

مثال

```
SELECT name, RPAD (sal, 7,'$')
FROM S_EMP;
```

NAME	RPAD SAL
Ali	968$$$$
Ahmed	1936$$$
SAMI	1512.5$
KHALED	3599.75

دالة ضبط ومحاذاة ناحية اليسار للبيانات LPAD Function

تستخدم هذه الدالة لمحاذاة البيانات ناحية اليمين حيث يتم ملء حرف معين (أو حروف) يسار البيانات

الشكل العام

LPAD (COL | VALUE, N, ' String')

حيث أن

COL اسم الحقل المطلوب محاذاة بياناته ، ووضع حرف (حروف) يمينه
VALUE يمكن وضع قيمة أو متغير حرفي بين علامتي تنصيص (" ")
N عدد مرات ظهور الف الجديد
String الحرف أو المتغير الذي سيتم ملء الفراغات به

مثال

```
SELECT name, LPAD (sal, 7,'#')
FROM S_EMP;
```

NAME	LPAD SAL
Ali	####968
Ahmed	###1936
SAMI	#1512.5
KHALED	3599.75

حذف و تعديل البيانات

دالة حذف بيانات ناحية يمين الحقل RTRIM Function

تستتخدم هذه الدالة لحذف بيانات من ناية يمين العمود او الحقل المحدد

الصيغة العامة

RTRIM (COL | VALUE,' String')

حيث أن

COL	اسم الحقل أو العمود
VALUE	القيمة البديلة للعمود
String	قيمة الحرف أو المتغير الذي سيتم البحث عنه

مثال

```
SELECT name, RTRIM (job،'man')
FROM S_EMP;
```

سيتم حذف man من يمين العمود job

name	RTRIM job
ALI	Sales
AHMED	Analyst
SAMI	Manager
Khaled	Manager

دالة حذف بيانات يسار الحقل LTRIM Function

تستخدم هذه الدالة لحذف بيانات من ناحية يسار العمود أو الحقل المحدد

الشكل العام

LTRIM (COL | VALUE,' String')

حيث أن

COL	اسم القل أو العمود
VALUE	القيمة البديلة للعمود
String	قيمة الحرف أو المتغير الذي سيتم البحث عنه

مثال

```
SELECT name, LTRIM (job،'man')
FROM S_EMP;
```

سيتم حذف man من يسار العمود job

name	**LTRIM job**
ALI	Salesman
AHMED	Analyst
SAMI	ager
Khaled	ager

دالة قياس طول بيانات الحقل LENGTH Function

تستخدم هذه الدالة لإيجاد طول بيانات متغير أو الحقل المحدد

الشكل العام

LENGTH (COL | VALUE)

مثال

```
SELECT LENGTH (name), LENGTH ('WELCOME')
FROM S_EMP;
```

سيتم حساب طول اسم الموظف وطول كلمة WELCOME.

LENGTH name	**LENGTH 'WELCOME'**
3	7
5	7
4	7
6	7

دالة تعديل بيانات في جدول TRANSLATE Function

تستخدم هذه الدالة لنعديل أو لتبديل بيانات موجودة في جدول معين

الشكل العام:

```
TRANSLATE (COL|VALUE, FROM, TO)
```

حيث أن

COL	اسم القل أو العمود
VALUE	القيمة البديلة للعمود (البيانات)
FROM	الحرف (الحروف) المطلوب تغييره
TO	الحرف (الحروف) المطلوب احلاله

مثال

```
SELECT name, TRANSLATE (name,'MI','WY')
FROM S_EMP WHERE sal=1512.5;
```

سيتم استبدال الحرفين MI بالحرفين WY

name	TRANSLATE(name)
SAMI	SAWY

دالة عرض اللفظ الصوتي SOUNDEX Function

تستخدم هذه الدالة لإيجاد اللفظ الصوتي للبيانات (المتغيرات) الموجودة في جدول معين حتى ولو كان هناك اختلاف في بعض الأحرف الهجائية

الشكل العام:

```
SOUNDEX (COL|VALUE)
```

حيث أن

COL	اسم القل أو العمود

VALUE القيمة البديلة للعمود (البيانات)

مثال

```
SELECT name, SOUNDEX (name)
FROM S_EMP WHERE sal=1512.5;
```

ملخص الفصل

تناولنا الدوال الحرفية الخاصة بتحويل حالة الحروف من حروف كبيرة إلى حروف صغيرة والعكس وكذلك ضبط محاذاة الحروف داخل النص ومن الدوال الهامة الني تناولناها ايضا دوال حذف البيانات ودوال عرض اللفظ الصوتي وتعديل بيانات الجداول .

دوال التاريخ والتحويل

أهداف الفصل

في نهاية هذا الفصل إن شاء الله تكون قادر على :

١- التعامل مع دوال التاريخ في SQL
٢- التعامل مع أدوات التحويل في SQL

دوال التاريخ

الوفت هو الحياة والتاريخ هو أصل الامم لذا لتا لنتعمق في كيفية استخدام دوال التاريخ

الدالة Sysdate

هذه الدالة تقوم بإعطاء تاريخ اليوم الحالي أي التاريخ المخزن في جهاز الكمبيوتر الذي ينفذ عليه هذا الأمر ويتم تخزين التاريخ وهمي يسمى Dual وهو موجود أصلا داخل لغة أوراكل لذلك يجب أن تتم عملية استدعاء التاريخ من هذا الجدول

مثال

لاستدعاء تاريخ اليوم الحالي نقوم بالآتي :

```
SELECT SYSDATE FROM DUAL;
```

الدالة NEXT_DAY

تستخدم هذه الدالة لعرض التاريخ الذي يوافق التاريخ التالي للتاريخ المعطى
فعلى سبيل المثال إذا كان التاريخ الموجود هو ٢٠٠٣/٧/٧ وطلب من الجهاز تحديد اليوم الذي يصادف يوم الجمعة من نفس الشهر فإنه يعطي ٢٠٠٣/٧/١١

الشكل العام

```
NEXT_DAY (DATE,CHAR)
```

حيث إن DATE هو التاريخ المعطى والمراد إيجاد التاريخ لليوم الذي يليه من خلال وضع اسم اليوم داخل المتغير CHAR

مثال

```
SELECT NEXT_DAY ('7/7/2003','MONDAY') FROM DUAL;
```

نتيجة التنفيذ

NEXT_DAY
11/7/2003

دالة تحديد اليوم الأخير من كل شهر LAST_DAY

تقوم هذه الدالة بتحديد آخر يم من كل شهر معطى

الشكل العام

```
LAST_DAY (DATE)
```

حيث أن DATE هو تاريخ الجهاز أو تاريخ تقوم بإدخاله

مثال

```
SELECT LAST_DAY (SYSDATE) FROM DUAL;
```

بفرض أن تاريخ الهاز SYSDATE هو ٢٠٠٣/٢/٢ فإن نتيجة للجملة السابقة هي كما يلي :

LAST_DAY(SYSDATE)
28/2/2003

ملاحظة
يمكنك أن تقوم بطرح تاريخ من تاريخ آخر كالمثال التالي `SELECT SYSDATE – HIRDATE FROM S_EMP` حيث HIRDATE حقل تاريخ .

الدالة MONTHES_BETWEEN

تستخدم هذه الدالة لعرض مدة الفرق بين شهرين

الشكل العام

```
MONTHES_BETWEEN   (DATE1,DATE2)
```

مثال

```
MONTHES_BETWEEN   ('01-SEP-95','11-JAN-94')
```

سيكون الناتج

1.9774194

الدالة ADD_ MONTHES

تستخدم هذه الدالة لإضافة عدد من الأشهر إلى التاريخ معطى

```
ADD_MONTHES (DATE,M)
```

حيث أن

DATE	التاريخ المعطى
M	عدد الاشهر

مثال

```
ADD_MONTHES ('7-JAN-99',6)
```

عند التنفيذ سيكون الناتج

7-JUL-99

الدالة ROUND

تستخدم هذه الدالة لعرض اقرب بداية شهر أو سنه لتاريخ معين تحدده

ملاحظة :تستخدم هذه الدالة أيضا للبيانات الرقمية للتقريب وسيتم شرحها في باب الدوال الرقمية

الصيغة العامة

```
ROUND (DATE, M|Y)
```

حيث أن

DATE	التاريخ المعطى
M	نكتب MONTH اذا اردنا ان يعود باقرب شهر
Y	نكتب YEAR اذا اردنا ان يعود باقرب سنة

مثال

```
ROUND ('07-MAY-96', 'MONTH')
```

عند التنفيذ سيكون الناتج

01-JUN-96

مثال

```
ROUND ('07-MAY-96', 'YEAR')
```

عند التنفيذ سيكون الناتج

01-JAN-96

الدالة TRANCATE Function

تستخدم هذه الدالة لعرض تاريخ اول يوم في شهر أو سنه لتاريخ معين تحدده

ملاحظة :تستخدم هذه الدالة أيضا للبيانات الرقمية للتقريب وسيتم شرحها في باب الدوال الرقمية

الصيغة العامة

```
TRUNC (DATE, M|Y)
```

حيث أن

DATE	التاريخ المعطى
M	نكتب MONTH اذا اردنا ان يعرض تاريخ اول يوم في شهر التاريخ المحدد
Y	نكتب YEAR اذا اردنا ان يعود لأول يوم لنفس سنة التاريخ

مثال

```
TRUNC ('07-MAY-96', 'MONTH')
```

عند التنفيذ سيكون الناتج

01-MAY-96

مثال

```
TRUNC ('07-MAY-96', 'YEAR')
```

عند التنفيذ سيكون الناتج

01-JAN-96

دوال التحويل

يتم تحويل البيانات من شكل إلى اخر وتوجد دوال خاصة بالتحويل ومنها

١- التحويل الى حروف TO_CHAR
٢- التحويل إلى أرقام TO_NUMBER
٣- التحويل إلى تاريخ TO_DATE

١- الدالة TO_CHAR

تستخدم هذه الدالة لتحويل التاريخ او الأرقام الى جملة حرفية حيث يتم تغيير شكل التاريخ او الارقام من صورة إلى اخرى

اولا من تاريخ الى جملة حرفية

الشكل العام

```
TO_CHAR (DATE,'FMT')
```

حيث أن

DATE قيمة التاريخ
FMT الصورة الجديدة

مثال

```
SELECT TO_CHAR (SYSDATE,'DAY, DD MON YY')
 FROM DUAL;
```

لنفرض أن تاريخ الجهاز (SYSDATE) هو ٢٠٠٣/٧/٧ فإن نتيجة التنفيذ للجملة السابقة هي كمايلي:

TO_CHAR (SYSDATE,'DAY, DD MON YY')
MONDAY 7 JUL 03

ملاحظة

١- اذا كتبنا YYYY فانه سوف يتم عرض السنة كاملة متلا٢٠٠٣
٢- اذا كتبنا MM فإنه سوف يتم عرض رقم الشهر مثلا 06
٣- اذا كتبنا MONTH فانه سوف يتم عرض اسم الشهر كاملا مثل July
٤- اذا كتبنا DYسيتم عرض أول ثلاثة حروف من اليوم
٥- اذا كتبنا DAY سيتم عرض اسم اليوم كاملا

٦- اذا اردنا عرض الساعة نكتب HH24:MI:SS AM مثل 15:45:32 PM

ثانيا التحويل من أرقام الى جملة حرفية

الشكل العام

TO_CHAR (NUMBER,'FMT')

حيث أن

NUMBER الرقم
FMT الصورة الجديدة

مثال

```
SELECT TO_CHAR (6500500,'$9,999,999)
FROM S_EMP;
```

عند التنفيذ سيكون الناتج

```
TO_CHAR (6500500,'$9,999,999)
---------------------------------
```

6,500,500

ملاحظة
١- عندما نكتب 9 فإننا نعني بها خانة رقم ٢- عندما نكتب (,) يتم طباعة الفاصلة ٣- عندما نكتب(.) يتم طباعة الفاصلة العشرية

الدالة TO_NUMBER

تقوم هذه الدالة بتحويل القيمة الرقمية المخزنة عى شكل حرف CHAR إلى قيمة رقمية فعليه NUMBER

الشكل العام

TO_NUMBER (VALUE)

حيث
VALUE قيمة مخزنة على شكل أرقام حرفية سيتم عرضها على شكل قيم رقمية

مثال

لنفرض أن لدينا حقل اسمه NUM في جدول DATA به ارقام ولكنها ارقام حرفيه أي انها تعامل معاملة الحروف فلا تدخل في العمليات الحسابية فعند تنفيذ هذه الدالة ستم تحويل القيم إلى فيم رقمية يمكن ان تدخل في العمليات الحسابية

```
SELECT NUM, TO_ NUMBER (NUM)
FROM DATA;
```

عند التنفيذ سيكون الناتج

NUM	TO_NUMBER(NUM)
9236	9236
7526	7526

ملاحظة
العمودان بهما نفس القيم ولكن يوجد اختلاف في النوع

الدالة TO_DATE

تقوم هذه الدالة بتحويل المتغير الحرفي داخل اشاؤة النص الى متعير بشكل تاريخ المثال التالي يوضح ذلك :

```
SELECT TO_DATE (‘JULY 7,2003’,’MONTH DD, YY’)
 FROM DUAL;
```

عند التنفيذ سيكون الناتج

TO_DATE (‘JULY 7,2003’,’MONTH DD, YY’)
07-JUL-03

ملخص الفصل

تناولنا في هذا الفصل دوال التاريخ الخاصة بالتعامل مع كامل الصيغ للتاريخ والوقت واستعراضها بعدة أشكالها ثم تطرقنا دوال التحويل التي تعمل تحويل حالات الكتابة من نص إلى تاريخ و بالعكس كذلك من أرقام الى نص

الدوال الرقمية

أهداف الفصل

في نهاية هذا الفصل ان شاء الله ستكن قادر على

١- التعامل مع دالة القيمة المطلقة
٢- إيجاد الجذر التربيعي للأعداد
٣- التعامل مع الدالة الأسية للأعداد
٤- تقريب الأعداد العشرية من خلال SQL
٥- تقريب الأعداد من خلال حذف المنزلة العشرية
٦- إيجاد باقي القسمة للأعداد
٧- ايجاد اشارة الأعداد السالبة

الدوال المطلقة والأسية والأسية والجذر التربيعي

دالة القيمة المطلقة ABS Function

تستخدم هذه الدالة لإيجاد القيمة المطلقة لرقم معين وغالبا يتم استخدان هذه الدالة مع جمل أخرى

الشكل العام

ABS (COL|VALUE)

حيث أن

COL **اسم الحقل أو العمود**
VALUE **القيمة البديلة للعمود (البيانات)**

مثال

```
SELECT name,job,hiredate-sysdate FROM S_EMP;
```

عند التنفيذ سيتم طباعة:

name	job	hierdate-sysdate
ALI	Salesman	-6906.3252
AHMED	Analyst	-6841.3252
SAMI	Manager	-4592.3252
Khaled	Manager	-6555.3252

نلاحظ أن قيم العمود hierdate-sysdate قيم سالبة ولكي نجعلها دائما موجبة نأخذ القيمة المطلقة لهذا العمود ونكتب

```
SELECT name,job,ABS(hiredate-sysdate) FROM S_EMP;
```

عند التنفيذ سيتم طباعة:

name	job	hierdate-sysdate
ALI	Salesman	6906.3252
AHMED	Analyst	6841.3252
SAMI	Manager	4592.3252
Khaled	Manager	6555.3252

٢- الدالة الأسية POWER Function

تستخدم هذه الدالة لإيجاد قيمة رقم مرفوع لأس

الشكل العام

```
POWER (COL|VALUE,P)
```

حيث أن

COL	اسم الحقل أو العمود
VALUE	القيمة البديلة للعمود (البيانات)
P	قيمة الأس

مثال

```
SELECT DIGIT, POWER (DIGIT1, 2) FROM DIGIT;
```

عند التنفيذ سيتم طباعة:

DIGIT1	POWER (DIGIT1, 2)
20	400
25	625
25.5	650.25
30.5	930.25

٣- دالة الجذر التربيعي SQRT Function

تستخدم هذه الدالة لإياد الجذر التربيعي لرقم معين

الشكل العام

SQRT (COL|VALUE)

حيث أن

COL	اسم الحقل أو العمود
VALUE	القيمة البديلة للعمود (البيانات)

مثال

```
SELECT name,sal, SQRT (sal) FROM S_EMP;
```

عند التنفيذ سيتم طباعة:

NAME	SAL	SQRT (SAL)
Ali	968	31.113
Ahmed	1936	44
SAMI	1512.5	38.9805
KHALED	3599.75	59.988

دوال التقريب وباقي القسمة والإشارة

دالة التقريب العشري ROUND Function

تستخدم هذه الدالة لتقريب العدد إلى اقرب رقم عشري أو صحيح

الشكل العام

ROUND (COL|VALUE, N)

حيث أن

COL	اسم الحقل أو العمود
VALUE	قيمة عددية
N	عدد المواقع العشرية

مثال

SELECT SQRT (sal), **ROUND** (SQRT (sal), 2) FROM S_EMP;

عند التنفيذ سيتم طباعة:

SQRT (SAL)	ROUND (SQRT (sal), 1)
31.113	31.11
44	44
38.9805	38.98
59.988	59.99

دالة التقريب بالحذف TRANCATE Function

تستخدم هذه الدالة لتقريب الرقم بحذف بعض المواقع العشرية

الشكل العام

TRUNC (COL|VALUE,N)

حيث أن

COL	اسم الحقل أو العمود
VALUE	قيمة عددية
N	عدد المواقع العشرية

مثال

```
SELECT DIGIT1, TRUNC (DIGIT1,-1),
TRUNC(POWER(DIGIT1,2),1) FROM DIGITS;
```

عند التنفيذ سيتم طباعة:

DIGIT1	TRUNC (DIGIT1, -1)	TRUNC (POWER (DIGIT1, 2), 1)
20	20	400
25	20	625
25.5	20	650.2
30.5	30	930.2

دالة باقي القسمة MOD Function

تستخدم هذه الدالة لإيجاد باقي قسمة عددين

الصيغة العامة

MOD (VALUE1, VALUE2)

حيث أن

VALUE1	قيمة عدد بة
VALUE2	قيمة عددية

مثال

```
SELECT DIGIT1, MOD (DIGIT1, 7) FROM DIGITS
```

عند التنفيذ سيتم طباعة:

DIGIT1	MOD (DIGI1, 7)
20	6
25	4
25.5	4.5
30.5	2.5

دالة الإشارة SIGN Function

تستخدم هذه الدالة لفحص إشارة الرقم فإذا كانت الإشارة موجبة تعود بالقيمة (١) أما اذا كانت الإشارة سالبة فتعود بالقيمة (١-)

الشكل العام انظر المثال التالي :

مثال

```
SELECT DIGIT1, SIGN (DIGIT1) FROM DIGITS
```

DIGIT1	MOD (DIGI1, 7)
20	1
25	1
25.5	1
30.5	1
-50	-1
-30.5	-1

ملخص الفصل

تناولنا في هذا الفصل الدوال الرقمية في لغة الاستعلام والاستفهام في نظام قواعد البيانات اوراكل وقد قسمنا هذه الدوال حسب وظائفها فمنها مايتعامل مع سجل أو صف من البيانات ومنها مايتعامل مع مجموعة من السجلات RECORDS او الأعمدة COLUMNS(الحقول) حيث تطرقنا الى دالة القيمة المطلقة والدالة الاسية والجذر التربيعي وكذلك دوال التقريب ودالة باقي القسمة والاشارة

استرجاع البيانات

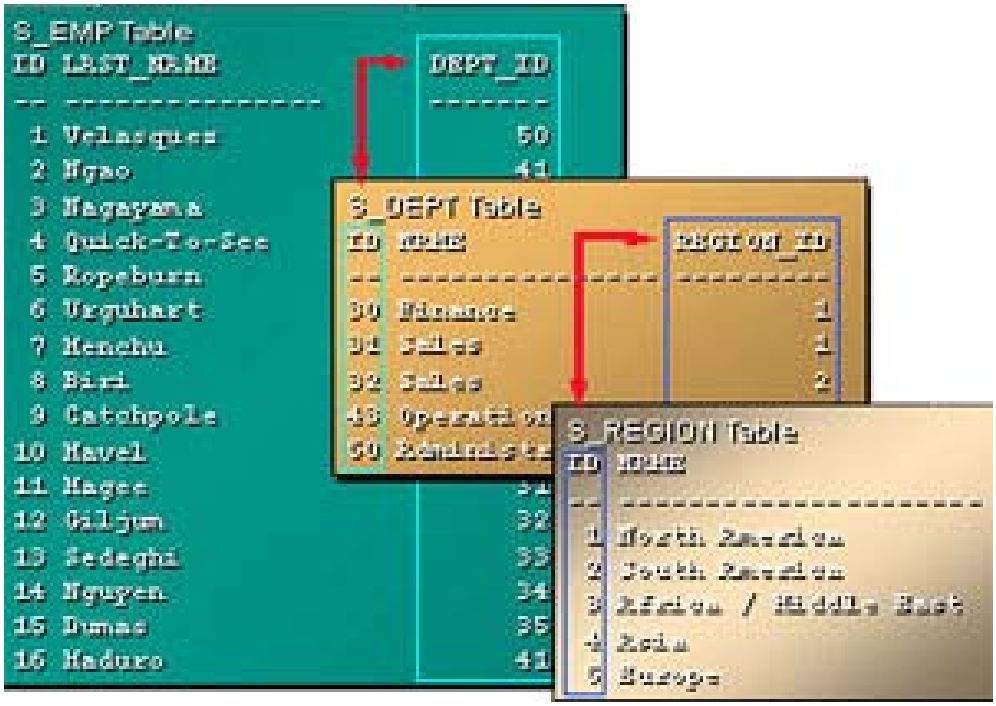
أهداف االفصل

ان شاء الله بعد دراسة هذا الفصل ستكون قادر على

١- استرجاع البيانات من أكثر من جدول
٢- استرجاع البيانات بأسماء مستعارة

استرجاع البيانات من أكثر من جدول

نستطيع استرجاع بيانات من أكثر من جدول اذا كان بينهم علاقة Relation حيث كل صف من احدهما يرتبط مع صف من الاخر عن طريق حقل مشترك



وتوجد عدة أنواع للربط :

١-EQUAI JOIN

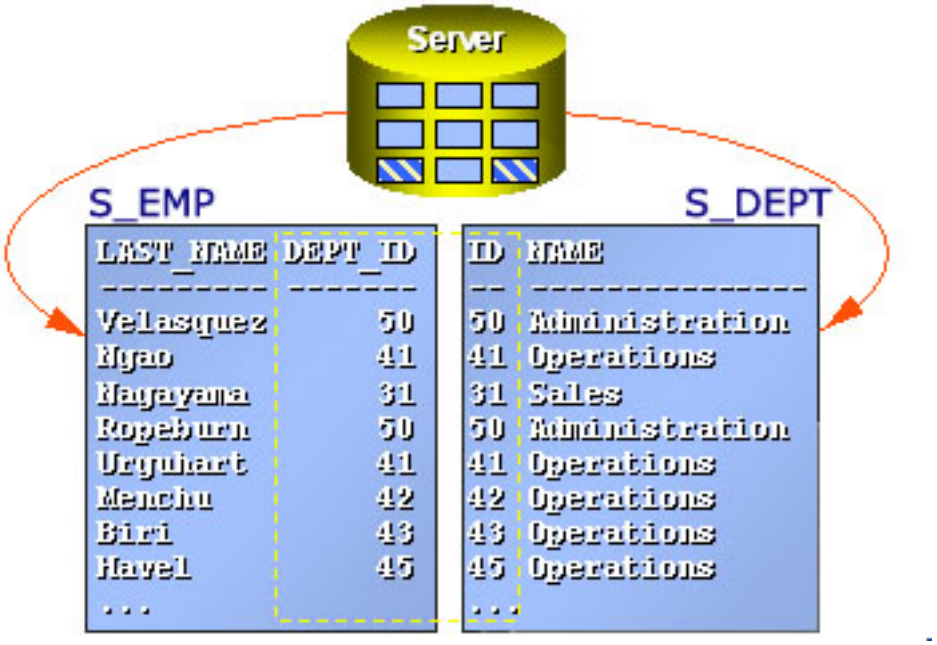
الصيغة العامة

```
SELECT table.column, table.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

حيث نكتب بعد SELEST الاعمدة التي نريد ظهورها ولاحظ كذلك أنه من الضروري كتابة الاعمدة التي عن طريقها تم الربط(التي يتساوى فيها الحقول في الجدولين) ونكتب بعد كلمة WHERE العمودان اللذان تم ربط الجدولين عن طريقهما

ملاحظة
نكتب اسم الجدول اولا ثم اسم العمود وبينهما نقطه `table.column`

مثال




```
SELECT S_EMP.DEPT_ID, S_EMP.LAST_NAME,S_DEPT.NAME
FROM S_EMP, S_DEPT
WHERE S_EMP.DEPT_ID = S_DEPT.ID;
```

عند التفيذ ينتج :

DEPT_ID	*LAST_NAME*	*NAME*
50	VELASQUEA	ADMINISTRATION
41	NGAO	OPERATOPNS
31	NAGAYAMA	SALES
50	ROPEBURN	ADMINISTRATION
41	URGIHART	OPERATOPNS
42	MENCHU	OPERATOPNS
43	BIRI	OPERATOPNS
45	HAVEL	OPERATOPNS

ويمكن اضافة شرط اخر بعد WHERE وذلك باستخدام AND فنكتب

```
SELECT S_EMP.DEPT_ID, S_EMP.LAST_NAME,S_DEPT.NAME
FROM S_EMP, S_DEPT
WHERE S_EMP.DEPT_ID = S_DEPT.ID AND S_DEPT = 41;
```

عند التنفيذ ينتج :

DEPT_ID	LAST_NAME	NAME
41	NGAO	OPERATOPNS
41	URGIHART	OPERATOPNS

٢-NON- EQUAI JOIN

يستخدم هذه النوع عندما لا يوجد اعمدة مشتركة بين الجدولين

مثال

```
SELECT S_EMP.ename, S_EMP.job, S_EMP.sal,
salgrade.grade
FROM S_EMP, salgrade
WHERE S_EMP.sal BETWEEN salgrade.losal AND
salgrade.hisal;
```

٣-OUTER JOIN

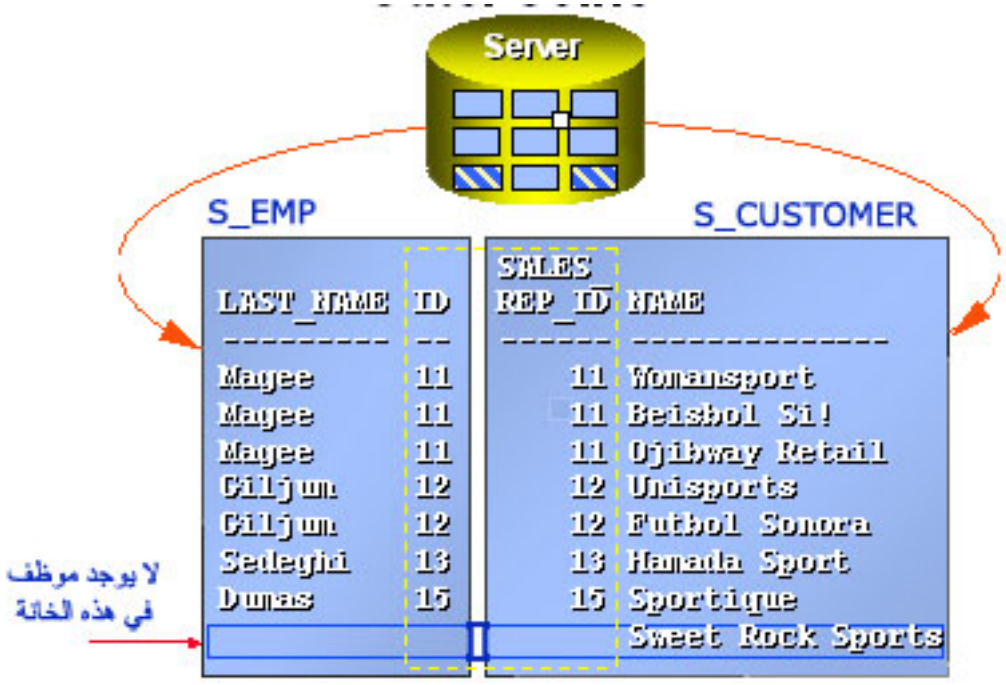
يستخدم هذا النوع لرؤية السجلات في الجدول الثاني التي لا يقابلها سجلات في الجدول الأول

الصيغة العامة

```
SELECT table.column, table.column
FROMtable1, table2
WHERE table1.column(+) = table2.column;
```

ملاحظة
نسنخدم المعامل (+) في هذا النوع من الربط حيث نكتبه بجوار الاسم الحقل الذي لا يحتوي على قيمة

مثال

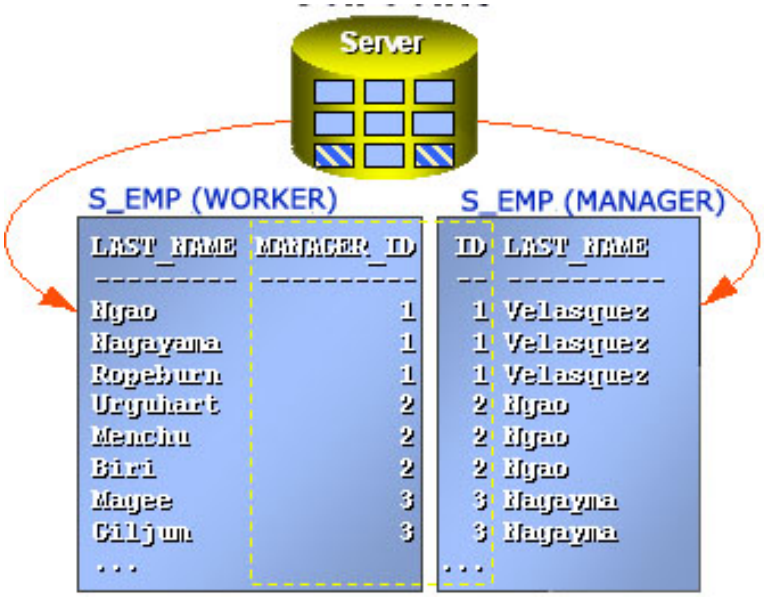


اذا اردنا الاستعلام عن الجزء الذي لا يتواجد به موظف نكتب

```
SELECT   s_emp.last_name, s_emp.id, ,
s_customer.name
FROM s_emp, s_customer
WHERE    s_emp.id (+) = s_customer.sales_rep_id
ORDER BY s_emp.id;
```

٤-SELF JOIN

تستخدم لربط صف من جدول بصف اخر من نفس الجدول



مثال :

```
SELECT   worker.last_name||' works for '||
manager.last_name
FROM s_emp worker, s_emp manager
WHERE    worker.manager_id = manager.id;
```

عند التنفيذ سيتم طباعة

```
worker.last_name||' works for '||
manager.last_name
-----------------------------------------------
Nago works for Velasquez
Nagayama works for Velasquez
Ropeburn works for Velasquez
Urguhart works for Nago
Biri works for Nago
Magee works for Nagayama
Giljum Nagayama
```

استرجاع البيانات بأسماء مستعارة

في اخر مثال قمنا باستدعاء الجدول s_emp باسم مستعار worker ومرة أخرى باسم manager وهذا الاسم يقوم محل اسم الجدول وقد استخدمناه للتسهيل

مثال

```
SELECT  e.ename, e.job, e.sal, s.grade
FROM    S_EMP e, salgrade s
WHERE   e.sal BETWEEN s.losal AND s.hisal;
```

مثال

```
SELECT   e.last_name, e.id, c.name
FROMs_S_EMP e, s_customer c
WHERE    e.id (+) = c.sales_rep_id
ORDER By e.id;
```

تناولنا في هذا الفصل جمل SQL الخاصة باسترجاع بيانات من اكثر من جدول كعرض رقم اقسام الموظفين من جدول الموظفين وتطرقنا كذلك إلى عملية استرجاع البيانات من خلال اسم مستعار

الدوال الرياضية والإحصائية ودوال المجموعات

أهداف الفصل

في نهاية هذا الفصل ان شاء الله ستكون قادر على :

١- التعامل مع الدوال الرياضية والاحصائية
٢- التعامل مع دوال المجموعات

الدوال الرياضية والإحصائية

هنالك مجموعة من الاقترانات الرياضية والاحصائية التي تتعامل مع مجموعة من السجلات لإخراج قيمة واحدة محدودة ومن هذه الاقترانات مايلي:

١- دالة العد COUNT
٢- دالة أكبر قيمةMAX
٣- دالة أصغر قيمة MIN
٤- دالة المجموع SUM
٥- دالة المتوسط الحسابي AVG
٦- دالة الانحراف المعياري STDDEV
٧- دالة المعيار VARIANCE

١- دالة العد COUNT

تقوم هذه الدالة بعملية حصر الحقول في عمود معين والتي لا تحمل القيمة NULL

الشكل العام

```
COUNT (DISTINCT|ALL|EXP)
```

حيث أن

DISTINCT	تستخدم لمنع احتساب الحقول التي تحمل القيمة NULL
ALL	يكون المدى هنا جميع الحقول حتى المكررة
EXP	يمكن استخدام اسم العمود

مثال

```
SELECT COUNT (DISTINCT NAME) FROM S_EMP;
```

تقوم الجملة السايقة بتحديد عدد الصفوف للحقل NAME من الجدول S_EMP

مثال
حساب عدد السجلات في الجدول

```
SELECT COUNT (*) ”TOTAL” FROM S_EMP;
```

٢- دالة أكبر قيمة MAX

تقوم هذه الدالة بعرض أكبر قيمة موجودة في عمود معين

الشكل العام

```
MAX (DISTINCT|ALL|EXP)
```

مثال

SELECT **MAX** (DISTINCT salary) FROM S_EMP;

تقوم هذه الجملة بعرض أكبر راتب في جدول الموظفين S_EMP

٣- دالة أكبر قيمة MIN

تقوم هذه الدالة بعرض أكبر قيمة موجودة في عمود معين

الشكل العام

MIN (DISTINCT|ALL|EXP)

مثال

SELECT **MIN** (DISTINCT salary) FROM S_EMP;

تقوم هذه الجملة بعرض أصغر راتب في جدول الموظفين S_EMP

٤- دالة المجموع SUM

تقوم هذه الدالة بعرض مجموع القيم الموجودة في حقل او عمود معين

SUM (DISTINCT|ALL|EXP)

مثال

SELECT **SUM** (salary) FROM S_EMP;

تقوم هذه الجملة بعرض مجموع رواتب الموظفين في جدول الموظفين S_EMP

٥- دالة المتوسط الحسابي AVG

تقوم هذه الدالة بعرض المعدل لمجموعة قراءات قد تكون موجودة في جدول معين

الشكل العام

AVG (COLUMN)

COLUMN هو اسم العمود الموجود في جدول معين

مثال

SELECT **AVG** (SALARY) FROM S_EMP;

تقوم هذه الجملة بحساب المتوسط الحسابي لمرتبات الموظفين

٦- دالة الانراف المعياري STDDEV

تقوم هذه الدالة بإيجاد الانحراف المعياري لمجموعة مشاهدات أو قراءات

الشكل العام

STDDEV (DISTINCT|ALL)

مثال

لإبجاد الانحراف المعياري للأرقام الواردة في حقل الرواتب في جدول الموظفين S_EMP

SELECT **STDDEV** (SALARY) FROM S_EMP;

٧- دالة المعيار VARIANCE

تقوم هذه الدالة بإيجاد المعيار لمجموعة مشاهدات أو قراءات

مثال

SELECT **VARIANCE** (SALARY) FROM S_EMP;

دوال المجموعات

هي عبارة عن جميع الدوال السابقة او أي دوال اخرى ولكن باستخداماها مع الدالة GROUP BY مما يؤدي إلى تجزئة الجدول الرئيسي إلى مجموعات اصغر منه حسب شرط مغين

الشكل العام

GROUP BY COL_NAME

COL_NAME هو اسم العمود في الجدول والذي سيتم تقسيم الجدول إلى مجموعات صغيرة بناء غليه

ولفهم هذه الدالة ننظر الى المثال التالي

اذا كتبنا الجملة التالية

```
SELECT   id, last_name, dept_id DEPARTMENT
FROM s_emp
WHERE dept_id = 41;
```

فان المخرجات هي

ID	LAST_NAME	DEPARTMENT
2	Ngao	41
6	Urguhart	41
16	Maduro	41
17	Smith	41

```
SELECT   dept_id, COUNT (*) ”Number”
FROM  s_emp
WHERE    dept_id = 41
GROUP BY dept_id;
```

DEPT_ID	Number
41	4

نلاحظ ظهور سطر واحد من البيانات للقسم ٤١ فالدالة GROUP BY قامت بتجميع القسم ٤١ مع بعضه (وكأنها عملت جدول خاص بها) وهذه واضح عندما استخدمنا دالة COUNT اظهرت ان عدد السجلات ٤ رغم أن عدد الاسطر هو واحد

لاستخدام شرط مع دالة المجموعات نكتب الامر HAVING
فيكون شكل جملة الاستعلام كالاتي

```
SELECT   column, group_functionFROM
    table[WHERE  condition]
[GROUP BY    group_by_expression][HAVING
    group_condition]
[ORDER BY    column];
```

اذا اردنا اظهر بيانات من جدول الموظفين EMP وعمل مجموعات بشرط أن تكون الوظيفة MANAGER

```
SELECT DEPTNO, JOB, COUNT (*), SUM (SAL)
FROM S_EMP
GROUP BY DEPARTNO, JOB
HAVING JOB='MANAGER';
```

مثال
اذا اردنا اظهار بيانات من جدول الموظفين EMP وعمل مجموعات بشرط أن تكون الوظيفة MANAGER أو ANALYSTاو SALESMAN

```
SELECT DEPTNO, JOB, COUNT (*), SUM (SAL)
FROM S_EMP
GROUP BY DEPARTNO, JOB
HAVING JOB IN ('MANAGER',' ANALYST',' SALESMAN');
```

ملخص الفصل

ركزنا في هذا الفصل على جمل SQL الخاصة بالدوال الرياضية والاحصائية التي تقوم بعرض قيمة واحدة من البيانات مثل MAX,MIN,AVG وغيرها ومن ثم دوال المجموعات التي تكون غالبا مع الامر GROUP BY والتي تقوم بفرز الجدول الى جدول اصغر منه أو يساويه

الاستعلامات الفرعية

أهداف الفصل

في نهاية هذا الفصل ان شاء الله ستكون قادر على

١- تحديد الحالات التي يمكن ان تستخدم فيها الاستعلامات الفرعية
٢- تعريف الاستعلامات الفرعية والمتداخلة
٣- معرفة أنواع الاستعلامات الفرعية
٤- كتابة استعلام فرعي احادي الصف وآخر متعدد الصفوف

الاستعلام الفرعي

مقدمة

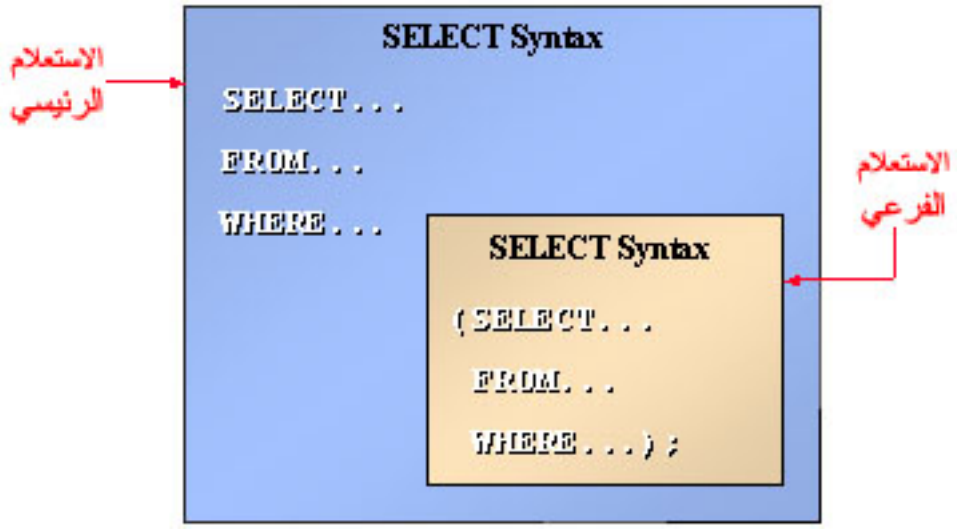
لنفترض أنك ترغب في كتابة استعلام لإيجاد بيانات الموظفين الذين تزيد رواتبهم عن راتب أحد الموظفين
في هذه الحالة تحتاج الى استعلامين

١- استعلام لايجاد راتب الموظف المعني
٢- استعلام لايجاد الموظفين الذين تزيد رواتبهم عن المبلغ المحسوب في الاستعلام الأول

ويمكنك دمج الاستعلامين وذلك بتركيب احد الاستعلامين في الاخر ، الاستعلام الداخلي سوف يعود بقيمة (قيم) والتي يستخدمها الاستعلام الخارجي (الرئيسي)
إن استخدام الاستعلامات الفرعية يشبه تماما تنفيذ الاستعلامين بشكل متتال واستلام نتيجة الاستعلام الاول كنتيجة بحث في الاستعلام الثاني

تعريف الاستعلام الفرعي

هو جملة استفسار SELECT مضمنة داخل جملة استفسار رئيسية لاسترجاع قيمة أو مجموعة من القيم ليتم استخدامها في الاستعلام الرئيسي انظر الشكل التالي



الشكل العام

```
SELECT select_listFROM tableWHERE expr operator
      (SELECT    select_list
       FROM      table);
```

١- الاستعلام الداخلي(الفرعي) ينفذ اولا
٢- ناتج الاستعلام الداخلي(الفرعي) يستخدم في الاستعلام الرئيسي

متطلبات الاستعلام الفرعي

١- وضع الاستعلام الفرعي بين قوسين (.....)
٢- وضع الاستعلام الفرعي يمين عملية المقارنة.
٣- لا يمكن استخدام ORDER BY داخل الاستعلام الفرعي
٤- استخدام المعاملات أحادية الصف مع الاستعلامات الفرعية الأحادية الصف
٥- استخدام المعاملات متعددة الصف مع الاستعلامات الفرعية المتعددة الصف

أنواع الاستعلامات الفرعية

يمكن تصنيف الاستعلام الفرعي حسب

١- استعلام فرعي أحادي الصف يسترجع صفا واحدا
٢- استعلام فرعي متعدد الصفوف يسترجع أكثر من صف واحد
٣- استعلام فرعي متعدد الأعمدة يسترجع اكثر من عمود

١- استعلام فرعي أحادي الصف يسترجع صفا واحدا

نستخدم هذا الاستعلام في حالة ان الاستعلام الفرعي لا يعود الا بسطر واحد انظر الى المثال التالي

```
SELECT   last_name, title, salary
FROMs_emp
WHERESalary <
         (SELECT  AVG(salary)
          FROM    s_emp);
```

توسط الحسابي ويأخذ
الاستعلام الرئيسي هذه القيمة ويستخرج اسماء ورواتب الموظفين الذين يعتبر راتبهم اعلى من المتوسط الحسابي

ملاحظة
١- أنواع معاملات المقارنة الخاصة بهذا النوع هي <>, >=, <=, <, >

٢- استعلام فرعي متعدد الصفوف يسترجع أكثر من صف واحد

نستخدم هذا الاستعلام في حالة ان الاستعلام الفرعي يعود بعدة اسطر انظر المثال التالي

```
SELECT   last_name, first_name, title
FROM  s_emp
WHERE dept_in IN
             (SELECT   ID
              FROM     s_dept
              WHERE name = 'Finance'
               OR region_id = 2);
```

يقوم الاستعلام الفرعي في هذا المثال باستخراج أرقام الموظفين الذين اسمهم 'Finance' او رقم منطقتهم ٢ من جدول الاقسام فهذا الاستعلام سيعود بعدة فيم يأخذها الاستعلام الرئيسي ويستخرج اسماء الموظفين على اساسها

ملاحظة
١- أنواع معاملات المقارنة الخاصة بهذا النوع هي
IN, ALL, ANY
٢- اذا استخدمنا معاملات الاستعلام احادي الصف مع استعلام متعدد فان البرنامج يعطيك رسالة خطأ
ORA-01427:single-row sub query returns more than one row

ملخص الفصل

في هذا الفصل تعرفنا على الاستعلامات الفرعية واهميتها وشروطها وانواعها الأحادية والمتعددة

أهداف الفصل

في نهابة هذا الفصل ستكون قادر انشاء الله على مايلي:

١- التعرف على الجداول وشروط تسميتها
٢- التعرف على أنواع البيانات
٣- إنشاء الجداول بواسطة الأمرCREAT E
٤- إنشاء مفتاح أساسي في الجدول
٥- إنشاء جدول بواسطة جدول اخر
٦- عرض مواصفات الجدول

تعريف الجدول

قلنا سايقا أن الجدول أو الملف يتكون من مجموعة من السجلات RECORDS وكل سجل يتكون من مجموعة من الحقول FIELDS كل حقل به مجموعة من الحروف او الارقام وهذه الحقول لها طول أو عرض

ويعرف الملف أو الجدول بأنه مجموعة من الأعمدة ويمثل كل عمود بحقل من الحقول ولكل حقل نوع بياناته

تسمية الجدول/ملف TABLE/FILE NAMES

١- لا يتجاوز طول اسم الجدول أو الملف عن ٣٠ حرف
٢- يمكن ان يكون حروفا كبيرة أو صغيرة
٣- الاسم يبدأ بحرف ولا يمكن أن يبدأ برقم
٤- يمكن أن يحتوي الاسم على رموز خاصة مثل #,$,% .... الخ

أنواع الحقول Fields Types

١- المتغير الحرفي CHAR

يتكون هذا الحقل من الحروف الأبجدية ، الأرقام ، والرموز الخاصة . وطول أو عرض هذا الحقل يصل الى ٢٤٠ حرف ولا بد من تحديد طول أو عرض الحقل وهو اجباري واذا لم يتم التحديد سيفترض البرنامج انه ١

الشكل العام

```
var CHAR (size)
```

مثال

```
FirstName CHAR (20)
```

٢- المتغير الحرفي VARCHAR2

يتكون هذا الحقل من الحروف الأبجدية ، الأرقام ، والرموز الخاصة . وطول أو عرض هذا الحقل يصل الى ٢٤٠ حرف لا يوجد طول افتراضي للحقل

مثال

```
FirstName VARCHAR2 (20)
```

٣- متغير رقمي صحيح Integer Number

ويتكون من الأرقام (0,1,2-9) والاشارة + أو _ ويمكن أن يحتوي على العلامة العشرية أو الفاصلة العشرية (.) والحد الأقصى لعدد خانات الرقم يصل إلى ٤٠ خانة وتحديد طول أوعرض الحقل اختياري وكذلك الاشارة أو الفاصلة العشرية

مثال

```
Serial_No NUMBER (10)
```

٤- المتغير الرقمي ذو العلامة العشرية

ويتكون من الأرقام (0,1,2-9)

مثال

Mark **NUMBER** (10,3)

٥- متغير طويل LONG

يستخدم للحقول التي تحتوي على بيانات تصل مساحتها الى ٢ جيجا

مثال

Detail **LONG**

٦- المتغير الدال على التاريخ (Date)

يتكون هذا الحقل من مجموعة الأرقام للدلالة على اليوم والشهر والسنة وطول هذا الحقل ٨ خانات على الأقل ويمكن أن يتغير شكل كتابة التاريخ بذكر اسم الشهر

مثال

HireDate **DATE**

إنشاء جدول

إنشاء جدول باستخدام أمر CREATE

الشكل العام

```
CREATE TABLE table-name (field type [(size [,
field2 type [(size]);
```

حيث ان

table-name	اسم الملف أو الجدول المطلوب
field1,field2	اسم الحقل أو الحقول المطلوب تصميمها مع وجود اسم حقل على الأقل
type	نوع حقل البيانات
Size	طول او عرض الحقل

مثال

```
CREATE TABLE SCHOOL (
S_no NUMBER (5) NOT NULL,
S_NAME CHAR (20).
S_LOC CHAR(15),
S_DATE DATE);
```

التقييد CONSTRAINT

يتم عن طريق التقييد فرض بعض القواعد على الحقول مثلا جعل حقل مفتاح أساسي منع ترك حقل فارغ ومن أنواعه

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FORIGN KEY

الشكل العام

```
column [CONSTRAINT constraint_name]
constraint_type,
```

constraint_name	اسم التقييد
constraint_type	نوع التقييد

١- غير فارغ NOT NULL

هذا الأمر لا يسمح بترك حقل فارغ

مثال

```
CREATE TABLE friend...
  phone  VARCHAR2(15) NOT NULL,...
  last_name  VARCHAR2(25)
    CONSTRAINT friend_last_name_nn NOT NULL,...
```

٢- فريد UNIQUE

هذا الأمريجعل الحقل (العمود) فريد فلا يسمح بتكرار صف من الصفوف فكل صف لا يوجد الا مرة واحدة

ملاحظة
١- يمكن أن يحتوي الحقل الفريد UNIQUE على قيمة فارغة بشرط أن يكون حقل واحد فقط فريد UNIQUE ٢- وبمكن أن يوجد في الجدول أكثر حقل فريد

مثال

```
... phone    VARCHAR2(10)
      CONSTRAINT s_emp_phone_uk UNIQUE,...
```

٣- مفتاح أساسي Primary Key

لا يمكن لأي جدول أن يحتوي الا على مفتاح أساسي واحد وإذا قمنا بتحويل حقل الى مفتاح أساسي فهذا الحقل يعتبر حقل فريد ولكن لا يمكن أن يحتوي على قيم فارغة

مثال

```
... id   NUMBER(7)
      CONSTRAINT s_emp_id_pk PRIMARY KEY,...
```

يمكن تعيين حقل (عمود) واحد أو أكثر ليكون مفتاح خارجي حيث يكون هذا العمود مرتبط بعمود اخر به مفتاح اساسي او فريد من نفس الجدول أو جدول اخر ومن الضروري أن يكون مرتبط بقيمة اخرى من عمود اخر او يكون فارغ
انظر المثال التالي

```
... dept_id  NUMBER(7)
      CONSTRAINT s_emp_dept_id_fk
       FOREIGN KEY (department_id)
         REFERENCES s_dept(id)
    ON DELETE CASCADE;
```

١- Foreign Key
تكتب بجوارها اسم الحقل المراد أن يكون مفتاح خارجي في الجدول الابن
٢- REFERENCES
نكتب بجوارها الحقل الذي سيتم الربط به من الجدول الاخر

٣- ON DELETE CASCADE
اذا تم مسح احد الصفوف في الجدول الاب سيمسح الصف المقابل له في الجدول الابن

انشاء جدول بواسطة جدول آخر

لإنشاء جدول بواسطة جدول آخر يجب أن يكون الجدول الآخر منشأ مسبقا ثم نكتب الأمر كالتالي

الشكل العام

```
CREATE TABLE tablename
  [column(, column...)]AS Select statement;
```

مثال

```
CREATE TABLE EMPLOYEE
AS
SELECT NAME, SAL, JOB FROM S_EMP
```

قمنا في هذه الجملة بعمل جدول جديد ثم نقلنا بيانات الجدول S_EMP اليه

انشاء جدول بواسطة استعلام فرعي

لإنشاء جدول بواسطة استعلام فرعي يجب أن يكون الاستعلام الفرعي قائم على الاستعلام من جدول منشأ مسبقا ثم نكتب الأمر كالتالي

الشكل العام

```
CREATE TABLE tablename
  [column(, column...)]AS subquery;
```

مثال

```
CREATE TABLE     emp_41
AS
SELECT   id, last_name, userid, start_date
FROM     s_emp
WHERE    dept_id = 41;
```

جملة عرض مواصفات جدول DESCIRBE

تستخدم هذه الجملة لغرض مواصفات الجدول ويمكن كتابتها كلمة كاملة أو مختصرة DESC

الشكل العام

```
DESCRIBE tablename;
```

مثال

DESCRIBE S_EMP;

ملخص الفصل

في هذا الفصل قمنا بالتعرف على الجداول وأنواع التقييد وكيفية إنشائها وإدراج المفتاح الأساسي والخارجي في الجدول وكيفية انشاء جدول من جدول اخر وكذلك من استعلام فرعي ومعرفة كيفية عرض معلومات الجدول

تعديل بنية الجدول وحذفه

أهداف الفصل

في نهاية هذا الفصل ستكون قادر ان شاء الله على:

١- التعرف على جميع عمليات التعديل على الجدول
٢- إضافة حقول جديدة إلى الجدول
٣- تعديل مواصفات حقول الجدول
٤- حذف المفتاح الأساسي من الجدول
٥- حذف الجدول

تعديل الجداول

تتيح لك الـSQL عملية تعديل الجدول بعد إنشائه في الحالات التالية

١- اضافة حقل جديد
٢- تعديل مواصغات حقل
٣- حذف المفتاح الأساسي من الجدول

ويجب الانتباه الى عدم إمكانية حذف حقل من الجدول

جملة تعديل الحقول ALTER Command

تستخدم هذه الجملة لإضافة حقول أو لتعديل الحقول في الجدول ويشمل هذا التعديل اسم الحقل ونوعه الحقل وحجم بياناته أو الحاق عبارة فارغ أو غير فارغ الى الحقل

اضافة حقل جديدNew Field

الشكل العام

```
ALERT TABLE tablename ADD (field Type (size) [NOT
NULL] | [NULL])
```

حيث أن

tablename	اسم الجدول المراد تعديله
field	اسم الحقل
Type	نوع البيانات
size	طول أو حجم الحقل

تعديل مواصفات حقل

يمكن عن طريق جملة ALTER تعديل مواصفات حقل

الشكل العام

```
ALERT TABLE tablename MODIFY Type (size) [NOT
NULL] | [NULL])
```

حذف الجداول

يمكن أن نحذف الجدول من قاعدة البيانات بواسطة الأمر DROP TABLE بشرط أن تكون مالك الجدول أو لك صلاحية للحذف

الشكل العام :

```
DROP TABLE tablename;
```

حيث

tablename; اسم الجدول المراد حذفه

حذف المفتاح الأساسي

نكتب الأمر التالي

```
ALTER TABLE tablename
DROP PRIMARY KEY CASCADE;
```

عبارة CASCADE هنا لالغاء أي علاقة تعتمد على المفتاح الأساسي

ملخص الفصل

في هذا الفصل تناولنا عملية تعدبل الجداول من إضافة وتعديل قول الجدول مع عدم امكانية حذف حقل موجود في الجدول وكذلك كيفية حذف المفتاح الأساسي من الجدول وحذف الجدول من قاعدة البيانات بواسطة الامر DROP

إدخال البيانات إلى الجداول

أهداف الفصل

إن شاء الله في نهاية هذا الفصل ستكون قادر على

١- الإلمام بالطرق المختلفة لإدخال البيانات إلى الجدول
٢- إدخال سجل واحد إلى جدول البيانات
٣- إضافة عدة سجلات الى الجدول

إضافة بيانات

مقدمة

لقد قمنا في الفصول السابقة بإنشاء الجداول والتعرف على أنواع البيانات وتعديل مواصفات الحقول وبهذا نكون قد أنشأنا جدول البيانات وحددنا خصائص الحقول من حيث نوع البيانات والحجم أي أننا أعددنا الوعاء الذي سيحوي البيانات وفي هذا الفصل سنتعلم بمشيئة الله كيفية إضافة البيانات إلى هذه الجداول والأمور المتعلقة بعمليات الإضافة وأنواعها ، كما أن هذا الفصل يشكل مدخلا وجزأ أساسيا للغة تناول البيانات (DML) في نظام قواعد بيانات أوراكل التي تعتبر إحدى النظم القوية في نظم إدارة قواعد البيانات (DBMS) لما لها من أثر فعال في معالجة البيانات داخل الجدول حيث تشمل لغة تناول البيانات (DML) على الأجزاء التالية :

١- إدخال البيانات إلى الجداول INSERT
٢- تعديل البيانات في الجداول UPDATE
٣- حذف البيانات من الجداول DELETE

في هذا الفصل سنتناول ادخال البيانات

إدخال البيانات إلى الجداول INSERT

هي العملية التي يتم بها إدخال سجل / سجلات إلى جدول البيانات من خلال جملة INSERT وتتم هذه العملية بعدة طرق

أولا : إضافة سجل واحد إلى جدول البيانات

الصيغة العامة:

```
INSERT INTO      table [(column1 [, column2...])]
VALUES           (value1 [, value2...]);
```

حيث أن

Table	اسم الجدول المراد إلحاق السجل به
Column1,Column2	اسماء الأعمدة(الحقول) المطلوب إدخال البيانات إليها
Value1, value2	القيم المطلوب إضافتها في حقول السجل الجديد وكل قيمة يتم إدراجها في الحقل المناسب في القائمة وبالتالي سيتم إضلفة القيمة١ في الحقل ١ وهكذا

الشرح

بعد تنفيذ جملة الإدخال بهذه الطريقة سوف يتم إضافة بيانات سجل واحد فقط في الجدول المذكور يحتوي على قيم تم إ دراجها في جملة الإدخال وللحقول المذكورة فقط

القواعد التي يجب التقيد بها في هذه الطريقة

١- يجب أن يكون عدد القيم التي سيتم إدخالها هو نفس عدد الحقول المذكور في جملة **INSERT**

٢- يجب أن يكون نوع بيانات القيم التي سيتم إدخالها من نفس نوع بيانات الحقول وأن تكون هذه القيم مرتبة حسب ترتيب الحقول في جملة **INSERT**

مثال

```
INSERT INTO  s_dept
VALUES              (11, 'Finance', 2);
```

٣- عند إدخال حقول التاريخ والنصوص يجب وضع القيم المدخلة بين علامتي تنصيص مفردتين

٤- يجب مراعاة وجوب إدخال قيم في الحقول الإجبارية التي تم تعريفها على أنها لا تحتوي فراغ **NOT NULL** ويتم ادخال NULL في الحقول التي بها فراغ

مثال

```
INSERT INTO  s_dept
VALUES   (13, 'Administration', NULL);
```

٥- يجوز عدم ذكر أسماء الحقول في جملة **INSERT** في حالة إدخال بيانات جميع الحقول لهذا السجل على أن تكون القيم المدخلة مرتبة حسب الترتيب الافتراضي للحقول في الجدول عند بنائه

ملاحظة
لمعرفة الترتيب الافتراضي للحقول في الجدول نستخدم الأمر **DESC**

مثال

ثانيا : اضافة عدة سجلات إلى جدول البيانات

من خلال هذه الطريقة يمكن ادخال أكثر من سجل واحد إلى جدول البيانات عن طريق استخدام متغيرات الادخال وهي عبارة عن متغيرات توضع في جملة الادخال بدلا من القيم نفسها ةيمكن أن نطلق على هذه الطريقة جملة الادخال متعددة السجلات

```
INSERT INTO  table [(column1 [, column2...])]
VALUES        (&variable1  [,&variable1  ...]);
```

حيث ان

Table	اسم الجدول المراد إلحاق السجل به
Column1,Column2	اسماء الأعمدة(الحقول) المطلوب إدخال البيانات إليها
Variable1, Variable 2	متغيرات الإدخال التي سوف يتم استبدالها بقيم حقيقية بعد تنفيذ جملة الادخال

الشرح

بعد تنفيذ جملة بهذه الطريقة سوف يطلب منك إدخال القيم للمتغيرات المذكورة في جملة الإدخال وبعد الانتهاء من ادخال القيمة تضغط مفتاح ENTER من لوحة المفاتيح وهكذا حتى تنتهي من إدخال حقول السجل الأول ولإدخال سجل آخر يمكنك الضغط على حرف (R) ثم مفتاح ENTER من لوحة المفاتيح وهو يعني تكرار الإدخال لسجلات اخرى

مثال

```
INSERT INTO  s_dept (id, name,
             region_id)
    VALUES           (&department_id,
'&department_name',
             &region_id);
```

```
Enter value for department_id: 61
Enter value for department_name: Accounting
Enter value for region_id: 2
```

تنطبق على هذه الطريقة جميع القواعد المذكورة في الطريقة الأولى ويضاف إليها مايلي :

١- تستبدل القيم في جملة الإدخال بمتغيرات
٢- يعود اختيار أسماء المتغيرات إلى المستخدم مع مراعاة شروط تسمية المتغيرات
٣- يجب أن توضع علامة & قبل متغير الادخال
٤- في جملة الإدخال يمكن وضع علامتى تنصيص مفردتين حول متغير الادخال الخاص بالحقول النصية

إضافة سجلات عن طريق نسخها من جدول آخر

من خلال هذه الطريقة يمكن إدخال أكثرمن سجل واحد إلى جدول البيانات عن طريق نسخ هذا السجل / السجلات من جدول آخر بواسطة جملة الاستفسار SELECT حيث نستخدم جملة INSERT مع استعلام فرعي انظر الى المثال التالي

```
INSERT INTO history(id, last_name, salary, title,
start_date)
SELECT id, last_name, salary,title, start_date
FROM s_emp
WHERE start_date < '01-JAN-94';
```

الشرح

بعد تنفيذ جملة الإدخال بهذه الطريقة ينم نسخ السجلات التي تحقق الشرط من الجدول ا لمصدر إلى الجدول الهدف وللحقول االمذكورة في جملة الإدخال

القواعد التي يجب التقيد بها في هذه الطريقة

١- كتابة جملة الإدخال INSERT محتوية على جملة استفسار SELECT
٢- تستبدل القيم في جملة الإدخال بأسماء حقول الجدول المصدر
٣- عدم استخدم العبارة VALUES
٤- يجب مراعاة وجوب إدخال قيم في الحقول الاجبارية التي تم تعريفها على أنها لا تحتوي فراغ NOT NULL
٥- مطابقة الحقول بين الجدولين من حيث ترتيب الحقول ونوع البيانات وعدد الحقول

ملخص الفصل

تناولنا من خلال هذا الفصل موضوع اضافة البيانات إلى الجداول بواسطة جملة INSERT وهنالك ثلاثة طرق لإجراء عملية الإضافة

أولا: إضافة سجل واحد إلى جدو ل البيانات

بهذه الطريقة سوف يتم إضافة بيانات سجل واحد فقط إلى جدول يحتوي على القيم التي ستذكر في جملة الإدخال

ثانيا : إضافة عدة سجلات إلى جدول البيانات

بهذه الطريقة يمكن إدخال أكثر من سجل واحد إلى جدول البيانات عن طريق استخدام متغيرات الادخال وهي عبارة عن متغيرات توضع في جملة الإدخال بدلا من القيم نفسها ويمكن أن نطلق على هذه الطريقة جملة الادخال متعددة السجلات

ثالثا: إضافة سجلات في جدول عن طريق نسخها من جدول آخر

بهذه الطريقة يمكن إدخال أكثر من سجل واحد إلى جدول البيانات عن طريق نسخ هذا السجل / السجلات من جدول آخر بواسطة جملة SELECT الموجودة بداخل جملة INSERT

تعديل وحذف بيانات الجداول

أهداف الفصل

بعد نهاية هذا الفصل ستكون ان شاء الله قادر على

١- الإلمام بالطرق المختلفة لتعديل البيانات في الجدول
٢- إتقان تعديل بيانات سجل واحد أو أكثر في جدول البيانات
٣- إتقان تعديل بيانات جميع السجلات في جدول البيانات
٤- إتقان حذف سجل أو أكثر من جدول البيانات
٥- إتقان حذف جميع السجلات من جدول البيانات

تعديل بيانات الجدول

في هذا الفصل سنتعلم كيفية تعديل البيانات وحذف السجلات كما أن هذا الفصل يشكل جزأ أساسيا مكملا للغة تناول البيانات DML في نظام أوراكل التي يعتبر احدى النظم القوية في نظم إدارة قواعد البيانات DBMS لما لها من أثر فعال في معالجة البيانات داخل الجدول حيث تشتمل لغة تناول البيانات DML على الاجزاء التالية

١- إدخال البيانات إلى الجداول INSERT
٢- تعديل البيانات في الجداول UPDATE
٣- حذف البيانات من الجداول DELETE

ولقد تناولنا ادخال البيانات في الفصل السابق وسندرس تعديل البيانات وحذفها في هذا الفصل

تعديل البيانات في الجدول

هي العملية التي يتم بها تعديل بيانات حقل / حقول في سجل واحد أو اكثر في جدول البيانات من خلال جملة UPDATEوتتم هذه العملية بطريقتين:

أولا: تعديل بيانات حقل / حقول لسجل واحد أو اكثر

الصيغة العامة

```
UPDATE   table SET    column1 = value [, column2 = value]
[WHERE   condition];
```

حيث أن

Table	اسم الجدول المراد تعديل البيانات فيه
Column1,column2	اسماء الحقول(الاعمده) المراد تعديل بياناتها
Value1, value2	القيم الجديدة التي سيتم إعطاؤها للحقول (الاعمدة) حقل ١ وحقل ٢ وبالتالي فإن قيمة ١ ستؤول الى الحقل ١ وهكذا
Condition	جملة شرط تحدد السجلات التي سيتم تعديل بياناتها هي السجلات التي تحقق الشرط

الشرح

بعد تنفيذ جملة التعديل بهذه الطريقة سوف يتم تعديل بيانات الحقل /الحقول المذكورة بحيث يتم إلغاء القيم الموجودة فيها لتحتوي القيم الجديدة المذكورة في جملة التعديل علما بأن السجلات التي سيتم تعديل بياناتها هي فقط السجلات التي تحقق الشرط في الجملة الشرطية

القواعد التي يجب التقيد بها في هذه الطريقة

١- قد تكون القيمة الجديدة عبارة عن تعبير حسابي يعتمد على قيم الحقول في الجدول
٢- قد تكون القيمة الجديدة ناتجة عن جملة استعلام فرعي SELECT
٣- يجب أن يكون نوع بيانات القيم الجديدة من نفس نوع بيانات الحقول التي سيتم تعديلها
٤- عند تعديل حقول التاريخ والنصوص يجب وضع القيم الجديدة بين علامتي تنصيص مفردتين
٥- يجب مراعاة وجوب إعطاء قيم للحقول التي سيتم تعديلها والتي تم تعريفها على أنها حقول إجبارية أي لا تحتوي فراغ NOT NULL

مثال

```
UPDATE   s_emp
SET      dept_id = 10
WHERE    id = 2;
```

ثانيا : تعديل بيانات حقل / حقول لجميع السجلات في الجدول

الصيغة العامة

UPDATE *table*SET *column1 = value* [, *column2 = value*]

الشرح
بعد تنفيذ جملة التعديل بهذه الطريقة سوف بتم تعديل بيانات الحقل / الحقول المذكورة بحيث يتم إلغاء القيم الموجودة فيها لتحتوي القيم الجديدة المذكورة في جملة التعديل علما بأن السجلات التي سيتم تعديل بياناتها هي حميع السجلات في الجدول

مثال

```
UPDATE        s_emp
SET           commission_pct = 10
```

حذف البيانات من الجداول

هي العملية التي يتم بها حذف بيانات سجل / سجلات من جدول البيانات من خلال جملة DELETE وتتم هذه العملية بطريقتين

أول:ا حذف سجل أو أكثر من الجدول

الصيغة العامة

```
DELETE [FROM] table[WHERE condition];
```

بعد تنفيذ جملة الحذف الطريقة سوف يتم حذف السجل / السجلات التي تحقق الشرط في الجملة الشرطية

.

القواعد التي يجب التقيد بها في هذه الطريقة :

١- يجب التأكد من بيانات السجلات التي سيتم حذفها قبل تنفيذ جملة الحذف وكذلك التأكد من الشرط في الجملة الشرطية لأن السجلات التي ستحذف تعتمد على الشرط المذكور

ثانيا: حذف جميع سجلات الجدول

الصيغة العامة

```
DELETE [FROM] table;
```

الشرح

بعد تنفيذ جملة الحذف بهذه الطريقة سوف يتم حذف جميع السجلات في الجدول

ملخص الفصل

تناولنا في هذا الفصل موضوع تعديل البيانات في الجدول بواسطة جملة UPDATE وهنالك طريقتان لاجراء عملية التعديل وهما

أولا تعديل بيانات حقل/ حقول لسجل واحد أو اكثر

حيث يتم الغاء القيم الموجودة بها لتحتوي على القيم الجديدة المذكورة في جملة التعديل علما بأن السجلات التي سيتتم التعديل بها هي السجلان التي تحقق الشرط.

ثانيا: تعديل بيانات حفل / حقول لجميع السجلات في الجدول

حيث يتم الغاء القيم الموجودة بها لتحتوي على القيم الجديدة المذكورة في جملة التعديل علما انه سيتم تعديل بيانات جميع الحقول لعدم وجود شرط

ثم تناولنا موضوع حذف السجلات عنDELETE وهنالك طريقتان لإجراء عمليةا الحذف

أولا: حذف سجل أو أكثر من الجدول

بهذه الطريقة سوف يتم حذف السجل / السجلات التي تحقق الشرط في جملة الشرط

ثانيا : حذف جميع سجلات الجدول

بهذه الطريقة سوف يتم حذف جميع السجلات في الجدول

المتسلسلات والفهارس والعروض

أهداف الفصل

في نهاية هذا الفصل ستكون ان شاء الله قادر على

١- التعامل مع المتسلسلات
٢- تكوين متسلسلة
٣- التعديل في متسلسلة
٤- حذف متسلسله
٥- معرفة الفرق بين الفهرسه التلقائية والفهرسه العادية
٦- عمل فهرسه للبيانات
٧- الغاء الفهرسه
٨- التعامل مع العروض View
٩- تكوين عرض
١٠- تعديل العروض
١١- الغاء العروض

المتسلسلات

المتسلسلة عبارة عن مجموعه من الارقام المتتالية تتولد تلقائيا مثل ١و٢و٣ ،،، حيث تستخدم كمفتاح أساسي Primary Key لانها عبارة عن قيم فريده وتتعامل معها الذاكره بسهولة وبكفائه.

الصيغة العامة

```
CREATE SEQUENCE name
          [INCREMENT BY n1]
          [START WITH n2]
          [{MAXVALUE n3 | NOMAXVALUE}]
          [{MINVALUE n4 | NOMINVALUE}]
          [{CYCLE | NOCYCLE}]
          [{CACHE n5 | NOCACHE}]
```

حيث أن

Name	اسم المتسلسلة
N1	مقدار الزيادة كل مرة
N2	اول رقم تبدأ منه المتسلسلة
N3	اقصى قيمه للمتسلسلة
N4	ادنى قيمة للمتسلسلة
N5	عدد الخانات في الذاكره

الشرح

- **السطر الاول قمنا بتعريف اسم المتسلسلة**
- السطر الثاني يتم تحديد مقدار الزيادة
- السطر الثالث الرقم الذي ستبدأ منه هذه المتسلسله
- السطر الرابع نحدد اقصى قيمه تصل اليها المتسلسلة والوضع الافتراضي عدم وجود قيمة قصوىNOMAXVALUE حيث يمكن للمتسلسلة ان تصل الى($١٠^{٢٧}$) للمتسلسلة التصاعديه وللتنازليه (-١)
- السطر الخامس يتم تحديد ادنى قيمه للمتسلسله والوضع الافتراضي عدم وجود قيمة NOMINVALUE حيث يمكن للمتسلسلة تصل الى (١) في حالة المسلسلة التصاعدية وحالة التنازليه الى (-$١٠^{٢٦}$)
- **السطر السادس cycle يسمح بتكرارا المتسلسلة اذا وصلت الى أقصى قيمه وأدنى قيمه والوضع الافتراضي هو عدم التكرار**
- السطر السادس يتم فيه حجز اماكن في الذاكره للمتسلسلة وذلك لسرعه التعامل مع المتسلسلة حيث نكتب في n5 عدد الخانات والوضوع الافتراضي أن اكون ٢٠ واذا اخترنا NOCACHE فانه لايتم حجز اماكن في الذاكره

مثال

```
CREATE SEQUENCE s_dept_id
  INCREMENT BY 1
  START WITH 51
  MAXVALUE 9999999
  NOCACHE
  NOCYCLE;
```

لكي نحصل على أفضل أداء للمتسلسلة نقوم بحجز اماكن في الذاكره لأكبر عدد من المصفوفات نعرفه ولكن اذا ستخدمنا عدد كبير فإننا سنشغل الذاكره ونبطئ التعامل مع الجهاز واذا ستخدمنا عدد قليل لن يكون كافي للمتسلسله

لذلك نستخدم مايسمى مولد المتسلسلات sequence generator

٩- استخدام المتسلسلات

اذا اردنا اضافة قيمه جديدة الى متسلسلة تستخدم

١٠- Table_name.NEXTVAL

واذا اردنا القيمة الحاليه نستخدم

Table_name.CURRVAL

ولنأخذ هذا المثال

```
INSERT INTO s_dept(id, name, region_id)
VALUES (s_dept_id.NEXTVAL, 'Finance', 2);
```

حيث قمنا بإضافة قيمه جديدة الى المتسلسلة في خانة id ثم قمنا باضافة finance الى خانة name و 2 في خانة region_id

واذا اردنا القيمة الحالية نستخدم

```
SELECT s_dept_id.CURRVAL
FROM SYS.dual;
```

لكن توجد ثغرات في المتسلسلات

١- اذا تم نقل سجل من مكان الى اخر فإنه يحدث خلل في المتسلسلة

٢- المتسلسلة يمكن أن تستخدم في اكثر من جدول

تعديل متسلسلة

يمكن تعديل مقدار الزياده أو القيمه الابتدائية للمتسلسله وتحديد اكبر وقيمه وادنى قيمه أو تغيير خيار التكراراو الذاكره عن طريق الامر ALTER

الصيغه العامه

```
ALTER SEQUENCE sequence
        [INCREMENT BY n]
        [{MAXVALUE n | NOMAXVALUE}]
        [{MINVALUE n | NOMINVALUE}]
        [{CYCLE | NOCYCLE}]
        [{CACHE n | NOCACHE}]
```

ولكن توجد قواعد لهذا التغيير

١- يجب أن تكون لك السماحية لتعديل هذه المتسلسله
٢- سيتم تطبيق التغيير الحادث على القيم الجديده
٣- اذا اردت ان يتم التغيير على جميع المتسلسلة يجب أن نقوم بمسحها اولا ثم اعادة انشائها

حذف متسلسلة

يتم حذف متسلسلة عن طريق الأمر DROP

مثال

```
DROP SEQUENCE s_dept_id;
```

الفهرسة

الفهرسة عبارة عن وسبلة مصممه لتسهيل الحصول على البيانات بسرعه من الجداول مثل الفهرس الموجود بأي كتاب فعندما نريد الوصول الى معلومة بسرعة ننظر الى الفهرس اولا وكذلك هو الحال في الأوراكل فعندما نقوم بفهرسة مجموعة من البيانات فاننا نستطيع الى احد هذه البيانات بسرعه ويتم كذلك البحث بين الجداول بسرعه ومن مميزات الفهرسه كذلك انها تقلل عدد البيانات المنتقله خلال وسائل الاخراج والادخال
من الضروري احيانا ترقيم الحقول ترقيم تلقائي مثل ارقام المناطق او ارقام السلع بدلا من الترقيم العادي

كيف تتم الفهرسة ؟

الفهرسه التلقائية

عندما نقوم بتعريف حقل على انه مفتاح أساسي أو اذا جعلناه حقل فريد فانه تلقائيا يفهرس

الفهرسة العادية

يمكن أن نقوم بفهرسة بيانات غير فريدة لتسهيل الوصول اليها

أنواع الفهرسة

توجد عدة أنواع للفهرسه حيث يمكن فهرسة حقل (عمود) واحد او مجموعة من الأعمدة منها

الفهرسة الفريدة ***unique index***
هي عبارة عن قيم مفهرسة ولكن بتقييد حيث ان كل قيمة لا تتكرر الا مرة واحدة

الفهرسة الغير فريدة ***nonunique index***
هي عبارة عن قيم مفهرسة ولكن بدون أي تقييد

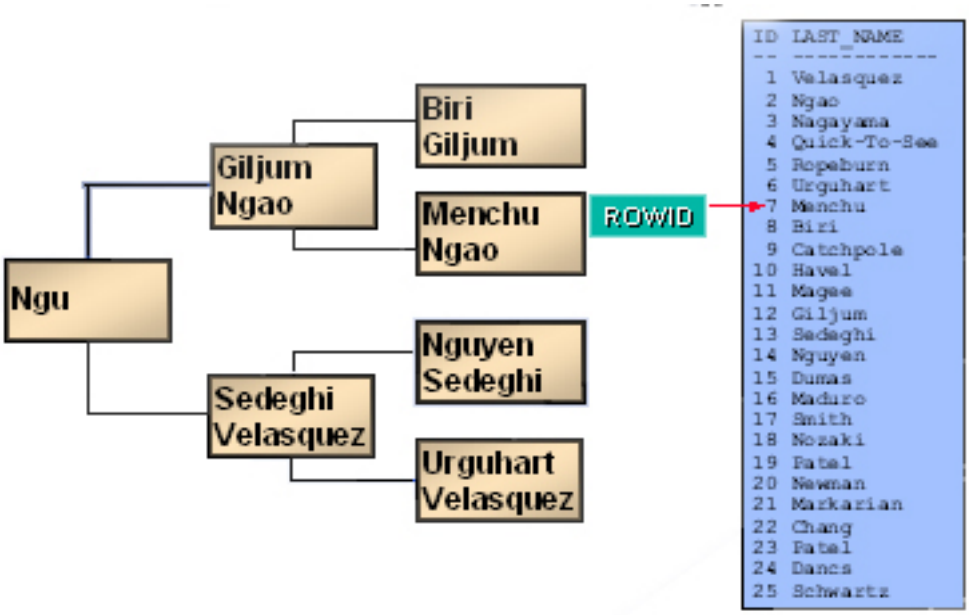
الفهرسة المركبة ***composite index***
حيث يتم فهرسة عدة اعمدة حيث أن كل عمود له قيم مختلفة

وتستخدم الفهرسة المركبه عندما تعود جملة SELECT بعدة قيم فيتم ترتيبها عن طريق استخدام الفهرسة المركبة

كيف يقوم الاوراكل بالتعامل مع الفهرسة

عندما نقوم بفهرسة حقل فإن الاوراكل يقوم بتكوين مايسمى index segment يتم فيها تخزين معلومات خاصة بالفهرسة والتي تسرع الوصول الى البيانات حيث أن هذه المعلومات توضح أماكن السجلات وتوضح أقصر الطرق للوصول اليها عن طريق وسائل الاخرج والادخال

والمصطلح الذي تعتمد عليه الفهرسة يسمى B*-tree index هي عبارة عن مخطط شجري لمجموعة من القيم مرتبة تنازليا حيث كلما ننتقل الى قيمة تاليه يتم مقارنتها مع كتلة القيم التي تعلوها وتسمى هذه الكتل branch blocks وناتج هذه المقارنه يتم مقارنته مع كتل اخرى إلى أدنى درجة في هذه الكتل ويسمى leaf blocks حيث تحتوي هذه الكتلة على كل البيانات المفهرسه ويوجد ما يسمى ROWID لهذه البيانات



اذا كانت البيانات فردية فإنه يوجد ROWID لكل قيمة ولكن اذا لمت تكن فردية فانه يمكن ان يتواجد أكثر من ROWID وفي هذه الحالة يتم ترتيبها البيانات حسب مفتاح الفهرسه ثم حسب ROWID

الصيغة العامة

```
CREATE INDEX index ON table (column[, column]...);
```

حيث أن

Index اسم الفهرس
Table اسم الجدول
Column اسن الحقل المراد فهرستة

مثال

```
CREATE INDEX     s_emp_last_name_idx
ON       s_emp(last_name);
```

عند الفهرسة يجب مراعاة مايلي

١- لا نفهرس الجداول التي استعلم عنها بعدد كبير من الصفوف حيث نستخدم الفهرسه عندما يكون الاستعلام لا يتجاوز ٥ % من الجدول
٢- لا نفهرس الحقول التي يتم التعديل فيها باستمرار
٣- لا نفهرس الجداول التي تحتوي على قيم متكررة بكثرة
٤- نفهرس الحقول التي يكون الاستعلام عنها سهل غالبا فلا نستخدم الفهرسه مع استعلامات شروطها معقدة
٥- لا نفهرس غالبا الحقول التي لا تحتوي على قيم فريدة
٦- نفهرس الحقول التي تعتبر حقول رابطه بين جدول واخر

حذف الفهرسة

لحذف الفهرسة نستخدم الأمر DROP INDEX

مثال

```
DROP INDEX s_emp_last_name_idx;
```

العروض

العرض عبارة عن استعلام من جدول واحد أو اكثر حيث يعتبر جدول وهمي يحتوي على البيانات التي يأخذها من جملة SELECT ولكن حقيقة ان هذا الجدول غير موجود

ويمكن استخدم العروض في عدة مهام منها :

- يحافظ على التحكم في مستوي الامان
- اخفاء تعقيد البينات عن المستخدم
- اعادة تسمية الأعمدة

الصيغة العامة

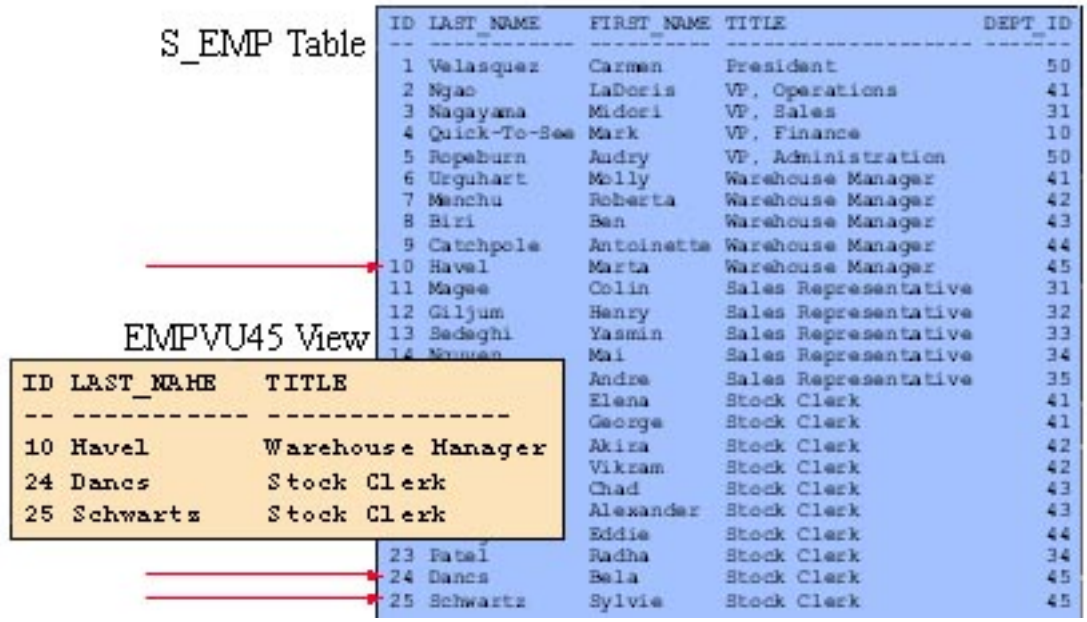
```
CREATE VIEW view-name
 (column1,...,columnN)
AS
select-statement
```

حيث أن

view-name اسم العرض
Column1 اسم الحقول التي ستم عرضها

مثال

```
CREATE VIEW empvu45
AS SELECT id, last_name, title
FROM   s_emp
WHERE  dept_id = 45;
```



ويمكنك مشاهدة المعلومات الخاصة بهذا العرض عن طريق الامر DESCRIBE فنكتب

```
DESC empvu45;
```

التحكم في مستوى الأمان

نحتاج احيانا الى أن نجعل بعض الجداول او العروض مسموحة لمستخدمين معينين كمدراء الشبكة فلذلك نقوم باضافة حقل جديد ونقيده بـ CHECK حيث اذا كان المستخدم مدير يعطي Y واذا لم يكن يعطي N ولعمل ذلك نقوم بعدة خطوات

اولا

```
alter table S_EMP
add
 (Manager char(1) check (Manager in ('Y','N')));
```

ثانيا

```
update S_EMP
set Manager = 'N'
where
ID != 1001;
```

ثالثا

```
update S_EMP
set Manager = 'Y'
where
ID = 1001;
```

في اخر خطوتين قمنا بتحديد شرط وهو في حالة اذاكان رقم الموظف ١٠٠١ فانه هو المدير لذلك في حقل Manager يعطي Y وغير ذلك يعطي N

رابعا

في هذه الخطوة والتي تليها سنقوم بإضافة حقل لاسم مستخدم حيث اسم المستخدم سيكون عبارة عن اول حرف من الاسم الأول للموظف بالاضافة الى اسمه الاخير

```
alter table S_EMP
add
(username varchar2(31));
```

خامسا

```
update S_EMP
set username = substr(first_name,1,1) ||
last_name;
```

سادسا

الان سنقوم بعمل عرض اسمه Re_SALARY مربوط بالجدول S_EMP

```
create view Re_SALARY as
select e.id, e.last_name, e.first_name,
decode(s.manager,'Y',e.salary,null) salary
from S_EMP e, S_EMP s
where
user = s.username(+);
```

user اسم المستخدم لقاعدة البيانات الحالي لذلك قيمة هذا المتغير تعتمد على المستخدم الحالي للقاعدة حيث قمنا بعمل عملية ربط خارجي outer-join شرطها أنه اذا كان المستخدم الحالي لقاعدة البيانات اسمه موجود في حقل اسماء المستخدمين في جدول الموظفين فانه يعود بقيمة في حقل Manager اما Y او N كما قلنا حسب رقم الموظف والدالة Decode تقوم مشاهدة قيمة Manager اذا كانت Y فإن المستخدم مدير وبتم عرض حقل المرتب واذا كانت N لا يتم عرض حقل المرتب

فلنفرض أنAhmed هو المدير فاذا قام بكتابة الاستعلام

```
select *
from Re_Salary;
```

فان المخرجات ستكون

ID	LAST_NAME	FIRST_NAME	SALARY
1001	AHMED	AHMED	1500
1002	HERNANDEZ	RANDY	3000
1003	EASON	PAUL	1500
1004	BARRETT	SARAH	1500
1005	HIGGINS	BEN	3000
1006	YEN	CINDY	1500
1007	GILROY	MAX	1500
1008	CARSON	BETH	3000
1009	SWANSON	HARRY	2500

ولكن اذا قام MAX بالاستعلام فإن المخرجات :

ID	LAST_NAME	FIRST_NAME	SALARY
1001	AHMED	AHMED	
1002	HERNANDEZ	RANDY	
1003	EASON	PAUL	
1004	BARRETT	SARAH	
1005	HIGGINS	BEN	
1006	YEN	CINDY	
1007	GILROY	MAX	
1008	CARSON	BETH	
1009	SWANSON	HARRY	

تعديل العرض

يتم التعديل في العرض عن طريق الامر

```
 11-CREATE OR REPLACE
```

مثال

```
CREATE OR REPLACE VIEW empvu45
       (id_number, employee, job)
AS SELECT id, last_name, title
FROM   s_emp
WHERE  dept_id = 45;
```

العرض من عرض اخر

يمكن أن نقوم بعمل عرض من عرض اخر

مثال

```
create view EMBVY45
as
select ID, Last_Name, First_Name,
Middle_Initial,HireDate,
from S_EMP;
```

حذف عرض

لالغاء عرض نستخدم الامر DROP

```
DROP VIEW empvu45;
```

ملخص الفصل

تناولنا في هذا الفصل كيفية تكوين المتسلسلات وتعدبلها وطريقة حذفها ثم تطرقنا الى الفهرسة وفوائدها وانواعها وكيفية عملها وتعديلها وكذلك الغائها وفي اخر الفصل تطرقنا الى العروض مفهومها وفوائدها وافكارها وكيفية تعديلها والغائها

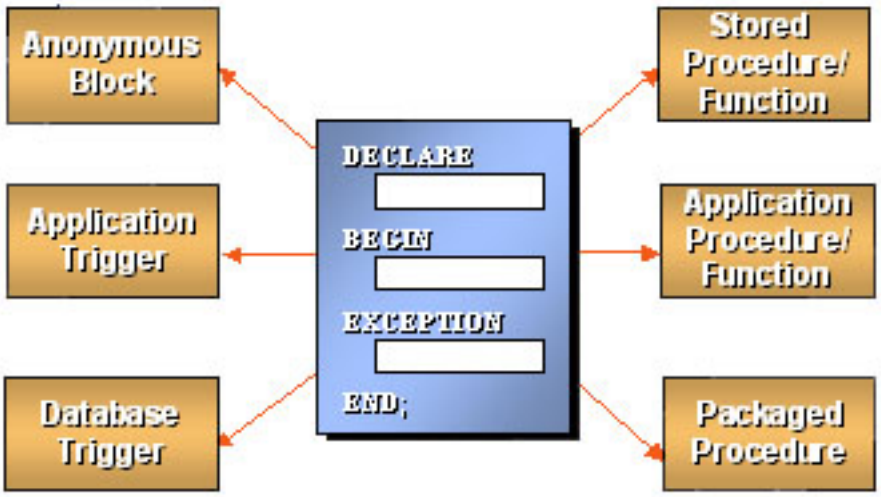
نبذه PL/SQL

أهداف الفصل.

١- التعرف على PL/SQL
٢- استخدام عبارة الشرط IF - THEN
٣- استخدام التكرار

التعرف على PL/SQL

هي عبارة عن كتل برمجية تشبه في في طريقة كتابتها لغة C بالإضافة لدعمها SQL وتحتوي هذه اللغة على اجراءات خاصة بها واومر مثل IF وغيرها وكذلك دوال وويمكن تمثيل بنية البرنامج في PL/SQL في الشكل التالي



برامج pl/sql تتم كتابتها في كتل من اوامر البرمجة تحتوي على مقاطع منفصلة للاعلان عن المتغيرات واوامر البرامج ومعالجة الاستثناءت(الاخطاء) .
ومن الممكن تخزين الاجراء في قاعدة البيانات كبرنامج فرعي له اسم محدد او كتابتها مباشرة في sql * plus ككتله مجهولة.

وكتلة البرنامج كمايلي:

سنبدا اولا كتابة الاجراء مباشرة في sql * plus وهي كمايلي:

```
DECLARE
هنا توجد تعريفات المتغيرات والموشرات
BEGIN
جسم البرنامج

EXCEPTION
رموز معالجة الاخطاء
END:
```

مع ملاحظة مايلي:

ان قسم declare وقسم exception هما اختيارين اي لايشترط وجودهما

اي اذا كان لايوجد لديك تعريف متغيرات لاتستخدم declare واذا كنت لان تتعامل مع الاخطاء لاتستخدم exception
والصيغة العامة لتعريف المتغيرات في declare

identifier **[CONSTANT]** ***datatype*** **[NOT NULL]**
[:= | DEFAULT ***expr*****];**

مثال

```
v_gender CHAR(1);
v_count  BINARY_INTEGER := 0;
v_total_sal  NUMBER(9,2) := 0;
v_order_date DATE := SYSDATE + 7;
c_tax_rate   CONSTANT NUMBER(3,2) := 8.25;
v_valid  BOOLEAN NOT NULL := TRUE;
```

بالإضافة الى المتغيرات التي توجد SQL توجد هنالك متغيرات تدعمها لغة ال PL\SQL

١- المتغير BINARY_INTEGER

وتشمل الأرقام الصحيحة بين –٢.١٤٧.٤٨٣.٦٤٧و ٢.١٤٧.٤٨٣.٦٤٧

٢- المتغير PLS_INTEGER

وتشمل الأرقام الصحيحة بين –٢.١٤٧.٤٨٣.٦٤٧و ٢.١٤٧.٤٨٣.٦٤٧ وهذا النوع لا يشغل مساحة كبيرة من الذاكر ة وأسرع من BINARY_INTEGER و Number

٣- NATURAL

وتشمل الارقام من ٠ الى ٢.١٤٧.٤٨٣.٦٤٧

٤- POSITIVE
وتشمل الارقام الموجبة فقط من ١ الى ٢.١٤٧.٤٨٣.٦٤٧

٣- BOOLEAN

وهو متغير يحمل القيمة TRUE أو FALSE أو NULL

٤-%TYPE

صمم هذا المتغير ليحمل نفس نوع البيانات التي يحملها حقل معين

٤- %ROWTYPE

صمم هذا المتغير ليكون متغير مركب يحمل نفس توع البيانات التي يحملها صف معين

طريقة تعريف المتغير من النوع %TYPE

الصيغة العامة

variable-name table-name.column-name%TYPE;

مثال

```
v_last_name       s_emp.last_name%TYPE;
```

متغير

last_name في جدول S_EMP ويحتوي على نفس القيم

طريقة تعريف المتغير من النوع % ROWTYPE

الصيغة العامة

variable-name table-name%ROWTYPE;

مثال

```
emp_record   s_emp%ROWTYPE;
```

يف المتغير المركب
s_emp

طرق الاسناد

مثل اذا اردت ان تقول ان قيمة i=5 فيتم ذلك كمايلي:

```
i:=5;
```

يجب وضع النقطتين قبل =

سوف ناخذ مثال على ذلك

يوجد ضمن اوامر sql الامر DBMS_OUTPUT.PUT_LINE
يستخدم لكي تعرض النتيجة في sql * plus وهو أمر تابع للحزمة DBMS_OUTPUT
فهذا الامر يقوم بعرض الجمل ويمكن أن يعرض كذلك التواريخ والارقام واذا اردنا ان يعرض انواع بيانات اخرى نستخدم الامر TO_CHAR

والصيغة العامة له :

DBMS_OUTPUT.PUT_LINE(massege)

حيث

massage هي النص او الشي الذي تريد عرضة

تم شرح هذا الامر لكي نبدأ به ونستخدمه لفهم اوامر ال pl/sql لكن في المستقبل سوف تعرف ان هذا الامر لايهمك كثيرا

.مثال :

نريد طباعة "hi all" على الشاشة sql * plus يتم ذلك كمايلي:

```
SET SERVEROUTPUT ON;
BEGIN
DBMS_OUTPUT.PUT_LINE('hi all');
End;
```

جرب هذا ولاحظ النتائج ولعلك تتسال عن سبب وجود السطر الاول
السطر الاول يخبر sql * plus بأن يكتب كل مايعود به الخادم.
ويكفي كتابته مرة واحدة عندما تدخل sql * plus

مثال اخر باستخدام المتغيرات

```
Declare
i   number(5);
BEGIN
i:=5;
DBMS_OUTPUT.PUT_LINE('i = ' || i);
END;
```

فائدة || الموجدة ضمن عملة الطباعة هي للوصل بين التعبيرين

عبارة الشرط IF -THEN

تستخدم هذه العبارة مثل اي العبارت الشرطية في لغة سي او سي++ او فيجوال بيسك وغيرها ، ولها استخدمات عديدة وسوف نعرف كيف نستخدمها مقدما مع حقول قواعد البيانات وذلك بعد اخذ المؤشرات

الصيغة العامة لها كمايلي:

```
IF condition THEN
   statement; ...  statement;
 [ELSIF condition THEN
statement; ...  statement;]
...
 [ELSIF condition THEN
statement; ...  statement;]
 [ELSE
statement; ...  statement;]
END IF;
```

حيث أن

conditional هي الشرط

ملاحظة
• ELSEو ELSIF اختيارية • جملة IF نحنوي على عدة عبارات ELSIF ولكن تحتوي على عبارة ELSE واحدة • انتبه الى أن طريقة كتابة ELSIF وليست ELSEIF

مثال على ذلك :

```
Declare
i   number(5);
BEGIN
i:=5;
IF i=5 then
DBMS_OUTPUT.PUT_LINE('i = ' || i);
ELSE
DBMS_OUTPUT.PUT_LINE('i not eqal 5 ');
    END IF;
END;
```

الشرط باستخدام اكثر من شرط

```
Declare
i   number(5);
BEGIN
i:=5;
IF i>1 then
DBMS_OUTPUT.PUT_LINE(i  || ' > 1');
ELSIF i<1 then
DBMS_OUTPUT.PUT_LINE(i  || ' < 1');
ELSIF i=1 then
DBMS_OUTPUT.PUT_LINE(i  || ' = 1');
   END IF;
END;
```

الـــتكرار

يوجد عدة اوامر للتكرار وهي:

١- loop-exit-end

وهنا لابد من وضع شرط لأنهاء الحلقة
نأخذ مثال بسيط على ذلك

```
Declare
i   number(5);
BEGIN
i:=1;
LOOP
IF i>10 then
EXIT;
END IF;
DBMS_OUTPUT.PUT_LINE('i =' || i);
i:=i+1;
End loop;
END;
/
```

الشرح

السطر الثاني : تعريف متغير من نوع رقم
السطر الثالث :البداية
السطر الرابع : اعطاء المتغير قيمة ابتدائية وهي i=1
السطر الخامس : شرط الانهاء
السطر السادس : الانهاء اذا كان i>10 ولا يكمل
السطر السابع :انها if
السطر الثامن : طباعة i
السطر التاسع :زيادة قيمة i بواحد
السطر العاشر : نهاية الحلقة

وبعد كتابة هذا الكود يكون الناتج كمايلي:

i =1
i =2
i= 3
i =4
i =5
i =6
i =7
i =8
i =9
i =10

٢- EXIT - END LOOP - WHEN

```
Declare
i   number(5);
BEGIN
i:=1;
LOOP
EXIT WHEN i>10;
DBMS_OUTPUT.PUT_LINE('i =' || i);
i:=i+1;
End loop;
END;
/
```

ويكون الناتج نفس السابق لكن لاحظ استخدم شرط الانهاء

EXIT WHEN i>10;

٣- END - LOOP - WHILE

```
Declare
i   number(5);
BEGIN
i:=1;
WHILE i <= 10 LOOP
DBMS_OUTPUT.PUT_LINE('i =' || i);
```

```
i:=i+1;
End loop;
END;
/
```

ويكون الناتج نفس المثال السابق

٤- FOR - IN - LOOP - END

وهذه ايضا طريقة اخرى لاستخدام حلقات التكرار وهي نفس عمل اسلوب حلقات for في اي لغة برمجة

والصيغة العامه لها هي على الاتي

```
FOR  i  IN  البداية..النهاية  LOOP
المراد تكرارها الجمل
END LOOP
```

مثـــال:

```
Declare
i   number(5);
BEGIN
FOR  i   IN  1..10 LOOP
DBMS_OUTPUT.PUT_LINE('i =' || i);
End loop;
END;
/
```

وسوف يكون الناتج كمايلي(نفس السابق):

١= i
i =2
i =3
i =4
i =5
i =6
i =7
i =8
i =9
١٠= i

جملة GOTO

كما في باقي لغات البرمجة فان GOTO تنقل تسلسل عمل البرامج من نقطه الى اخرى

الصيغة العامة

GOTO <<my_label>>

حيث أن

my_label عنوان النقطة التي سيتم الانتقال اليها

مثال

```
declare
i  positive := 1;
max_loops constant positive := 100;
begin
i := 1;
loop
i := i + 1;
if i > max_loops then
goto more_processing;
end if;
end loop;
<<more_processing>>
i := 1;
end;
  /
```

التعليقات

يمكن كتابة التعليقات في PL/SQL بعدة طرق

١- كتابة (--) ثم التعليق

مثال

```
-- DON'T FORGET  MY NAME
```

٢- كتابة (***) ثم التعليق ثم (***) مرة أخرى

مثال

```
*** DON'T FORGET  MY NAME ***
```

٣- كتابة (*/) ثن التعليق ثم (/*) مرة اخرى

مثال

```
/* DON'T FORGET  MY NAME */
```

استخدام أوامر SQL في كتلPL/SQL

يمكنك استعمال جمل SQL داخل الكتل البرمجية الخاصة بـ PL/SQL ولكن مع وجود اختلافات حيث كل جملة SQL تنتهي بفاصلة منقوطة (;)

مثال

```
DECLARE
max_records CONSTANT int := 100;
i           int := 1;
BEGIN
FOR i IN 1..max_records LOOP
  if (mod(i,10) = 0) then
    INSERT INTO test_table
           (record_number, current_date)
    VALUES
           (i, SYSDATE);
  else
    NULL;
  end if;
END LOOP;
COMMIT;
END;
/
```

في هذا المثال قمنا باستخدام الامرINSERT وsysdate وهي من أوامر SQL

المؤشرات

اهداف الفصل

قي مهاية هذا الفصل إن شاء الله ستكون قادر على

١- التعامل مع المؤشرات الصريحه
٢- النعامل مع المؤشرات الضمنية

المؤشرات CURSORS

تستخدم pl/sql المؤشرات cursors لأدارة عبارات التحديد select في لغة sql وكما لاحظنا الاوامر السابقة مثل if والتكرار لم نستخدمها مع بيانات الجداول المخزنه ولعمل ذلك لابد من استخدام هذه المؤشرات.
وهناك نوعين من المؤشرات هي الضمنية والصريحة وسوف نتطرق لك واحد بالتفصيل والامثلة اللازمة.

١- المؤشرات الصريحه :

يتم تعريف هذا النوع من المؤشرات كجزء من الاعلان declare ويجب ان تشتمل عبارة sql المعرفه على عبارة التحديد select فقط حيث لايمكن استخدام الكلمات الاساسية insert,update,delete
وعند استخدام المؤشرات الصريحه دائما ماستكتب اربعة مكونات كمايلي:
١- يتم تعريف المؤشر في الجزء declare
٢- يتم فتح المؤشر بعد عبارة begin

الصيغة العامة لتعريف المؤشر الصريح كمايلي:

```
DECLARE
      IS   اسم المؤشر   CURSOR
الاستعلام
```

تقوم باستبدال اسم المؤشر باسم مؤشر حقيقي
وتقوم بوضع جملة الاستعلام select في مكان الاستعلام

ولكي تقوم بفتح هذا المؤشر وتستخدمه نقوم بفتحه باستخدام الامر open كمايلي:

OPEN اسم المؤشر

وبعد فتح المؤشر تقوم باسترجاع او تحميل البيانات سطر(سجل) واحد من المؤشر الذي تم تعريفه باستخدام الامر FETCH كمايلي:

FETCH اسم المؤشر INTO متغير١،متغير٢،.....

ومعنى هذا اي قم باسترجاع البيانات من المؤشر المعطى اسمه وحملها into الى المتغيرات كع ملاحظة ان عدد المتغيرات يساوي عدد الحقول الموجودة في استعلام المؤشر.
وبعد الانتهاء من اجراء العمليات على المؤشر يجب عليك اغلاقه ويتم اغلاقه كمايلي:

```
close cursor_name
```

مثال على طريقة تعريف مؤشر :

افرض انه لدينا هذا الجدول

age	name	no
23	mohammed	111
22	talal	222
24	majed	333

اولا قم بانشاء هذا الجدول كمايلي:

```
create table stud(
     no number(4),
name varchar2(40),
  age number(2));
```

ثانيا قم بادخال البيانات السابقة في هذا الجدول كمايلي:

```
insert into stud values(111,'mohammed',23);
insert into stud values(222,'talal',22);
insert into stud values(333,'majed',24);
```

ثم قم بتنفيذ مايلي

```
set serveroutput on;
DECLARE
name_stu varchar2(40);
CURSOR  name_student IS
select name from stud
where no=111;
BEGIN
OPEN name_student;
FETCH name_student INTO name_stu;
DBMS_OUTPUT.PUT_LINE(name_stu);
CLOSE name_student;
END;
/
```

بعد التنفيذ سوف يظهر لك الناتج كمايلي:

mohammed

وهذا الشي صحيح

لاحظ اننا اتبعنا نفس الخطوات التي ذكرناه لكي نتعامل مع مؤشر صريح

لاحظنا في المثال السابق ان الاستعلام في cursor سوف يعود بسجل واحد لكن ماذا يحدث لو اعاد المؤشر اكثر من سجل واردنا المرور على كافة السجلات ؟

لحل السؤال السابق لابد من استخدام حلقة بها شرط وهذا هو هل سجلات المؤشر انتهت ام لا ونعرف ذلك من خلال خاصية found للمؤشر كمايلي:

found%mycur

حيث :

mycur : هي اسم المؤشر.

% : توضح انا مايلي اسم المؤشر هي احد خصائصه.

found : خاصية التي من خلالها نعرف هل تم الانتهاء من جميع السجلات ام لا

مثال :

اسم الطالب	كود المقرر	الدرجة	النتيجة
NO_STU	SUBJECT	MARK	RESULT
111	216CS	88	
222	225CS	75	
333	225CS	40	

نريد انشاء اجراء يقوم بالمرور على الجدول وينظر الى درجة الطالب اذا كان ناجح في المقرر ام لا فاذا كان mark اكبر او يساوي ٥٠ ضع قيمة true في حقل result والا ضع قيمة false في حقل result

نقوم اولا بانشاء هذا الجدول :

```
create table stu_study(
NO_STU number(4),
SUBJECT varchar2(8),
MARK number(3),
RESULT varchar2(20));
```

المدخلات السابقة وبعد انشاء الجدول نقوم بادخال

```
insert into stu_study (NO_STU,SUBJECT,MARK)
values (111,'216CS',88);

insert into stu_study (NO_STU,SUBJECT,MARK)
values (222,'225CS',75);

insert into stu_study (NO_STU,SUBJECT,MARK)
values (333,'225CS',40);
```

بعد ذلك نقوم بانشاء الاجراء:

```
declare
mar number(3);
no number(3);
cursor res_stu is
select no_stu,mark
from stu_study;
begin
open res_stu;
loop
fetch res_stu into no,mar;
exit when res_stu%notfound;
if mar>=50 then
update stu_study set result='TRUE' where
no_stu=no;
else
update stu_study set result='FALSE' where
no_stu=no;
end if;
end loop;
close res_stu;
end;
/
```

وبهذا تكون النتائج في الجدول كمايلي :

NO_STU	SUBJECT	MARK	RESULT
111	216CS	88	TRUE
222	225CS	75	TRUE
333	225CS	40	FALSE

هناك طريقة اخرى لتعريف المتغيرات لاحظ في الجدول السابق ان الحقل no_Stu تم تعريفه على انه من نوع number وتم تعريف المتغير no في الاجراء على انه number ايضا لكي يتم وضع رقم الطالب فيه لكن لاحظ لو تم تغير نوع الحقل في الجدول من number الى varchar2 فانه يجب عليك تغير نوع المتغير no في الاجراء ايضا لكن هناك طريقه تجعلك لاتعدل الاجراء كل مرة وهي استخدام الامر التالي لتعريف المتغير no في الاجراء

type%no_stu.stu_study NO

حيث :

NO هي اسم المتغير

stu_study : اسم الجدول

no_stu : الحقل المطلوب في الجدول

type% : خاصية نوع الحقل

ومعنى ماسبق قم بتعريف متغير اسمه no له نفس نوعية الحقل الذي اسمه NO_STU الموجود في الجدول . stu_study

وبهذا لان تقوم بتغير نوع العنصر في الاجراء في كل مرة تغير النوع وهكذا مع جميع المتغيرات التي لها صله بالجداول

وبذلك يصبح الاجراء بعد التعديل كمايلي:

```
declare
mar stu_study.mark%type;
no stu_study.no_stu%type;
cursor res_stu is
select no_stu,mark
from stu_study;
begin
open res_stu;
loop
fetch res_stu into no,mar;
exit when res_stu%notfound;
if mar>=50 then
update stu_study set result='TRUE' where
no_stu=no;
else
update stu_study set result='FALSE' where
no_stu=no;
end if;
```

```
end loop;
close res_stu;
end;
/
```

المؤشرات الضمنية

تعرفنا سابقا على فائد المؤشرات cursors ودرسنا النوع الاول منها وفي هذا الفصل عندنا نوع اخر وهو المؤشرات الضمنية وهي اسهل من المؤشرات الصريحه

وتوجد نقطتين هامتين عند التعامل مع المؤشرات الضمنيه :

* يظهر المؤشر الضمني في جسم الاجراء body وليس في declare الخاص بالاجراء كمافي المؤشرات الصريحه

* لابد ان يسترجع مؤشر select الضمني سطر واحد.

والصيغة العامه للمؤشر الضمني كمايلي:

```
SELECT COLUM1,COLUM2,.......  INTO
VARIABLE1,VARIABLE2,.....

FROM  table_name
```

ومعنى هذا قم باختيار الحقل١ و الحقل٢ وضعها في المتغيرات منغير١ و متغير٢ من الجدول table_name

سوف نأخذ مثال على ذلك وسوف نستخدم الجدول الذي انشئناه سابق في الدرس الثاني عندما تعاملنا مع المؤشرات الصريحة وكان اسم الجدول stud

no	name	age
111	mohammed	23
222	talal	22
333	majed	24

واردنا مثلا كتابة اجراء يقوم بحساب متوسط اعمار الطلاب (قد يقول البعض انه لايحتاج ذلك الى اجراء فمجرد استخدام جملة select نستطيع عمل ذلك انا اقول نعم هذا صحيح لكن احب استخدام الاجراء في هذا المثال لكي نرى طريقة عمل المؤشر الضمني ولكن سوف نرى بعد قليل مثال شامل يتم فيه استخدام الموشرات الصريحه والضمنية في نفس الوقت) والان نقوم بكتابة الاجراء كمايلي :

```
set serveroutput on;
declare
aveage number(4,2);
begin
select avg(age)
into aveage
from stud;
DBMS_OUTPUT.PUT_LINE(aveage);
end;
/
```

*** مثال شامل لاستخدام الموشرات الصريحة والضمنيه في نفس الوقت :

لنفرض انه لدينا الجدولين التاليين : الجدول الاول اسمه courses (المقررات):

رقم المقرر	اسم المقرر	عدد الساعات
code	course_name	hours
216CS	NETWORK	3
225CS	ASSEMBLY	3
325CS	DATABASE	4

وقم بانشاء الجدول كمايلي:

```
create table courses(
code varchar2(8),
course_name varchar2(40),
hours number(3),
primary key(code));
```

وقم بادخال البيانات الموجود بالجدول كمايلي:

```
insert into courses values('216CS','NETWORK',3);

 insert into courses
values('225CS','ASSEMBLY',3);
```

```
 insert into courses
values('325CS','DATABASE',4);
```

ثم نقوم بتكوين الجدول الثاني وهو studys :

اسم الطالب	كود المقرر	الدرجة	عدد النقاط
NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	
222	225CS	75	
333	225CS	40	
111	225CS	90	
222	216CS	78	
333	216CS	85	

ويتم الانشاء كمايلي:

```
create table studys(
NO_STU varchar2(6),
COURSE_CODE varchar2(8),
MARK number(3),
point number(5,2),
primary key(NO_STU,COURSE_CODE));
```

بالبيانات الموجودة بالجدول كمايلي: ويتم ادخال

```
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('111','216CS',88);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('222','225CS',75);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('333','225CS',40);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('111','225CS',90);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('222','216CS',75);
```

```
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('333','216CS',85);
```

بعد الانتهاء من انشاء وادخال البيانات المطلوب انشاء اجراء يقوم بحساب عدد النقاط لكل طالب وفي كل مادة وهو الحقل عدد النقاط الذي لم ندخل فيه اي شيء ويجب نعلم ان :

MARK	average
95-100	5
90-94	4.75
85-89	4.5
80-84	4
75-79	3.5
70-74	3
65-69	2.5
60-64	2
1-59	1

ويتم حساب النقاط كمايلي :

عدد النقاط في اي مقرر = معدل المادة (وليس الدرجة كمافي الجدول السابق) * عدد ساعات المقرر

مثال لحساب معدل الطالب الذي رقمه ١١١ في المقرر 216CS

نلاحظ من جدول studys ان الطالب قد تحصل على درجة ٨٨ ونلاحظ ان الدرجة من الجدول السابق هي بين ٨٥- ٨٩ وبالتالي فإن معدل الطالب في هذا المقرر هو ٤.٥ (وهي الطريقة المتبعة في اغلب الجامعات) ، ومن جدول courses نحصل على عدد الساعات للمقرر وبالتالي فان :

عدد النقاط = ٤.٥ * ٣ = ١٣.٥ وهكذا في جميع الطلاب وهذا هو المطلوب من الاجراء عمله.

وبالتالي فان الاجراء سوف يكون كمايلي:

```
DECLARE
no_Student studys.NO_STU%type;
hou courses.hours%type;
mark studys.mark%type;
cou_code courses.code%type;
poi studys.point%type;
cursor st_point is
```

```
select NO_STU,COURSE_CODE,MARK from studys;
BEGIN
open st_point;
loop
exit when st_point%notfound;
fetch st_point into no_Student,cou_code,mark;
select hours
into hou
from courses
where code=cou_code ;
if (mark>=95)and(mark<=100) then
poi:=5 * hou;
elsif mark>=90 then
poi:=4.75 * hou;
elsif mark>=85 then
poi:=4.5 * hou;
elsif mark>=80 then
poi:=4 * hou;
elsif mark>=75 then
poi:=3.5 * hou;
elsif mark>=70 then
poi:=3 * hou;
elsif mark>=65 then
poi:=2.5 * hou;
elsif mark>=60 then
poi:=2 * hou;
else
poi:=1 * hou;
end if;
update studys set POINT=poi
where NO_STU=no_Student and COURSE_CODE=cou_code
;
end loop;
close st_point;
end;

/
```

لاحظ هنا اننا استخدمنا المؤشرات الصريحة والمؤشرات الضمنيه والصريحة استخدمناه لكي تقوم بفتح سجلات الجدول studys والمؤشر الضمنلي استخدمنا لكي يعود بعدد الساعات في كل مرة يدور بالحلقة.

شرح الاجراء :

في التعريفات اتوقع انه لاتوجد هناك مشكلة لديكم ، اما جسم البرنامج ابتداً من begin فهو كمايلي :

اولا يفتح المؤشر الصريح والذي يحتوي على جميع سجلات الجدول studys ثم يكون حلقة دورانية لكي يمر على جميع سجلات الطلاب الموجودة في المؤشر الصريح وطبعا شرط الانهاء لهذه الحلقة هو الوصول الى اخر سجل . ثم يقوم بعملية تحديث سجلات الطالب الاولى في المتغيرات كمايلي:

```
fetch st_point into no_Student,cou_code,mark;
```

وطبعا المتغيرات هي رقم الطالب ورقم المقرر والدرجة في المقرر ولنفرض الان نحن الان عند السجل الاول ووهو الطالب الذي رقمه ١١١ ورقم المقرر CS٢١٦ ودرجته هي ٨٨ سوف يضع هذه البيانات في المتغيرات، ثم يستخدم مؤشر ضمني لكي يحضر عدد ساعات المادة التي درسها الطالب ١١١ وهي CS٢١٦ و المؤشر هو

```
select hours
into hou
from courses
where code=cou_code ;
```

ومعنى هذا احضر عدد ساعات المقرر الذي رقمه هو cou_code وهذا المتغير هو معروف من المؤشر الصريح الاول وسبب استخدامنا هذا المؤشر هو ان عدد ساعات المقرر موجودة في جدول اخر ولابد من استخدام هذا المؤشر لكي يحضر عدد الساعات. وبما اننا فرضنا اننا عند السجل الاول فسوف يحضر عدد ساعات المقرر CS٢١٦ وهي ٣ ساعات ثم بدأ يختبر الدرجة وذلك طبقا للجدول الدرجات والمعدلات حيث كانت درجة الطالب الذي رقمه ١١١ في المقرر cCS٢١٦ هي ٨٨ وبالتالي يكون عدد النقاط كمايلي = ٤.٧٥ * ٣ = ١٣.٥ ، وبعد الانتهاء من حساب المعدل يقوم بنعديل الجدول وتحديث قيمة point بقيمتها الجديدة ، وهكذا يمر على كل طالب بنفس الطريقة السابقة الى ان يصل الى نهاية السجلات.وبالتالي تكون النتائج كمايلي في الجدول studys :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3

111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

بعد الانتهاء من هذا المثال نكون انهينا المؤشرات بنوعيها بشكل تام وبامثله واقعيه

المصفوفات و الاجرائيات و الدوال المخزنة

اهداف الفصل

سنتعلم في هذا القصل باذن الله

١- انشاء المصفوفات والتعامل معها

٢- كيفية انشاء اجراء

٣- التعامل مع الوظائف المخزنة

الجداول في pl/sql (المصفوفات)

تستخدم هذه الجداول (المصفوفات) مثل المصفوفات في هي لغة من لغات البرمجة مثل لو كانت لديك سلسلة من الارقام وتريد تخزينها فانك تستخدم هذه الجداول للتخزين ويتم تعريف متغير من هذا النوع كمايلي اولا يتم تعريف هذا النوع :

TYPE اسم_النوع IS TABLE OF نوع_المتغير INDEX BY BINARY_INTEGER

مثال على ذلك :

```
DECLARE
TYPE num_array IS TABLE OF number(4) INDEX BY
BINARY_INTEGER;
num num_array;
BEGIN
........
........
END;
```

لاحظ اولا تم تعريف نوع واسماه num_array ، ثم قام بتعريف متغير num واعطاه نوع num_array وهو النوع الجديد الذي قمنا بانشاءه.

مثال عملي /

```
set serveroutput on;
DECLARE
TYPE num_array IS TABLE OF number(4) INDEX BY
BINARY_INTEGER;
i   number(4);
num    num_array;
BEGIN
FOR  i  IN  1..10 LOOP
num(i) := i * i ;
END LOOP;
FOR  i  IN  1..10 LOOP
DBMS_OUTPUT.PUT_LINE(i || '*' || i || '= ' ||
num(i) );
END LOOP;
END;
/
```

وويكون عمل هذا الاجراء كمايلي : الحلقة الاولى تقوم بضرب العدد i في نفسه وتخزنه في المتغير num برتبه i وهكذا والحلقة الثانية للطباعة ويكون الناتج كمايلي :

1*1= 1
2*2= 4
3*3= 9
4*4= 16
5*5= 25
6*6= 36
7*7= 49
8*8= 64
9*9= 81
10*10= 100

الاجرائيات المخزنة

شاهدنا في الدروس الماضية ان اي اجراء نقوم بكتابة اني اذا اردت استخدامة اكثر من مرة فاني اقوم بكتابة كل مرة في sql * plus لكي احصل على النتائج لكن ماهو رأيك لو نقوم بتخزين هذا الاجراء في قاعدة البيانات ونعطية اسم وحينما نحتاجه نستدعية باسمه وهذا يوفر علينا الشيء الكثير لذلك فصلنا هو الاجرائيات المخزنة.

ولكي نقوم بانشاء اجراء مخزن نقوم بمايلي :

```
CREATE [OR REPLACE] PROCEDURE procedure-name
 [(argument1 ... [, argumentN) ] IS
 [local-variable-declarations]
BEGIN
executable-section
 [exception-section]
END [procedure-name];
```

حيث أن

procedure_name اسم الاجراء المستخدم.

اما OR REPLACE فهي توضع حينما تعلم ان الاجراء موجود من السابق.

اما عن المغيرات التي بين القوسين فهي اما متغيرات مدخله مثل اذا كان لديك اجراء حساب معدل طالب وتريد تمرير رقم الطالب الذي تريد حساب معدله فهذه هي تعتبر كمدخلات ولتعريف متغير بهذا الشكل يكون كمايلي :

```
student_id  in number(9)
```

لاحظ اسم المتغير هو student_id ثم بعده وضعنا الكلمة in ومعنى ان هذا المتغير يعتبر كمدخل

اما لتعريف متغير يعود بقيمة من الاجراء مثلا لو اردنا تعرف متغبر يرجع بمعدل الطالب يتم التعريف كمايلي :

```
ave out number(5,2)
```

بعد تنفيذ الاجراء يكون هذا المتغير يحتوي على معدل الطالب الذي تم تمرير رقمه مثلا.

مع العلم انه يمكن تعريف متغير للمدخلات والمخرجات حيث تمرر به القيمه اولا وبعد تنفيذ الاجراء يتم وضع القيمه في نفس المتغير وتتم كمايلي :

```
ave in out number(5,2)
```

ومعنى هذا اي مدخل ومخرج في نفس الوقت .

مـــثال :

في الجدول الذي قمنا بدراسته في الفصول السابقة وكان بأسم studys

وكان كمايلي :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3
111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

لو اردنا تصميم اجراء مخزن لكي يقوم بطباعة درجة الطالب بعد تمرير رقم الطالب ورقم المقرر.

الاجراء المخزن سوف يكون كمايلي :

```
create or replace procedure stu_mark(
stu_id in studys.NO_STU%type,
cou in studys.COURSE_CODE%type)
as
mar studys.mark%type;
begin
select mark
into mar
from studys
where NO_STU=stu_id
and COURSE_CODE=cou;
DBMS_OUTPUT.PUT_LINE(mar);
end;
/
```

بعد الانتهاء من تنفيذ الاجراء يكون الاجراء مخزن في قاعدة البيانات ولكي نقوم باستدعاءه نقوم بمايلي

```
begin
stu_mark(111,'216CS');
end;
/
```

لاحظ كيف تم استدعاء الاجراء السابق من خلال اسم الاجراء وبذلك سوف يكون الناتج على الشاشة كمايلي 88 وهي صحيحة بعد تمرير رقم الطالب 111 ومقرر 216CS

لكن لاحظ اننا لم نستخدم متغيرات اخراج لكن مأرايك ان نصمم اجراء اخر يقوم بنفس الوظيفة التي يقوم بها الاجراء السابق لكن عملية الطباعة تكون بعد الاستدعاء لكي نجعل الاجراء يقوم بارجاع درجة الطالب بمتغير لذلك فان الاجراء كمايلي :

```
create or replace procedure stu_mark22(
stu_id in studys.NO_STU%type,
cou in studys.COURSE_CODE%type,
mara out studys.mark%type)
as
begin
select mark
into mara
from studys
where NO_STU=stu_id
and COURSE_CODE=cou;
end;
/
```

بعد ذلك نقوم باستدعاء الاجراء ومن ثم طباعة الدرجة لان لو لاحظت الاجراء لايقوم بالطباعة ولاحظ ايضا ان الدرجة تم وضعها في المتغير mara ولذلك سوف يعود بهذه القيمة وسوف يكون الاستدعاء كمايلي :

```
declare
m studys.mark%type;
begin
stu_mark1(111,'225CS',m);
DBMS_OUTPUT.PUT_Line(m);
end;
/
```

وسوف يكون الناتج هو ٩٠ وهذا صحيح بناء على الجدول.

الوظائف المخزنة

كان درسا السابق عن الاجراءات المخزنة واليوم لدينا درس مشابه له وهو الوظائف المخزنة لكن الفرق ان الوظائف لابد ان تعيد قيمة

والصسغة العامة لتكوين وظيفة كمايلي:

```
CREATE [OR REPLACE] FUNCTION function-name
 [(argument1 ... [, argumentN) ]
RETURN function-datatype IS
 [local-variable-declarations]
BEGIN
executable-section
 [exception-section]
RETURN function-value
END [function-name];
```

حيث أن

function_name اسم الوظيفة المستخدمه.

اما REPLACE OR فهي توضع حينما تعلم ان الاجراء موجود من السابق.

اما عن المغيرات التي بين القوسين فهي اما متغيرات مدخله مثل اذا كان لديك اجراء حساب معدل طالب وتريد تمرير رقم الطالب الذي تريد حساب معدله فهذه هي تعتبر كمدخلات ، وهي بنفس الطريقة التي تعاملنا بها مع الاجراءت المخزنة لاتغير على المتغيرات وطرق تعريفها.اما RETURN datatype فهي تدل على نوع القيمة المعادة من الوظيفة .

مثـــــال : في الجدول الذي قمنا بدراسته في الدرس الرابع وكان بأسم studys وكان كمايلي :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3
111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

لو اردنا تصميم وظيفة ترجع بمعدل الطالب الفصل اي يتم تمرير رقم الطالب الى الوظيفة ثم يتم حساب المعدل الفصلي للطالب

ويتم حساب المعدل الفصل للطالب كمايلي =مجموع النقاط ÷ مجموع عدد الساعات لمقررات

ولانشاء الوظيفة كمايلي :

```
1  create or replace function stu_avea(stnum in
2 studys.NO_STU%type)
3  return real
4  as
5  hour courses.hours%type;
6  avrage number(4,2);
7  sum_hours courses.hours%type:=0;
8  point studys.POINT%type;
9  total_Point studys.POINT%type:=0;
10 codem courses.CODE%type;
11 cursor sumpoint
12 is
13 select COURSE_CODE,POINT
14 from studys
15 where NO_STU=stnum;
16 begin
17 open sumpoint;
18 loop
19 fetch sumpoint into codem,point;
20 exit when sumpoint%notfound;
21 select hours
22 into hour
23 from courses
24 where code=codem;
25 total_Point:=total_Point+point;
26 sum_hours:=sum_hours+hour;
27 end loop;
28 close sumpoint;
29 avrage:=total_Point/sum_hours;
30 return avrage;
   end;
```

الشرح :

السطر رقم ١: لتعريف الوظيفة
السطر رقم ٢: نوع القيمة التي سوف ترجع بها الوظيفة
السطر رقم ٤: تعريف متغير عدد الساعات وهو نفس حقل عدد ساعات المقرر الموجودة في جدول courses

الان بعد الانتهاء من شرح طريقة تصميم الوظيفة جاء دور طريقة الاستدعاء :

لكن قبل الاستدعاء لنحسب يدويا معدل الطالب الذي رقمه ١١١ مثل لكي نقارنه بالنتائج بعد الاستعلام :

مجموع نقاط الطالب = ١٣.٥+١٤.٢٥= ٢٧.٧٥

مجموع عدد الساعات = (عدد ساعات المقرر CS٢١٦) + (عدد ساعات المقرر CS ٢٢٥)
= ٣ + ٣ = ٦

وبالتالي فان معدل الطالب = ٢٧.٧٥ ÷ ٦ = ٤.٦٣

لكن الان دعنا نستدعي الدالة ونشاهد النتائج

```
SELECT distinct(NO_STU),stu_avea(no_stu)
from studys
where no_stu=111;
```

لاحظ كيف تم استدعاء الدالة من خلال الاستعلام ولاحظ استخدام الدالة distinct وهي لعدم تكرار السجل واليك النتائج :

```
NO_STU          STU_AVEA(NO_STU)
------    ;     ----------------
111               4.63
```

لاحظ لو كان الاستعلام بدون وجود الدالة distinct فسوف يتكرر رقم الطالب عدد ظهوره في الجدول لذلك لوكان كمايلي :

```
SELECT NO_STU,stu_avea(no_stu)
from studys
where no_stu=111;
```

فان النتائج ستصبح هكذا

```
NO_STU                  STU_AVEA(NO_STU)
------      ;           ----------------
111                        4.63
111                        4.63
```

وهذا سبب ظهور الدالة distinct

الحزم البرمجية و الزنادات

أهداف الفصل

سنتناول في هذا الفصل ان شاء الله :

١- ماهية الحزم البرمجية
٢- كيفية انشاء الحزم البرمجية ومكوناتها
٣- ماهية الزنادات
٤- التعامل مع الزنادات
٥- قوائد الزنادات

الحزم البرمجية

تعلمنا سابقا كيفية انشاء الاجراءت والوظائف المخزنة. لكن مارأيك لو وجد لدينا قاعدة بيانات كبيرة جدا ولنضرب مثال انها تحتوي على ٥٠ اجراء او وظيفة وظيفة او اجراء لها عمل خاص ولنفرض ان هذه القاعدة هي لمحل تجاري ضخم يحتوي على بيانات العملاء وبيانات الموظفين وبيانات الاصناف التجارية وبيانات المخزون وغيرها من بيانات ، ولذلك فان بعض هذه الاجرائيات والوظائف المخزنة مختص بالعملاء مثلا وجود اجراء لحساب اجمالي عميل وغيرها من الاجرائيات ، ومثل وجود اجرئيات خاصة بالموظفين مثلا اجرائية خاصة بحساب راتب الموظف بعد حذف الحسومات واضافة العلاوات وغيرها ايضا ، لكن وضعها في هذا الشكل في قاعدة البيانات قد يسبب لك بعض الارباك لذلك مارأيك بان تجمع كل الوظائف والاجرائيات الخاصة بكل قسم في مجموعة لوحدها وهذه المجموعة تدعي الحزمه package مثلا نجمع كل اجرائيات والوظائف الخاصة بالعملاء في حزمة خاصة

فوائد استخدام الحزمة

١- تجميع وحدات pl/sql المرتبطة.
٢- اداء افضل.
٣- تكون السرية افضل.
٤- اهم شيء هو في عملية الصيانه حيث تسهل عملية الصيانة باستخدامم الحزم.

مكونات الحزم

تتكون الحزمة من جزءين الاول وهو الوصف specification ويحتوي على التعاريف مثل متغيرات او مؤشرات او اسماء الاجراءت ومتحولتها.
اما الجز الثاني فهو جسم الحزمة ويحتوي على تفاصيل الاجراءت والعمليات وغيرها
والصيغة العامة لانشاء الجزء الاول كمايلي :

```
CREATE OR REPLACE PACKAGE pack_name AS
.........
.........
.........
end;
```

والصيغة العامة لانشاء الجزء الثاني كمايلي :

```
CREATE OR REPLACE PACKAGE BODY pack_name AS
.........
الحزمة جسم
.........
end;
```

لكن يجب ان يكون اسم الحزمة في الجزء الاول هو نفس اسم الحزمة في الجزء الثاني.

مـثال :

لنقم بانشاء حزمة تحتوي على وظيفة لحساب معدل طالب واجراء لطباعة المعدل
ولذلك سوف نستخدم نفس الوظيفة التي انشناها في الدرس السادس والتي اسمها stu_avea
والتي تقوم بحساب معدل الطالب والان نبداء بانشاء الحزمة . الجزء الاول من الحزمة
specification كمايلي :

```
CREATE OR REPLACE PACKAGE student AS
function stu_avea(stnum in
studys.NO_STU%type)return real;
procedure print_ave(avrage in real);
end;
```

الان نقوم بانشاء جسم الحزمة والتي تحتوي على التفاصيل.كمايلي

```
CREATE OR REPLACE PACKAGE BODY student AS
function stu_avea(stnum in studys.NO_STU%type)
return real
as
hour courses.hours%type;
avrage number(4,2);
sum_hours courses.hours%type:=0;
point studys.POINT%type;
total_Point studys.POINT%type:=0;
codem courses.CODE%type;
cursor sumpoint
is
select COURSE_CODE,POINT
from studys
where NO_STU=stnum;
begin
open sumpoint;
loop
fetch sumpoint into codem,point;
exit when sumpoint%notfound;
select hours
into hour
from courses
where code=codem;
total_Point:=total_Point+point;
sum_hours:=sum_hours+hour;
end loop;
```

```
close sumpoint;
avrage:=total_Point/sum_hours;
return avrage;
end;

procedure print_ave(avrage in real)
as
begin
DBMS_OUTPUT.PUT_LINE(avrage);
end;
end;
```

ويحتوي جسم الحزمة كمانلاحظ على مكونات الوظيفة والاجراء الذي تم تعريفهما في وصف الحزمة حيث ان الوظيفة لحساب المعدل والاجراء لطباعة المعدل.

طريقة استدعاء اجراء او وظيفة موجود داخل حزمة :

تتم عملية الاستدعاء كمايلي :

pack_name.func_proc_name

اي اسم الحزمة اولا ثم نقطة ثم اسم الاجراء او الوظيفة مـــــثال :

```
set serveroutput on
declare
aa real;
begin
aa:=student.stu_avea(111);
student.print_ave(aa);
end;
/
```

وبعد التفيذ يكون الناتج هو معدل الطالب الذي رقمه ١١١ لاحظ اول شي استدعينا داله حساب المعدل ووضعناها في المتغير aa ثم استدعينا اجراء الطباعة ليتم طبعة على الشاشة.

والان و بعد ان تعرفت على فائدة الحزم مارأيك من الان فصاعد ان تستخدم الحزم في كتابة الاجرائيات والوظائف

الزنادات TRRIGERs

تتشابه الزنادات مع البرامج الفرعية الا في الطرق التالية :

* يتم تنفيذ الزنادات ضمنيا،عندما يعدل الجدول بالرغم من عمل المستخدم او التطبيقات على الجدول .

* يتم تعريف الزنادات للجدول الخاص بقاعدة البيانات

* لاتقبل الزنادات المعاملات

تعد الزنادات هامة جدا في تطوير نظم البيانات الموجهة الخاصة بالانتاج .

تركيب الزناد :

```
create [or replace] Trigger <TRRIGER_NAME>
<before|after] [instead of] trigger event on <table name>
[for Each row [whene triggering restriction]]
<trigger body>
```

كماهو مع الاجراءات المخزنة امكانية استخدام replace لكي تقوم بالتعديل على الزناد اذا كان موجود ولاتقوم بانشاءه من جديد.
ينفذ التوقيت الخاص بالزناد سواء نفذ الزناد قبل او بعد اغلاق الزناد بواسطة الخيارين before و after ، لكن خيار after اكثر كفاءة لان قطع البيانات المؤثرة يجب ان تقراء منطقيا مرة للزناد ومرة لعبارة trigger
ملاحظة/ ان حدث اطلاق الزناد هو جملة sql التي تجعل الزناد وحدث الاطلاق اما update او delete او insert او بكليهما..
ويوجد اربعة انواع من الزنادات:
١- صف after.
٢- جملة after.
٣- صف before.
٤- جملة before.
وكل زناد من اجل جملة update او insert او delete كل زناد يعد نوع واحد من (instead of , after,before) ويمكن تعريف تسع زنادات للجدول الواحد..

* معالجة احداث اطلاق الزناد :
او على توليفة من هذه delete او update او insertيحتوي حدث اطلاق الزناد على عملية العمليات عندما يتعامل زناد واحد مع اكثر من عملية واحدة، فيمكنك ان تستخدم دعائم شرطية للتعرف على نوع العبارة التي تستخدم لتنفيذ الجزء الخاص بالرمز في الزناد
والدعائم هي كمايلي:

```
IF inserting then .....end if;
IF updating then .....end if;
IF deleting then .....end if;
```

قيد الزناد

يحدد هذا القيد تعبير منطقي يجب ان يكون صحيح كي يطلق الزناد.
على سبيل المثال الزناد التالي stduent_trigger لا يتم حدوثة الا اذا كان رقم الطالب student_id اقل من ١٠٠

```
create or replace trigger stduent_trigger
before insert or update on student
for each row
when(new.student_id<100)
```

قيود على انشاء الزنادات:

١- يمكن للنص ان يحتوي على جمل dml sql لكن جمل select يجب ان تكون جمل select into
٢- لايسمح بجمل التحكم (commit,savepoint,rollback)
٣-لايمكن لبرنامج فرعي مخزن ان يتضمن جمل التحكم السابقة اذا تم استدعائه بواسطة الزناد.

مـــــــــــثال :

لنفرض انه لدينا الثلاث جداول التالية:

الاول : هو جدول player بيانات جميع اللاعبين في النادي سواء درجة شباب او درجة ممتاز :

no_player	name	date_birth	phone	address	levels
1	talal	11/11/1973	123456	riyadh1	1
2	mohammed	1/1/1982	654321	riyadh2	2
3	sami	1/1/1988	123789	riyadh3	2
4	yosif	12/3/1970	123123	riyadh4	1

حيث level تمثل الدرجة التي يلعب بها اللاعب حيث ١ تمثل الدرجة الاولى الممتاز - و ٢ تمثل الشباب .

ولانشاء الجدول كمايلي :

```
create table player(
no_player varchar2(6) primary key,
name varchar2(50),
date_birth date,
phone varchar2(9),
address varchar2(20),
levels number(2));
```

الثاني : هو جدول اللاعبين في درجة الممتاز وهو خاص بالرواتب واسم الجدول larg_player

no_player	level_no	salary
1	1	
4	1	

ولانشاء الجدول كمايلي :

```
create table larg_player(
no_player varchar2(6) primary key,
level_no number(2),
salary number(7,2));
```

الثالث : هو جدول اللاعبين في درجة الشباب وهو خاص بالرواتب واسم الجدول youth

no_player	level_no	salary
2	2	
3	2	

ولانشاء الجدول كمايلي :

```
create table youth(
no_player varchar2(6) primary key,
level_no number(2),
salary number(7,2));
```

الان نريد عمل زناد بحيث حينما يقوم المستخدم بادخال اسم لاعب جديد وتحديد مستواه (شباب او ممتاز) يقوم الزناد باختبار المستوى فاذا كان شباب اضاف رقم اللاعب في جدول الشباب وكذلك لوكان مستواه درجة اولى الممتاز فانه يضيف رقم اللاعب في جدول larg_player وبذلك يكون الزناد كمايلي :

```
create or replace trigger player_age
before insert on player
for each row
begin
if inserting then
if :new.levels=1 then
insert into larg_player(no_player,level_no) values
(:new.no_player,:new.levels);
elsif :new.levels=2 then
insert into larg_player(no_player,level_no) values
(:new.no_player,:new.levels);
end if;
end if;
end;
```

بعد ذلك قم بادخال مايلي :

```
insert into player values('1','talal','11/11/1973','123456','riyadh1',1);
```

لاحظ ان المدخلات تمت على جدول player بعد ذلك اذهب وقما بالاستعلام في جدول larg_player سوف تجد انه اضاف رقم اللاعب هناك.

التجميعات

انواع التجميعات في قاعدة البيانات:

التجميعة هي مجموعة من العناصر من نفس النوع

والتجميعات هي على نوعين:

- التجميعة varray

وهي كمصفوفة متغيرة ومشابهة للمصفوفات في اي لغة من لغات البرمجة مثل c و++c وتتم الاشارة الى اي عنصر في هذه التجميعة باستخدام الارقام السفلية ويتم التخزين في هذه التجميعة بصورة خطية inline

- التجميعة nested table

تعتبر كجدول موجود في قاعة البيانات والاشارة الى اي عنصر في هذه التجميعة ايضا باستخدام الارقام السفلية ويتم تخزين البيانات في جدول تخزين منفصل.

--اولا :التعامل مع التجميعات في sql plus

أ -التجميعة varray من النوع البسيط

مثال/ نفرض انك تريد انشاء جدول الاقسام في مستشفى والذي سوف يحتوي على رقم القسم ، اسم القسم ، واسم القسم ، ومن ثم اسماء موظفين القسم .مع العلم ان اسماء الموظفين سوف تكون في تجميعةvarray .

نقوم اولا بانشاء التجميعة كمايلي:

```
Create type namev as varray(30) of varchar2(50);

/
```

حيث لو فرضنا ان اكبر عدد للموظفين هو 30 واكبر طول للاسم هو 50 .

ثم نقوم بأنشاء الجدول ونقوم بانشاءه كمايلي/

```
Create table deptv
(nodept number(5) primary key,
namedept varchar2(50),
emp namev);
```

ونقوم فيمايلي بتنفيذبعض اوامر sql على الجدول

١- الادراج insert :

Insert into deptv values(10,'medical',namev('ali','sami','fahad','fady'));

٢- التحديث update :

لتحديث العناصر في التجميعة يتطلب استخدام pl/sql ولايمكن تنفيذ ذلك من خلال sql القياسية مثال:

```
Declare
Editname namev;
I   number:=1;
Begin
Select emp into editname
From deptv where nodept=10;
Loop
If (i=editname.count+1) then
Exit;
Elsif (editname(i)='sami') then
Editname(i):='mohammed';
End if;
i:=i+1;
end loop;
update deptv set emp=editname where nodept=10;
end;
```

شرح المثال السابق:

سوف يقوم بتغير اسم الموظف samiالذي قمنا بادخال وتبديلة الى mohammed وشرح الخطوات كمايلي

اولا قمنا بتعريف متغير editname من نفس نوع التجميعة namev وذلك لكي نقوم بتخزين المؤشر والذي يحتوي على اسماء الموظفين فيه ونعرف ايضا متغير I وهو من يستخدم كعداد.

ثم نقوم بعمل مؤشر لاستخراج اسماء الموظفين وهي كمايلي

Select emp into editname

From deptv where nodept=10;

يقوم هنا باستخراج اسماء الموظفين للقسم ١٠ وتخزين ناتج الاستعلام في المتغير editname والذي هو من نفس نوع التجميعة

ثم يبدأ حلقة ومن ثم يختبر هل I وصلت الى نهاية التجميعة اذا كان نعم قام بانهاء الاجراء واذا لم يصل الى نهاية التجميعة يختبر عنصر التجميعة الحالي هل هو يساوي sami ام لا اذا كان يساوي samiيقوم بتغير هذه القيمة الى mohammed ومن ثم يزيد العداد بواحد ومن ثم يعود من جديد الى ان يصل الى نهاية التجميعة وبعد الانتهاء من جميع العناصر يقوم بعمل التحديث للجدول

update deptv set emp=editname where nodept=10;

ومن ثم يقوم بانهاء الاجراء.

٢- الحذف delete(Trim) :

حذف عنصر من التجميعة يتطلب عمل اجراء pl/sql

والحذف في التجميعات varray يتم على اخر عنصر في التجميعة اي لو حذفنا عنصر واحد فانه يتم على اخر عنصر ولا يمكن تحديد العنصر

مثال:

```
Declare
Namedel namev;
Begin
Select emp into namedel
From deptv where nodept=10;
Namedel.trim(1);
update deptv set emp=namedel where nodept=10;
end;
/
```

٤- التحديث بالاضافة update(append)

هذا الامر يستخدم للاضافة

ألم يتبادر الى ذهنك كيف نضيف مزيدا من الموظفين الى القسم ١٠ يمكن ان تقول نستخدم الامر insert لكن هذا غير صحيح لاننا عندما نستخدم الامر insert وندخل رقم القسم ١٠ يظهر لنا خطاء لان حقل رقم القسم مفتاح رئسي

لذلك اذا اردنا اضافة المزيد من الموظفين اولا نقوم بعمل توسع extend للتجميعة لكي تسمح لنا باضافة عنصر جديد .ونستخدم الاجراء لذلك

ويكون الاجراء كمايلي:

```
Declare
Newname namev;
Begin
Select emp into newname
From deptv where nodept=10;
Newname.extend;
Newname(newname.last):='khaled';
update deptv set emp=newname where nodept=10;
end;
/
```

ب - التجميعة varray من النوع الشيء:

هذه التجميعة هي تجميعة معرفة بواسطة المستخدم

مثال ذلك :

لو اردنا انشاء جدول يحتوي على مسمى الوظيفة وفي حقل اخر نكّون تجميعة تحتوي على اسم الموظف وراتبة لجميع موظفين هذه الوظيفة

JOB_NAME	EMPLOYEE
manager	(ali,5000),(sami,6000),(fahad,4000)
Analysis	(laui,7500),(mohammed,7500)
Programming	(fady,8000),(saed,6000)

وبالتالي فإن الخطوة الاولى هي انشاء object كمايلي

```
Create type empobj

as object (nameemp varchar2(50),salary number(6));

/
```

ثم نقوم بانشاء التجميعة

```
Create type employeeobj as varray(20) of empobj;

/
```

لاحظ الفرق اننا استخدمنا object الذي انشئناه ولذلك سمي هذا النوع بهذا الاسم

وبعد انشاء التجميعة نقوم بإنشاء الجدول كمايلي:

```
Create table jobobj
(job_name varchar2(50),
```

```
employee employeeobj);
```

وبعد انشاء الجدول سوف نتعرف الان على كيفية التعامل مع هذا الجدول من خلال sql * plus

١ـ الاضافة :

لاضافة صف إلى الجدول السابق نقوم بمايلي :

```
Insert into jobobj values
('manager',employeeobj (
                empobj('ali',5000),
                empobj('sami',6000),
                empobj('fahad',4000)));
```

وبهذا نكون قد اضفنا الصف الاول في الجدول السابق

ولو ذهبنا الى sql * plus وطلبنا منه مايلي

```
Select * from jobobj
```

```
JOB_NAME
--------------------------------------------------
EMPLOYEEobj(NAMEEMP, SALARY)
-----------------------------------------------------------
manager
EMPLOYEE1(EMPOBJ('ali', 5000), EMPOBJ('sami', 6000),
EMPOBJ('fahad', 4000))
```

ونلاحظ لقد تم اضافة الصف الجديد

٢ـ التحديث :

يجب تطبيق قطعة pl/sql كمايلي:

```
                                    Declare
                            Editname employeeobj;
                                  Editobj empobj;
                                     i number:=1;
                                          Begin
                          Select employee into editname
                  From jobobj where job_name='manager';
                                         Loop
                                Editobj:=editname(i);
                           If (i=editname.count) then
                                        Exit ;
                       Elsif editobj.nameemp='sami' then
                                Editobj.salary:=10000;
                                Editname(i):=editobj;
                                        End if;
                                        i:=i+1;
                                    End loop;
                    Update jobobj set employee=editname
                            Where job_name='manager';
                                         End;
```

وبهذا يتم تعديل راتب الموظف الذي اسمه sami الموجود في قسم manager وجعل راتبه ١٠٠٠٠

٣- الحذف:

```
                                    Declare
                                Editemp employeeobj;
                                      begin
          select Select employee into editemp
               From jobobj where job_name='manager';
                                  Editemp.trim(1);
                     Update jobobj set employee=editemp
                            Where job_name='manager';
```

وبهذا يتم حذف سجل واحد وهو اخر صف من التجميعة

ج- التجميعة NESTED TABLE من النوع البسيط:

تعتبر هذه التجميعة نفس التجميعة varray من النوع البسيط والتي نقوم بتعريفها بواسطة احد انواع البيانات الموجودة مثل number و varchar2 وسوف نستخدم هنا نفس المثال الذي استخدمناه في التجميعة varray من النوع الشيئ .

مثال/ نفرض انك تريد انشاء جدول الاقسام في مستشفى والذي سوف يحتوي على رقم القسم ، اسم القسم ، واسم القسم ، ومن ثم اسماء موظفين القسم .مع العلم ان اسماء الموظفين سوف تكون في تجميعةnested table

اولا نقوم بانشاء التجميعة وتكون كمايلي:

```
Create type namenested as table of varchar2(50);
/
```

ثم نقوم بأنشاء الجدول كمايلي:

```
Create table deptnested
(nodept number(5) primary key,
namedept varchar2(50),
emp namenested)
nested table emp store as nestedtablesimple;
```

لاحظ السطر الاخير الذي تم اضافنته

فيجب دائم وضعه اذا استخدمنا nested table مع تغير المكتوب باللون الاحمر والذي يمثل اسم الحقل الذي هو من النوع nested table.

والذي تحته خط يعتبر كأسم لهذا nested table

****التعامل مع الجدول السابق من خلال اوامر sql**

١- الاضافة :

```
Insert into deptnested
values(1,'medical',namenested('ali','sami','fahad','fady'));
```

٢- التحديث update :

لتحديث العناصر في التجميعة يتطلب استخدام pl/sql ولايمكن تنفيذ ذلك من خلال sql القياسية مثال:

```
                                    Declare
                                Editname namenested;
                                      I   number:=1;
                                           Begin
                                  Select emp into editname
                              From deptnested where nodept=1;
                                            Loop
                                 If (i=editname.count+1) then
                                           Exit;
                              Elsif (editname(i)='sami') then
                                   Editname(i):='mohammed';
                                          End if;
                                          i:=i+1;
                                       end loop;
                    update deptnested set emp=editname where
                                       nodept=1;
                                           end;
                                               /
```

شرح المثال السابق:

سوف يقوم بتغير اسم الموظف samiالذي قمنا بادخال وتبديلة الى mohammed وشرح الخطوات كمايلي

اولا قمنا بتعريف متغير editname من نفس نوع التجميعة namenested وذلك لكي نقوم بتخزين المؤشر والذي يحتوي على اسماء الموظفين فيه ونعرف ايضا متغير I وهو يستخدم كعداد.

ثم نقوم بعمل مؤشر لاستخراج اسماء الموظفين وهي كمايلي

Select emp into editname

From deptnested where nodept=1;

يقوم هنا باستخراج اسماء الموظفين للقسم ١ وتخزين ناتج الاستعلام في المتغير editname والذي هو من نفس نوع التجميعة

ثم يبدأ حلقة ومن ثم يختبر هل I وصلت الى نهاية التجميعة اذا كان نعم قام بانهاء الاجراء واذا لم يصل الى نهاية التجميعة يختبر عنصر التجميعة الحالي هل هو يساوي sami ام لا اذا كان يساوي samiيقوم بتغير هذه القيمة الى mohammed ومن ثم يزيد العداد بواحد ومن ثم يعود من جديد الى ان يصل الى نهاية التجميعة وبعد الانتهاء من جميع العناصر يقوم بعمل التحديث للجدول

update deptnested set emp=editname where nodept=1;

ومن ثم يقوم بانهاء الاجراء.

٣- التحديث بالاضافة update(append) :

اذا اردنا اضافة المزيد من الموظفين اولا نقوم بعمل توسع extend للتجميعة لكي تسمح لنا باضافة عنصر جديد .ونستخدم الاجراء لذلك

ويكون الاجراء كمايلي:

```
                                Declare
                          Newname namenested;
                                  Begin
                        Select emp into newname
                   From deptnested where nodept=1;
                                Newname.extend;
                  Newname(newname.last):='khaled';
update deptnested set emp=newname where nodept=1;
                                    end;
                                       /
```

٤-الحذف :

هناك طريقتان الاولى باستخدام trim وهذه الطريقة تحذف اخر صف في التجميعة ولايمكن اختيار اي عنصر في التجميعة:

```
                                Declare
                          Namedel namenested;
```

```
                                        Begin
                                Select emp into namedel
                        From deptnested where nodept=1;
                                        Namedel.trim(1);
update deptnested set emp=namedel where nodept=1;
                                                end;
                                                   /
```

اما الطريقة الثانية فهي باستخدام الامر delete(m) وهذه الطريقة لا تحذف من الاخير بل يتم اختيار اي العناصر الذي تريد حذف فلو وضعنا m=2 وكانت المدخلات الجدول كما فعلا سابقا سيتم حذف العنصر الثاني والذي هو الموظف sami

```
                                        Declare
                                Namedel namenested;
                                        Begin
                                Select emp into namedel
                        From deptnested where nodept=1;
                                        Namedel.delete(2);
update deptnested set emp=namedel where nodept=1;
                                                end;
                                                   /
```

وهذه الطريقة(طريقة delete) هي من ابرز الفوارق بين varray و nested table حيث varray لا يمكنها حذف اي عنصر ولكن تحذف فقط العنصر الاخير بعكس nested table

د- التجميعة NESTED TABLE من النوع الشيئ:

كما لاحظنا ان التجميعة varray يتم انشاءها من النوع الشيء فإن NESTED TABLE يتم إنشاءه ايضا من النوع الشيء المعرف بواسطة المستخدم.

سوف نستخدم لشرح هذا المثال نفس المثال الذي استخدمناه في varray من النوع object

مثال ذلك :

لو اردنا انشاء جدول يحتوي على مسمى الوظيفة وفي حقل اخر نكّون تجميعة تحتوي على اسماء الموظف وراتبة لجميع موظفين هذه الوظيفة

اول خطوة هي انشاء الشيء empobj والذي تم انشاءه سابق عندما قمنا بشرح vaaray من النوع الشيء واذا كنت لم تقم بانشاءه فهذا الكود :

```
Create type empobj
as object (nameemp varchar2(50),salary
number(6));
/
```

ثاني خطوة هي إنشاء التجميعة nested table باستخدام الشيء empobj كمايلي

```
Create type empnestedobj as table of empobj;
/
```

ثالث خطوة نقوم بإنشاء الجدول كمايلي:

```
Create table jobnested
(job_name varchar2(50),
employee empnestedobj)
nested table employee store as nestedtablesimple;
```

* التعامل مع الجدول السابق من خلال اوامر sql

١- الاضافة :

لكي تقوم بادراج بيانات في الجدول السابق نقوم بمايلي:

```
Insert into jobnested
Values('manager',
             empnestedobj(empobj('ali',6000),
empobj('sami',7000),
     empobj('fahad',6500)));
```

١- التحديث :

يتم التحديث باستخدام pl/sql كمايلي:

مثلا لتغير راتب الموظف ali الموظف الاداري اي في مسمى وظيفتها manager الى ٩٠٠٠

```
Declare
Editsal empnestedobj;
Editempobj empobj;
i number:=1;
Begin
Select employee into editsal
From jobnested where job_name='manager';
Loop
Editempobj:=editsal(i);
If (i=editsal.count) then
Exit;
Elsif (editempobj.nameemp='fahad') then
Editempobj.salary:=9900;
Editsal(i):=editempobj;
End if;
i:=i+1;
end loop;
update jobnested set employee=editsal where
job_name='manager';
end;
/
```

٣- التحديث بالاضافة update(append) :-

```
Declare
Editemp empnestedobj;
Begin
Select employee into editemp
From jobnested where job_name='manager';
Editemp.extend;
Editemp(editemp.last):=empobj('mohammed',7000);
update jobnested set employee=editemp where
job_name='manager';
end;
/
```

٤- الحذف delete(trim) :

هناك طريقتان الاولى باستخدام trim وهذه الطريقة تحذف اخر صف في التجميعة ولايمكن اختيار اي عنصر في التجميعة:

```
declare
empdel empnestedobj;
begin
Select employee into empdel
From jobnested where job_name='manager';
Empdel.trim(1);
update jobnested set employee=empdel where
job_name='manager';
end;
/
```

اما الطريقة الثانية فهي باستخدام الامر delete(m) وهذه الطريقة لا تحذف من الاخير بل يتم اختيار اي العناصر الذي تريد حذف فلو وضعنا m=2 وكانت المدخلات الجدول كما فعلا سابقا سيتم حذف العنصر الثاني وهو الموظف sami وفيمايلي الكود

الطريقة الثانية

```
declare
empdel empnestedobj;
begin
Select employee into empdel
```

```
From jobnested where job_name='manager';
Empdel.delete(2);
update jobnested set employee=empdel where
job_name='manager';
end;
/
```

عبارة الشرط IF -THEN

تستخدم هذه العبارة مثل اي العبارت الشرطية في لغة سي او سي++ او فيجوال بيسك وغيرها ، ولها استخدمات عديدة وسوف نعرف كيف نستخدمها مقدما مع حقول قواعد البيانات وذلك بعد اخذ المؤشرات

الصيغة العامة لها كمايلي:

```
IF condition THEN
   statement; ...  statement;
 [ELSIF condition THEN
statement; ...  statement;]
...
 [ELSIF condition THEN
statement; ...  statement;]
 [ELSE
statement; ...  statement;]
END IF;
```

حيث أن

conditional هي الشرط

ملاحظة
• ELSEو ELSIF اختيارية • جملة IF نحنوي على عدة عبارات ELSIF ولكن تحتوي على عبارة ELSE واحدة • انتبه الى أن طريقة كتابة ELSIF وليست ELSEIF

مثال على ذلك :

```
Declare
i   number(5);
```

```
BEGIN
i:=5;
IF i=5 then
DBMS_OUTPUT.PUT_LINE('i = ' || i);
ELSE
DBMS_OUTPUT.PUT_LINE('i not eqal 5 ');
    END IF;
END;
```

الشرط باستخدام اكثر من شرط

```
Declare
i   number(5);
BEGIN
i:=5;
IF i>1 then
DBMS_OUTPUT.PUT_LINE(i  || ' > 1');
ELSIF i<1 then
DBMS_OUTPUT.PUT_LINE(i  || ' < 1');
ELSIF i=1 then
DBMS_OUTPUT.PUT_LINE(i  || ' = 1');
   END IF;
END;
```

الـــتكرار

يوجد عدة اوامر للتكرار وهي:

١- loop-exit-end

وهنا لابد من وضع شرط لأنهاء الحلقة
نأخذ مثال بسيط على ذلك

```
Declare
i   number(5);
BEGIN
i:=1;
LOOP
IF i>10 then
EXIT;
END IF;
DBMS_OUTPUT.PUT_LINE('i =' || i);
i:=i+1;
End loop;
END;
/
```

الشرح

السطر الثاني : تعريف متغير من نوع رقم
السطر الثالث :البداية
السطر الرابع : اعطاء المتغير قيمة ابتدائية وهي i=1
السطر الخامس : شرط الانهاء
السطر السادس : الانهاء اذا كان i>10 ولا يكمل
السطر السابع :انها if
السطر الثامن : طباعة i
السطر التاسع :زيادة قيمة i بواحد
السطر العاشر : نهاية الحلقة

وبعد كتابة هذا الكود يكون الناتج كمايلي:

i =1
i =2
i= 3
i =4
i =5
i =6
i =7
i =8
i =9
i =10

٢- EXIT - END LOOP - WHEN

```
Declare
i   number(5);
BEGIN
i:=1;
LOOP
EXIT WHEN i>10;
DBMS_OUTPUT.PUT_LINE('i =' || i);
i:=i+1;
End loop;
END;
/
```

ويكون الناتج نفس السابق لكن لاحظ استخدم شرط الانهاء

EXIT WHEN i>10;

٣- END - LOOP - WHILE

```
Declare
i   number(5);
BEGIN
i:=1;
WHILE i <= 10 LOOP
DBMS_OUTPUT.PUT_LINE('i =' || i);
```

```
i:=i+1;
End loop;
END;
/
```

ويكون الناتج نفس المثال السابق

٤- FOR - IN - LOOP - END

وهذه ايضا طريقة اخرى لاستخدام حلقات التكرار وهي نفس عمل اسلوب حلقات for في اي لغة برمجة

والصيغة العامه لها هي على الاتي

```
FOR  i  IN  البداية..النهاية  LOOP
المراد تكرارها الجمل
END LOOP
```

مثـــال:

```
Declare
i   number(5);
BEGIN
FOR  i   IN  1..10 LOOP
DBMS_OUTPUT.PUT_LINE('i =' || i);
End loop;
END;
/
```

وسوف يكون الناتج كمايلي(نفس السابق):

١= i
i =2
i =3
i =4
i =5
i =6
i =7
i =8
i =9
١٠= i

جملة GOTO

كما في باقي لغات البرمجة فان GOTO تنقل تسلسل عمل البرامج من نقطه الى اخرى

الصيغة العامة

GOTO <<my_label>>

حيث أن

my_label عنوان النقطة التي سيتم الانتقال اليها

مثال

```
declare
i  positive := 1;
max_loops constant positive := 100;
begin
i := 1;
loop
i := i + 1;
if i > max_loops then
goto more_processing;
end if;
end loop;
<<more_processing>>
i := 1;
end;
  /
```

التعليقات

يمكن كتابة التعليقات في PL/SQL بعدة طرق

٤- كتابة (--) ثم التعليق

مثال

```
-- DON'T FORGET  MY NAME
```

٥- كتابة (***) ثم التعليق ثم (***) مرة أخرى

مثال
```
*** DON'T FORGET  MY NAME ***
```

٦- كتابة (*/) ثن التعليق ثم (/*) مرة اخرى

مثال
```
/* DON'T FORGET  MY NAME */
```

استخدام أوامر SQL في كتلPL/SQL

يمكنك استعمال جمل SQL داخل الكتل البرمجية الخاصة بـ PL/SQL ولكن مع وجود اختلافات حيث كل جملة SQL تنتهي بفاصلة منقوطة (;)

مثال

```
DECLARE
max_records CONSTANT int := 100;
i           int := 1;
BEGIN
FOR i IN 1..max_records LOOP
  if (mod(i,10) = 0) then
    INSERT INTO test_table
           (record_number, current_date)
    VALUES
           (i, SYSDATE);
  else
    NULL;
  end if;
END LOOP;
COMMIT;
END;
/
```

في هذا المثال قمنا باستخدام الامرINSERT وsysdate وهي من أوامر SQL

المؤشرات

اهداف الفصل

قي مهاية هذا الفصل إن شاء الله ستكون قادر على

٣- التعامل مع المؤشرات الصريحه

٤- النعامل مع المؤشرات الضمنية

المؤشرات CURSORS

تستخدم pl/sql المؤشرات cursors لأدارة عبارات التحديد select في لغة sql وكما لاحظنا الاوامر السابقة مثل if والتكرار لم نستخدمها مع بيانات الجداول المخزنه ولعمل ذلك لابد من استخدام هذه المؤشرات.
وهناك نوعين من المؤشرات هي الضمنية والصريحة وسوف نتطرق لك واحد بالتفصيل والامثلة اللازمة.

١- المؤشرات الصريحه :

يتم تعريف هذا النوع من المؤشرات كجزء من الاعلان declare ويجب ان تشتمل عبارة sql المعرفه على عبارة التحديد select فقط حيث لايمكن استخدام الكلمات الاساسية insert,update,delete
وعند استخدام المؤشرات الصريحه دائما ماستكتب اربعة مكونات كمايلي:
١- يتم تعريف المؤشر في الجزء declare
٢- يتم فتح المؤشر بعد عبارة begin

الصيغة العامة لتعريف المؤشر الصريح كمايلي:

```
DECLARE
      IS   اسم المؤشر   CURSOR
الاستعلام
```

تقوم باستبدال اسم المؤشر باسم مؤشر حقيقي
وتقوم بوضع جملة الاستعلام select في مكان الاستعلام

ولكي تقوم بفتح هذا المؤشر وتستخدمه نقوم بفتحه باستخدام الامر open كمايلي:

OPEN اسم المؤشر

وبعد فتح المؤشر تقوم باسترجاع او تحميل البيانات سطر(سجل) واحد من المؤشر الذي تم تعريفه باستخدام الامر FETCH كمايلي:

FETCH اسم المؤشر INTO متغير١،متغير٢،.....

ومعنى هذا اي قم باسترجاع البيانات من المؤشر المعطى اسمه وحملها into الى المتغيرات كع ملاحظة ان عدد المتغيرات يساوي عدد الحقول الموجودة في استعلام المؤشر.
وبعد الانتهاء من اجراء العمليات على المؤشر يجب عليك اغلاقه ويتم اغلاقه كمايلي:

```
close cursor_name
```

مثال على طريقة تعريف مؤشر :

افرض انه لدينا هذا الجدول

age	name	no
23	mohammed	111
22	talal	222
24	majed	333

اولا قم بانشاء هذا الجدول كمايلي:

```
create table stud(
     no number(4),
name varchar2(40),
  age number(2));
```

ثانيا قم بادخال البيانات السابقة في هذا الجدول كمايلي:

```
insert into stud values(111,'mohammed',23);
insert into stud values(222,'talal',22);
insert into stud values(333,'majed',24);
```

ثم قم بتنفيذ مايلي

```
set serveroutput on;
DECLARE
name_stu varchar2(40);
CURSOR  name_student IS
select name from stud
where no=111;
BEGIN
OPEN name_student;
FETCH name_student INTO name_stu;
DBMS_OUTPUT.PUT_LINE(name_stu);
CLOSE name_student;
END;
/
```

بعد التنفيذ سوف يظهر لك الناتج كمايلي:

mohammed

وهذا الشي صحيح

لاحظ اننا اتبعنا نفس الخطوات التي ذكرناه لكي نتعامل مع مؤشر صريح

لاحظنا في المثال السابق ان الاستعلام في cursor سوف يعود بسجل واحد لكن ماذا يحدث لو اعاد المؤشر اكثر من سجل واردنا المرور على كافة السجلات ؟

لحل السؤال السابق لابد من استخدام حلقة بها شرط وهذا هو هل سجلات المؤشر انتهت ام لا ونعرف ذلك من خلال خاصية found للمؤشر كمايلي:

found%mycur

حيث :

mycur : هي اسم المؤشر.

% : توضح انا مايلي اسم المؤشر هي احد خصائصه.

found : خاصية التي من خلالها نعرف هل تم الانتهاء من جميع السجلات ام لا

مثال :

اسم الطالب	كود المقرر	الدرجة	النتيجة
NO_STU	SUBJECT	MARK	RESULT
111	216CS	88	
222	225CS	75	
333	225CS	40	

نريد انشاء اجراء يقوم بالمرور على الجدول وينظر الى درجة الطالب اذا كان ناجح في المقرر ام لا فاذا كان mark اكبر او يساوي ٥٠ ضع قيمة true في حقل result والا ضع قيمة false في حقل result

نقوم اولا بانشاء هذا الجدول :

```
create table stu_study(
NO_STU number(4),
SUBJECT varchar2(8),
MARK number(3),
RESULT varchar2(20));
```

المدخلات السابقة وبعد انشاء الجدول نقوم بادخال

```
insert into stu_study (NO_STU,SUBJECT,MARK)
values (111,'216CS',88);

insert into stu_study (NO_STU,SUBJECT,MARK)
values (222,'225CS',75);

insert into stu_study (NO_STU,SUBJECT,MARK)
values (333,'225CS',40);
```

بعد ذلك نقوم بانشاء الاجراء:

```
declare
mar number(3);
no number(3);
cursor res_stu is
select no_stu,mark
from stu_study;
begin
open res_stu;
loop
fetch res_stu into no,mar;
exit when res_stu%notfound;
if mar>=50 then
update stu_study set result='TRUE' where
no_stu=no;
else
update stu_study set result='FALSE' where
no_stu=no;
end if;
end loop;
close res_stu;
end;
/
```

وبهذا تكون النتائج في الجدول كمايلي :

NO_STU	SUBJECT	MARK	RESULT
111	216CS	88	TRUE
222	225CS	75	TRUE
333	225CS	40	FALSE

هناك طريقة اخرى لتعريف المتغيرات لاحظ في الجدول السابق ان الحقل no_Stu تم تعريفه على انه من نوع number وتم تعريف المتغير no في الاجراء على انه number ايضا لكي يتم وضع رقم الطالب فيه لكن لاحظ لو تم تغير نوع الحقل في الجدول من number الى varchar2 فانه يجب عليك تغير نوع المتغير no في الاجراء ايضا لكن هناك طريقه تجعلك لاتعدل الاجراء كل مرة وهي استخدام الامر التالي لتعريف المتغير no في الاجراء

type%no_stu.stu_study NO

حيث :

NO هي اسم المتغير

stu_study : اسم الجدول

no_stu : الحقل المطلوب في الجدول

type% : خاصية نوع الحقل

ومعنى ماسبق قم بتعريف متغير اسمه no له نفس نوعية الحقل الذي اسمه NO_STU الموجود في الجدول . stu_study

وبهذا لان تقوم بتغير نوع العنصر في الاجراء في كل مرة تغير النوع وهكذا مع جميع المتغيرات التي لها صله بالجداول

وبذلك يصبح الاجراء بعد التعديل كمايلي:

```
declare
mar stu_study.mark%type;
no stu_study.no_stu%type;
cursor res_stu is
select no_stu,mark
from stu_study;
begin
open res_stu;
loop
fetch res_stu into no,mar;
exit when res_stu%notfound;
if mar>=50 then
update stu_study set result='TRUE' where
no_stu=no;
else
update stu_study set result='FALSE' where
no_stu=no;
end if;
```

```
end loop;
close res_stu;
end;
/
```

المؤشرات الضمنية

تعرفنا سابقا على فائد المؤشرات cursors ودرسنا النوع الاول منها وفي هذا الفصل عندنا نوع اخر وهو المؤشرات الضمنية وهي اسهل من المؤشرات الصريحه

وتوجد نقطتين هامتين عند التعامل مع المؤشرات الضمنيه :

* يظهر المؤشر الضمني في جسم الاجراء body وليس في declare الخاص بالاجراء كمافي المؤشرات الصريحه

* لابد ان يسترجع مؤشر select الضمني سطر واحد.

والصيغة العامه للمؤشر الضمني كمايلي:

```
SELECT COLUM1,COLUM2,.......  INTO
VARIABLE1,VARIABLE2,.....

FROM  table_name
```

ومعنى هذا قم باختيار الحقل١ و الحقل٢ وضعها في المتغيرات منغير١ و متغير٢ من الجدول table_name

سوف نأخذ مثال على ذلك وسوف نستخدم الجدول الذي انشئناه سابق في الدرس الثاني عندما تعاملنا مع المؤشرات الصريحة وكان اسم الجدول stud

no	name	age
111	mohammed	23
222	talal	22
333	majed	24

واردنا مثلا كتابة اجراء يقوم بحساب متوسط اعمار الطلاب (قد يقول البعض انه لايحتاج ذلك الى اجراء فمجرد استخدام جملة select نستطيع عمل ذلك انا اقول نعم هذا صحيح لكن احب استخدام الاجراء في هذا المثال لكي نرى طريقة عمل المؤشر الضمني ولكن سوف نرى بعد قليل مثال شامل يتم فيه استخدام الموشرات الصريحه والضمنية في نفس الوقت) والان نقوم بكتابة الاجراء كمايلي :

```
set serveroutput on;
declare
aveage number(4,2);
begin
select avg(age)
into aveage
from stud;
DBMS_OUTPUT.PUT_LINE(aveage);
end;
/
```

*** مثال شامل لاستخدام الموشرات الصريحة والضمنيه في نفس الوقت :

لنفرض انه لدينا الجدولين التاليين : الجدول الاول اسمه courses (المقررات):

رقم المقرر	اسم المقرر	عدد الساعات
code	course_name	hours
216CS	NETWORK	3
225CS	ASSEMBLY	3
325CS	DATABASE	4

وقم بانشاء الجدول كمايلي:

```
create table courses(
code varchar2(8),
course_name varchar2(40),
hours number(3),
primary key(code));
```

وقم بادخال البيانات الموجود بالجدول كمايلي:

```
insert into courses values('216CS','NETWORK',3);

 insert into courses
values('225CS','ASSEMBLY',3);
```

```
 insert into courses
values('325CS','DATABASE',4);
```

ثم نقوم بتكوين الجدول الثاني وهو studys :

اسم الطالب	كود المقرر	الدرجة	عدد النقاط
NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	
222	225CS	75	
333	225CS	40	
111	225CS	90	
222	216CS	78	
333	216CS	85	

ويتم الانشاء كمايلي:

```
create table studys(
NO_STU varchar2(6),
COURSE_CODE varchar2(8),
MARK number(3),
point number(5,2),
primary key(NO_STU,COURSE_CODE));
```

بالبيانات الموجودة بالجدول كمايلي: ويتم ادخال

```
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('111','216CS',88);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('222','225CS',75);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('333','225CS',40);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('111','225CS',90);
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('222','216CS',75);
```

```
insert into studys(NO_STU,COURSE_CODE,MARK)
values ('333','216CS',85);
```

بعد الانتهاء من انشاء وادخال البيانات المطلوب انشاء اجراء يقوم بحساب عدد النقاط لكل طالب وفي كل مادة وهو الحقل عدد النقاط الذي لم ندخل فيه اي شيء ويجب نعلم ان :

MARK	average
95-100	5
90-94	4.75
85-89	4.5
80-84	4
75-79	3.5
70-74	3
65-69	2.5
60-64	2
1-59	1

ويتم حساب النقاط كمايلي :

عدد النقاط في اي مقرر = معدل المادة (وليس الدرجة كمافي الجدول السابق) * عدد ساعات المقرر

مثال لحساب معدل الطالب الذي رقمه ١١١ في المقرر 216CS

نلاحظ من جدول studys ان الطالب قد تحصل على درجة ٨٨ ونلاحظ ان الدرجة من الجدول السابق هي بين ٨٥- ٨٩ وبالتالي فإن معدل الطالب في هذا المقرر هو ٤.٥ (وهي الطريقة المتبعة في اغلب الجامعات) ، ومن جدول courses نحصل على عدد الساعات للمقرر وبالتالي فان :

عدد النقاط = ٤.٥ * ٣ = ١٣.٥ وهكذا في جميع الطلاب وهذا هو المطلوب من الاجراء عمله.

وبالتالي فان الاجراء سوف يكون كمايلي:

```
DECLARE
no_Student studys.NO_STU%type;
hou courses.hours%type;
mark studys.mark%type;
cou_code courses.code%type;
poi studys.point%type;
cursor st_point is
```

```
select NO_STU,COURSE_CODE,MARK from studys;
BEGIN
open st_point;
loop
exit when st_point%notfound;
fetch st_point into no_Student,cou_code,mark;
select hours
into hou
from courses
where code=cou_code ;
if (mark>=95)and(mark<=100) then
poi:=5 * hou;
elsif mark>=90 then
poi:=4.75 * hou;
elsif mark>=85 then
poi:=4.5 * hou;
elsif mark>=80 then
poi:=4 * hou;
elsif mark>=75 then
poi:=3.5 * hou;
elsif mark>=70 then
poi:=3 * hou;
elsif mark>=65 then
poi:=2.5 * hou;
elsif mark>=60 then
poi:=2 * hou;
else
poi:=1 * hou;
end if;
update studys set POINT=poi
where NO_STU=no_Student and COURSE_CODE=cou_code
;
end loop;
close st_point;
end;

/
```

لاحظ هنا اننا استخدمنا المؤشرات الصريحة والمؤشرات الضمنيه والصريحة استخدمناه لكي تقوم بفتح سجلات الجدول studys والمؤشر الضمنلي استخدمنا لكي يعود بعدد الساعات في كل مرة يدور بالحلقة.

شرح الاجراء :

في التعريفات اتوقع انه لاتوجد هناك مشكلة لديكم ، اما جسم البرنامج ابتداً من begin فهو كمايلي :

اولا يفتح المؤشر الصريح والذي يحتوي على جميع سجلات الجدول studys ثم يكون حلقة دورانية لكي يمر على جميع سجلات الطلاب الموجودة في المؤشر الصريح وطبعا شرط الانهاء لهذه الحلقة هو الوصول الى اخر سجل . ثم يقوم بعملية تحديث سجلات الطالب الاولى في المتغيرات كمايلي:

```
fetch st_point into no_Student,cou_code,mark;
```

وطبعا المتغيرات هي رقم الطالب ورقم المقرر والدرجة في المقرر ولنفرض الان نحن الان عند السجل الاول ووهو الطالب الذي رقمه ١١١ ورقم المقرر CS٢١٦ ودرجته هي ٨٨ سوف يضع هذه البيانات في المتغيرات، ثم يستخدم مؤشر ضمني لكي يحضر عدد ساعات المادة التي درسها الطالب ١١١ وهي CS٢١٦ و المؤشر هو

```
select hours
into hou
from courses
where code=cou_code ;
```

ومعنى هذا احضر عدد ساعات المقرر الذي رقمه هو cou_code وهذا المتغير هو معروف من المؤشر الصريح الاول وسبب استخدامنا هذا المؤشر هو ان عدد ساعات المقرر موجودة في جدول اخر ولابد من استخدام هذا المؤشر لكي يحضر عدد الساعات. وبما اننا فرضنا اننا عند السجل الاول فسوف يحضر عدد ساعات المقرر CS٢١٦ وهي ٣ ساعات ثم بدأ يختبر الدرجة وذلك طبقا للجدول الدرجات والمعدلات حيث كانت درجة الطالب الذي رقمه ١١١ في المقرر cCS٢١٦ هي ٨٨ وبالتالي يكون عدد النقاط كمايلي = ٤.٧٥ * ٣ = ١٣.٥ ، وبعد الانتهاء من حساب المعدل يقوم بنعديل الجدول وتحديث قيمة point بقيمتها الجديدة ، وهكذا يمر على كل طالب بنفس الطريقة السابقة الى ان يصل الى نهاية السجلات.وبالتالي تكون النتائج كمايلي في الجدول studys :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3

111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

بعد الانتهاء من هذا المثال نكون انهينا المؤشرات بنوعيها بشكل تام وبامثله واقعيه

المصفوفات و الاجرائيات و الدوال المخزنة

اهداف الفصل

سنتعلم في هذا القصل باذن الله

٤- انشاء المصفوفات والتعامل معها

٥- كيفية انشاء اجراء

٦- التعامل مع الوظائف المخزنة

الجداول في pl/sql (المصفوفات)

تستخدم هذه الجداول (المصفوفات) مثل المصفوفات في هي لغة من لغات البرمجة مثل لو كانت لديك سلسلة من الارقام وتريد تخزينها فانك تستخدم هذه الجداول للتخزين ويتم تعريف متغير من هذا النوع كمايلي اولا يتم تعريف هذا النوع :

TYPE اسم_النوع IS TABLE OF نوع_المتغير INDEX BY BINARY_INTEGER

مثال على ذلك :

```
DECLARE
TYPE num_array IS TABLE OF number(4) INDEX BY
BINARY_INTEGER;
num num_array;
BEGIN
........
........
END;
```

لاحظ اولا تم تعريف نوع واسماه num_array ، ثم قام بتعريف متغير num واعطاه نوع num_array وهو النوع الجديد الذي قمنا بانشاءه.

مثال عملي /

```
set serveroutput on;
DECLARE
TYPE num_array IS TABLE OF number(4) INDEX BY
BINARY_INTEGER;
i   number(4);
num    num_array;
BEGIN
FOR  i  IN  1..10 LOOP
num(i) := i * i ;
END LOOP;
FOR  i  IN  1..10 LOOP
DBMS_OUTPUT.PUT_LINE(i || '*' || i || '= ' ||
num(i) );
END LOOP;
END;
/
```

وويكون عمل هذا الاجراء كمايلي : الحلقة الاولى تقوم بضرب العدد i في نفسه وتخزنه في المتغير num برتبه i وهكذا والحلقة الثانية للطباعة ويكون الناتج كمايلي :

1*1= 1
2*2= 4
3*3= 9
4*4= 16
5*5= 25
6*6= 36
7*7= 49
8*8= 64
9*9= 81
10*10= 100

الاجرائيات المخزنة

شاهدنا في الدروس الماضية ان اي اجراء نقوم بكتابة اني اذا اردت استخدامة اكثر من مرة فاني اقوم بكتابة كل مرة في sql * plus لكي احصل على النتائج لكن ماهو رأيك لو نقوم بتخزين هذا الاجراء في قاعدة البيانات ونعطية اسم وحينما نحتاجه نستدعية باسمه وهذا يوفر علينا الشيء الكثير لذلك فصلنا هو الاجرائيات المخزنة.

ولكي نقوم بانشاء اجراء مخزن نقوم بمايلي :

```
CREATE [OR REPLACE] PROCEDURE procedure-name
 [(argument1 ... [, argumentN) ] IS
 [local-variable-declarations]
BEGIN
executable-section
 [exception-section]
END [procedure-name];
```

حيث أن

procedure_name اسم الاجراء المستخدم.

اما OR REPLACE فهي توضع حينما تعلم ان الاجراء موجود من السابق.

اما عن المغيرات التي بين القوسين فهي اما متغيرات مدخله مثل اذا كان لديك اجراء حساب معدل طالب وتريد تمرير رقم الطالب الذي تريد حساب معدله فهذه هي تعتبر كمدخلات ولتعريف متغير بهذا الشكل يكون كمايلي :

```
student_id  in number(9)
```

لاحظ اسم المتغير هو student_id ثم بعده وضعنا الكلمة in ومعنى ان هذا المتغير يعتبر كمدخل

اما لتعريف متغير يعود بقيمة من الاجراء مثلا لو اردنا تعرف متغبر يرجع بمعدل الطالب يتم التعريف كمايلي :

```
ave out number(5,2)
```

بعد تنفيذ الاجراء يكون هذا المتغير يحتوي على معدل الطالب الذي تم تمرير رقمه مثلا.

مع العلم انه يمكن تعريف متغير للمدخلات والمخرجات حيث تمرر به القيمه اولا وبعد تنفيذ الاجراء يتم وضع القيمه في نفس المتغير وتتم كمايلي :

```
ave in out number(5,2)
```

ومعنى هذا اي مدخل ومخرج في نفس الوقت .

مـــثال :

في الجدول الذي قمنا بدراسته في الفصول السابقة وكان بأسم studys

وكان كمايلي :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3
111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

لو اردنا تصميم اجراء مخزن لكي يقوم بطباعة درجة الطالب بعد تمرير رقم الطالب ورقم المقرر.

الاجراء المخزن سوف يكون كمايلي :

```
create or replace procedure stu_mark(
stu_id in studys.NO_STU%type,
cou in studys.COURSE_CODE%type)
as
mar studys.mark%type;
begin
select mark
into mar
from studys
where NO_STU=stu_id
and COURSE_CODE=cou;
DBMS_OUTPUT.PUT_LINE(mar);
end;
/
```

بعد الانتهاء من تنفيذ الاجراء يكون الاجراء مخزن في قاعدة البيانات ولكي نقوم باستدعاءه نقوم بمايلي

```
begin
stu_mark(111,'216CS');
end;
/
```

لاحظ كيف تم استدعاء الاجراء السابق من خلال اسم الاجراء وبذلك سوف يكون الناتج على الشاشة كمايلي 88 وهي صحيحة بعد تمرير رقم الطالب 111 ومقرر 216CS

لكن لاحظ اننا لم نستخدم متغيرات اخراج لكن مأرايك ان نصمم اجراء اخر يقوم بنفس الوظيفة التي يقوم بها الاجراء السابق لكن عملية الطباعة تكون بعد الاستدعاء لكي نجعل الاجراء يقوم بارجاع درجة الطالب بمتغير لذلك فان الاجراء كمايلي :

```
create or replace procedure stu_mark22(
stu_id in studys.NO_STU%type,
cou in studys.COURSE_CODE%type,
mara out studys.mark%type)
as
begin
select mark
into mara
from studys
where NO_STU=stu_id
and COURSE_CODE=cou;
end;
/
```

بعد ذلك نقوم باستدعاء الاجراء ومن ثم طباعة الدرجة لان لو لاحظت الاجراء لايقوم بالطباعة ولاحظ ايضا ان الدرجة تم وضعها في المتغير mara ولذلك سوف يعود بهذه القيمة وسوف يكون الاستدعاء كمايلي :

```
declare
m studys.mark%type;
begin
stu_mark1(111,'225CS',m);
DBMS_OUTPUT.PUT_Line(m);
end;
/
```

وسوف يكون الناتج هو ٩٠ وهذا صحيح بناء على الجدول.

الوظائف المخزنة

كان درسا السابق عن الاجراءات المخزنة واليوم لدينا درس مشابه له وهو الوظائف المخزنة لكن الفرق ان الوظائف لابد ان تعيد قيمة

والصسغة العامة لتكوين وظيفة كمايلي:

```
CREATE [OR REPLACE] FUNCTION function-name
 [(argument1 ... [, argumentN) ]
RETURN function-datatype IS
 [local-variable-declarations]
BEGIN
executable-section
 [exception-section]
RETURN function-value
END [function-name];
```

حيث أن

function_name اسم الوظيفة المستخدمه.

اما REPLACE OR فهي توضع حينما تعلم ان الاجراء موجود من السابق.

اما عن المغيرات التي بين القوسين فهي اما متغيرات مدخله مثل اذا كان لديك اجراء حساب معدل طالب وتريد تمرير رقم الطالب الذي تريد حساب معدله فهذه هي تعتبر كمدخلات ، وهي بنفس الطريقة التي تعاملنا بها مع الاجراءت المخزنة لاتغير على المتغيرات وطرق تعريفها.اما RETURN datatype فهي تدل على نوع القيمة المعادة من الوظيفة .

مثـــــال : في الجدول الذي قمنا بدراسته في الدرس الرابع وكان بأسم studys وكان كمايلي :

NO_STU	COURSE_CODE	MARK	POINT
111	216CS	88	13.5
222	225CS	75	10.5
333	225CS	40	3
111	225CS	90	14.25
222	216CS	78	10.5
333	216CS	85	13.5

لو اردنا تصميم وظيفة ترجع بمعدل الطالب الفصل اي يتم تمرير رقم الطالب الى الوظيفة ثم يتم حساب المعدل الفصلي للطالب

ويتم حساب المعدل الفصل للطالب كمايلي =مجموع النقاط ÷ مجموع عدد الساعات لمقررات

ولانشاء الوظيفة كمايلي :

```
1  create or replace function stu_avea(stnum in
2 studys.NO_STU%type)
3  return real
4  as
5  hour courses.hours%type;
6  avrage number(4,2);
7  sum_hours courses.hours%type:=0;
8  point studys.POINT%type;
9  total_Point studys.POINT%type:=0;
10 codem courses.CODE%type;
11 cursor sumpoint
12 is
13 select COURSE_CODE,POINT
14 from studys
15 where NO_STU=stnum;
16 begin
17 open sumpoint;
18 loop
19 fetch sumpoint into codem,point;
20 exit when sumpoint%notfound;
21 select hours
22 into hour
23 from courses
24 where code=codem;
25 total_Point:=total_Point+point;
26 sum_hours:=sum_hours+hour;
27 end loop;
28 close sumpoint;
29 avrage:=total_Point/sum_hours;
30 return avrage;
   end;
```

الشرح :

السطر رقم ١: لتعريف الوظيفة
السطر رقم ٢: نوع القيمة التي سوف ترجع بها الوظيفة
السطر رقم ٤: تعريف متغير عدد الساعات وهو نفس حقل عدد ساعات المقرر الموجودة في جدول courses

الان بعد الانتهاء من شرح طريقة تصميم الوظيفة جاء دور طريقة الاستدعاء :

لكن قبل الاستدعاء لنحسب يدويا معدل الطالب الذي رقمه ١١١ مثل لكي نقارنه بالنتائج بعد الاستعلام :

مجموع نقاط الطالب = ١٣.٥+١٤.٢٥= ٢٧.٧٥

مجموع عدد الساعات = (عدد ساعات المقرر CS٢١٦) + (عدد ساعات المقرر CS ٢٢٥)
= ٣ + ٣ = ٦

وبالتالي فان معدل الطالب = ٢٧.٧٥ ÷ ٦ = ٤.٦٣

لكن الان دعنا نستدعي الدالة ونشاهد النتائج

```
SELECT distinct(NO_STU),stu_avea(no_stu)
from studys
where no_stu=111;
```

لاحظ كيف تم استدعاء الدالة من خلال الاستعلام ولاحظ استخدام الدالة distinct وهي لعدم تكرار السجل واليك النتائج :

```
NO_STU          STU_AVEA(NO_STU)
------    ;     ----------------
111               4.63
```

لاحظ لو كان الاستعلام بدون وجود الدالة distinct فسوف يتكرر رقم الطالب عدد ظهوره في الجدول لذلك لوكان كمايلي :

```
SELECT NO_STU,stu_avea(no_stu)
from studys
where no_stu=111;
```

فان النتائج ستصبح هكذا

```
NO_STU                  STU_AVEA(NO_STU)
------      ;           ----------------
111                        4.63
111                        4.63
```

وهذا سبب ظهور الدالة distinct

الحزم البرمجية و الزنادات

أهداف الفصل

سنتناول في هذا الفصل ان شاء الله :

٦- ماهية الحزم البرمجية
٧- كيفية انشاء الحزم البرمجية ومكوناتها
٨- ماهية الزنادات
٩- التعامل مع الزنادات
١٠- قوائد الزنادات

الحزم البرمجية

تعلمنا سابقا كيفية انشاء الاجراءت والوظائف المخزنة. لكن مارأيك لو وجد لدينا قاعدة بيانات كبيرة جدا ولنضرب مثال انها تحتوي على ٥٠ اجراء او وظيفة وظيفة او اجراء لها عمل خاص ولنفرض ان هذه القاعدة هي لمحل تجاري ضخم يحتوي على بيانات العملاء وبيانات الموظفين وبيانات الاصناف التجارية وبيانات المخزون وغيرها من بيانات ، ولذلك فان بعض هذه الاجرائيات والوظائف المخزنة مختص بالعملاء مثلا وجود اجراء لحساب اجمالي عميل وغيرها من الاجرائيات ، ومثل وجود اجرئيات خاصة بالموظفين مثلا اجرائية خاصة بحساب راتب الموظف بعد حذف الحسومات واضافة العلاوات وغيرها ايضا ، لكن وضعها في هذا الشكل في قاعدة البيانات قد يسبب لك بعض الارباك لذلك مارأيك بان تجمع كل الوظائف والاجرائيات الخاصة بكل قسم في مجموعة لوحدها وهذه المجموعة تدعي الحزمه package مثلا نجمع كل اجرائيات والوظائف الخاصة بالعملاء في حزمة خاصة

فوائد استخدام الحزمة

١- تجميع وحدات pl/sql المرتبطة.
٢- اداء افضل.
٣- تكون السرية افضل.
٤- اهم شيء هو في عملية الصيانه حيث تسهل عملية الصيانة باستخدامم الحزم.

مكونات الحزم

تتكون الحزمة من جزءين الاول وهو الوصف specification ويحتوي على التعاريف مثل متغيرات او مؤشرات او اسماء الاجراءت ومتحولتها.
اما الجز الثاني فهو جسم الحزمة ويحتوي على تفاصيل الاجراءت والعمليات وغيرها
والصيغة العامة لانشاء الجزء الاول كمايلي :

```
CREATE OR REPLACE PACKAGE pack_name AS
.........
.........
.........
end;
```

والصيغة العامة لانشاء الجزء الثاني كمايلي :

```
CREATE OR REPLACE PACKAGE BODY pack_name AS
.........
الحزمة جسم
.........
end;
```

لكن يجب ان يكون اسم الحزمة في الجزء الاول هو نفس اسم الحزمة في الجزء الثاني.

مـثال :

لنقم بانشاء حزمة تحتوي على وظيفة لحساب معدل طالب واجراء لطباعة المعدل
ولذلك سوف نستخدم نفس الوظيفة التي انشناها في الدرس السادس والتي اسمها stu_avea
والتي تقوم بحساب معدل الطالب والان نبداء بانشاء الحزمة . الجزء الاول من الحزمة
specification كمايلي :

```
CREATE OR REPLACE PACKAGE student AS
function stu_avea(stnum in
studys.NO_STU%type)return real;
procedure print_ave(avrage in real);
end;
```

الان نقوم بانشاء جسم الحزمة والتي تحتوي على التفاصيل.كمايلي

```
CREATE OR REPLACE PACKAGE BODY student AS
function stu_avea(stnum in studys.NO_STU%type)
return real
as
hour courses.hours%type;
avrage number(4,2);
sum_hours courses.hours%type:=0;
point studys.POINT%type;
total_Point studys.POINT%type:=0;
codem courses.CODE%type;
cursor sumpoint
is
select COURSE_CODE,POINT
from studys
where NO_STU=stnum;
begin
open sumpoint;
loop
fetch sumpoint into codem,point;
exit when sumpoint%notfound;
select hours
into hour
from courses
where code=codem;
total_Point:=total_Point+point;
sum_hours:=sum_hours+hour;
end loop;
```

```
close sumpoint;
avrage:=total_Point/sum_hours;
return avrage;
end;

procedure print_ave(avrage in real)
as
begin
DBMS_OUTPUT.PUT_LINE(avrage);
end;
end;
```

ويحتوي جسم الحزمة كمانلاحظ على مكونات الوظيفة والاجراء الذي تم تعريفهما في وصف الحزمة حيث ان الوظيفة لحساب المعدل والاجراء لطباعة المعدل.

طريقة استدعاء اجراء او وظيفة موجود داخل حزمة :

تتم عملية الاستدعاء كمايلي :

pack_name.func_proc_name

اي اسم الحزمة اولا ثم نقطة ثم اسم الاجراء او الوظيفة مـــــثال :

```
set serveroutput on
declare
aa real;
begin
aa:=student.stu_avea(111);
student.print_ave(aa);
end;
/
```

وبعد التفيذ يكون الناتج هو معدل الطالب الذي رقمه ١١١ لاحظ اول شي استدعينا داله حساب المعدل ووضعناها في المتغير aa ثم استدعينا اجراء الطباعة ليتم طبعة على الشاشة.

والان و بعد ان تعرفت على فائدة الحزم مارأيك من الان فصاعد ان تستخدم الحزم في كتابة الاجرائيات والوظائف

الزنادات TRRIGERs

تتشابه الزنادات مع البرامج الفرعية الا في الطرق التالية :

* يتم تنفيذ الزنادات ضمنيا،عندما يعدل الجدول بالرغم من عمل المستخدم او التطبيقات على الجدول .

* يتم تعريف الزنادات للجدول الخاص بقاعدة البيانات

* لاتقبل الزنادات المعاملات

تعد الزنادات هامة جدا في تطوير نظم البيانات الموجهة الخاصة بالانتاج .

تركيب الزناد :

```
create [or replace] Trigger <TRRIGER_NAME>
<before|after] [instead of] trigger event on <table name>
[for Each row [whene triggering restriction]]
<trigger body>
```

كماهو مع الاجراءات المخزنة امكانية استخدام replace لكي تقوم بالتعديل على الزناد اذا كان موجود ولاتقوم بانشاءه من جديد.
ينفذ التوقيت الخاص بالزناد سواء نفذ الزناد قبل او بعد اغلاق الزناد بواسطة الخيارين before و after ، لكن خيار after اكثر كفاءة لان قطع البيانات المؤثرة يجب ان تقراء منطقيا مرة للزناد ومرة لعبارة trigger
ملاحظة/ ان حدث اطلاق الزناد هو جملة sql التي تجعل الزناد وحدث الاطلاق اما update او delete او insert او بكليهما..
ويوجد اربعة انواع من الزنادات:
١- صف after.
٢- جملة after.
٣- صف before.
٤- جملة before.
وكل زناد من اجل جملة update او insert او delete كل زناد يعد نوع واحد من (instead of , after,before) ويمكن تعريف تسع زنادات للجدول الواحد..

* معالجة احداث اطلاق الزناد :
او على توليفة من هذه delete او update او insertيحتوي حدث اطلاق الزناد على عملية العمليات عندما يتعامل زناد واحد مع اكثر من عملية واحدة، فيمكنك ان تستخدم دعائم شرطية للتعرف على نوع العبارة التي تستخدم لتنفيذ الجزء الخاص بالرمز في الزناد
والدعائم هي كمايلي:

```
IF inserting then .....end if;
IF updating then .....end if;
IF deleting then .....end if;
```

قيد الزناد

يحدد هذا القيد تعبير منطقي يجب ان يكون صحيح كي يطلق الزناد.
على سبيل المثال الزناد التالي stduent_trigger لا يتم حدوثة الا اذا كان رقم الطالب student_id اقل من ١٠٠

```
create or replace trigger stduent_trigger
before insert or update on student
for each row
when(new.student_id<100)
```

قيود على انشاء الزنادات:

١- يمكن للنص ان يحتوي على جمل dml sql لكن جمل select يجب ان تكون جمل select into
٢- لايسمح بجمل التحكم (commit,savepoint,rollback)
٣-لايمكن لبرنامج فرعي مخزن ان يتضمن جمل التحكم السابقة اذا تم استدعائه بواسطة الزناد.

مـــــــــــثال :

لنفرض انه لدينا الثلاث جداول التالية:

الاول : هو جدول player بيانات جميع اللاعبين في النادي سواء درجة شباب او درجة ممتاز :

no_player	name	date_birth	phone	address	levels
1	talal	11/11/1973	123456	riyadh1	1
2	mohammed	1/1/1982	654321	riyadh2	2
3	sami	1/1/1988	123789	riyadh3	2
4	yosif	12/3/1970	123123	riyadh4	1

حيث level تمثل الدرجة التي يلعب بها اللاعب حيث ١ تمثل الدرجة الاولى الممتاز - و ٢ تمثل الشباب .

ولانشاء الجدول كمايلي :

```
create table player(
no_player varchar2(6) primary key,
name varchar2(50),
date_birth date,
phone varchar2(9),
address varchar2(20),
levels number(2));
```

الثاني : هو جدول اللاعبين في درجة الممتاز وهو خاص بالرواتب واسم الجدول larg_player

no_player	level_no	salary
1	1	
4	1	

ولانشاء الجدول كمايلي :

```
create table larg_player(
no_player varchar2(6) primary key,
level_no number(2),
salary number(7,2));
```

الثالث : هو جدول اللاعبين في درجة الشباب وهو خاص بالرواتب واسم الجدول youth

no_player	level_no	salary
2	2	
3	2	

ولانشاء الجدول كمايلي :

```
create table youth(
no_player varchar2(6) primary key,
level_no number(2),
salary number(7,2));
```

الان نريد عمل زناد بحيث حينما يقوم المستخدم بادخال اسم لاعب جديد وتحديد مستواه (شباب او ممتاز) يقوم الزناد باختبار المستوى فاذا كان شباب اضاف رقم اللاعب في جدول الشباب وكذلك لوكان مستواه درجة اولى الممتاز فانه يضيف رقم اللاعب في جدول larg_player وبذلك يكون الزناد كمايلي :

```
create or replace trigger player_age
before insert on player
for each row
begin
if inserting then
if :new.levels=1 then
insert into larg_player(no_player,level_no) values
(:new.no_player,:new.levels);
elsif :new.levels=2 then
insert into larg_player(no_player,level_no) values
(:new.no_player,:new.levels);
end if;
end if;
end;
```

بعد ذلك قم بادخال مايلي :

```
insert into player values('1','talal','11/11/1973','123456','riyadh1',1);
```

سوف larg_player بعد ذلك اذهب وقما بالاستعلام في جدول playerلاحظ ان المدخلات تمت على جدول تجد انه اضاف رقم اللاعب هناك.

المراجع

١- Oracle 8 Unleashed Second Edition – Sams
٢- Introduction to Oracle: SQL and PL/SQL Using
Procedure Builder (Electronic Presentation)1995
٣- Oracle8 How-To
٤- Teach Yourself Oracle 8 In 21 Days

Mohamed ismael Mohamed

moonbook@live.com