Oracle

Part 1

الجزء الأول

تعريفات

ما هي الأوركل

الهيكل المادي للملفات في أوراكل Files Structure

وحدات تخزين البيانات في أوراكل

العمليات Processes

هيكلية الذاكرة Memory Structure

تنفيذ المعاملات في أوراكل

فتح قاعدة البيانات Database Startup

إغلاق قاعدة البيانات Database Startup

الكتالوغ The Data Dictionary

معلومات لابد منها

تنصيب قاعدة بيانات أوراكل

إن أردنا تعريف الأوراكل نقول إنه برنامج قاعدة بيانات وهي شبيهة إلى مايكروسوفت أكسس وأقرب إلى مايكروسوفت إس كيو إل سريفر يعني أنها قاعدة بيانات وليست لغة برمجة مستقلة بحد ذاتها فمثلا فيجوال بيسك تعتبر لغة برمجة لأنها تمكنك من عمل برامج ذات أهداف متعددة ولا يشترط أن تكون برامج قواعد بيانات فيمكنك عمل برنامج رسم كما برنامج الرسام في ويندوز ويمكنك التحكم بها على حد كبير من المرونة والإمكانيات التي تسمح لك حتى بالارتباط بقواعد بيانات متعددة ولكن أوراكل هي قاعدة بيانات قوية و آمنة ولكن يوجد لديها أدوات تساعدها للتعامل معها وإظهارها في أشكال متعددة مثل الدفلوبر بحيث يمكنك إدخال البيانات واستخراجها عن طريق نماذج وتقارير ورسوم بيانية ولكن لا يمكنها التعامل مع قاعدة بيانات غير أوراكل كما أنها لا تمنك من عمل برامج مثل الرسام دعنا نأخذ أمثلة لكي يتبين الفرق لديك. ولنفترض مثلاً بأن مديرك طلب منك ثلاثة مشاريع وهي:

المشروع الأول

برنامج يقوم بعمل ملصقات خاصة بالرسائل لكي يتم إلصاقها على مظاريف المراسلات يستند إلى قاعدة بيانات العملاء.

الحل

يمكن للبرنامج الاستناد على قاعدة بيانات أوراكل أو عمل قاعدة بيانات صغيرة على أكسس ويمكن عمل الملصقات كتقرير مخرج من نفس أوراكل ولكن في الفيجوال بيسك يمكنك إضافة لمسات فنية وإضافة فلاتر معينة تعمل على الملصقات لإظهارها بأشكال جمالية أكثر ففي حال أوراكل تحتاج إلى نسخة قاعدة بيانات أوراكل لتحميلها على الجهاز كما تحتاج للملفات التشغيلية الخاصة بالنماذج والتقارير كما أن الحمايات يجب أن توزع للمستخدمين بحيث لا يسمح لكل مستخدم الدخول قبل إعطائه هذه السماحيات فبرنامج كهذا لا يحتاج إلى حماية لأن أي موظف من الممكن استخدامه ولذلك يفضل فيجوال بيسك

المشروع الثاني

برنامج يقوم بعمل مفكرة للهاتف الشخصي أو لعناوين وهواتف العملاء

الحل

هذا البرنامج ينطبق عليه نفس ما ينطبق على المشروع الأول

المشروع الثالث

برنامج خاص بالحسابات يستند على قاعدة بيانات حسابات الشركة في ثلاثة فروع موزعة جغرافياً وهذه المؤسسة تتعامل أيضا بالتجارة الإلكترونية على الإنترنت كما تحتاج لتقارير وإحصاءات دورية تزود للعملاء من قبل البرنامج دون الحاجة لعمل يدوي.

الحل

هذا البرنامج يمكن أن يحل بفيجوال بيسك ولكن قاعدة البيانات لا يمكن أن تكون أكسس بل مستحيل وذلك لكثرة البيانات وكثرة العمليات وتوزيع قواعد البيانات بين فروع الشركة المختلفة فيمكن أن يربط برنامج فيجوال بيسك بقاعدة بيانات أوراكل أو أي كيو أل سيرفر من مايكروسوفت ولكن كان من الأحرى استعمال الديفلوبر لهذا التطبيق وذلك لمرونة البرنامج مع قاعدة البيانات التي وضع الديفلوبر لأجلها كما أن من أدوات أوراكل إمكانية التعامل مع الإنترنت وإعطاء حماية وأمان للبيانات الموجودة كما أن البرنامج لا يمكن أي مستخدم من الدخول إليه والعبث بمحتوياته سوى من هم مختصين بذلك وبسماحيات متفاوتة.

يشير الاسم اليوناني القديم أوراكل Oracle إلى الكاهنة المقدسة التي يطلبها الناس للتبوء بالغيب ولتفسير حكمة الإلهة المقدسة إن كل ما تقوله هو حكمة ولكنها دائماً حكمة مقدسة مغلفة بالغموض. ويصف اللفظ أيضاً كل ما لديه الكثير من العلم بموضوع معين ويستطيع أن يقدمه كنصيحة غالية. أما الصفة فهي oracular وهى تشير إلى الصعوبة في الفهم. وإذا كانت أوراكل جديرة بان تتتوج أميرة للسحر والغموض إلا إننا ننوي بإذن الله أن ننظر لنرى كيف يعمل هذا البناء الضخم المسمى بأوراكل من الداخل وذلك الأمر يعد مقدمة لا غنى عنه لمن يريد أن ينتفع

بمزايا نظام من أقوى نظم قواعد البيانات وان يتقدم قليلاً في دراسته كي يصبح بعون الله مديراً ومسيطراً على أوراكل الأميرة الساحرة.

وقد حاولت قدر المستطاع أن أقدم أوراكل بأسلوب مبسط ولم التزم حرفياً بالتقسيم الموجود في كتابات أوراكل المعقدة كي أحاول بقدر الإمكان تبسيط المعلومة ومن ثَم ندخل إلى التعامل مع إصداراتها.

ما هي الأوركل

تشير قاعدة البيانات Databaseإلى مجموعة منظمة من البيانات والتي تخزن بطريقة معينة بحيث يسهل الوصول إليها والتعامل معها بكفاءة. ويتم التعامل مع تلك البيانات بواسطة نظم قواعد البيانات RDBMS وهى برامج معقدة وأدوات تمكننا من الوصول إلى البيانات بسهولة.

إن إدارة قواعد البيانات تتطلب العديد من العمليات والإجراءات ومن أمثلتها:

إنشاء قواعد البيانات: وهي الحاوية أو الوعاء التي ستنتظم فيه البيانات وكيفية تنظيم البيانات نفسها داخل تلك الحاوية وطرق استرجاعها كما تتضمن أيضاً إجراءات تأمين البيانات وطرق الاحتفاظ بتا سليمة وحمايتها من التلف.

إن أوراكل كنظام لإدارة لقواعد البيانات قد تم تصميمه خصيصاً لكي يعمل في بيئة عمل تحوي العديد من المستخدمين الذين يريدون إنجاز إعمالهم بسرعة وسهولة وبالتالي ضمان الوصول في نفس الوقت تقريباً إلى قاعدة البيانات والاستفادة من مواردها المتاحة وتتميز بيئة العمل تلك بالتالي:

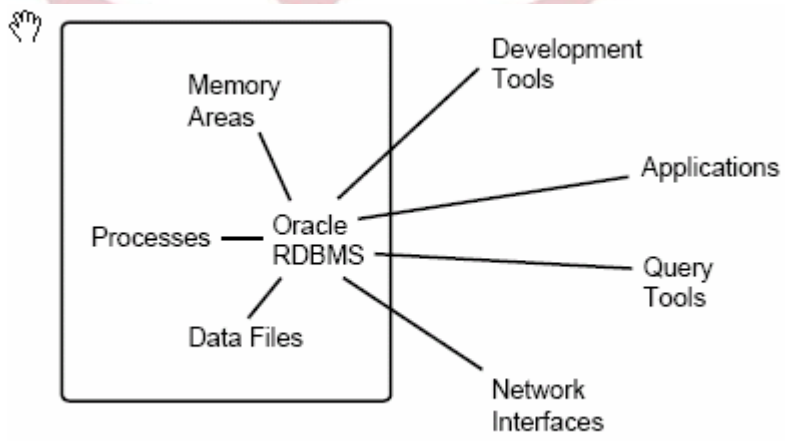
- يعمل نظام التشغيل على تنفيذ العديد من العمليات Process في نفس الوقت.
- يقوم نظام التشغيل بتحديد وقت معين لكل عملية وطريقة تنفيذها وأولوية التنفيذ.
- يقوم نظام التشغيل بتقسيم الذاكرة Memory إلى مناطق أو أجزاء محجوزة للمستخدمين ومناطق تتشارك فيها مجموعة من العمليات.

قبل أن نصور الكيفية التي يعمل بها سيرفر الأوركل ينبغي توضيح معنى السيرفر والذي هو ببساطة مجموعة من البرامج والتي تشكل نظاماً متكاملاً تركب على جهاز

كمبيوتر بمواصفات خاصة كي يستطيع تقديم خدمات معينة إلى برامج أخرى أو إلى عدة مستخدمين في نفس الوقت تقريباً.

إطلالة على سيرفر الأوركل :

سيرفر أوراكلOracle Server هو مجموعة من البرامج التي تشكل نظاماً شاملاً لإدارة قاعدة البيانات. من خلال تلك البرامج يتم إجراء عمليات عديدة في الذاكرة والتي تتعامل معها الأوركل بشكل خاص حيث تسمح تلك العمليات بالوصول إلى الملفات المادية التي تشكل قاعدة البيانات والتعامل معها بكفاءة. يمكن توضيح تلك الفكرة عن طريق الشكل التالي :



من خلال العرض السابق يمكن تصور معمارية السيرفر لأوراكل على النحو التالي:

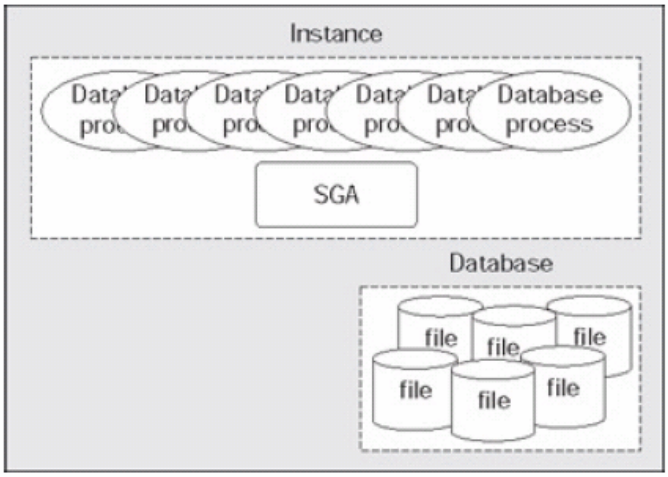
1. العمليات Processes
2. معمارية الذاكرة Memory Structure
3. الملفات المادية المكونة لقاعدة البيانات Physical File Structure

ينبغي أن نعي التفرقة بين الأوركل كداتابيز Database حيث يشير المصطلح إلى مجموعة منظمة من الداتا مخزنة في ملفات أما المثال Instance فهو مجموعة من

العمليات Processes والتي تتم في مناطق معينة من الذاكرة تحددها أوراكل كي نتمكن من الوصول إلى الملفات المادية التي تشكل قاعدة البيانات والتعامل معها.
أي أن:

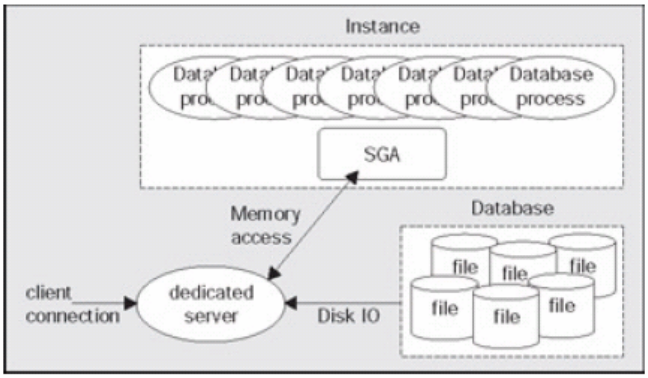
- معمارية الذاكرة + العمليات التي تتم فيها = Oracle Instance
- الملفات المادية المكونة لقاعدة البيانات يشار إليها عادة بـ Database
- الملفات المادية المكونة لقاعدة البيانات + مثال الأوركل = Oracle Server

الشكل التالي يبين اكثر المقصود مما قلناه :



ولكن ما الذي يحدث عندما يقوم المستخدم بالاتصال بأوراكل ؟

يمكننا توضيح ذلك بالرسم التالي :



ABAHE

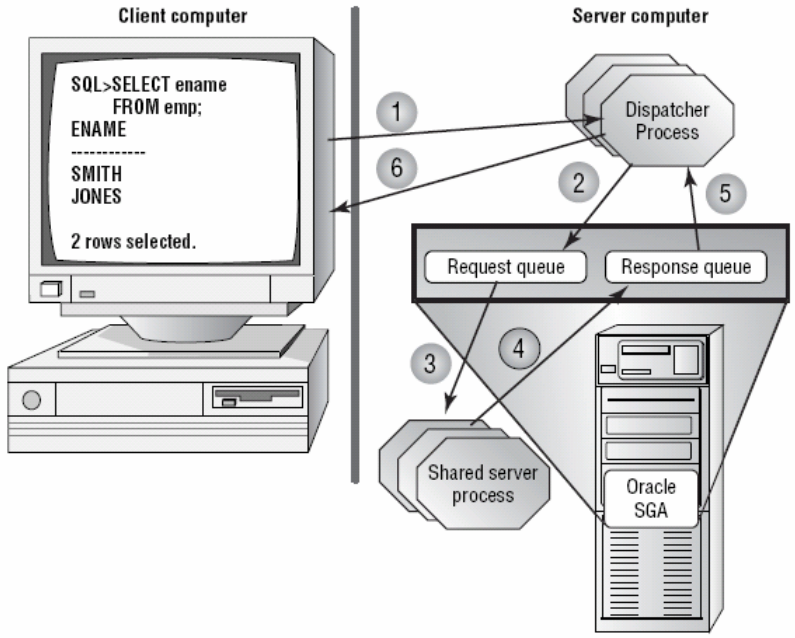
عندما يقوم المستخدم بالاتصال بأوراكل من خلال تطبيق ستقوم أوراكل بتوليد عملية جديدة مخصصة بالكامل له Dedicated Server Process وستظل موجودة طوال فترة الاتصال ويكون الشاغل الأساسي لتلك العملية هو خدمة المستخدم فإذا قمنا بتنفيذ استعلام مثلاً فإنها سوف تستقبل جملة الاستعلام وتنفذها حيث تبدأ في إعطاء الأوامر للسيرفر للبحث عن الداتا سواء في ذاكرة الكاش أو الملفات ثم تقوم بجلبها إلى التطبيق الذي طلبها. أن الغرض الأساسي منها هو الاستجابة إلى طلبات Sql الخاصة بالمستخدم .

إن أوراكل تقدم بديلاً آخر للاتصال يسمح لمجموعات ضخمة من المستخدمين بالعمل في نفس الوقت بكفاءة وهو Shared Server وهو ميكانيزم للاتصال يمكن تشبيه بجدول كبير من المياه التي يتم توزيعها على عدد كبير من الأفراد فيكون لكل واحد منها حصته وهذا الأسلوب يتيح لزيادة عدد المستخدمين لقاعدة البيانات تقريباً بلا حدود فبدلاً من تخصيص عملية لكل مستخدم ولنفرض أنهم 10آلاف مستخدم يمكن بذلك الطريقة فتح 100عملية من أوراكل تظل مسؤولة على خدمة ذلك العدد الكبير من المستخدمين.

وطبقا لهذا البديل تقوم أوراكل بفتح مجموعة من العمليات تسمى مستقبلات **dispatchers** والتي ستقوم بوضع طلبات المستخدمين على هيئة طابور جاهز للتنفيذ في منطقة للذاكرة المؤقتة التي تعينها أوراكل وتسمى منطقة النظام الشاملة System Global Area(SGA) ويتم الانتقاء من العمليات الغير شاغرة لتنفيذ

ذلك الطابورQuee فلو كانت العملية الأولى غير مشغولة بتنفيذ طلب آخر فإنها ستتولى الانتقاء من الطابور لتنفيذه وهكذا كما يتضح من الشكل التالي.

FIGURE 5.1 Request processing in Shared Server



يتضح من الشكل السابق التالي :

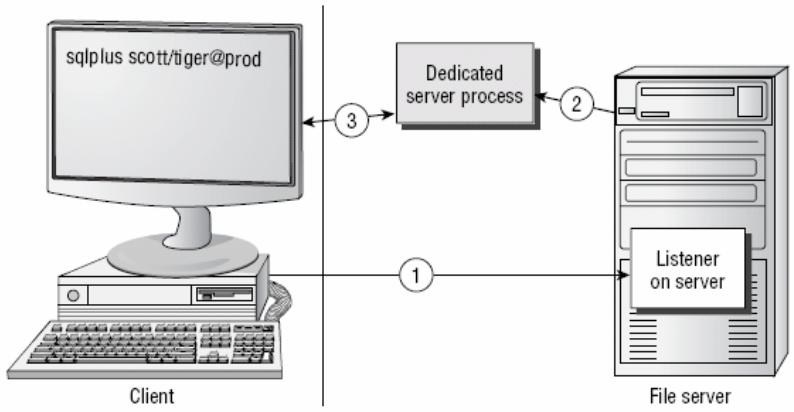
- أن المستخدم سوف يقوم بالاتصال بالمستقبل dispatcher وعندها يقوم المستقبل بوضع أمر المستخدم في طابور الانتظار في منطقة الذاكرة SGA.
- أول عملية تكون جاهزة سوف تقوم بأخذ الطلب من الطابور لتنفيذه.
- طلب المستخدم تم تنفيذه وتم وضع نتائج التنفيذ في طابور الانتظار مرة أخرى تمهيداً لعرضه على المستخدم يقوم المستقبل بأخذ نتائج الطلب من الطابور ثم يقدمه إلى المستخدم ثم يغلق قناة الاتصال.

ولكن كيف يقوم المستخدم Client بالاتصال بأوراكل :

تناولنا في الفقرة السابقة ما الذي يحدث عندما يقوم المستخدم بالاتصال بأوراكل وإعطاء طلباته ليتم تنفيذها ولكن كيف يقوم فعليا بإجراء ذلك الاتصال أو بمعنى آخر ما الذي يمكنه من إجراء مثل ذلك الاتصال ؟

يمكن توضيح ذلك من الشكل التالي وهو حالة تخصيص عملية لكل مستخدم Dedicated Server Process

FIGURE 4.6 Dedicated connection: direct handoff method



في معظم الحالات يتم الاتصال من خلال عن طريق برتوكولات الاتصال الشبكي TCP/IP حيث يكون المستخدم على جهاز يقوم بالاتصال بالسيرفر والذي يكون مركباً على جهاز آخر من خلال برتوكول TCP/IP يقوم المستخدم باستخدام تطبيق للاتصال بأوراكل.

هنا يقوم بكتابة اسم المستخدم الذي يريد الدخول إليه في أوراكل وكذلك كلمة السر كما يكتب اسم الخدمة الشبكية التي يريد الاتصال بها TNS Names.

أن TNS هي برنامج للاتصال موجود على جهاز المستخدم وهو اختصار لـTransparent Network Substrate ويقوم بالتعامل مع الاتصالات عن بعد Remote وكذلك الاتصال من نقطة لنقطة Peer to peer communication.

إن اسم الخدمة الشبكية هو اختصار Connect String يحل إلى مجموعة عبارات تخبر عن مكان الداتابيز الذي يراد الاتصال بها. بكتابة اسم الخدمة الشبكية وهو وفقا للمثال السابق ora816.us.oracle.com يقوم برنامج الاتصال بمحاولة حل اسم الخدمة الشبكية أو معرفة ما يقابلها من عبارات توضح مكان وكيفية الاتصال بقاعدة البيانات وهناك اكثر من طريقة لمعرفة تلك المعلومات ولنأخذ أسهلها وهي الاستعانة بملف نص يسمى TNSNAMES.ORA ويكون على الشكل التالي حيث تتم قراءته بالنسبة للخدمة السابقة على النحو التالي

```
ORA816.US.ORACLE.COM =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = aria.us.oracle.com)(PORT = 1521))
  )
  (CONNECT_DATA =
   (ORACLE_SID = ora816)
  )
 )
```

هنا يقدم الملف السابق مجموعة من المعلومات المفيدة والتي تعين على تحديد موقع قاعدة البيانات المراد الاتصال بها على النحو التالي:

- اسم الجهاز الموجود به أوراكل HostNmae
- اسم البوابة المفتوحة للاتصال بأوراكل على الجهاز Port
- اسم الداتابيز SID

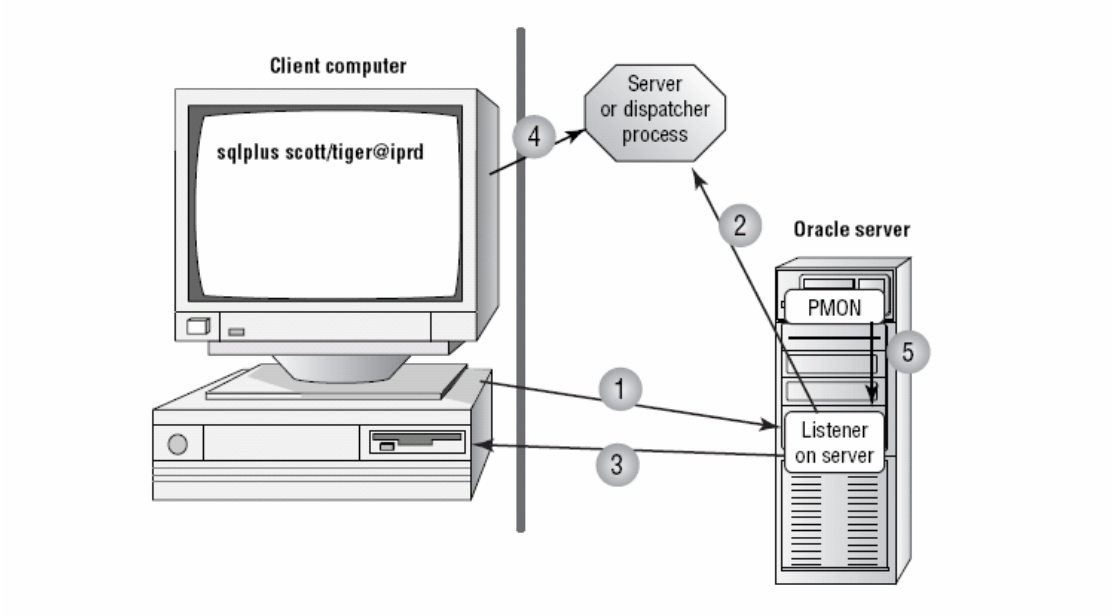
الآن عُرف التطبيق في جهاز المستخدم أين سوف يتصل فهو يستخدم بروتوكول الاتصال TCP/IP في فتح قناة اتصال بينه وبين قاعدة البيانات الموجودة على

الجهاز aria.us.oracle.com بواسطة فتح البوابة 1521 الموجود على الجهاز المذكور.

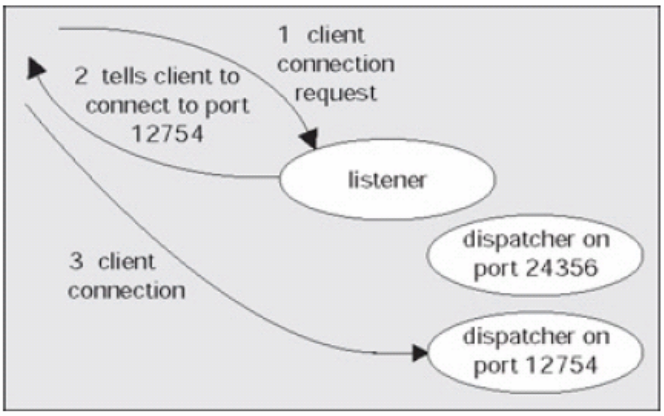
في الجهاز الموجود عليه السيرفر توجد عملية تعمل في الخلفية طوال الوقت مهمتها استقبال طلبات الاتصال من المستخدمين وتسمى TNS Listener. يمكن تشبيه تلك العملية بأذن كبيرة مهمتها أن تتنصت على الشبكة لترى هل هناك أحد يطلب الاتصال بالسيرفر أم لا. فإذا كان هناك طلب يبحث في مشروعيته أولاً فإذا كان من الجهات المسموح لها بالدخول يقوم بامرار الاتصال.

يقوم البرنامج السابق وهو Listner على حسب إعداده بالسماح بفتح قناة اتصال بين المستخدم وبين السيرفر وطبقاً لإعدادات السيرفر فهو يقوم بفتح عملية من عمليات السيرفر لخدمة طلب العميل Didicated Server Process أو بتوصيله بالمستقبل Dispatcher في حالة Shared Server. وفى حالة نجاحه في تلك العملية تكون مهمة Listener قد انتهت بفتح تلك القناة ويبقى في حالة تسمع آخر بينما يكون المستخدم قد اتصل مباشرة بقاعدة البيانات ويكون بخدمته عملية من عمليات السيرفر كما أوضحنا من قبل.

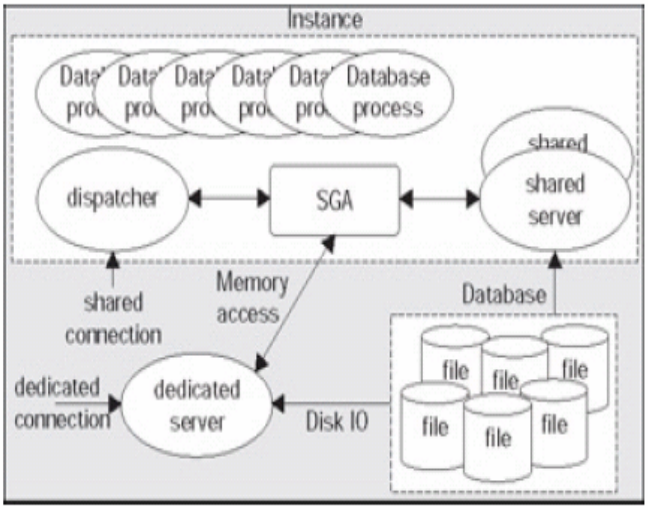
دور الليسنر Listner في حالة Shared Server



يعلم الليسنر بالضبط المستقبلات التي تعمل على الداتابيز وعندما يطلب المستخدم الاتصال يقوم الليسنر باختيار مستقبل من تلك المستقبلات التي تعمل ثم يعود ليقدم للمستخدم معلومات عن كيفية الاتصال بالمستقبل ومنها رقم البوابة المفتوحة وإلى هنا ينتهي دور الليسنر فينقطع الاتصال بينه وبين المستخدم ليبدأ المستخدم الاتصال مباشرة بالمستقبل كما يتبين ذلك من الشكل التالي.



والشكل التالي هو توضيح لما تم ذكره قبل الآن وهو يوضح التفاعل الذي يتم بين المستخدم وبين عمليات السيرفر



الهيكل المادي للملفات في أوراكل Files Structure

ببساطة يمكن النظر إلى أوراكل سيرفر على أنه عبارة عن مثال مربوط بملفات مادية Physical Files تشكل الجزء المادي من قاعدة البيانات فالمثال كما ذكرنا من قبل هو معمارية معينة للذاكرة (تقسيم واستخدام معين تستخدمه أوراكل في إدارة الذاكرة) يتم فيه عمليات تهدف إلى ضمان الوصول وإدارة الملفات المادية المكونة لقاعدة البيانات. وهنا سنتكلم عن مجموعة الملفات التي تشكل الجزء المادي لقاعدة البيانات وتتحكم في فتح المثال نفسه. حيث تستخدم ملفات الداتا Data Files في التخزين المادي للجداول والفهارس بما تحتويه من سجلات منظمة كما يمكنها أيضاً تخزين المناظير Views والإجراءات procedures.. الخ ويتولى كاتب البلوك DBWR. وهو عملية تتم في الخلفية كما سنرى بعد نقل المعاملات التي تم تأكيدها من مكانها المؤقت بالذاكرة إلى الملفات المادية لقاعدة البيانات.

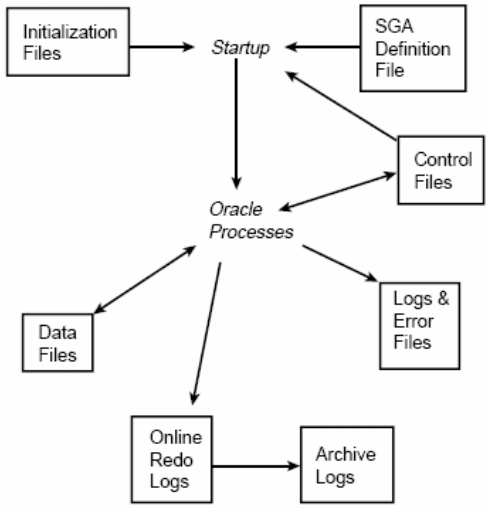
تحوى ملفات التراجع والإعادة على سجلات مسجل عليها معلومات كافية عن أية معاملة Transaction تتم على الداتابيز والمنطق في ذلك بسيط وهو تسجيل تلك المعاملات بصورة مستقلة عن الملفات الأساسية بحيث لو حدث تلف في تلك الملفات يمكن استرجاع التغيرات التي حدثت من ملفات الـ Redo Log يتولى كاتب اللوغ LGWR وهو عملية تتم في الخلفية كتابة التغيرات التي حدثت حيث يقوم بنقل السجلات المعبرة عن تلك التغيرات من الذاكرة الكاش والتي تعينها أوراكل لمعلومات الإعادة (الوج Redo) وتسمى Redo Log Buffer إلى ملفات اللوغ Redo Log Files بصورة مستمرة ويوجد عادةً مجموعتين من تلك الملفات ويبدأ الكتابة في أحد الملفات فإذا امتلئ تماماً يتم التحويل Switching إلى الملف الآخر وهكذا فإذا امتلئ الملف الآخر يتم التبديل وإعادة الكتابة على الملف الأول بطريقة دائرية والسؤال هنا ماذا لو حدث تلف بتلك الملفات؟ أوراكل تتيح وسيلة حماية أيضاً عن طريق أرشفة تلك الملفات قبل إعادة الكتابة عليها حيث تقوم عملية ARCH

بعمل نسخ احتياطية لملفات الريدو لوغ ولا تسمح بإعادة الكتابة على أية من تلك الملفات إلا بعد إتمام أرشفة الملف.

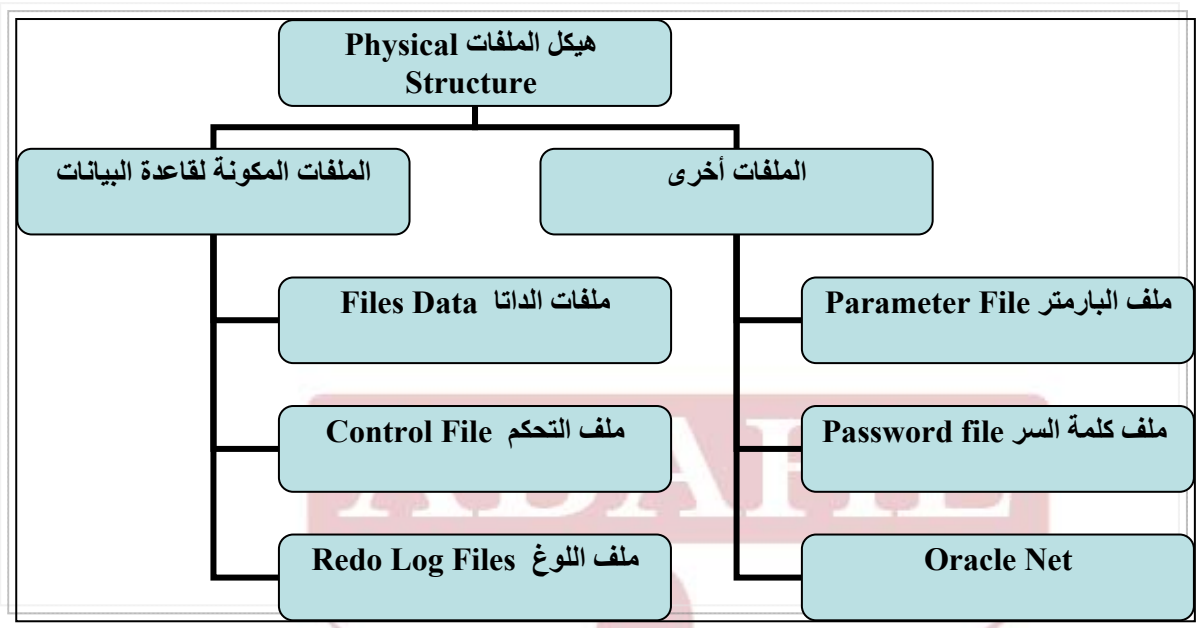
تعادل ملفات البارمتر ملفات الأوامر في نظام الدوس وهى ملفات تسمح بإعطاء قيم تحدد الطريقة التي سيعمل بها مثال الأوركل بينما يلزم ملف التحكم Control File لفتح قاعدة البيانات والتعامل معها. وتقوم أوراكل بتسجيل الأخطاء التي تحدث أثناء التشغيل وكذلك معلومات التشغيل بصورة تفصيلية في ملفات Log & Alert . Files والشكل التالي يوضح الهيكل المادي لأوراكل وعلاقته بالعمليات.

والشكل التالي يوضح أنواع الملفات المختلفة التي تشكل قاعدة البيانات:

Figure 6.4.
The basic types of Oracle data files.



حيث يمكن تصور هيكل الملفات المادي لأوراكل على النحو التالي:



- **ملفات البارمتر Parameter files**

يتحكم ملف البارمتر في فتح المثال حيث يعطى التعريفات الخاصة بـ SGA وتعيين مساحتها كما يتضمن اسم ملف التحكم الخاص بقاعدة البيانات وأيضاً في وضع تعريفات العمليات الخاصة بأوراكل في الخلفية. ويوجد نوعين من الملفات الأول يسمى ملف البارمتر PFILE وملف SPFILE (Server Paramater file).

يمكن تحديد اكثر من 250 قيمة في كلاً من الملفين وفي أوراكل g10 تقسم القيم إلى نوعين أولهما أساسي ويحتوى على 30 قيمة ينبغي تحديدها وبقية القيم للاختيارات المتقدمة.

- **ملف التحكم Control File**

الملف الرئيسي للتحكم في فتح قاعدة البيانات حيث يحدد ذلك الملف مواضع الملفات المادية لقاعدة البيانات واسم قاعدة البيانات ولغة قاعدة البيانات وحجم البلوك المستخدم في تخزين الملفات بالإضافة إلى معلومات لا غنى عنها لعمليات الإصلاح وإعادة استرجاع البيانات في حالة فشل قاعدة البيانات. وتعتبر ملفات التحكم من الأجزاء التي لا يمكن الاستغناء في قاعدة البيانات حيث تقوم بتخزين معلومات حيوية عن الداتابيز ومنها :

- اسم قاعدة البيانات
- أسماء ومواقع وحجم ملفات الداتا واللوغ
- معلومات لاستعادة وإصلاح قاعدة البيانات في حالة وقوع أخطاء

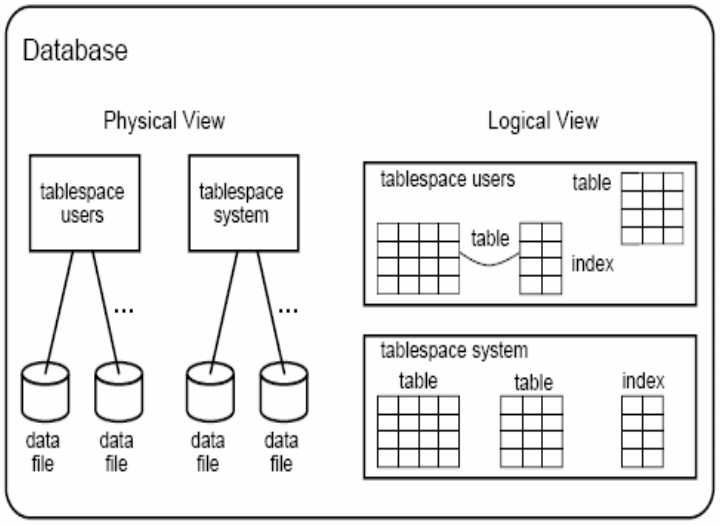
ملفات التحكم يتم إنشاءها أثناء عمل قاعدة البيانات وفى المكان الذي يحدده البارمتر Control Files في ملف البارمتر. ومن أجل أهمية الملف يراعى وجود ما يسمى ازدواج ملف التحكم Multiplex حيث يتم إنشاء نسخ من ملفات التحكم في أماكن مختلفة ويتم تحديثها أتوماتيكيا بآخر التعديلات والذي تقوم به خصيصاً عملية بأوراكل تتم في الخلقية تسمى CHKP. يمكن من خلال View الاستعلام عن ملفات التحكم وهي

```
SQL> select name from v$controlfile;
```

- **ملفات الداتا المكونة لقاعدة البيانات Data Files**

تعتبر ملفات الداتا هي أهم مكون من مكونات قاعدة البيانات حيث تحوى جميع الجداول والفهارس والتي تخزن بها البيانات على هيئة صفوف. تلك الملفات لا يمكن قراءتها مباشرة بواسطة أية وسيلة من محررات النصوص العادية وإنما يتم التعامل معها بواسطة أوراكل عن طريق تنفيذ جمل استعلام معينة. إن تلك المعلومة هامة للغاية فإننا لا نملك السيطرة على كيفية تخزين الداتا في الملفات المادية فهذا دور أوراكل ولكننا فقط نستطيع التعامل مع البيانات من خلال تنفيذ جمل SQL على هيكل افتراضي مكون من المساحات الجدولية TableSpaces والتي تنشأ فيها جداول Tables وفهارس Indexes كما يتضح من الشكل التالي:

Figure 8.3.
Physical versus logical data structures.



يتضح من الشكل السابق ما يلي:

- الهيكل المادي للملفات مكون من عدة ملفات Data Files وهى ملفات توجد على نظام التشغيل تشكل الوعاء المادي للبيانات.
- لا يمكن إدارة تلك الملفات بصورة مباشرة بل تتم من خلال أوراكل حيث تنشئ هيكل افتراضي للملفات مكوناً بصفة أساسية من مساحات جدولية TableSpaces.
- المساحات الجدولية تحوى على الجداول والفهارس والمناظير Views.
- من خلال تلك الجداول والمناظير وبواسطة تنفيذ جمل SQL يتم تنظيم البيانات الفعلية وإدارتها.
- كل مساحة جدولية تنظم واحد أو اكثر من الملفات المادية DataFiles.
- كل ملف داتا DataFile يرتبط بمساحة جدولية واحدة فقط بينما المساحات الجدولية نفسها يمكن أن تكون مرتبطة بأكثر من ملف داتا.
- لاحظ أننا لا يمكنا السيطرة على مكان وضع الجداول أو الفهارس على ملفات الداتا المادية DataFiles.

أما عن ملفات الداتا نفسها فأهم ما يمكن أن يقال عنها هو أن كل ملف داتا يخزن به رقم مولد بواسطة أوراكل System Change Number (SCN) يدل على آخر عملية تغيير حدثت وتمت على مستوى قاعدة البيانات وهذا الرقم له أهميته الكبيرة فعن طريقه يمكن لأوراكل أن تعرف أية تغييرات ينبغي تأكيدها وتخزينها على الملفات وبالتالي فان ذلك الرقم لا يمكن الاستغناء عنه في حالة النسخ الاحتياطي للملفات وإصلاح واستعادة قاعدة البيانات في حالة حدوث أخطاء.

بقي أن نعرف أن ملفات الداتا يتم تحديدها وحجزها على نظام التشغيل بالحجم الكلى والذي تم تحديده في أمر إنشاءها فلو حددنا مثلاً أن حجم المساحة الجدولية Tablespace سيكون M100 فانه سيتم إنشاء وحجز ملف داتا على نظام التشغيل بنفس القيمة حتى ولو كان الملف لا يوجد به أية بيانات بعد.

- **ملفات الريدو لوغ Redo Log Files**

تخزن فيها أية تغيرات تحدث لقاعدة البيانات فمثلاً إدخال معاملة جديدة New transaction أو التعديل فيها أو إلغاءها وهذه الملفات أساسية في حالة فشل الوصول إلى قاعدة البيانات فتستخدم تلك الملفات في الإصلاح وإعادة الوضع إلى ما كان عليه.

- **ملف التحكم في كلمة السر Password File**

يستخدم لتسجيل أسماء المستخدمين والذين لهم صلاحيات SYSDBA و SYSOPR لإدارة لقاعدة البيانات.

وحدات تخزين البيانات في أوراكل

أن أوراكل تستخدم وحدات خاصة افتراضية بها Logical لحجز وتخصيص مساحات التخزين على نظام التشغيل كالتالي:

القطاع :Segment

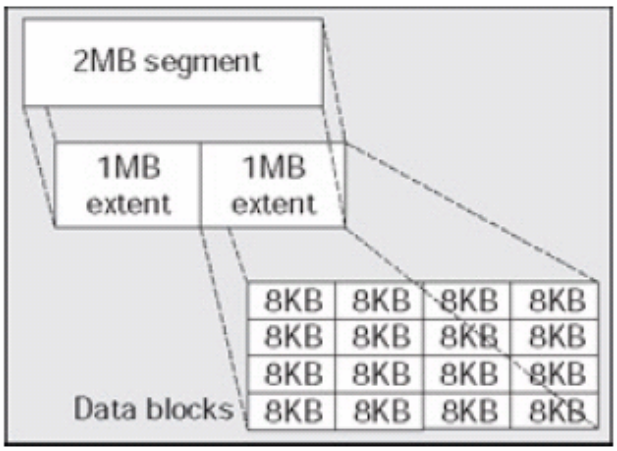
يمكن تمثيل القطاع Segment بالوعاء الذي يشغل حيزاً معيناً لتخزين البيانات. وكما أننا يمكننا نخزن ماءاً في وعاء معين فيكون وعاء يحتوى على الماء وربما احتجنا آخر لتخزين سائلاً آخر فيه فيكون أيضاً وعاءاً لتخزين الزيت مثلاً كذلك كل كائن يتطلب تخزينه في قاعدة البيانات الوعاء الخاص به فالجداول تكون مكونة من العديد من الامتدادات القطاعية فيكون لدينا ما يسمى Table Sgment والفهارس تحتاج إلى امتدادات مختلفة أخرى فتمسى امتدادها Index Sgments ومعلومات التراجع تسجل على امتدادات تسمى Rollback Sgments وهكذا.

الامتداد Extents

القطاعات نفسها مؤلفة من امتدادات Extents وهى عبارة عن مساحات تخزينية متجاورة. أن كل قطاع يتألف على الأقل من واحد امتداد وبعض القطاعات تتطلب أن يكون هناك أكثر من امتداد مثل قطاع التراجع Segment rollback. الامتدادات قد تتفاوت في الحجم من 1 بلوك وحتى 2 جيجا .

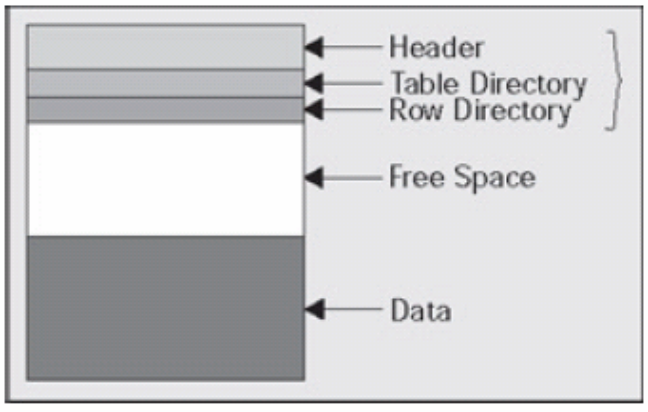
البلوك Block

الامتدادات تتكون من بلوكات Bloack وهى أصغر وحدة تخزينية يمكن حجزها وتخصيصها في أوراكل. أن البلوكات هي ما سوف تخزن به أوراكل الداتا فعلياً Physical في ملفات على نظام التشغيل. أن البلوك هو أصغر وحدة يمكن قراءتها أو كتابتها من الديسك ويلاحظ أن البلوك في أوراكل يختلف عن البلوك في نظام التشغيل مثلاً فهو يتراوح بين (2او 4 أو 8 أو حتى 16 كيلو بايت أو 32 كيلو بايت)



- كما يعبر الشكل فان القطاع يكون من واحد أو اكثر من امتداد Extents و يتشكل كل امتداد من بلوكات متجاورة.

- كل بلوك في الداتابيز يكون حجما ثابتا كما أن البلوكات لها نفس التكوين العام حيث يتكون البلوك كما يوضح الشكل التالي:



- **رأس البلوك Bloch Header**

o يحتوى رأس البلوك Header على معلومات توضح نوع البلوك سواء كان بلوك جداول Table Block أو بلوك فهرس Index Block.

- معلومات عن المعاملة الحالية Active transaction والتي تتم على البلوك أخر واحدة تمت على ذلك البلوك.
- معلومات عن عنوان البلوك على الديسك.

- **دليل الجداول Table Directory**

- يحتوى الدليل إن وجد على معلومات عن الجداول والتي تخزن صفوف في ذلك البلوك أو مجموعة الجداول إذا كانت أكثر من جداول تخزن على صفوف على هذا البلوك.

- **دليل الصفوف Table Directory**

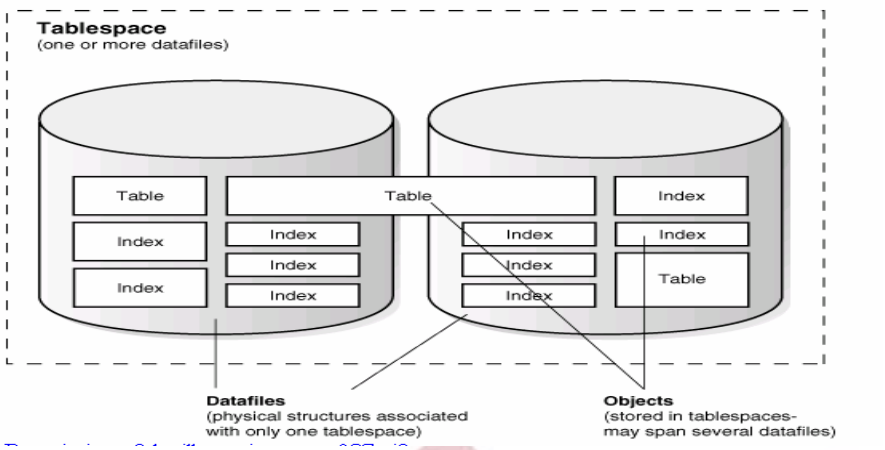
- يحتوى على معلومات تشير إلى أماكن وجود الصفوف في البلوك عن الصفوف الموجودة في البلوك.

تسمى العناصر السابقة بسقف البلوك Block Overhead وهى مساحة من البلوك مخصصة فقط لكي تستخدمها أوراكل في إدارة البلوك نفسه والباقي من مساحة البلوك أما أن يحتوى على الداتا نفسها أو جزء فارغ يمكن ملئه .

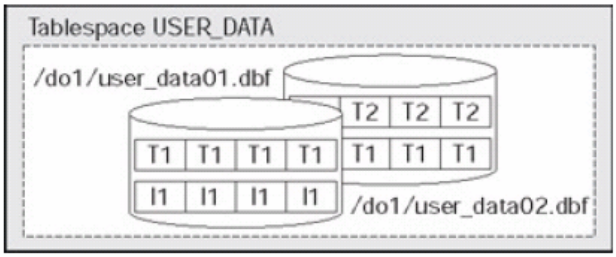
المساحة الجدولية TableSpace :

- يمكن تشبيهها بوعاء كبير حاوي لجميع القطاعات فكل قطاع Segment يجب أن ينتمي إلى مساحة جدولية خاصة به التي تحوى الامتدادات Extents والتي تحوى البلوكات Block.
- كل مساحة جدولية على المستوى المادي Phiscal مكونة من واحد أو أكثر من ملفات الداتا Data Files المرتبطة بها. الشكل التالي يوضح العلاقة بين المساحات الجدولية وملفات الداتا وأنواع الكائنات التي تحتويها المساحة الجدولية.

Figure 3-1 Datafiles and Tablespaces



من الشكل التالي نتعلم بعض الحقائق الهامة:



- من الشكل يتضح لنا وجود مساحة جدولية TableSpace تسمى **USER_DATA** تتكون من الناحية المادية من عدد (2) ملف داتا Data Files وهما (user_data01.dbf و user_data02.db).
- المساحة الجدولية السابقة تحتوى على ثلاثة قطاعات وهى (T1, T2,I1) من المحتمل انهما يكونان عدد (2) جداول وهما T1, T2 وعدد (1) فهرس وهو المعبر عنه بـ I1
- المساحة الجدولية مخصص لها عدد (4) امتدادات Extents وكل امتداد عبارة عن مجموعة متجاورة من البلوكات.

- القطاع Segment (T1)يتكون من عدد (2) امتداد بحيث يكون كل امتداد Extent يوجد مادياً على ملف داتا خاص به.
- القطاع (T2, I1) كلا منهما مكون من امتداد وحيد.
- بفرض أننا نحتاج إلى زيادة مساحة إضافية فيكون أمامنا أحد أمرين :
  - تكبير ملفات الداتا الموجودين بالفعل.
  - إضافة ملف داتا جديد إلى المساحة الجدولية.

ABAHE

إدارة أروا كل للمساحات الجدولية:

قبل أوراكل 8.1.5 لم يكن هناك أسلوبا لإدارة الامتدادات داخل المساحة الجدولية إلا بما يعرف الإدارة عن طريق الكاتلوغ **dictionary-managed tablespace** ويمكننا تصور تلك الطريقة كما في إدارة حساب لك في البنك حيث يحتوى الحساب على جانبين جانب مدين وجانب دائن كما في الشكل التالي:

الجانب المدين	الجانب الدائن
يتم وضع جميع الامتدادات Extents المخصصة للكائن كجداول مثلاً	جميع الامتدادات المتاحة من النظام

عندما يحتاج الكائن لمساحة إضافية يتم طلب ذلك من أوراكل والتي تصدر أمر بحث عن المساحات المتاحة والتي يمكن إعطاءها للكائن وذلك عن طريق تنفيذ عدة جمل SQL على الكاتلوغ Dictionary tables وفى هذه الحالة تعدل الجداول الخاصة بالكتالوغ بنتائج البحث عن المساحات الخالية وربما تضاف أو تلغى أو تحدث صفوف إلى بعض الجداول الموجودة في الكتالوغ الخاص بأوراكل. يؤدى أداء ذلك بصفة مستمرة إلى وجود عبء على النظام مما كان يؤثر تأثيراً كبيراً على اعتبارات الكفاءة . في الإصدار 7.3 من أوراكل قدمت أوراكل أول مرة مفهوم

المساحة الجدولية المؤقتة Temporary TableSpace والتي لا يمكن عمل أية كائنات بصورة دائمة عليها وهنا خصصت أوراكل كل الامتدادات Extents المتاحة إلى تلك المساحة المؤقتة وعندما يطلب أي كائن مساحة إضافية تقوم أروا كل بتنفيذ جمل في الكاتلوغ ويتم البحث عن المساحات الخالية وتضاف إلى تلك المساحة الجدولية المؤقتة وتظل فيها فإذا ما احتاج الكائن أو غيره إلى مساحة إضافية تبدأ أوراكل بالبحث في الذاكرة عن مساحات خالية من الامتدادات في الذاكرة فإذا وجدتها تقوم باستغلالها وإذا لم تجدها فإنها ترجع لاستخدام الأسلوب القديم. ويعيب الأسلوب السابق بان المساحة الجدولية المؤقتة سرعان ما تشغل وبالتالي تؤثر على أداء النظام.

قدمت أوراكل مفهوم جديد ابتداء من الإصدار 8.1.5 وهو الإدارة المحلية locally managed tablespace في مقابل إدارة المساحات الجدولية عن طريق الكاتلوغ dictionary managed السابق فيقوم الأسلوب الجديد على أن المساحة الجدولية TableSpace تقوم بإدارة الامتدادات extents الخاصة بها حيث يتم الاحتفاظ بقيمة Bitmap على كل ملف داتا لتدل على حالة البلوكات المخزنة على تلك الملفات هل هي خالية أم شاغرة فإذا كانت خالية فانه يتم استغلالها مرة أخرى. أن أوراكل تقوم باستمرار بتحديث تلك القيمة دون الاستعانة بالكتالوج.

ملفات التراجع أو الإعادة Redo Log Files

تعتبر تلك الملفات حيوية بالنسبة لقاعدة البيانات فهي تحوي سجلات العمليات التي تمت على قاعدة البيانات ولا تستخدم تلك الملفات إلا في عمليات إصلاح واستعادة أية أخطاء في قاعدة البيانات فعلى سبيل المثال لو حدث انقطاع التيار الكهربائي عن الجهاز الموجود به السيرفر فان ذلك سيسبب في الأغلب في حدوث أخطاء في المثال Instance Failure وهنا تستخدم أوراكل ملف التراجع OnLine Redo Log لإعادة استرجاع النظام إلى النقطة التي كانت قبل انقطاع التيار الكهربائي. وإذا حدث على سبيل المثال أن قمنا بتأكيد إلغاء جداول فانه يمكن مع استخدام

نسخة احتياطية لقاعدة البيانات بالإضافة إلى ملفات التراجع أن نعيد ما فقد من بيانات.

كل عملية تقريباً تتم في أوراكل يتم تسجيلها في ملفات الريدو ONLINE Redo فعلى سبيل المثال عندما ندخل صفاً جديداً أو عندما نلغي صفاً فإن ذلك يسجل في ملفات الريدو. يوجد نوعين من ملفات الريدو وهما

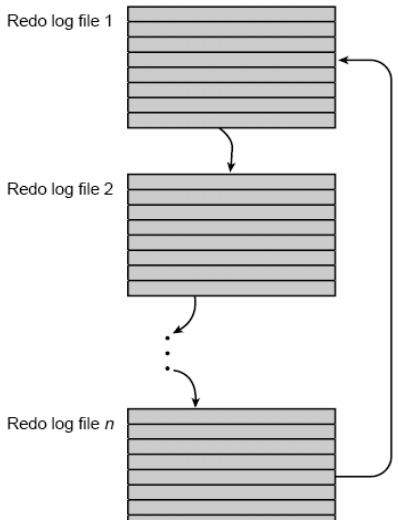
- **ملفات التراجع Online Redo Log**

تحتوى قاعدة بيانات أروا كل على اثنين من ملفات الريدو على الأقل وهي ملفات ذات حجم ثابت ويتم التسجيل عليها بطريقة دائرية بمعنى أن قاعدة البيانات سوف تسجل في الملف الأول حتى امتلائه ثم يتم التحويل إلى الملف الثاني وفى حالة امتلائه يتم إعادة الكتابة على الملف الأول وهكذا. كما يتضح من الشكل التالي:

Warning

Carefully monitor the file system where the archive log files are being written to prevent locking up Oracle's capability of accepting transactions.

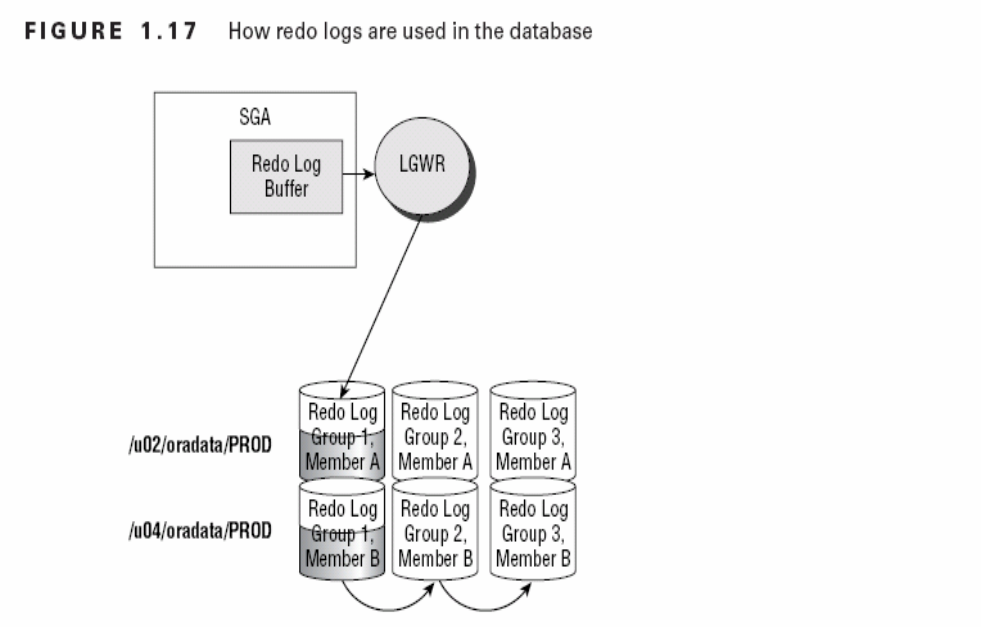
Figure 8.4. Online redo log file recycling.



تسمى عملية الانتقال من اللوغ إلى اللوغ الآخر بالتبديل أو التحويل Switch 0Log ولفهم كيفية عمل ملفات الريدو فإنه يتعين علينا فهم عملية التأكد Check pointing (CHKP) وكاتب البلوك Database Block Writer (DBWn).

تبدأ قصتنا بجزء معين من الذاكرة الكاش تسمى الذاكرة المؤقتة لتخزين بلوكات الداتابيز Database buffer cache والتي تمثل منطقة معينة من الذاكرة تستخدمها أوراكل في تخزين بلوكات قاعدة البيانات المراد التعامل معها حيث يتم استدعاء بلوكات الداتا من الملفات المادية ليتم التعامل معها في تلك المنطقة من الذاكرة سواء بالقراءة أو التعديل. ويتم بالتوازي تسجيل معلومات كافية عن تلك التعديلات في منطقة أخرى من الذاكرة SGA تسمى الذاكرة المؤقتة للريدو لوغ redo log buffe وعندما تتم التعديلات ويراد تأكيدها وذلك بإصدار الأمر Commit فان أوراكل لا تبدأ في الحال بكتابة البلوكات المعدلة على الملفات المادية Data files مرة أخرى بل تنتظر قليلا حيث يتم أولا نقل المعلومات الخاصة بالتعديلات من ذاكرة الريدو Redo Log Buffer إلى ملف الريدو ONLine Redo Log بواسطة عملية لأوراكل تتم في الخلفية تسمى كاتب اللوغ LGRW كما يوضحه الشكل التالي

FIGURE 1.17 How redo logs are used in the database



وبعد إتمام تلك العملية بنجاح يتم نقل التعديلات من الذاكرة المؤقتة لتخزين بلوكات الداتابيز Database buffer cache إلى ملفات الداتا حيث تتولى مسؤولية ذلك عملية أخرى تعمل في الخلفية تسمى DBWR.

والمنطق في هذا التصرف بسيط هو تجنب الآثار الناتجة عن فشل المثال فجأة كحالة انقطاع التيار الكهربائي عن السيرفر فبوجود ملف الريدو فإنه في تلك الحالة يتم الاستعانة بمعلومات التراجع الموجودة به حيث ستقوم أوراكل بشكل تلقائي بإرجاع المعاملات Transactions التي لم تتم بشكل صحيح بالاستعانة بملفات الريدو ثم تقوم مرة أخرى بمحاولة التعديل في البلوكات ثم تتم العملية بنقل تلك التعديلات إلى الملفات المادية فتتم بذلك تأكيد التعديلات . ولكن ما العمل في حالة امتلاء ذاكرة الكاش Database buffer cache بالبلوكات المعدلة وتسمى Dirty Buffer. هنا يأتي دور لعملية هامة للغاية تتم في الخلفية بدون أن يشعر المستخدم بوجودها وهى عملية التحقق (Checkpoints (CHKP والتي تتأكد من مدى امتلاء ذاكرة الكاش فإذا وصلت إلى حد معين وجب تفريغ تلك الذاكرة لاستقبال بلوكات جديدة ويتم تفريغ ذاكرة الكاش بنقل البلوكات المعدلة إلى الملفات المادية وهو دور كاتب البلوك (DBWR) ثم بعد إتمام النقل تقوم عملية التحقق CHKP بإتمام تفريغ الذاكرة. يوجد حدث Event يؤدى إلى تنشيط عملية التحقق وهو عادةً ما يتحقق عندما يحدث تبديل لملف اللوغ. وحتى يتم كاتب البلوك عمله في إعادة البلوكات المعدلة مرة أخرى إلى الملفات المادية تكون البلوكات محمية من إعادة الاستخدام بواسطة معلومات التراجع المسجلة في ملف الريدو لوغ فإذا ما حاولنا إعادة استخدامها فان رسالة تظهر كالتالي:

```
...
Thread 1 cannot allocate new log, sequence 66
Checkpoint not complete
 Current log# 2 seq# 65 mem# 0: C:\ORACLE\ORADATA\TKYTE816\REDO02.LOG
...
```

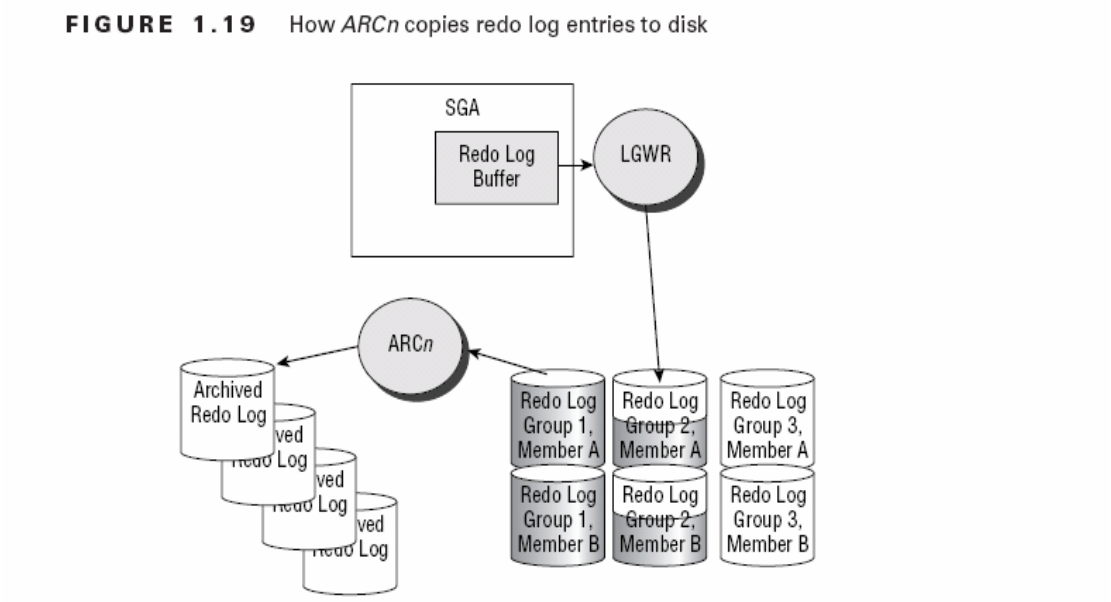
وهذا يعنى أن كاتب اللوغ لم ينته من عمله بعد وقد امتلئ ملف اللوغ فلا يستطيع استقبال سجلات جديدة وهنا يكون على مدير قاعدة البيانات أن يزيد بقدر المستطاع من سعة ملفات الريدو لوج.

ملفات اللوغ الاحتياطية Archived Redo Log:

يمكن أن تعمل قاعدة البيانات بإحدى أسلوبين وهما إما عدم السماح بتكوين نسخ احتياطية على الديسك من ملفات الإعادة والتراجع Online Redo Log ويسمى ذلك الوضع بـ NOARCHIVELOG أو بالسماح بتخزين وأرشفة الملفات السابقة قبل إعادة الكتابة عليها من جديد ARCHIVELOG وهو الوضع المحبذ لتخفيض احتمالات فقد البيانات.

إن أوراكل لا يمكنها السماح بتكبير حجم ملف اللوغ Online Redo إلا ما لا نهاية فلابد وان تعيد الكتابة عليه بعد امتلائه وقلنا مما سبق أن ملفات اللوغ لا غنى عنها في حالة الإصلاح فما العمل في حالة تلف ملف اللوغ نفسه. إذا لم يكن ملف اللوغ موجوداً فانه لا مفر سوف تفقد الكثير من الداتا ولذلك تعمل أروا كل في حالة Archivelog أن تكون نسخ احتياطية من ملفات التراجع وتحرص على تجديدها باستمرار تحسباً لتلف ملفات اللوغ Online Redo log كما يتضح من الشكل التالي:

FIGURE 1.19 How *ARCn* copies redo log entries to disk



Processes العمليات

تمثل العملية سلسلة من الأوامر أو الخطوات التي يتم تنفيذها بصورة متابعة وتحجز مكاناً خاصاً بها في الذاكرة حيث يتم تنفيذها وفى بيئة العمل التي تتميز بالعديد من المستخدمين يعلمون في نفس الوقت يتم تنفيذ العديد من العمليات في نفس الوقت بصورة متوازية (كل منها يعمل بصورة مستقلة عن الأخرى وتقريباً في نفس الزمن). يمكن بصورة عامة وضع تصور عام لما تقوم به أوراكل من مهام وعمليات رئيسية على النحو التالي:

- عمليات تهدف إلى خدمة طلبات المستخدم Processes servicing user requests
- عمليات تقوم بكتابة البيانات على ملفات الداتابيز Processes writing data to the data files
- عمليات تقوم بتسجيل أية أحداث أو تغييرات تحدث Processes recording transactions in log files
- عمليات تقوم بمراقبة أداء قاعدة البيانات ومحاولة اكتشاف الأخطاء وعلاجها Processes monitoring the functioning of the database

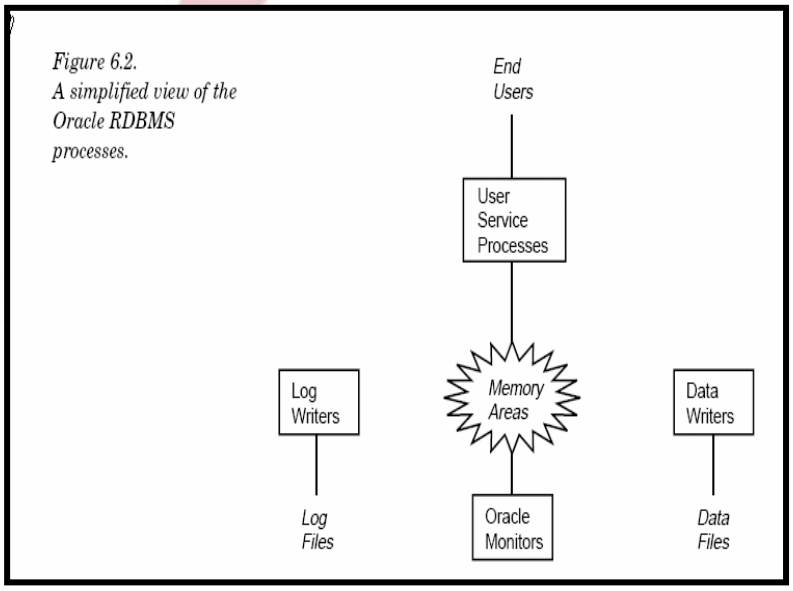


Figure 6.2. A simplified view of the Oracle RDBMS processes.

تقـــــوم

المجموعة الأولى من العمليات تقوم بتلبية طلبات المستخدم حيث يمكن النظر إليها كرابطة أو حبل يشدك إلى أوراكل فعندما تحتاج إلى معلومات من قاعدة البيانات سوف تأمر تلك المجموعة من العمليات كي تقوم بربطك بقاعدة البيانات ثم تحضر ما طلبته من معلومات وإذا ما قمت بتعديلات أو تحديثات فإنها تأمر باستدعاء عمليات أخرى لتسجل تلك التعديلات في مناطق خاصة في الذاكرة تمهيداً لنقل تلك التحديثات فيما بعد إلى الملفات الخاصة بالداتابيز.

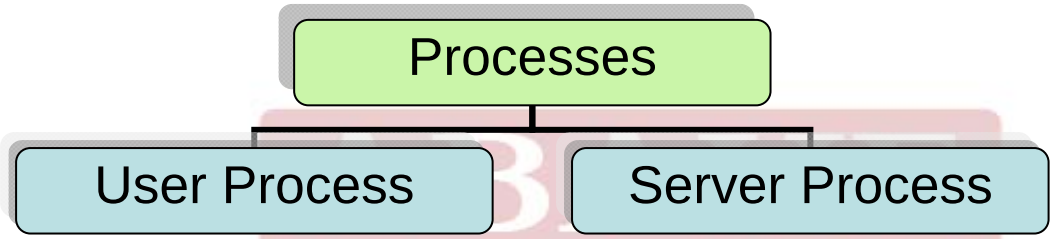
أما **المجموعة الثانية** من تلك العلميات فهي تعنى بوضع وكتابة البيانات على الملفات المكونة للداتابيز Data Files فكما قلنا من قبل فان المعلومات تخزن مبدئياً في مناطق تحددها الأوركل في الذاكرة Shared Memory ثم تنقل بواسطة تلك العمليات بعد وقت معين إلى الملفات ويتم إخلاء المنطقة لتوفير قدر اكبر من مساحة الذاكرة.

وتختص **المجموعة الثالثة** بكتابة ما يحدث على البيانات من تعديلات في ملفات اللوغ حيث تسمح تلك الملفات باستعادة آخر ما تم على البيانات من تعديلات في حالة فشل الوصول إلى الداتابيز والذي يحدث نتيجة العديد من الأسباب ومنها تلف وحدات التخزين.

وعليه فإن أوراكل تسجل كل معاملة Transaction (أية إدخال جديد لبيانات أو التعديل في بيانات موجودة سواء بالإضافة أو بالإلغاء) في ملفات اللوغ Log Files وهى ملفات مستقلة عن ملفات الداتا وفى حالة فقدان أية من ملفات الداتا المكونة للداتابيز أو فشل التعامل معه فيمكن استدعاء ملف اللوغ للمساعدة في عمليات الإصلاح حيث يمكن استدعاء نسخة احتياطية من ملف الداتا وتطبيق ما هو موجود في ملف اللوغ من آخر معاملات حتى يمكن استعادة الوضع إلى ما هو عليه.

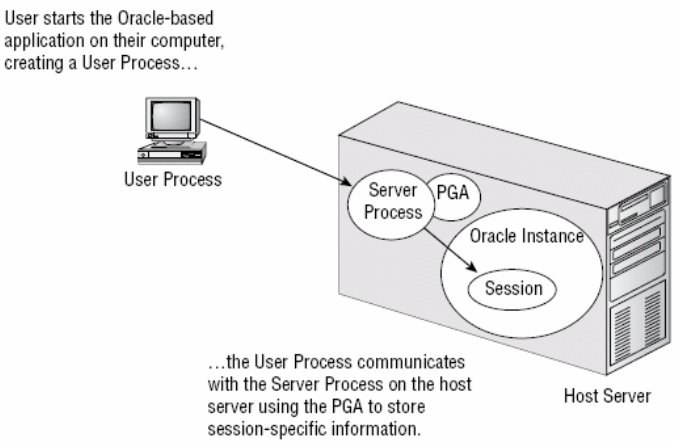
وتختص آخر مجموعة من العمليات بمراقبة أداء الداتابيز.

تُقسم العمليات التي تقوم بها أوراكل إلى نوعين أساسين من وجهة المستخدم الذي يريد الاتصال بأوراكل:



وإذا نظرنا إلى المستخدم الذي يريد الاتصال بقاعدة البيانات نجد أن هناك نوعين من العمليات مخصصين لضمان اتصال المستخدم بمثال الأوركل وبالتالي للاتصال بقاعدة البيانات والتعامل معها. يقوم المستخدم بتشغيل تطبيق وليكن مثلاً برنامج لشؤون الأفراد والذي يتطلب الحصول على داتا من قاعدة البيانات فتقوم الأوركل بتشغيل عمليات خاصة لخدمة هذا المستخدم User Process حيث تقوم تلك العلميات بعمل قناة اتصال تسمح بربط المستخدم بمثال الأوركل وعندما يتم فتح تلك القناة تقوم أوراكل بتشغيل عمليات إضافية Server Process على سيرفر الأوركل نفسه والتي تكون مسؤولة فعلياً عن تعامل المستخدم مع قاعدة البيانات ويتضح ذلك من الرسم التالي:

FIGURE 1.12 The relationship between User and Server processes



كما يتضح من الشكل السابق الآتي:

- المستخدم يبدأ في تشغيل تطبيق يحتاج إلى قاعدة البيانات فتبدأ عمليات المستخدم في التشغيل User Process.
- تقوم عمليات المستخدم User Process بالاتصال بعمليات السيرفر على السيرفر لفتح قناة اتصال وتسجل منطقة خاصة في الذاكرة PGA بيانات الاتصال ومعلومات المستخدم.
- بمجرد تأكيد الاتصال يتم التعامل المباشر بين المستخدم وقاعدة البيانات عن طريق عمليات السيرفر Server Process والتي تتعامل مع المثال Oracle Instance و تكون في خدمته.

العمليات التي تعمل بهدوء في الخلفية:

Oracle Processes In Background

تقوم أوراكل بتشغيل مجموعة من العمليات في الخلفية وذلك لخدمة كل مثال. تؤدى تلك العمليات وظائف هامة للغاية فهي تتولى إرسال البيانات من الذاكرة إلى ملفات الداتابيز والعكس I/o كما تراقب أداء الوظائف والعمليات الأخرى من أجل تحسين الأداء ويوجد منها خمسة عمليات لا غنى عنها والباقي يعتمد على خيارات إعداد أوراكل للعمل وهي:

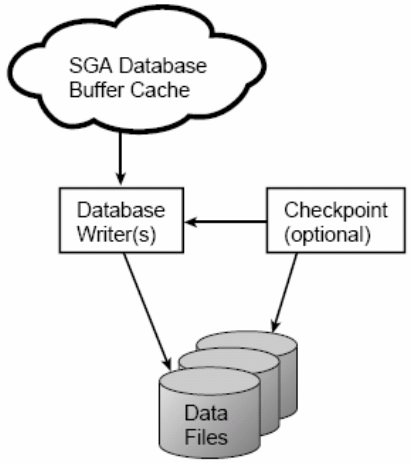
1. وظيفة كتابة الداتا إلى الملفات:

يوجد نوعين رئيسين من العمليات التي تختص بتلك الوظيفة الهامة أولهما:

- كاتب الداتا Database Writer (DBWRn)

حيث تقوم تلك العملية بكتابة بلوكات البيانات المعدلة من الذاكرة إلى الملفات المكونة للداتابيز كما يظهر في الشكل التالي والذي يوضح انتقال البيانات من الذاكرة المؤقتة Database Buffer cache إلى الملفات المادية المكونة لقاعدة البيانات Data files0 ويلاحظ أنه بالنسبة للمستخدم يمكن أن يكون هناك اكثر من كاتب يعمل في الخلفية لتفادي الإختناقات التي ممكن أن تحدث أثناء انتقال البيانات.

Figure 9.2.
Processes that write to the data files.

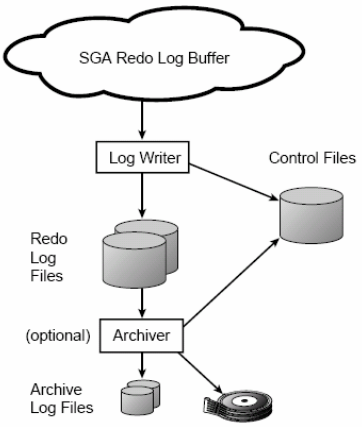


- وظيفة التحقق Checkpoint (CKPT)

تقوم تلك الوظيفة بتحديث ملف التحكم Control File ورأس ملفات الداتا Data files والمكونة لقاعدة البيانات برقم يسمى System Change Number (SCN) وهو رقم يتم توليده من النظام ليدل على آخر معاملة تمت بنجاح على قاعدة البيانات. إن هذه الوظيفة تُفَعل تلقائياً بناءً على حدث يسمى التحقق Checkpoint والذي يحدث كلما حدث تبديل للكتابة بين ملف لوغ إلى ملف لوغ آخر.

- كاتب اللوغ Log Writer (LGWR)/

Figure 9.3.
The Oracle log writing processes.

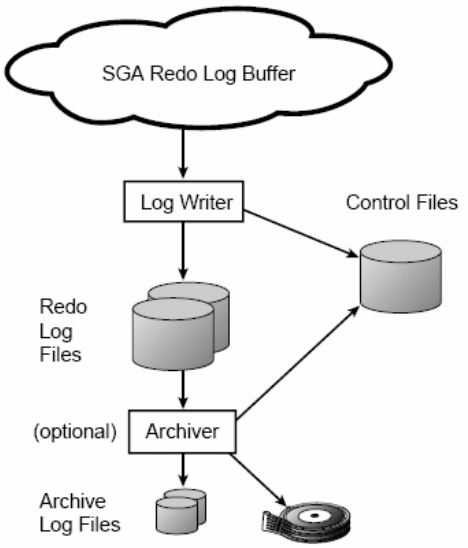


وهو المسئول عن نقل المعاملات Transactions التي حدثت من المنطقة المؤقتة بالذاكرة الخاصة بتسجيل التغيرات التي تمت Log Buffer إلى ملفات اللوغ ويتبع ذلك انه معني بتحديد أي من ملفات اللوغ المستعدة لاستقبال تلك التغيرات كما يكون مسؤولا عن التأكد من نقل نسخ التعديلات من ملف اللوغ إلى ملفات الأرشيف قبل إعادة الكتابة على ملف اللوغ من جديد. وإذا لم تكن هناك عمليات CHKP إضافية لتوليد آخر رقم تغيير حدث SCN فان كاتب اللوغ في هذه الحالة يحمل بجهد إضافي في توليد الرقم وتعديل رأس ملفات الداتا.

- **كاتب الأرشيف (ARCH)**

مهمته الأساسية تتركز في عمل نسخ احتياطية من ملفات اللوغ إلى ملفات الأرشيف والتي تحفظ على شريط أو ديسك آخر. تبدأ تلك الوظيفة عندما تكون الداتابيز في وضعية الأرشيف وذلك عن طريق الأمر Alter log System Archive أو أتوماتيكياً عندما يتم فتح مثال الأوركل بواسطة ملف البارمتر والذي يكون موجوداً فيه البارمتر log_archive_start = true

Figure 9.3.
The Oracle log writing processes.



2. وظائف الرقابة ومتابعة الأداء Monitoring Processes:

توجد ثلاثة عمليات رئيسية في تلك المجموعة فآما الأولى فهي (SMON) ودورها تحرير مثال الأوركل من أية اختناقات وأخطاء أثناء بدء تشغيل المثال أو في أي وقت يتطلب ذلك أما الوظيفة الثانية (PMON) فهي الوظيفة التي دورها تحرير ومسح ما ينجم عن فشل أية عملية من عمليات المستخدم

- **مراقبة النظام System Monitor (SMON)**

وظيفة العملية مراقبة مثال الأوركل والتعامل مع أية أخطاء تحدث نتيجة فتح المثال فهي على تقوم على سبيل المثال بالتأكد من فتح المثال قد تم بصورة طبيعية وإذا لم يكن تحاول إجراء إصلاح واستعادة الوضع عما كان عليه قبل حدوث الخطأ فإذا لم يكن المثال قد تم إغلاقه بصورة طبيعية ShutDown normal فإنه قد يكون هناك بعض المعاملات والتي تسجل بعد على ملفات الداتا وبالتالي تتطلب إعادة الاسترجاع وتحرير عمليات الفهرسة ونقل تلك

المعاملات كما ينبغي تحرير القطاعات المؤقتة Temporary Segments ونقل ما عليها إلى ملفات الداتا وهكذا.

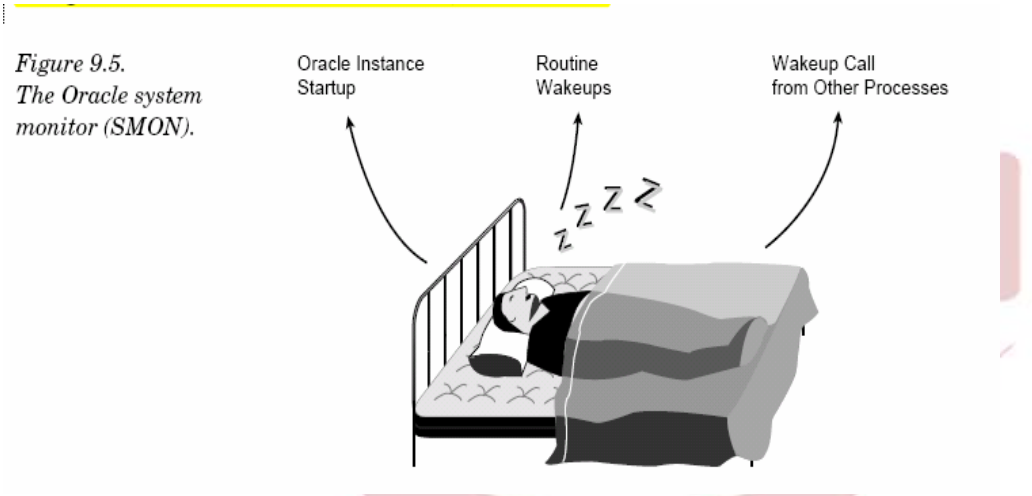


Figure 9.5. The Oracle system monitor (SMON).

كما يتضح من الشكل فالوظيفة السابقة معدة لكي تعمل في حالة إذا ما كان المثال في حاجة إلى بعض التحرير والإخلاء فمثلاً تقوم الوظيفة بالتحقق من وجود قطاعات مؤقتة غير مستغلة فتقوم بتحرير تلك المساحات ودمج المساحات كما تعمل الوظيفة أيضا في حالة احتياج كاتب الداتا Database Writer إلى مساحات مؤقتة Tempory Sgments فهو يطلب من الوظيفة تحرير وإخلاء بعض المساحات المستغلة اللازمة لأداء عمله بكفاءة.

- **مراقب العمليات Process Monitor (PMON))**

يخصص وقته كله لعملية الإخلاء والتحرير بعد إتمام عمل عمليات المستخدم User Processes فهي على سبيل المثال تقوم بالتالي:

- إزالة أرقام العمليات التي انتهت.
- إخلاء وإزالة أية أقفال Locks تكون قد عملتها تلك العمليات.
- إزالة وإخلاء أية عناصر من ذاكرة الكاش تكون ناتجة عن عمليات سابقة.

Memory Structure هيكلية الذاكرة

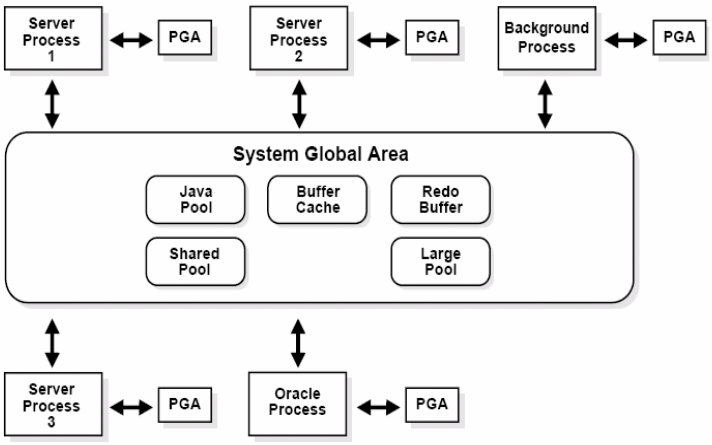
تستخدم أوراكل معمارية وتقسيماً معيناً للذاكرة في السيرفر الذي يحتوى على قاعدة البيانات وتمكن تلك المعمارية من أن تعمل مجموعة من العمليات في الذاكرة على تأمين وصول العديد من المستخدمين إلى قاعدة البيانات والتعامل معها بشكل فعال.

تتكون معمارية الذاكرة لأوراكل كما يتضح من الشكل التالي:

- منطقة النظام الشاملة System Global Area (SGA) وتتكون من:
  - الذاكرة المؤقتة لتخزين بلوكات الداتابيز Database buffer cache
  - الذاكرة المؤقتة لتخزين الريبدو لوغ Redo Log buffer
  - المنطقة المشتركة Shared pool
  - المنطقة Large pool
- منطقة البرامج الشاملة Program Global Areas (PGA) وتتكون من:
  - Stack areas
  - Data areas

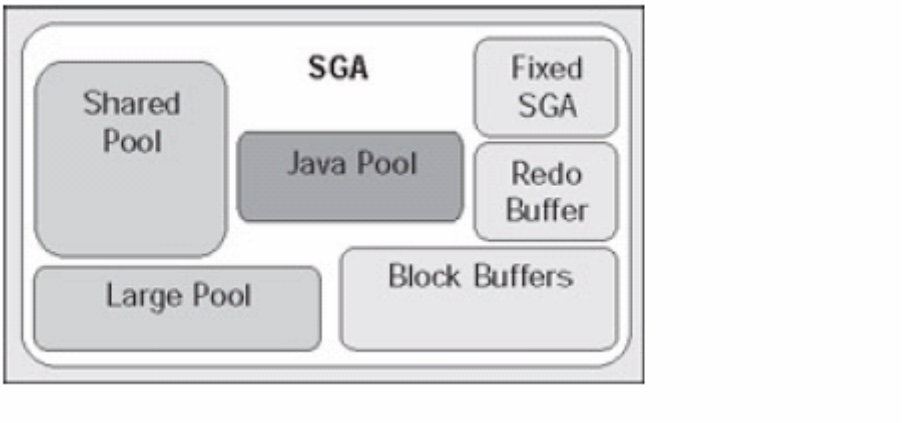
وتتضح معمارية الذاكرة وعلاقتها بالعمليات في الشكل التالي :

Figure 7–1 Oracle Memory Structures



منطقة النظام الشاملة SGA (System Global Area):

كل مثال من أوراكل يتكون من جزء ضخم من الذاكرة تعينه أوراكل يسمى منطقة النظام الشاملة System global Area ويتم إجراء عمليات أروا كل فيه كي يستطيع العديد من المستخدمين التعامل مع قاعدة البيانات. وتأخذ تلك المنطقة حيزاً من الذاكرة يتراوح بين عدد محدود من الميجا بايت في حالة قاعدة البيانات لأغراض الاختبار بينما يبلغ في الأنظمة الكبيرة المعقدة مساحات تقدر بالجيجا بايت. تتكون منطقة النظام الشاملة من مجموعة من مناطق الذاكرة والتي يتم مشاركتها من قبل المستخدمين والرسم التالي يوضح ذلك:



- منطقة الجافا Java Pool :

مقدراً ثابت من الذاكرة يخصص للجافا التي تعمل على قاعدة البيانات JVM

- منطقة :Large Pool

تخصص تلك المنطقة لعمليات دخول العديد من المستخدمين Session على قاعدة البيانات كما يستخدمها برنامج RMAN في عملية اخذ النسخ الاحتياطية.

- منطقة Shared Pool :

تحتوى تلك المنطقة على الإجراءات التي يتشارك فيها العديد من المستخدمين Shared Procedures و معلومات الكاتلوغ caches dictionary و الكيرسور.... الخ

والجدول التالي يوضح أهم البارمتر التي تؤثر على تحديد شكل SGA:

البارمتر	التأثير
JAVA_POOL_SIZE	تحدد حجم منطقة الجافا
SHARED_POOL_SIZE	تحدد حجم المنطقة المشتركة Shared Pool
LARGE_POOL_SIZE	تتحكم في حجم منطقة Large Pool
DB_BLOCK_BUFFERS	تحدد حجم ذاكرة الكاش لبلوكات الداتابيز
LOG_BUFFER	تحدد حجم ذاكرة اللوغ

منطقة محددة الحجم Fixed SGA

تشغل حيزاً ثابتاً من منطقة النظام الشاملة SGA ومقدار ذلك الحيز يتفاوت من نظام تشغيل لآخر أو حتى على حسب إصدارات أوراكل نفسها. إن أوراكل تستخدم تلك المنطقة لتضع فيها متغيرات وقيم تشير إلى مكونات أخرى من منطقة النظام الشاملة. إنها منطقة تستخدمها أوراكل ذاتها ولا نستطيع التحكم فيها تماماً.

ولننتقل الآن للتعرف على أقسام منطقة النظام الشاملة SGA:

- الذاكرة المؤقتة لتخزين بلوكات الداتابيز Database buffer cache
- الذاكرة المؤقتة لتخزين الريبدو لوغ Redo Log buffer

الذاكرة المؤقتة لتخزين بلوكات الداتابيز Database buffer cache

نحن الآن نزور أكبر وأهم قسم إنها **غرفة الكاش** المخصصة لتخزين بلوكات قاعدة البيانات Database buffer cache مؤقتاً. يمكننا تصور عملها كساحة انتظار أو نزل بين المستخدم وبين ملفات الداتا المكونة لقاعدة البيانات ففيها تنتظر بصورة مؤقتة بلوكات الداتا والمكونة من سجلات من الجداول المختلفة التي يتم استدعاءها لحساب مستخدم معين أو البلوكات التي سيتم إرجاعها مرة أخرى إلى الملفات بعد إتمام وتأكيد التعديلات التي تمت عليها. وهنا يتعين تحديد تلك الغرفة بكل دقة فلو كان الحجم على سبيل المثال أقل مما ينبغي فسيتعين علينا أن ننتظر طويلاً لنرى نتيجة تنفيذ جمل استعلام وغيرها ولو كان الحجم أكبر من اللازم فسنرهق بقية عمليات أوراكل وربما لا تعمل من الأصل!.

وحيث أن ساحة الانتظار لا توجد بها أماكن تسع للكل يبدو القرار صعباً وهو محاولة طرد بعض النزلاء واستقبال قادمين جدد.

هنا تتخذ أوراكل القرار الصعب وفقاً لمنطق بسيط وهو أن يبقى كل ما كان له فائدة ويرحل من لا يعمل 0LRU فتنشئ أوراكل قائمة تسمى الأكثر وعندما يستدعى سجل من قاعدة البيانات بواسطة جملة استعلام فإنه يذهب مباشرة إلى منطقة الذاكرة والتي أطلقنا عليها ساحة الانتظار ويوضع في قائمة الأكثر استخداما في نهاية القائمة. وكلما استخدم ذلك السجل يتم ترقيته إلى ترتيب أعلى في القائمة فإذا امتلئ الكاش ببلوكات جديدة نتيجة جمل استعلام أخرى وأوشك الكاش على الامتلاء تماماً فان أوراكل تتخذ قراراً بناءً على النظر في القائمة السابقة بإعادة الكتابة على الأجزاء الغير مستخدمة حالياً.

يتولى كاتب البلوك وهو عملية من عمليات الأوركل التي تعمل في الخلفية كما ذكرنا من قبل بكتابة تلك البلوكات إلى الملفات المادية Data Files و المكونة لقاعدة البيانات وهنا تدار تلك البلوكات التي تحتاج لإعادتها إلى الملفات بواسطة الكاتب DBWR ويطلق عليها مصطلح Dirty Blocks.

الذاكرة المؤقتة لتخزين الريبدو لوغ Redo Log buffer:

تخزن فيها بصورة مؤقتة معلومات التراجع والإعادة Redo قبل نقلها بواسطة كاتب الوغ LGWR إلى ملفات الريدو OnLine Redo Log وحيث أن التعامل مع الذاكرة يكون أسرع من التعامل مع الديسك فان استخدام تلك الذاكرة المؤقتة لمعلومات التراجع يؤدى إلى سرعة التعامل مع قاعدة البيانات. أن الداتا لن تبقى في تلك المنطقة للابد وفى الحقيقة فان محتويتها تفرغ كل ثلاث ثوان أو عندما يصدر أمر التأكيد Commit أو في حالة امتلاءها بثلث حجمها. حجم تلك المنطقة من الذاكرة يتحكم فيه البارمتر **LOG_BUFFER** ويعتبر الحد الأقل من تلك المساحة هو أربعة أضعاف حجم بلوكات الداتابيز ويمكن تقديره بالكيلو بايت عن طريق المعادلة (128* عدد CPUs)

منطقة الجافا Java Pool:

تخزن فيها بصورة مؤقتة كل ما يتعلق بأكواد الجافا والداتا. تستخدم تلك المنطقة بطرق مختلفة ويتوقف ذلك على الأسلوب الذي يعمل به السيرفر وتوجد وسيلة لأوراكل تعطينا إحصائيات عن طريقة استخدام تلك المنطقة للذاكرة وتتنبأ بمدى تأثير التغيير في حجم تلك المنطقة على الأداء وتفعل تلقائياً عندما يكون البارمتر statistics_level على الوضع TYPICAL

المنطقة المشتركة Shared Pool:

تعتبر تلك المنطقة من الذاكرة من أهم مناطق الـ SGA والتي تؤثر على أداء النظام ككل فصغر حجم تلك المنطقة عن الحد المعقول يؤدى التي تقليل الأداء بدرجة ملحوظة وربما تعطل النظام تماماً بينما تخصيص حيز أكبر من الحد الملائم يؤدي إلى نفس الكارثة. ولكن ما هي تلك المنطقة. إنها ببساطة المكان الذي تخزن فيه أوراكل بصفة مؤقتة الأكواد والبيانات الخاصة المتعلقة بالبرامج فعلى سبيل المثال عندما ننفذ جملة استعلام فإن أوراكل سوف تجهز جملة الاستعلام تلك وتحقق من سلامة الأمر ثم ترى هل يوجد استعلام آخر بنفس الصيغة وتخزن نتائج ذلك مؤقتا في تلك المنطقة. لو فرضنا أن هناك أكثر من 1000 اتصال بقاعدة البيانات وكلها

تنفذ نفس جملة الاستعلام SQL فالمطلوب فقط هو نسخة واحدة توضع في تلك المنطقة بصورة مؤقتة ثم يتم إجراءها على الكل ويتم تشارك نتائجها.

أن اكواد لغة PL/sql والتي يتم تنفيذها تخزن أيضا في تلك المنطقة. أوراكل أيضا تخزن بصورة مؤقتة جميع البارمتر الخاصة بالنظام في تلك المنطقة كما تخزن أيضاً معلومات عن الكاتلوغ The data dictionary في تلك المنطقة أيضاً. وبعبارة أخرى إنها بمثابة المطبخ الذي يتم طبخ كل الداتا المتعلقة بالأكواد فيه. وسندرس منها التالي:

- **منطقة The Library Cache**

تستخدم لتخزين جمل SQL المتشاركة فهنا تخزن بصورة خطة تنفيذ جمل الاستعلام parse tree and the execution لكل جملة. ولو كانت هناك عدة من التطبيقات أصدرت نفس الجملة فإن تلك المنطقة تشارك نفس الجملة لتقليل حجم الذاكرة فيما لو تم إصدار نفس الجملة لعدة تطبيقات على حدة.

- **منطقة تخزين معلومات الكاتلوغ The Data-Dictionary Cache**

يحتوى الكتالوغ على مجموعة من الجداول والمناظير والتي تستخدمها أوراكل لإدارة قاعدة البيانات حيث تخزن في تلك الجداول معلومات عن الهيكل المادي والهيكل الافتراضي لقاعدة البيانات ومن أمثلتها:

- معلومات عن صلاحيات المستخدم user privileges
- معلومات عن قيود الجداول الموجودة بقاعدة البيانات Integrity constraints
- أسماء الأعمدة ونوعها المكونة للجداول الموجودة بقاعدة البيانات columns in database tables
- معلومات عن حجم المساحات المخصصة لكل مخطط space allocated and used for schema objects

وأوراكل على اتصال دائم بالكتاوج من خلال تنفيذ جمل SQL للاستعلام والتعديل في تلك البيانات فاحرص على تخصيص مساحة كافية من الذاكرة للتخزين المؤقت لتلك البيانات لان ذلك يؤثر تأثيراً كبيراً في الاداء.

منطقة البرامج The Program Global Area (PGA):

تحتوي تلك المنطقة على الداتا ومعلومات التحكم عن عمليات السيرفر وحجمها ومكوناتها تتحدد بناء على اختيارات السيرفر التي تم تحديدها وتتكون من المناطق التالية :

- منطقة الـ Stack space :

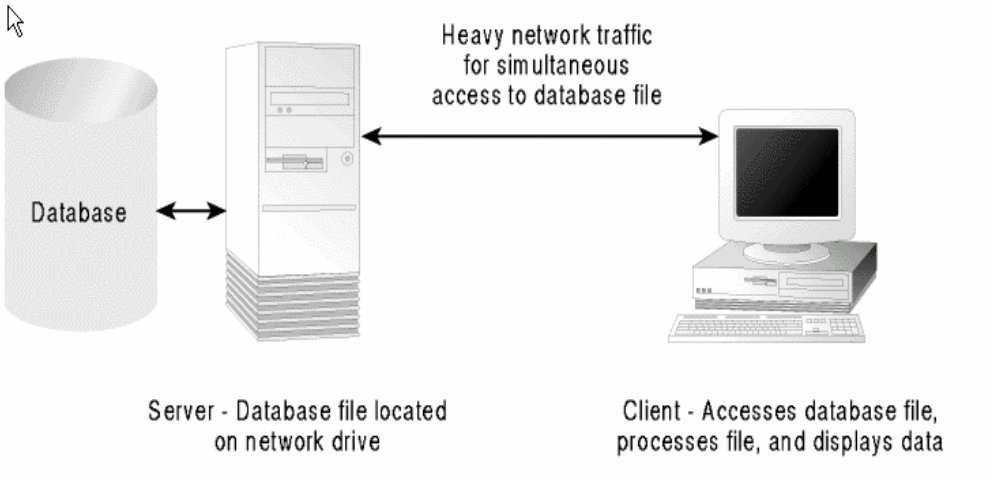
تحتوى على متغيرات الاتصال والدخول session's variables وكذلك المصفوفات arrays, and so on

- معلومات الاتصال الخاصة بالوصول لأوراكل Session تخزن في تلك المنطقة ولو كان السيرفر يعمل على اسلوب multithreaded server فان تلك المعلومات تسجل في الـ SGA
- منطقة SQL الخاصة Private SQL area : وتسجل فيها متغيرات binding variables and runtime buffers is kept.

تنفيذ المعاملات في أوراكل

المثال الحالي يلخص كيفية تعامل أوراكل لمعاملة Transaction وقبل أن نبدأ في المثال يتعين علينا توضيح معنى جديد وهو مفهوم المعاملة.

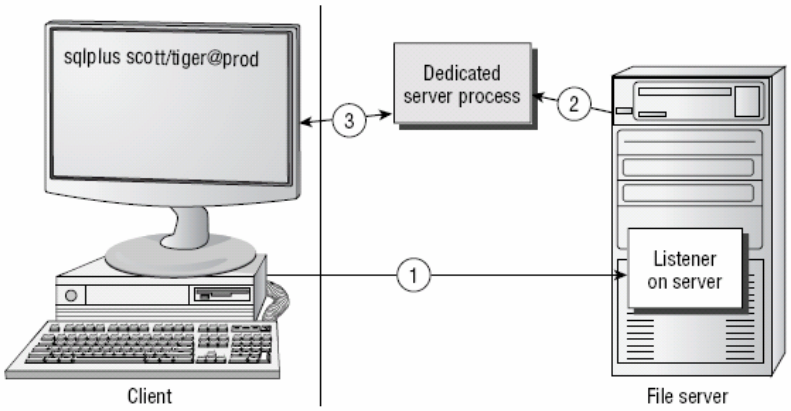
يصف مصطلح المعاملة Transaction مجموعة من الأعمال المرتبطة كوحدة واحدة نريد أن تنفذها أوراكل وتتكون المعاملة من واحد أو اكثر من جمل الـ SQL والتي تنتهي دائماً بجملة لتأكيد تنفيذها تسمى جملة التأكيد Commit أو للتراجع عن تنفيذها Rollback. إن مثلنا يفترض أننا نعمل في بيئة عمل (المزود / الخادم client /server) كما يتضح من الشكل التالي:



ولذلك يكون من الضروري هنا استخدام برتوكول الاتصال الشبكي SQL*Net.

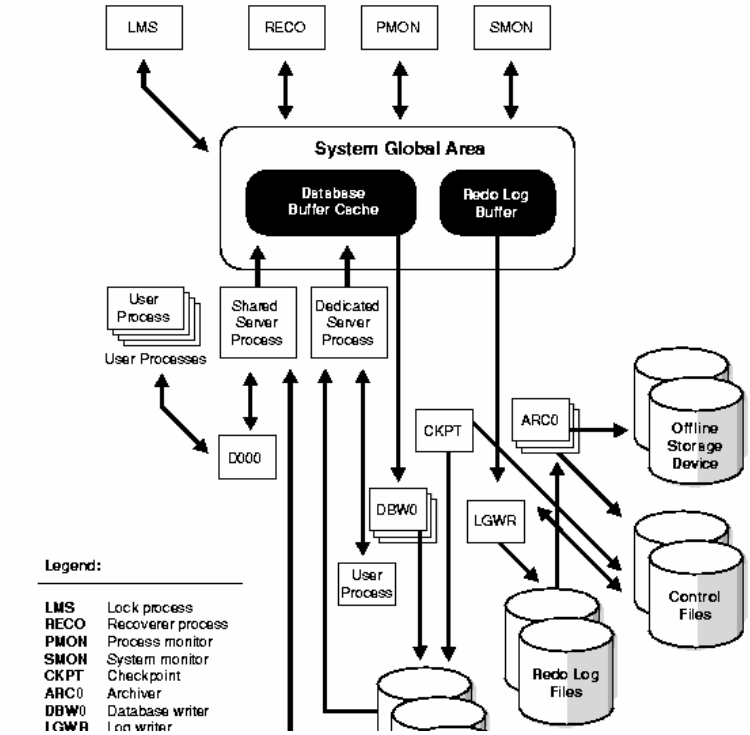
- يشغل مستخدم تطبيق من التطبيقات وليكن SQLPLUS ويحاول الاتصال بقاعدة البيانات أوراكل وهنا التطبيق يحاول عمل قناة اتصال بين أوراكل وبين المستخدم من خلال بروتوكول الاتصال الشبكي SQL*Net.

FIGURE 4.6 Dedicated connection: direct handoff method



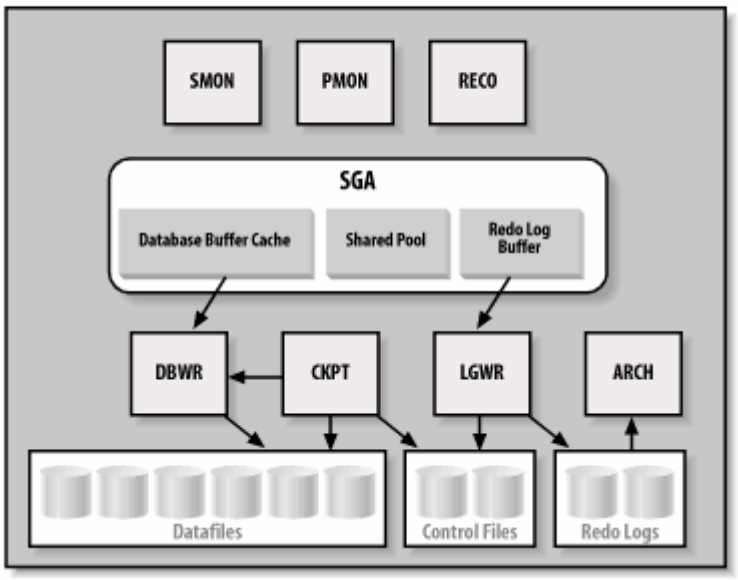
- السيرفر يتلقى اتصال المستخدم ويولد علمية من عمليات السيرفر لخدمة المستخدم.
- المستخدم يطلب تنفيذ جملة SQL وفى مثالنا يطلب المستخدم تغير قيمة في صف موجود فى جداول موجود بقاعدة البيانات.

Figure 1-3 Memory Structures and Processes of Oracle

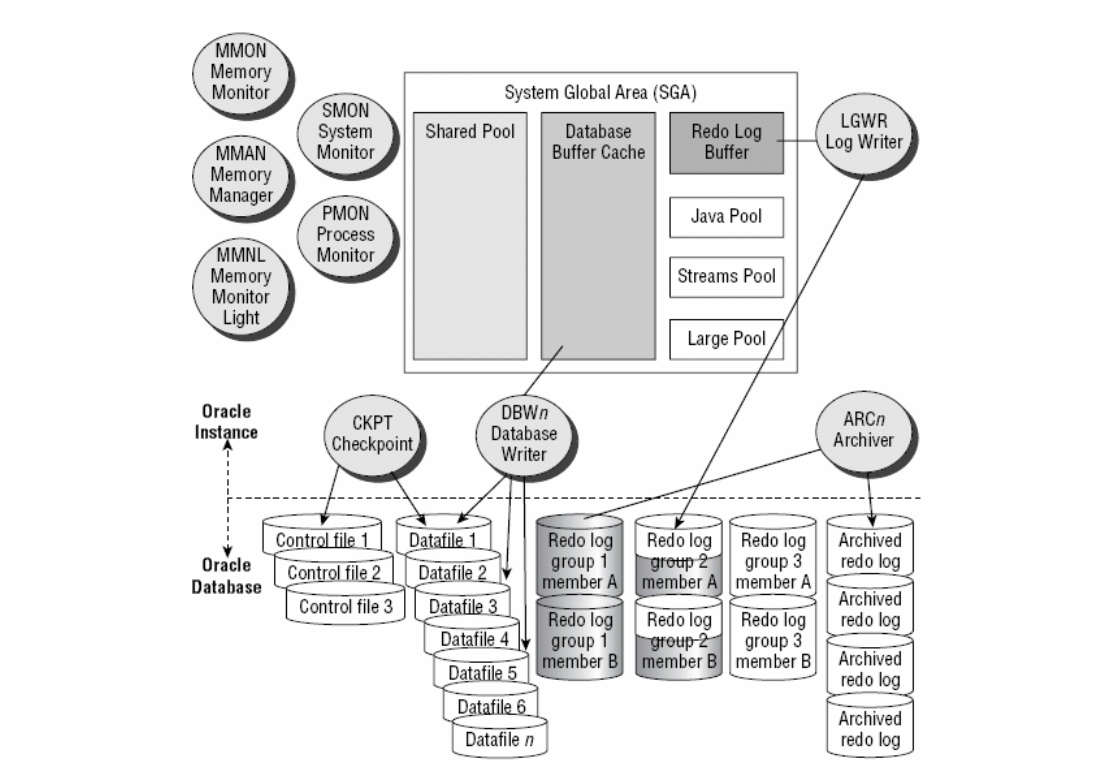


- يقوم السيرفر بالبحث في منطقة الذاكرة المشتركة Shared Pool عما إذا كانت هناك منطقة sql مشتركة Shared SQL Area وبها نفس الجملة الذي يريد المستخدم تنفيذها وإذا وجد مثل تلك المنطقة من الذاكرة فانه يتأكد من صلاحيات المستخدم في المشاركة في تلك المنطقة والوصول إلى الداتا الموجودة بها فإذا ما تأكد من ذلك فسوف يستخدم السيرفر تلك المنطقة لتنفيذ طلب المستخدم وإذا لم يجدها فانه يحاول تخصيص منطقة جديدة لتهيئة تنفيذ الجملة ثم تنفيذها.
- تصدر جملة الاستعلام للبحث عن الداتا المطلوبة فيتم البحث عنها أولاً في الذاكرة SGA فان لم يجدها يصدر أمر البحث داخل ملفات قاعدة البيانات Data files فإن وجدها توضع بلوكات الداتا في منطقة النظام الشاملة SGA. لاحظ أن عملية السيرفر تكون قادرة على قراءة ملفات الداتا المادية مباشرة وفى وقت آخر يقوم كاتب البلوك بكتابة بلوكات الداتا المعدلة إلى الملفات المادية مرة أخرى. وبالفعل توجد الداتا وتجرى عليها التعديلات.
- المستخدم إما أن يقوم بتأكيد العملية بإصداره أمر Commit أو لا يرد إتمامها فيصدر جملة Rollback والتي تلغى ما أجراه المستخدم من تعديلات.
- في نفس الوقت تسجل معلومات التغيير في ذاكرة الريدو وبمجرد تأكيد المستخدم للعملية يقوم كاتب اللوغ بتسجيل المعاملة من ذاكرة الريدو Redo Log Buffer إلى ملف اللوغ OnLine Redo Logملف اللوغ.
- بعد إتمام كاتب اللوغ بتسجيل معلومات العملية يبدأ كاتب البلوك في نقل التعديلات من ذاكرة الكاش لبلوكات الداتا Database buffer cash إلى ملفات الداتا Datafiles ويتم إصدار رسالة إلى المستخدم بنجاح تنفيذ العملية.

Figure 2-8. An Oracle instance



فتح قاعدة البيانات Database Startup



كما ذكرنا من قبل يمثل المثال معمارية معينة للذاكرة بالإضافة إلى عمليات تتم في الخلفية والتي تجعل المستخدم قادراً على الاتصال بقاعدة البيانات والتعامل معها. عندما تبدأ أوراكل في العمل فإنها تبدأ في تهيئة معمارية الذاكرة وتشغيل عمليات الخلفية استعدادا لدخول المستخدمين على قاعدة البيانات. وهكذا فان هناك عدد من المراحل حتى يتم التأكد من أن قاعدة البيانات جاهزة تماماً لتلقى طلبات المستخدمين. أن قاعدة البيانات تمر على الآتي:

- **التشغيل قبل الارتقاء STARTUP NOMOUNT:**

يتم فتح المثال وتجهيزه لفتح قاعدة البيانات قبل الارتقاء إلى قاعدة البيانات حيث يتم قراءة ملف البارمتر المسؤول عن إعدادات المثال وإعداد الذاكرة وفقاً للمعمارية التي تتطلبها أوراكل كما تأخذ عمليات الخلفية وضع الاستعداد للفتح والتشغيل ولكن دون

الاتصال بالهيكل المادي للملفات 0Data Files وفى هذه المرحلة لا تكون قاعدة البيانات متاحة بعد للاستخدام. عندما نكون في تلك الوضعية نستطيع إجراء بعض المهام وأكثرها شيوعا تشغيل الأوامر Scripts والتي نستطيع من خلالها عمل قاعدة بينات جديدة أن لم توجد واحدة من قبل. في تلك المرحلة من الممكن أن تحدث بعض المشاكل في فتح قاعدة البيانات أن كانت موجودة وهى على سبيل المثال عدم الوصول بشكل سليم إلى ملف التحكم في قاعدة البيانات Control File وهنا لا يمكن فتح قاعدة البيانات إلا بعد معالجة تلك المشكلة

- **الفتح مع الارتقاء STARTUP MOUNT:**

تتم جميع عمليات المرحلة السابقة بالإضافة إلى إمكانية الاتصال بهيكل الملفات. في هذه المرحلة يتم الوصول إلى ملف التحكم وقراءته والحصول على جميع المعلومات عن هيكل قاعدة البيانات المراد فتحها. يوجد بعض وظائف إدارة قاعدة البيانات المحددة التي يمكن إجراءها في تلك الوضعية ومنها وظائف الإصلاح فعلى سبيل المثال تغيير أماكن الملفات على الديسك أو إعداد قاعدة البيانات في الوضعية archive log mode

- **فتح قاعدة البيانات تماماً STARTUP OPEN:**

إذا مرت المراحل السابقة كلها بدون مشاكل تكون قاعدة البيانات مفتوحة وجاهزة تماماً لدخول المستخدمين وعلى الرغم من أن قاعدة البيانات متاحة للجميع إلا انه يمكن تشغيل قاعدة البيانات ببعض الاختيارات الإضافية إذا واجهتنا مواقف محددة وهى :

- **إجبار قاعدة البيانات على الفتح والعمل STARTUP FORCE:**

يمكنك تبني ذلك الاختيار إذا واجهتك بعض المشاكل أثناء فتح قاعدة البيانات بطريقة طبيعية فعلى سبيل المثال إذا حدث انقطاع للكهرباء مفاجئ

وتوقفت قاعدة البيانات بعد الفتح فهنا تكون محاولة تجربة تلك الخطوة أمراً ضرورياً يستحق المحاولة. وهنا لا يتوقف هذا الاختيار على أية وضعية كانت عليها قاعدة البيانات سواء كانت أغلقت بطريقة غير طبيعية shutdown abort and then restarts the database

- **الفتح مع وجود قيود STARTUP RESTRICT**

هنا تفتح قاعدة البيانات للمستخدمين الذين لهم صلاحية RESTRICTED SESSION دون بقية المستخدمين ويكون هذا في حالة إجراء الصيانة أو التصدير أو الاستيراد لقاعدة البيانات حيث يكون مطلوباً تعطيل وصول بقية المستخدمين لحين إتمام الصيانة. وبعد ذلك نعطل تلك الوضعية بالأمر التالي لتمكين بقية المستخدمين من الدخول العادي لقاعدة البيانات.

إغلاق قاعدة البيانات Database Startup

يتم إغلاق قاعدة البيانات لظروف عديدة منها إجراء عمليات الاستعادة والإصلاح أو لتبديل السيرفر وهنا لكي تغلق قاعدة البيانات ينبغي أن الدخول إلى قاعدة البيانات Connect مثل SYSDBA or SYSOPER وهى صلاحيات إدارية واسعة وهناك عدة أنواع أو أساليب لغلق قاعدة البيانات كما يلي:

الإغلاق الطبيعي العادي Shutting Down with the NORMAL Option

يستخدم الأمر التالي لإغلاق قاعدة البيانات بطريقة طبيعية :

SHUTDOWN NORMAL

يتبع الإغلاق العادي التالي:

- لا يسمح بأية اتصالات جديدة لقاعدة البيانات بعد إصدار الأمر.
- تنتظر قاعدة البيانات خروج جميع المتصلين حالياً بها قبل أن تغلق.
- يتم إعادة فتح قاعدة البيانات من جديد بصورة طبيعية فلا تحتاج لأي نوع من الإصلاح.

غلق قاعدة البيانات في الحال:

Shutting Down with the IMMEDIATE Option

يستخدم ذلك الأسلوب في الحالات التالية:

- عند توقع حدوث انقطاع قريب للطاقة.
- لكي نبدأ عملية نسخ احتياطي بصورة أتوماتيكية لقاعدة البيانات .
- عندما تعمل قاعدة البيانات بصورة غير طبيعية ونكون غير قادرين على خروج المستخدمين بصورة طبيعية Log Off

نصدر الأمر التالي لغلق قاعدة البيانات بصورة فورية

SHUTDOWN IMMEDIATE

يتبع الإغلاق الفوري التالي:

- لا يسمح بأية اتصالات جديدة على قاعدة البيانات كما لا يسمح بإجراء أية عمليات جديدة.
- أية عمليات غير مؤكدة Committed سوف يتم التراجع عنها rolled back .
- لا تقوم أروا كل بالانتظار حتى يخرج المستخدمين الحاليين لقاعدة البيانات عنها بل تقطع عنهم الاتصال.
- عند إعادة الفتح من جديد لا يحتاج المثال لإعادة إصلاح instance recovery.

الإغلاق مع الاختيار:

Shutting Down with the TRANSACTIONAL Option

يستخدم ذلك الاختيار عندما نريد إغلاق قاعدة البيانات ولكن بعد إتمام المعاملات الحالية Active Transaction قبل الإغلاق حيث نصدر الأمر التالي:

SHUTDOWN TRANSACTIONAL

الإغلاق عن طريق الإجهاض:

Shutting Down with the ABORT Option

يعتبر هذا إغلاقاً غير طبيعياً لقاعدة البيانات حيث يتم الخروج فوراً من المثال ولا يستخدم إلى في الحالات التالية:

- قاعدة البيانات لا تعمل بصورة طبيعية ولم تفلح الطرق السابقة في غلقها.
- حالة الانقطاع الفوري للطاقة فمثلا إذا كنا نعلم أن التيار الكهربائي سينقطع في خلال دقيقة واحدة.
- حدوث أخطاء في فتح المثال.
- نصدر الأمر التالي :

SHUTDOWN ABORT

وفى تلك الحالة يحدث :

- لا يسمح بأية اتصالات جديدة أو عمليات بقاعدة البيانات.
- يخرج المستخدمون الحاليين من قاعدة البيانات على الفور ويتم إنهاء أية عمليات كانوا يقومون بها حتى ولو كانت غير مكتملة.
- أية عمليات غير مؤكدة لا يتم التراجع عنها not rolled back

وفى هذه الحالة سنحتاج إعادة إصلاح المثال Instance recovery عند إعادة الفتح.

The Data Dictionary الكتالوغ

يعتبر الكتالوغ **data dictionary** واحداً من أهم مكونات قاعدة البيانات وهو عبارة عن مجموعة من الجداول تقوم أوراكل بإدارتها بنفسها وتخزن بها معلومات عن قاعدة البيانات ومن أمثلتها:

- جميع التعريفات الخاصة بكل المخططات Schema والكائنات Objects المتعلقة بها مثل (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on
- معلومات عن صلاحيات المستخدم user privileges
- معلومات عن قيود الجداول الموجودة بقاعدة البيانات Integrity constraints
- أسماء الأعمدة ونوعها المكونة للجداول الموجودة بقاعدة البيانات columns in database tables
- معلومات عن حجم المساحات المخصصة لكل مخطط space allocated and used for schema objects

وتقوم أوراكل نفسها بالوصول إلى معلومات الكتالوغ وقراءتها والتعديل فيها أن لزم الأمر فهي على سبيل المثال تقوم بالوصول إلى معلومات الكاتلوغ لكي تحصل على معلومات خاصة المستخدمين والمخططات وما تحتويه كما تقوم بتعديل معلومات الكتالوغ في كل مرة تصدر فيها جمل SQL التي من النوع DDL فمثلا في حالة إنشاء المستخدم لجداول جديد.

إن المستخدم SYS, Owner of the Data Dictionary هو المالك لمجموعة الجداول والمناظير Viewsالمكونة للكتالوغ ولا يجب لمستخدم آخر أن يقوم بالتعديل أو الإلغاء في ذلك المخطط وبالتالي احرص على كلمة السر للمستخدم SYS

أن معلومات الكتالوغ تكون متاحة لأوراكل عند فتح قاعدة البيانات وهي توجد على المساحة الجدولية المسماة SYSTEM tablespace ولذلك فتلك المساحة تبقى Online باستمرار.

يتكون الكاتلوغ من:

- مجموعة الجداول الأساسية Base Table :

والتي تخزن فيها قيم وداتا خاصة بما قلناه سابقا. أروا كل فقط هي وحدها التي تملك قراءة وتعديل تلك الجداول نظرا لخطورتها وتأثيرها على فتح وسلامة قاعدة البيانات.

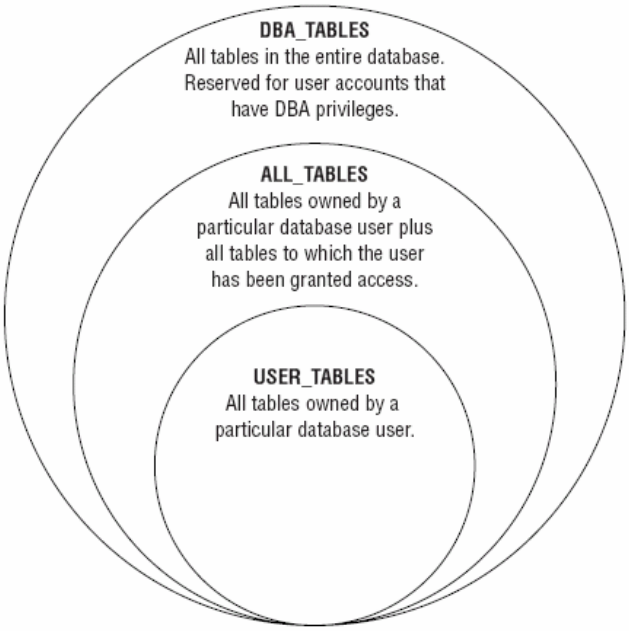
- **مناظير User-Accessible Views**

تستخدم في تلخيص وعرض المعلومات عن مجموعة الجداول المكونة للكتالوغ فهي تترجم معلومات الجداول السابقة إلى صورة يستطيع قراءاتها المستخدم. ومن أمثلتها أسماء المستخدمين الموجودين على قواعد البيانات والصلاحيات الممنوحة لهم.

استخدام المناظير الذي يوفرها الكتالوغ Data Dictionary Views

توفر أوراكل Oracle 10*g* عدد هائل من المناظير للتعامل مع معلومات الكتالوغ ويبلغ العدد حسب اختيارات التنزيل اكثر من 1300 منظار. إن تلك المناظير تسمى بأسماء تبدأ بـ **DBA_** و **USER_** , **ALL_** , الفارق بينها يمكن توضحه كما في الرسم التالي:

FIGURE 1.2 A comparison of data dictionary views



كما يظهر من الرسم فان المناظير DBA_TABLES يظهر جميع الجداول التي تحتوى قاعدة البيانات بينما يظهر المنظار ALL_TABLES جميع الجداول التي يمتلكها مستخدم معين مسجل لدى قاعدة البيانات بالإضافة إلى جميع الجداول التي يمتلك صلاحيات تمكنه من الوصول إليها بينما يظهر المنظار USER_TABLES جميع الجداول المملوكة فقط للمستخدم كما يتضح من المثال التالي:

```
SQL> select table_name from user_tables ;

TABLE_NAME
----------------------------
AVL_HOL
AVL_HOL_BACK
BOUNS
CHNG
CHNG_CODES
CHNG_LIST
DEPT
DOC_TYP
EMP
EMP_DOC
EMP_IMG
```

يظهر المثال جميع الجداول المملوكة للمستخدم emp

والجدول التالي يظهر بعض المناظير جرب واكتشف بنفسك

TABLE 1.2 Examples of Data Dictionary Views

Dictionary View	Description
DBA_TABLES	Shows the names and physical storage information about all the tables in the database.
DBA_USERS	Shows information about all the users in the database.
DBA_VIEWS	Shows information about all the views in the database.
DBA_TAB_COLUMNS	Shows all the names and datatypes of the table columns in the database.

للحصول على معلومات عن كل المناظير يمكن الذهاب إلى:

A complete list of the Oracle 10g data dictionary views can be found in Chapter 4 of the *Oracle Database Reference 10g Release 1 (10.1) Part Number B10755-01* available at http://tahiti.oracle.com.

هناك نوع آخر من المناظير تسمى **Dynamic Performance Views**

وفي أوراكل Oracle 10g يوجد حوالي 350 من تلك المناظير ومعظم تلك المناظير تسمى بأسماء تبتدئ بـ V$ الجدول التالي يبين أمثلة منها:

TABLE 1.3 Examples of Dynamic Performance Views

Dynamic Performance View	Description
V$DATABASE	Contains information about the database itself, such as the database name and when the database was created.
V$VERSION	Shows which software version the database is using.
V$OPTION	Displays which optional components are installed in the database.
V$SQL	Displays information about the SQL statements that database users have been issuing.

ومثال عليها المنظار V$VERSIONوالذي يبين نوع إصدار الأوركل المستخدم

```
SQL> select * from v$version ;

BANNER
----------------------------------------------------------------
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod
PL/SQL Release 10.2.0.1.0 - Production
CORE    10.2.0.1.0      Production
TNS for 32-bit Windows: Version 10.2.0.1.0 - Production
NLSRTL Version 10.2.0.1.0 - Production

SQL>
```

وعلى الرغم من كلا النوعين من المناظير يشتركان كلاهما في تقديم معلومات هامة عن عمل أوراكل و بينهما تشابه إلا انه توجد بعض اختلافات تظهر في الجدول التالي:

TABLE 1.4 A Comparison of Data Dictionary and Dynamic Performance Views

Dictionary Views	Dynamic Performance Views
The DBA_ views usually have plural names (for example, DBA_DATA_FILES).	The names of the V$ views are generally singular (for example, V$DATAFILE).
The DBA_ views are available only when the database is open and running.	Some V$ views are available even when the database is not fully open and running.
The data contained in the DBA_ views is generally uppercase.	The data contained in the V$ views is usually lowercase.
The data contained in the DBA_ views is static and is not cleared when the database is shut down.	The V$ views contain dynamic statistical data that is lost each time the database is shut down.

معلومات لابد منها

مسؤوليات ووظائف مدير قواعد البيانات:

على مدير قواعد البيانات أوراكل بعض المهام الرئيسية الواجب عليه الاهتمام بها لضمان صلاحية قاعدة البيانات:

- ❖ تركيب برامج أوراكل والتطبيقات الخاصة به
- ❖ تحديد مواصفات الأجهزة المستخدمة والتعامل معها
- ❖ تخطيط قاعدة البيانات وتصميمها
- ❖ إنشاء وفتح قواعد البيانات
- ❖ إنشاء البرامج والتطبيقات
- ❖ إنشاء نسخ احتياطية لقواعد البيانات
- ❖ إضافة مستخدمين لقواعد البيانات و إعطاء امتيازات الدخول
- ❖ تحسين أداء قواعد البيانات

وظائف نظم إدارة قواعد البيانات:

إليك بعض المواصفات التي يجب أن تتوفر في قواعد البيانات القياسية والتي تقدمها أوراكل مع العديد من المزايا الأخرى:

- ❖ إدارة حجم كبير جدا من المعلومات وبشكل آمن
- ❖ إمكانية العمل في بيئة متعددة المستخدمين مثل الشبكات
- ❖ حماية البيانات من عدم العبث بها
- ❖ تقديم الأدوات لإصلاح الأعطال والتخزين الاحتياطي

نظام إدارة قواعد البيانات أوراكل:

يمتاز نظام قواعد البيانات أوراكل بالوثوقية والأداء العالي بسبب التقيد بالمقاييس والكثير من الأسباب التي تم شرح بعضها هنا:

- ❖ أن تكون إمكانية التخزين كبيرة جداً
- ❖ إمكانية التعامل مع عدد كبير من المستخدمين بشكل متزامن ومتوازي
- ❖ نظام أمن من الممكن عدم توقف التعامل معه على مدار اليوم وقدرته على إصلاح الأعطال حتى الفيزيائية منها
- ❖ حماية البيانات من العبث بها أو الحصول عليها من غير المستخدمين المحددين
- ❖ الأداء العالي إذ أن زيادة البيانات والمستخدمين لا تؤثر على سرعة وعمل أوراكل
- ❖ المواصفات القياسية الخاصة بأوراكل مواصفات عالمية
- ❖ إمكانية العمل على أكثر من نظام تشغيل
- ❖ إمكانية الاتصال القوية عبر الشبكات
- ❖ التعريب إذ يمكن التعامل معه باللغة العربية

أهداف قواعد البيانات:

هناك أهداف معينة وضعت لأجلها قواعد البيانات والتي قامت الشركات بعد ذلك بتطوير قواعد البيانات هذه وأكبرها هي شركة أوراكل وإليك أهم أهداف قواعد البيانات التي استدعت وضعها لها.

مركزية البيانات:

وهي إحدى أهم أهداف الذي وجدت قواعد البيانات لها والتي توفر ترتيب البيانات لسهولة الوصول إليها وعدم تضاربها مع التوفير في المساحات المستخدمة في التخزين وللمشاركة في البيانات بين عدة تطبيقات

استقلالية البيانات عن التطبيقات:

والتي تفيد في عدم ضياع البيانات بسبب انهيار البرنامج التطبيقي الذي يقوم بإدارتها.

ربط البيانات بعلاقات:

وهي هدف مهم في توفير الجهد والوقت وللحصول على البيانات التي ترتبط أو تشترك فيما بينها للحصول على معلومات قيمة.

تكامل وانسجام البيانات:

إحدى أهداف قواعد البيانات التي تقوم بتحديد نوع البيانات ووضع الشروط المحددة لهذه البيانات.

أمان البيانات:

وهي إحدى أهداف قواعد البيانات التي تساعد على عدم ضياع البيانات وسرعة إصلاحها واستردادها.

الوثوقية:

وهي إحدى أهم الأهداف والتي تساعد في سرية البيانات وعدم فضحها وذلك بتحديد السماحيات للوصول إليها

المشاركة في البيانات:

وهي من أهم الأهداف في البيئات الشبكية والتي تحتوي على العديد من المستخدمين الذين لهم سماحيات وصول لقواعد البيانات حيث تساعد في عدم تضارب البيانات بين المستخدمين.

بعض أهم الأدوات الخاصة بمديري قواعد البيانات أوراكل:

Server Manager: مدير الخادم . تستخدم هذه الأداة لمراقبة قاعدة البيانات كما تساعدك على التحكم في إدارة قاعدة البيانات.

Enterprise Manager: مدير المؤسسات . تستخدم هذه الأداة في إدارة الصلاحيات والتحكم في سماحيات المستخدمين بواجهة رسومية.

SQL Loader: معالج لغة الاستعلامات . تستخدم هذه الأداة في استيراد البيانات من ملفات نصية إلى قاعدة بيانات أوراكل.

Export Manager: مدير التصدير . تستخدم هذه الأداة في تصدير قاعدة بيانات أوراكل أو جدول محدد من قاعدة البيانات لاستخدامات النسخ الاحتياطي أو لنقل البيانات من قاعدة بيانات إلى أخرى.

Import Manager: مدير الاستيراد . تستخدم هذه الأداة في استيراد قاعدة بيانات أو جداول تم تصديرها من قبل عملية تصدير سابقة إما لإصلاح قاعدة البيانات أو لنقل البيانات من قاعدة بيانات أخرى.

وتعتبر الأوراكل من الرواد الذين استطاعوا تقديم حلول متكاملة للتجارة الإلكترونية في عالم الإنترنت حول العالم والتي قامت بتطوير البنى التحتية لخدمة تقنية الخادم / الزبون في تطبيقات الإنترنت وقدمت الأدوات والبرامج والحلول حول الشبكة العالمية العنكبوتية، وتعمل برامج أوراكل على الحاسبات الشخصية محطات العمل الفرعية الحاسبات المتوسطة حاسبات رئيسية وبشكل كبير بسبب تقنية التوازي وتقديمها الدعم لمعظم نظم التشغيل انتشاراً.

إصدارات الأوراكل:

وقد تم بيع الإصدار الأول عام 1979 وبعد أن لاقى رواجاً سريعاً توالت الشركة بطرح الإصدارات الجديدة والمتطورة على التوالي وهي كالتالي:

الإصدار الثاني: فقد تم بنائه من أجل العمل مع حوا سيب الـ (بي دي بي الرقمية) والتي تعمل على نظام التشغيل (أر إس إكس) والتي عملت فيما بعد على نظام (دي اي سي فاكس).

الإصدار الثالث: من النظام تم طرحه عام 1983 حيث أجريت عليه الكثير من التحسينات خاصة تلك المتعلقة بصيغة (لغة الاستفسارات القياسية) وتم كتابته بلغة (سي) وتم تغيير اسم الشركة من . أر إس أي . إلى مجموعة أوراكل.

الإصدار الرابع: من أوراكل تم إنجازه عام 1984 ولقد دعم هذه الإصدار نظامي التشغيل (فاكس) و(أي بي أم في أم) كما كان أول إصدار يدعم خاصية تناسق القراءة.

الإصدار الخامس: من أوراكل فقد تم إنجازه عام 1985 وقدم هذا الإصدار دعماً لتقنية الزبون/الخادم باستخدام (لغة الاستفسارات القياسية) كما انه أول منتج يعمل ضمن نظام التشغيل والذي استطاع تجاوز حاجز 640 كيلو بايت من نظام التشغيل DOS .

الإصدار السادس: من أوراكل تم إنجازه عام 1988 ولقد أضاف تقنية القفل على مستوى منخفض إضافة إلى العديد من التحسينات والوظائف والمنصات كما أضيف إليه خيار التوازي والذي يعمل على نظام (دي اي سي فاكس) وذلك عام 1991 ومن ثم أصبح هذا الخيار متاحا ضمن العديد من المنصات.

الإصدار السابع: تم طرحه عام 1992 وتم عليه إجراء العديد من التغييرات والإضافات مثل منطقة الذاكرة والمعالجة المركزية واستخدام الدخل والخرج واحتوى أيضا على الكثير من الأدوات الخاصة بمديري قواعد البيانات

الإصدار الثامن: والذي يتضمن مفهوم الأغراض بالإضافة إلى العديد من الميزات والتقنيات وأدوات إدارة قواعد البيانات والسماح لوجود حقول تتسع إلى حد 4 جيجابايت للحقل الواحد كحد أقصى.

أخيراً وحتى يومنا هذا فقد تم طرح الإصدار الأخير من أوراكل وهو الإصدار العاشر.

تنصيب قاعدة بيانات أوراكل i9 المخصصة للويندوز NT/2000/XP

للحصول على أغلب منتجات أوراكل من موقع أوراكل الرسمي "التسجيل مجانا" ورابط التسجيل هو:

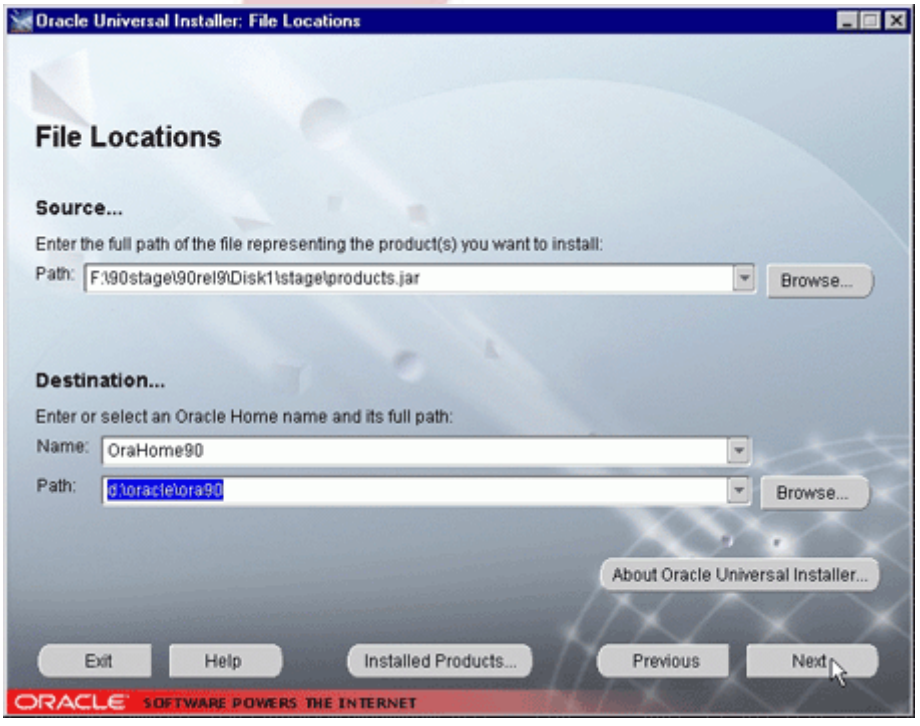
https://profile.oracle.com/jsp/reg/createUser.jsp?act=5&src=1180588&tid=262&owner=3&nexturl=http%3A//www.oracle.com/technology/software/index.html&language=en

تنصيب قاعدة بيانات أوراكل i9 :

الاسم : OraHome90

المسار :\Oracle\ora90: .. or <D> or < C>

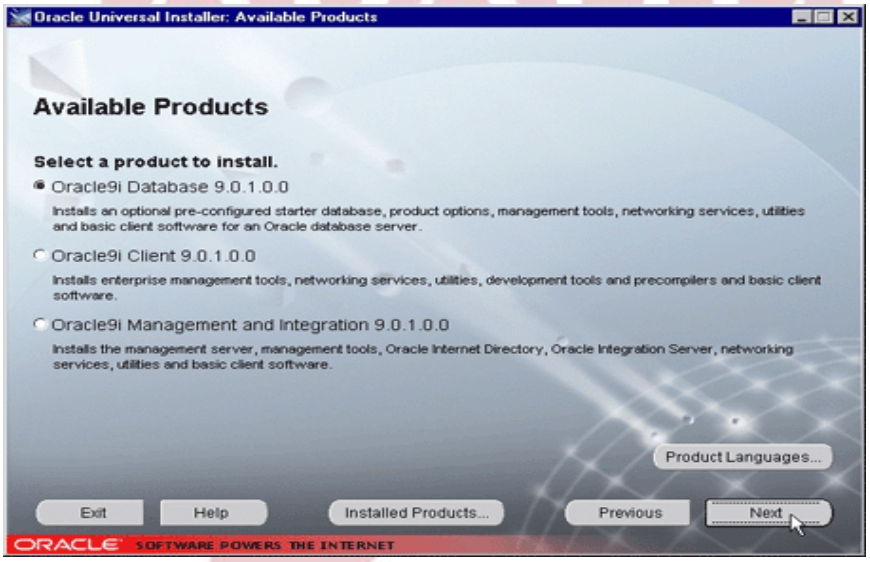
وذلك كما في الشاشة التالية :-



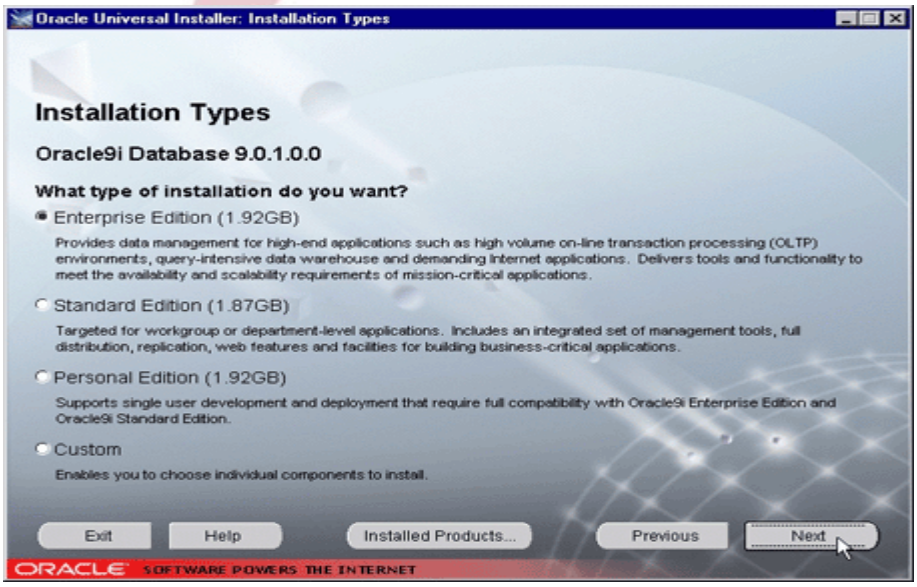
ثم Next

يمكن تغيير أسماء وموقع الإعداد لكن أهم شي أن لا تكون قاعدة البيانات والديفلوبر في مكان واحد " أي في نفس الملف أو المجلد" ويمكن أن يكونوا الاثنان في جزء واحد في الهارد ديسك مثل الجزء :C أو :D ولكن هنا تم اختيار الأماكن الافتراضية لكي يتم فهم المثال وتوصيله بطريقه واضحة للجميع , كما يجب أن يتم إعداد الديفلوبر قبل قاعدة البيانات.

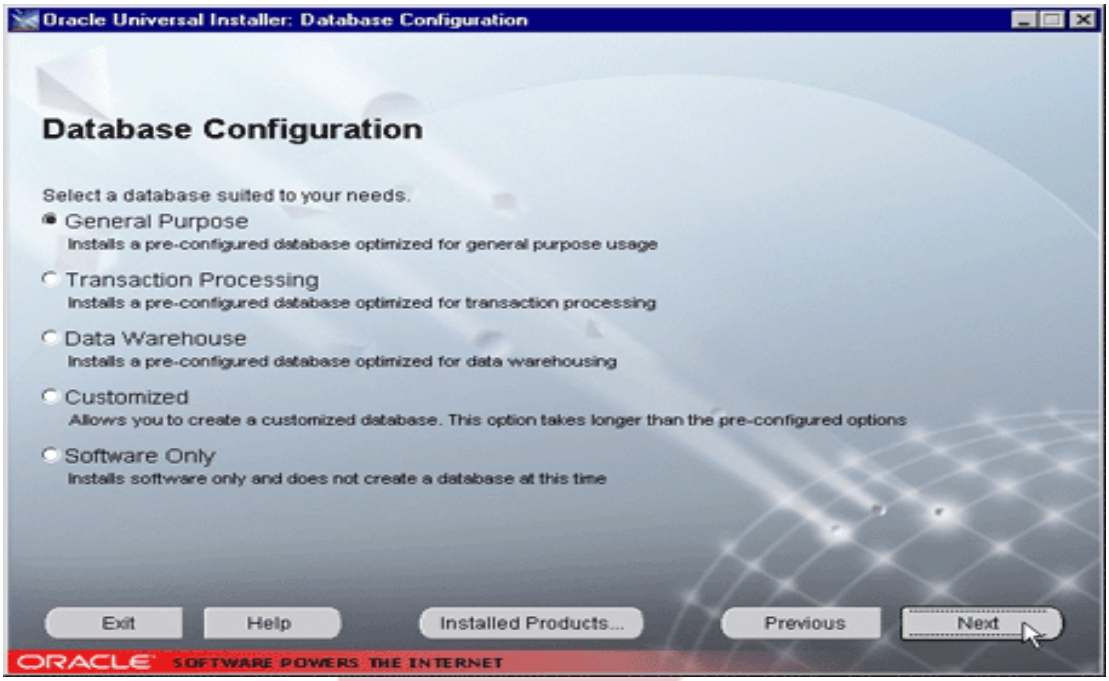
ثم قم باختيار تنصيب قاعدة بيانات أوراكل i9 :



قم باختيار نوع قاعدة بيانات أوراكل الذي تريد تنصيبه في جهازك:



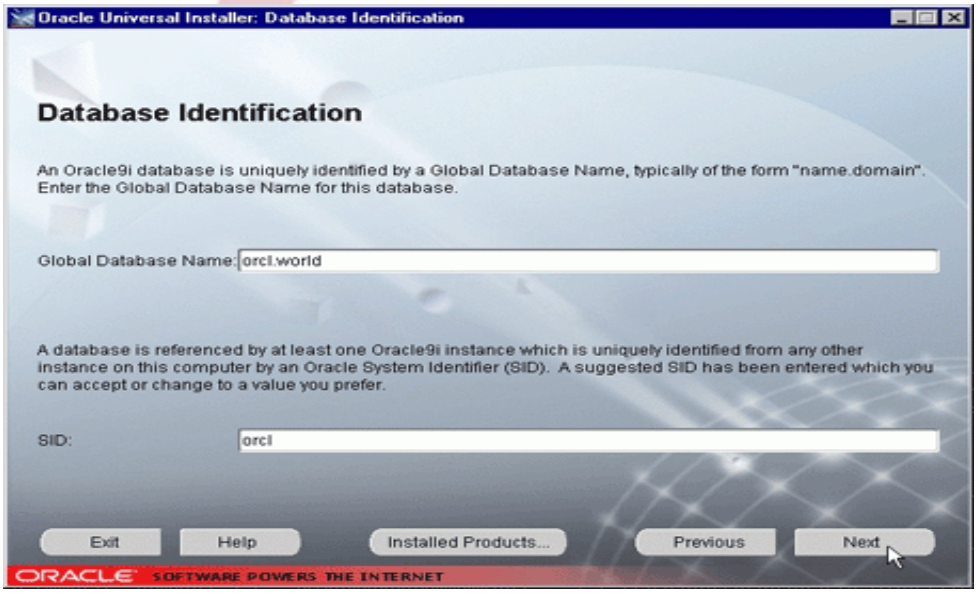
ثم قم باختيار نوع التنصيب الاختيار الأول أفضل لغير الخبراء في أنواع التنصيب المتقدمة General Purpose



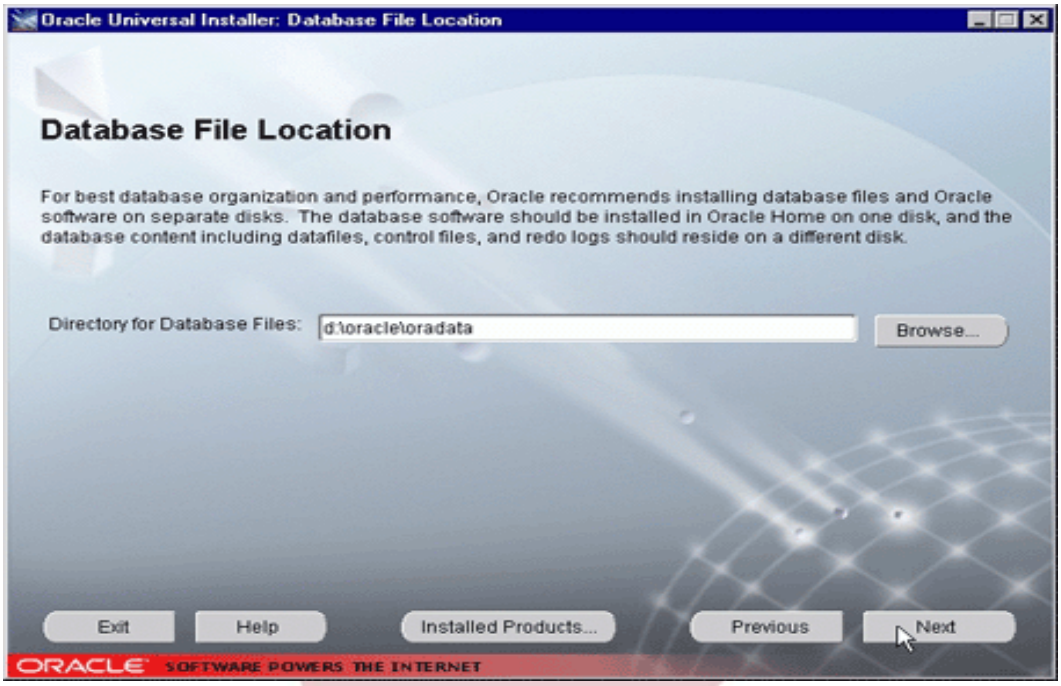
قم بتحديد أسم قاعدة البيانات (SID) و Global database name لقاعدة البيانات

وليكون global database name هو ORACLE.US.COM

وطبعا سيكون أسم قاعدة بيانات أوراكل (SID) هو ORACLE



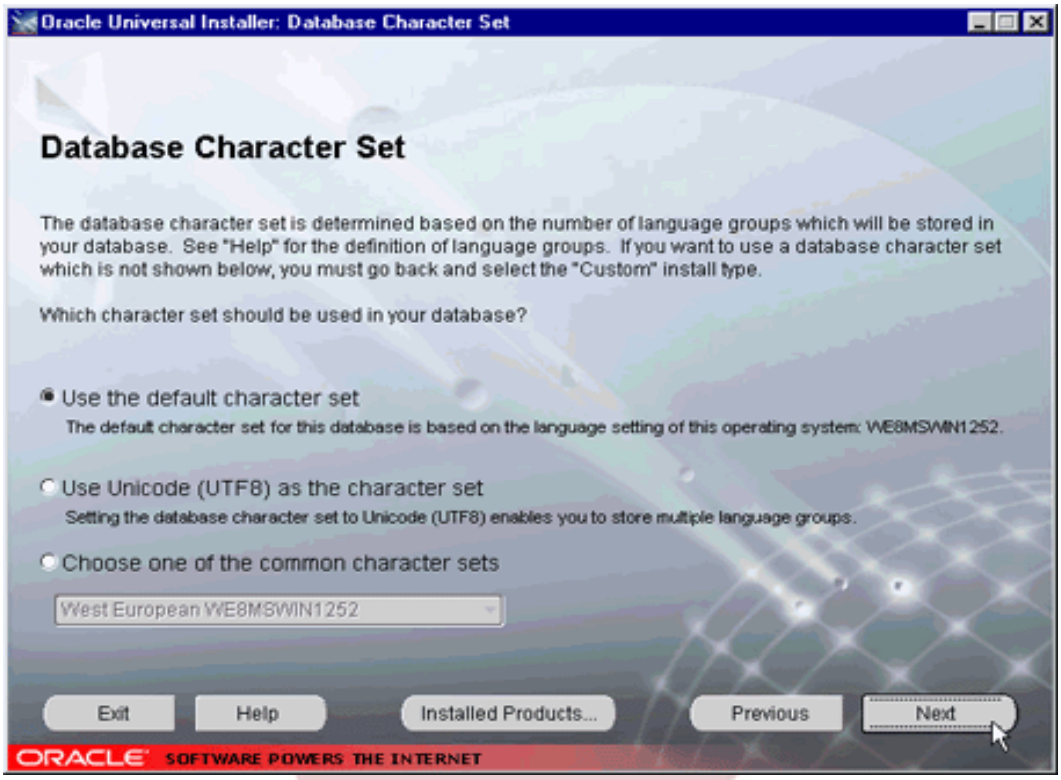
يتم طلب تحديد موقع Database file فلا تغيير الموقع.



تظهر لنا الآن ثلاثة خيارات لتحديد نوع الأحرف:

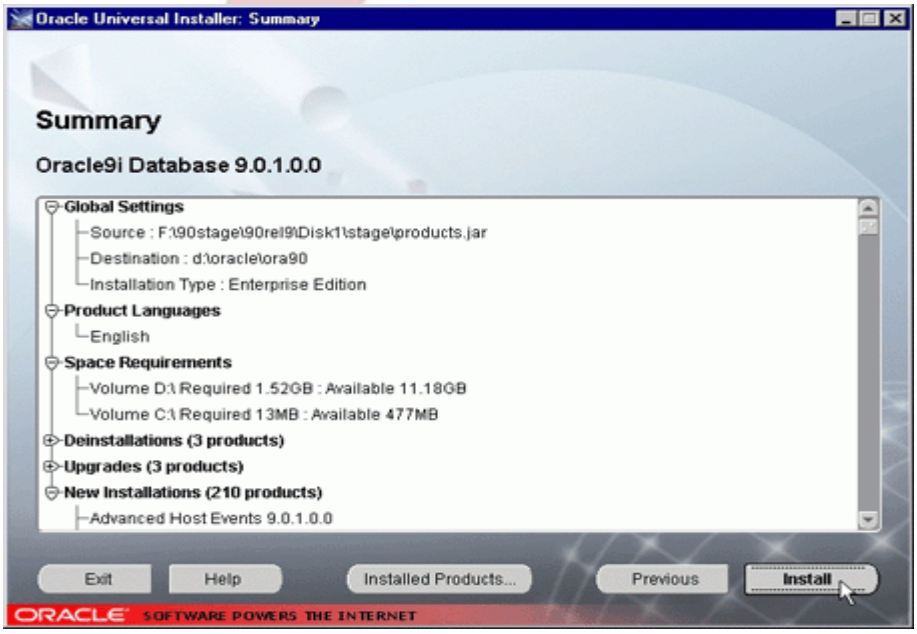
- النوع الافتراضي . نفس المستخدم في نظام التشغيل .
- استخدام(تقنية اليونيكود) وهي تقنية تستخدم لأول مره في أوراكل وهي تدعم كل اللغات . يفضل اختيارها .
- الاختيار من القائمة النوع الذي تريده.

وذلك حسب الشاشة أدناه.



ثم أضغط على زر Next

نحدد الآن نوع قاعدة بيانات أوراكل المراد تنصيبها والأدوات التي سوف يتم تنصيبها معها كما في الشكل:

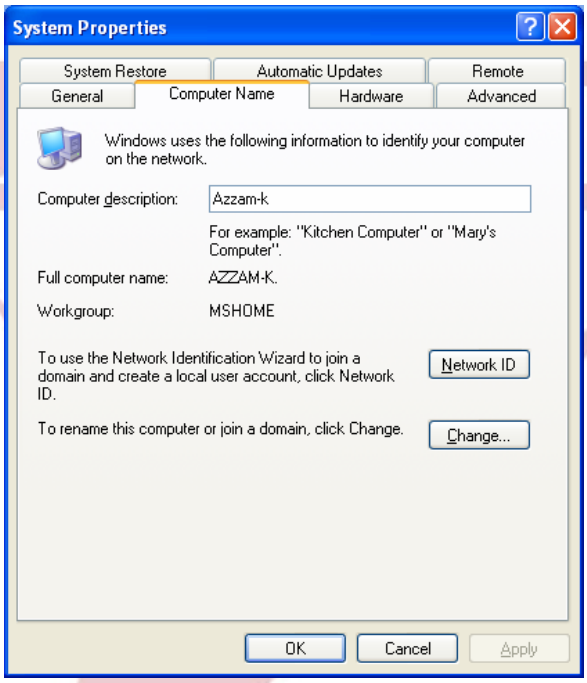


نضغط Install لبدء التنصيب

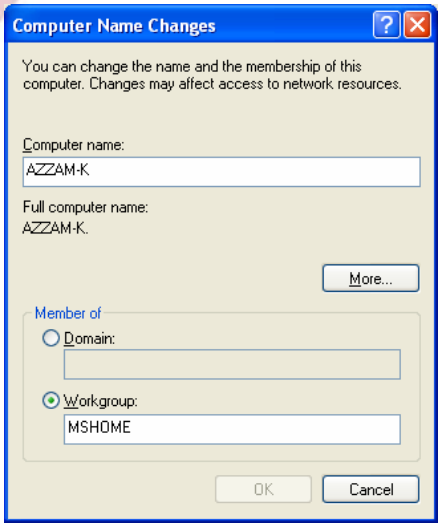
أثناء التنصيب سيتم طلب الـ CD رقم 2 ورقم 3 وبعد الانتهاء من عملية التنصيب بنجاح سوف تظهر هذه الشاشة:

بعد الانتهاء من التنصيب علينا القيام ببعض الإعدادات:

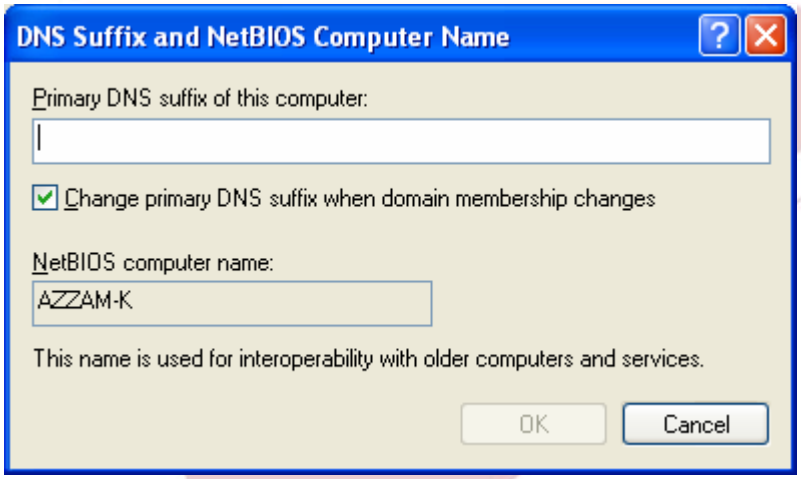
من جهاز الكمبيوتر الموجودة في سطح المكتب نضغط زر الماوس اليمين ومن ثم اختار Properties وسوف تظهر لك شاشة properties System اختار منها Computer Name



ثم نضغط على زر Change ستظهر شاشة Computer Name Changes نضغط على زر More



ستظهر شاشة DNS Suffix and NetBIOS Computer Name وفى خانة computer suffix of this Primary DNS نكتب ORACLE.US.COM ونضغط على زر OK وسوف يُطلب إعادة تشغيل الجهاز. بعد إعادة تشغيل الجهاز نكمل طريقة اتصال الديفلوبر 2000 Release 6.0 OR i6 بقاعدة بيانات أوراكل i9



اتصال الديفلوبر 2000 Release 6.0 OR i6 بقاعدة بيانات أوراكل i9

1- قم بإنشاء configuration Net service Name Local وذلك كالتالي :

قم بتشغيل الأداة التالية Net Configuration Assistant والموجودة في:

Start -> programs -> Oracle - Orahome90 -> Configuration and Migration Tools -> Net Configuration Assistant

- اختار Local Net service Name configuration ثم اضغط على زر Next.
- اختار Add ثم اضغط على زر Next.
- اختار Oracle 8i or later database or service ثم اضغط على زر Next.
- حدد اسم Service Name الذي قمت بتحديده عند إعداد (تنصيب) قاعدة البيانات وهو نفس اسم الـ " ORACLE.US.COM global database name " ثم اضغط على زر Next.

- قم بتحديد البروتوكول وهو TCP ثم اضغط على زر Next .
- قم بتحديد Host Name وهو إما اسم الكمبيوتر أو رقم TCP/IP للكمبيوتر. لا تقم بتغيير رقم الـ Port ثم اضغط على زر Next.
- اختار Yes, perform a test ثم اضغط على زر Next.

يجب أن تكون نتيجة الامتحان .Connecting...Test successful وإلا يجب التأكد من البيانات التي قمت بإدخالها في السابق أو قم بالضغط على زر Change login وقم بوضع التالي:

Username: system
Password: manager

إذا تم الاتصال بنجاح اضغط على زر Next.

– قم بتحديد اسم للـ Net Service التي قمت بإنشائها وليكن developer ثم اضغط على زر Next.

– ثم اضغط على Next ثم Next ثم Finish .

قم بأخذ نسخه من الملف المسمى tnsnames.ora الموجود في الموقع التالي : Oracle\Ora90\network\admin

قم بلصق الملف السابق في الموقع التالي الموجود به ملف بنفس الاسم والموقع هو ORANT\net80\admin

قم بتغيير اسم الملف إلى sqlnet.old الموجود في الموقعين التاليين:

أ– Oracle\Ora90\network\admin

ب– ORANT\net80\admin

قم بتشغيل أي أداه من أدوات الديفلوبر

بيانات الدخول كالتالي:

Username: system

Password: manager

Host String Or Database: developer.oracle.us.com

وسوف يتم الاتصال بين الديفلوبر وقاعدة البيانات.

Oracle

Part 2

مقدمـة:

نرجو أن تكون عزيزي الطالب قد كونت فكرة جيدة عن الأوراكل من خلال الجزء الأول. وسنتعرض في هذا الجزء إلى التعرف على الأوراكل بشكل عام من خلال التعرف على الأوامر الخاصة به وكيفية التعامل مع الجداول و. . . . . علنا نستطيع الدخول إلى عالم الأوراكل الواسع بكل ثقة ودراية.

الأمر CREATE:

يقوم هذا الأمر بإنشاء USER على قاعدة البيانات ويجب وضع اسم الـ USER والذي هو في هذا المثال Azzam ثم يأتي بعدها BY IDENTIFIED وهي كلمة السر وهي هنا 236.

```
CREATE USER Azzam IDENTIFIED BY "236"
```

يجب أن تخرج لك الرسالة التالية User created. أي أنه تم إنشاء المستخدم. إذا قاعدة استخدام الإنشاء المستخدم هي:

```
CREATE USER NAMEOF USER IDENTIFIED BY "YOUR PASSWORD"
```

الأمر GRATN:

وحتى نستطيع الدخول إلى المستخدم يجب إعطاءه الصلاحيات وهذه الصلاحيات تسمى GRATNS حيث يتم من خلالها تفويض المستخدم للدخول إلى قاعدة البيانات.

```
GRANT OBJECTNAME TO YOURUSERNAME
```

وهي تعني إعطاء صلاحية الاتصال إلى المستخدم وتعطى صلاحيات أخرى للمستخدم منها RECOURCE وتعني مورد وكذلك DBA وتعني DATABASE ADMINISTRATION مدير قاعدة البيانات:

لاحظ الكود أدناه:

```
GRANT RESOURCE,DBA TO Azzam
```

لقد تم إعطاء المستخدم خاصتين معاً من خلال الفاصلة , هذا يعني أنه يمكننا أن نعطي عدد من الصلاحيات بسطر واحد وباستخدام الفاصلة.

الأمر CONNECT:

وهو للدخول إلى المستخدم حسب التالي:

```
CONNECT Azzam/236
```

سيتم الاتصال هنا بالمستخدم المسمى Azzam وكلمة السر 236 وسوف تظهر لنا رسالة بأنه تم الاتصال.

ويمكننا معرفة ذلك من الأمر التالي:

```
SHOW USER
```

فهذا الأمر يبين لنا اسم المستخدم

DUMMY TABLE:

هو جدول أنشأته أوراكل لكي نستعين به ببعض العمليات فهو يتعامل مع قاعدة البيانات مباشرة مثل استعراض التاريخ واسم هذا الجدول هو DUAL.

لاستعراض تاريخ اليوم نكتب الأمر التالي:

```
SELECT SYSDATE FROM DUAL
```

وأهم ما سنتناوله الآن هو طرق إنشاء الجداول والربط بينها وتعريف المحددات.

الأمر CREATE TABLE:

عند القيام بإنشاء جدول يعني أن نهيأ قاعدة البيانات وبعد ذلك يجب وضع اسم الجدول فنقول CREATE TABLE STUDENTS مثلاً ونضع بعدها قوس ليشمل عدد الحقول هذا الجدول وينصح دائماً باتخاذ أول ثلاثة حروف من اسم الجدول عند تسمية أي حقل تابع لهذا الجدول والهدف هو معرفة أن هذا الحقل تابع للجدول ثم نحدد نوع الحقل أي DATATYPE وسوف نتناول ثلاثة أنواع هي:

VARCHAR2: وهي تأخذ أرقام وحروف في الحقل ويجب تحديد طول الحقل ونقصد بطول الحقل هو عدد الأحرف في هذا الحقل فنقول VARCHAR2 20 أي أن طول الحقل 20 حرف.

NUMBER: وهي تأخذ أرقام عادية أو أرقام عشرية ويجب أيضاً تحديد طول الحقل فيه وهي عدد الخانات فنقول NUMBER4 ويعني رقم مكون من أربع خانات كالتالي 1234.

DATE: وهو التاريخ كما هو معروف والقاعدة لإنشاء الجدول هي:

```
CREATE TABLE table_name
(column1 datatype ,
column2 datatype ,
..
)
```

بعد أن تعرفنا على طريقة إنشاء الجداول سنرى الآن طريقة ربطها نستخدم المحددات أو ما يسمى بـ CONSTRAINTS طبعا المحددات أنواع وأشكال سوف نطرق إلى ما يلزمنا **وهو الـ PRIMARY KEY وهو المفتاح الأساسي** وهو عبارة عن حقل وحيد يتم تحديده بالجدول وهذا الحقل نقوم بتعيينه بهدف عدم تكرار البيانات وهو لا يأخذ قيمة فارغة وإنما يأخذ مجموعة وكذلك يمكن تحديد اكثر من حقل في الجدول لتعينهم على ألا يتكرروا ولكن كوحدة واحدة وعموماً والجدول دائماً يحتوي على PRIMARY KEY واحد.

:CONSTRAINT

هنا نهيأ الجدول ونذكر اسمه ثم نذكر نوعه وما هي الحقول التي تريد تحديدها مع الأخذ بعين الاعتبار أن هذه الحقول لا يمكن أن تتكرر ولنوضح شيئاً مهما أننا لو حددنا مثلاً رقم الموظف وتاريخ ميلاده على أنها مفتاح أساسي فهنا نلاحظ مايلي:

لو أدخلنا رقم الموظف مثلاً Ammar وتاريخ الميلاد 08-03-2000

وأدخلنا رقم الموظف مثلاً Ammar وتاريخ الميلاد 08-03-2001

هل ستقبل (قاعدة البيانات) هذه العملية الجواب..؟ نعم والسبب أننا حددنا في المفتاح الأساسي أن رقم الموظف وتاريخ ميلاده هما واحد ويمنع التكرار في حالة تشابههما لذلك يجب أن نكون حذرين في حالة تحديد المفتاح الأساسي.

```
CONSTRAINT constraint_name PRIMARY KEY (column1, column2, . column_n)
```

أما النوع الثاني فهو FOREIGN KEY وهو المفتاح المرجعي وفي هذا النوع فقط تحدد أن الحقل الذي بالجدول الأول تابع للحقل الأساسي في الجدول الثاني وللتوضيح أكثر مثلاً رقم الجنسية في جدول الموظفين تابع رقم الجنسية في جدول الجنسيات لذلك فالفروض أن يكون أساسي والهدف من هذا كله هو إلزام المستخدم بإدخال أرقام أو بيانات محدده وتفادي إدخال بيانات غير موجود وتصبح بياناتك أقوى وبرنامجك أقوى مع ملاحظة انه عند إنشاء هذا النوع من المحددات يجب أن يكون الجدول المنشأ موجود على قاعدة البيانات. فمثلاً عندما نريد ربط رقم الجنسية بجدول الموظفين برقم الجنسية بجدول الجنسيات يجب أن يكون جدول الجنسيات منشأ قبل جدول الموظفين وهكذا ويمكن الرجوع إلى أكثر من حقل في كلا الجدولين وكذلك يجب أن يكون من نفس النوع.

سنطبق ما سبق بشكل عملي:

طريقة إنشاء الجدول تتم بوضع:

```
CONSTRAINT CONSTRAINT_NAME FOREIGN KEY (column1, column2, ... column_n)
```

هنا نحدد أسماء الحقول في الجدول الحالي

```
REFERENCES parent_table (column1, column2, ... column_n)
```

وهنا نحدد أسماء الحقول مع ذكر اسم الجدول المراد الرجوع إليه

وتصبح القاعدة بالشكل التالي:

```
CONSTRAINT fk_column
 FOREIGN KEY (column1, column2, ... column_n)
 REFERENCES parent_table (column1, column2, ... column_n)
);
```

جدول الجنسيات: وهو يتكون من:

1 . رقم الجنسية 2 . وصف الجنسية 3 . تاريخ الإنشاء 4 . اسم المستخدم

جدول الإدارات: وهو يتكون من:

1 . رقم الإدارة 2 . اسم الإدارة 3 . تاريخ الإنشاء 4 . اسم المستخدم

جدول الموظفين : وهو يتكون من:

1 . رقم الموظف 2 . اسم الموظف 3 . تاريخ الميلاد 4 . الجنس

5 . الجنسية 6 . تاريخ التعيين 6 . الراتب الأساسي 7 . بدلات أخرى

8 . الإدارة التابع لها 9 . تاريخ الإنشاء 10 . اسم المستخدم

كود جدول الجنسيات:

```
CREATE TABLE NATIONALITY(
NAT_NO VARCHAR2(5),
NAT_NAME VARCHAR2(20),
NAT_CRE_DATE DATE,
NAT_CRE_NAME VARCHAR2(50),
CONSTRAINT NAT_PK PRIMARY KEY(NAT_NO)
)
/
```

أما بالنسبة إلى NAT_CRE_DATE , فهي تفيد في حالة تاريخ إنشاء الحقل أما NAT_CRE_NAME فتفيد بإضافة اسم المستخدم الذي قام بإنشاء هذا الحقل.

كود جدول الإدارات:

```
CREATE TABLE DEPARTMENTS(
DPT_NO VARCHAR2(5),
DPT_NAME VARCHAR2(20),
DPT_CRE_DATE DATE,
DPT_CRE_NAME VARCHAR2(50),
CONSTRAINT DPT_NO_PK PRIMARY KEY(DPT_NO)
)
/
```

كود جدول الموظفين:

```
CREATE TABLE EMPLOYEES(
EMP_ID VARCHAR2(10),
EMP_NAME VARCHAR2(50),
EMP_BIRTH_DATE DATE,
EMP_SEX VARCHAR2(1),
NAT_NO VARCHAR2(5),
EMP_HIRE_DATE DATE,
EMP_BASIC_SALARY NUMBER(4),
EMP_ADD_EXCHANGE NUMBER(4),
DPT_NO VARCHAR2(5),
EMP_CRE_DATE DATE,
EMP_CRE_NAME VARCHAR2(20),
CONSTRAINT EMP_ID_PK PRIMARY KEY(EMP_ID),
CONSTRAINT NAT_NO_FK FOREIGN KEY (NAT_NO)
REFERENCES NATIONALITY (NAT_NO),
CONSTRAINT DPT_NO_FK FOREIGN KEY (DPT_NO)
REFERENCES DEPARTMENTS (DPT_NO)
)
/
```

ونلاحظ إن NAT_NO وهو رقم الجنسية وضع باسم مختلف لأنه تابع لجدول أساسي وهو جدول الجنسيات وكذلك الحال بالنسبة ل DPT_NO وهو يرمز إلى رقم الإدارة. أما بالنسبة إلى EMP_CRE_DATE فهي تفيد في حالة تاريخ إنشاء الحقل أما EMP_CRE_NAME فتفيد بإضافة اسم المستخدم الذي قام بإنشاء هذا الحقل طبعاً يجب الأخذ بعين الاعتبار أننا عندما نريد أن نربط حقل في جدول معين بحقل آخر فيجب أن يكون من نفس النوع.

DESCRIBE YOUR_TABLENAME :

وهو أمر يقوم بعرض الحقول التي بالجدول ونوع كل حقل وسوف نلاحظ هذه القيمة NOT NULL أي انه لا يقبل قيمة فارغة وهو المفتاح الأساسي الذي قمنا بتحديده

سابقاً ويمكن كتابته بالاختصار: DESC EMPLOYEES وسوف يعرض بالشكل التالي:

```
DESCRIBE EMPLOYEES <SQL
```

NAME	NULL?	TYPE
EMP_ID	NOT NULL	VARCHAR2(10)
EMP_NAME		VARCHAR2(50)
EMP_BIRTH_DATE		DATE
EMP_SEX		VARCHAR2(1)
NAT_NO		VARCHAR2(5)
EMP_HIRE_DATE		DATE
EMP_BASIC_SALARY		NUMBER(4)
EMP_ADD_EXCHANGE		NUMBER(4)
DPT_NO		VARCHAR2(5)
EMP_CRE_DATE		DATE
EMP_CRE_NAME	VARCHAR2(20)	

وهكذا تم إنشاء وربط الجداول مع بعضها

الأسئلة:

1–بين كيف يتم إنشاء جدول للطلاب مكون من ثلاثة حقول حيث يقبل رقم الطالب أحرف وأرقام وطوله 5 واسم الطالب يقبل أحرف وأرقام ومكون من 20 حرف وتاريخ ميلاد الطالب ؟

2 . ما المقصود بالمحددات التالية:

أ . المفتاح الأساسي PRIMARY KEY

ب . المفتاح المرجعي FOREIGN KEY ؟

3 . بيّن بمثال كيف يتم ربط جدولين ببعضهما باستخدام الـ FOREIGN KEY.

4 . اجب بنعم أو لا مع ذكر السبب:

يوجد لدينا جدول مكون من حقلين أساسيين PRIMARY KEY ولنفرض انهما رقم الموظف ورقم إدارته فهل تقبل قاعدة البيانات الإضافة لو قمنا بإضافة:

رقم الموظف = e0001 ورقم الإدارة = 10

وإضافة رقم الموظف = e0002 ورقم الإدارة= 10

العمليات التي تتم على الجداول

إضافة Insert تعديل Update حذف Delete

1 . الإضافة insert:

المقصود بالإضافة هو عملية إضافة مجموعة من البيانات على مجموعة من الحقول في جدول معين وتكون طريقة الإضافة بطباعة الأمر insert ثم نكتب into ثم اسم الجدول ونفتح قوس ثم نكتب أسماء الحقول المراد إدخال البيانات فيها مع الأخذ بعين الاعتبار أن نضيف أيضاً إلى الجداول التي لا تقبل القيم الفارغة مثل المفتاح الأساسي Primary key وكذلك الحقول المربوطة بحقول أخرى الـ key foreign وبعدها نكتب الأمر values ونفتح قوس ويجب أن يكون ترتيب القيم بنفس ترتيب الحقول في البداية مع الأخذ بعين الاعتبار أن نوع الحقل VARCHAR2 يوضع بين علامتين 'VALUE' والتاريخ DATE يجب أن تضعه أيضاً بين علامتين 'VALUE ' أما الأرقام فتوضع بدون علامات CUTAION SINGLE ويجب عمل commit; وهو أمر يطبع لتخزين معلومة على قاعدة البيانات وهي تستخدم بعد الإضافة أو الحذف أو التعديل لتأكيد العملية ويمكن استخدامها أيضاً بعد مجموعة من العمليات مثلا إضافة عدد واحد من السطور أو مجموعة من السطور.

القاعدة:

```
INSERT INTO table
(column-1, column-2, ... column-n)
VALUES
(value-1, value-2, ... value-n);
```

ونلاحظ هنا إننا نستخدم الفاصلة بين كل قيمة وأخرى.

مثال:

إضافة على جدول الجنسيات حيث يتألف من :

NAT_NO رقم الجنسية

NAT_NAME اسم الجنسية

NAT_CRE_DATE تاريخ الإضافة

NAT_CRE_NAME تاريخ التعديل

```
INSERT INTO
NATIONALITY(NAT_NO,NAT_NAME,NAT_CRE_DATE,NAT_CRE
_NAME)
VALUES('001','SAUDI',SYSDATE,USER);
INSERT INTO
NATIONALITY(NAT_NO,NAT_NAME,NAT_CRE_DATE,NAT_CRE
_NAME)
VALUES('002','JORDAN',SYSDATE,USER);
INSERT INTO
NATIONALITY(NAT_NO,NAT_NAME,NAT_CRE_DATE,NAT_CRE
_NAME)
VALUES('003','EGYPT',SYSDATE,USER);
INSERT INTO
DEPARTMENTS(DPT_NO,DPT_NAME,DPT_CRE_NAME,DPT_CRE
_DATE)
VALUES ('DP01','EMPLOYEES',USER,SYSDATE);
INSERT INTO
DEPARTMENTS(DPT_NO,DPT_NAME,DPT_CRE_NAME,DPT_CRE
_DATE)
VALUES ('DP02','ACCOUNT',USER,SYSDATE);
INSERT INTO
DEPARTMENTS(DPT_NO,DPT_NAME,DPT_CRE_NAME,DPT_CRE
_DATE)
VALUES ('DP03','COMPUTER',USER,SYSDATE)
COMMIT;
```

2 . الحذف DELETE:

حذف مجموعة من السجلات ضمن شرط معين أو من غير شرط:

نكتب الأمر DELETE ثم اسم الجدول

أو تكتب الأمر DELETE ثم تكتب FROM ثم تكتب اسم الجدول

```
DELETE FROM table name
DELETE TBALE_NAME
```

مثال:

هنا سنقوم بحذف جميع سجلات جدول الإدارات وجدول الجنسيات.

```
DELETE DEPARTMENTS;
DELETE FROM NATIONALITY;
COMMIT;
```

ولكن يفضل استخدام الشرط حيث تحدد الحقل الذي تريد حذفه.

استخدام WHERE CONTION : وهو شرط لا تتم عملية الحذف إلا بتحققه ويمكن وضع أكثر من شرط والفصل بينهما عن طريق AND

```
DELETE FROM DEPARMENTS
WHERE DPT_NO='DP02'
COMMIT;
```

3 . التعديل UPDATE:

هو القيام بعمل تعديل على الحقول أيضاً وعلى الجدول كاملاً وللقيام بالتعديل نكتبUPDATE ثم نكتب اسم الجدول ثم نكتب SET ثم اسم الحقل المراد تعديله.

```
update table_name set field_name = value
```

مثال:

```
UPDATE DEPARTMETS SET DPT_NAME='ALL'
```

كما يمكن أن نعدل أكثر من قيمة باستخدام الفاصلة وكذلك باستخدام الـ where condition لنحدد الحقول المراد التعديل عليها كما في الشكل التالي:

```
update table name set field name = value,field_name2=value
```

مثال:

```
UPDATE DEPARTMETS SET DPT_NAME='ALL' ,DPT_NO='02'
WHERE DPT_NO='DPT02'
```

مع ملاحظة مراعاة أنواع الحقول في الجداول كما ذكرنا سابقاً

Select Statement أنواعها وكيف يمكن التعامل معها:

تعرّف الـ Select Statement على إنها أمر من خلاله يتيح لنا إحضار بيان أو مجموعة بيانات من جدول واحد أو أكثر وبطرق مختلفة وحسب الشرط الذي تضعه كما يمكن استخدام الـ statement select في معالجة العمليات على الجداول وهي Update Delete Insert وكذلك يمكن استخدامها مع عدة function واقصد بها معادلات جاهزة.

ABAHE

1 . جملة SELECT البسيطة:

نكتب SELECT ثم اسم الحقول أو إذا كنت تريد عرض كل الحقول اكتب * أي نجمة ثم FROM وهنا تعطي إيعاز أن تحدد اسم الجدول بعد الـ FROM فتصبح بالشكل التالي:

```
SELECT *
FROM <table name>;
```

مثال :

```
SELECT * FROM NATIONALITY
```

هذا المثال يعرض لنا جميع محتويات جدول الجنسيات.

2 . جملة الـ SELECT التي تحدد فيها أسماء الحقول في الجدول:

فهي لا تختلف عن الحالة الأولى وإنما تستبدل النجمة * بأسماء حقول

```
SELECT <column name, column name, ..., <column name>
FROM <table name>;
```

مثال:

```
SELECT DPT_NO,DPT_NAME FROM DEPARTMENTS
```

والفرق هنا هو أننا حددنا بالـSELECT اسم الإدارة ورقمها فقط .

3 . جملة الـ SELECT مع الـ WHERE CONDTION:

وهي تخضع للشرط مع WHERE CONDITION

```
SELECT *
FROM <table_name>
WHERE ....
```

مثال:

```
SELECT * FROM DEPARTMENTS
WHERE DPT_NO='DP01'
```

4 . استخدام الـ SELECT مع UPDATE و INSERT و DELETE وكذلك إنشاء الجدول CREATE TABLE:

أ . استخدام الـ SELECT مع INSERT:

يتم الإضافة على الجدول باستخدام جملة الـ INSERT وتحدد بالمقابل جملة الـ SELECT وهذه الحالة تؤخذ إذا كنا نريد نسخ بيانات جدول من آخر للإضافة داخل نفس الجدول وذلك حسب البيان الذي يأتي من SELECT.

```
INSERT INTO TABLE_NAME
(COLOUMN1...,COLOUMN2...)
SELECT COLOUMN1...,COLOUMN2...)
FROM OTHER_TABLE_NAME
WHERE .....
```

مثال:

```
INSERT INTO DEPARTMENTS(DPT_NO,DPT_NAME)
SELECT NAT_NO,NAT_NAME
FROM NATIONALITY;
```

على افتراض أن نوع الحقول في كل جدول متساوية

ب . استخدام الـ SELECT مع DELETE:

```
DELETE FROM TABLE_NAME
WHERE COLOUMN_NAME [COLOR=blue]IN[/COLOR]
(SELECT COLUMN FROM TABLE NAME
WHERE ....
```

هنا استخدمنا FUNCTION تستخدم مع WHERE CONDTION وهي IN ويقصد بها أن هل القيمة معينة موجودة ضمن جملة الـ SELECT طبعا سوف نأتي لها بالتفصيل ان شاء الله ولكن نستعرضها هنا فقط بشكل مختصر.

ج . استخدام الـ SELECT مع الـ UPDATE:

```
UPDATE TABLE_NAME SET COUMN_NAME= SELECT
CLOUMN_NAME FROM TABLE_NAME WHERE ....
```

على أساس أن جملة الـ SELECT ترجع قيمة واحدة لا أكثر

د . استخدام الـ SELECT مع TABLE CREATE:

القاعدة:

```
CREATE TABLE new_table
 AS (SELECT column_1, column2, ... column_n
 FROM old_table_1, old_table_2, ... old_table_n);
```

وهنا تنشئ جدول باستخدام جملة الـ SELECT

مثال:

```
CREATE TABLE DEPT
AS (SELECT DPT_NO,DPT_NAME FROM DEPARTMENTS
WHERE DPT_NO='DP001')
```

الدوال functions في الأوراكل

تساعد الدوال في فرز وتصنيف وترتيب البيانات:

1 . الدوال التي تساعد في فرز وتصنيف البيانات هي:

ALIASES: وتسمى بالأسماء المستعارة يعني نستبدل اسم الحقل بأي اسم نحن نحدده كي تعرض في العناوين.

```
select sysdate as "My Date" from dual;
My Date
--------
22-10-06
```

DISTINCT: وتستخدم لتصنيف البيانات المتكررة وتأتي بعد جملة الـ SELECT مباشرة ومن مميزاتها إنها تتعامل مع البيانات المتشابه بالحقول التي تأتي بجملة الـ SELECT. فلو كانت لدينا البيانات التالية:

اسم الموظف عمار تاريخ ميلاده 02-07-1980

اسم الموظف عمار تاريخ ميلاده 02-07-1981

واستخدمنا جملة الـ SELECT في إحضار اسم الموظف وتاريخ ميلاده مع DISTINCT نلاحظ انه سوف يرجع القيمتين والسبب لأن تاريخ الميلاد مختلف بينما الاسم متشابه. لكن لو طلبنا إحضار اسم الموظف فقط فسوف يحضر قيمة واحدة وهي اسم الموظف وقاعدة استخدامها هي:

```
SELECT DISTINCT CLOUMN1...,COLUMN2.. FROM
TABLE_NAME
```

```
SELECT DISTINCT DPT_NAME FROM DEPARTMENTS
```

UPPER و LOWER: وتستخدم لتحويل الأحرف الإنجليزية إلى CAPITAL باستخدام UPPER وإلى SMMAL باستخدام LOWER

القاعدة:

```
SELECT UPPER(CLOUMN OR STRING) FROM TABLE NAME
SELECT LOWER(CLOUMN OR STRING) FROM TABLE NAME
```

مثال:

```
SELECT LOWER('ADMIN05') FROM DUAL;
SELECT UPPER('admin05') FROM DUAL;
```

وجدول الـ DUAL شرحنا عنه سابقاً.

```
SELECT UPPER('DPT_NAME'),LOWER(DPT_NAME FROM
DEPARTMENTS
```

هنا نعرض اسم الإدارة مرة CAPITAL ومرة SMALL مع ملاحظة إن هذه الدوال لا تنطبق بالطبع على أحرف اللغة العربية.

initcap: وتستخدم لتحويل حرف من كل كلمة إلى capital

```
select initcap(cloumname or string) from table_name
```

```
select initcap('admin05') from dual;
```

NVL: وتستخدم للتعويض عن القيم الفارغة الـ NULL إلى قيم نحن نحددها واستخداماتها في الأرقام والحروف.

```
SELECT NVL(CLUMONAME,YOUR_VALUE) FROM
TABLE NAME
```

مثال:

```
SELECT NVL(dpt_name,'not found') from departments
select NVL(number,0) from dual
```

to char : وهي تستخدم للتحويل كل شيء إلى قيمة string أو ترتيب مثلاً تاريخ حسب قناع معين.

```
select to_char(cloumn,'ur mask') from table name
select to_char(cloumn) from table name
select to_char(sysdate,'dd-mm-yyyy') from dual
select to_char(sysdate) from dual;
```

ORDER BY: وهي دالة تعنى بترتيب البيانات بشكل منظم إما تصاعدي أو تنازلي وتستخدم بجملة الـ SELECT وتكتب في آخر جملة الـ SELECT حيث نكتبها ونكتب اسم الحقل ويكتب بعدها إذا كنت الترتيب تصاعدي ويرمز لها بـ ASC ثم تنازلي ويرمز لها بالرمز DESC.

```
ORDER BY COLUMN_NAME ASC

SELECT DPT_NO,DPT_NAME FROM DEPARTMENTS
ORDER BY DPT_NO ASC
```

GROUP BY : وهو ما يقصد به بالمجموعات ويستخدم الـGROUP BY لتمثيل أو تقسيم المعلومات على شكل مجموعات سواء مجموعة واحدة أو عدة مجموعات.

```
SELECT column1, column2, ... column_n, aggregate_function
(expression)
FROM tables
WHERE predicates
GROUP BY column1, column2, ... column_n;
```

ونلاحظ كيف يمكن استخدامه وسوف تتضح فكرته عند استخدامه مع معادلات أخرى فالهدف هو مثلا إيجاد عدد الأقسام في كل إدارة.

Count: وهو عبارة عن عدد يستخدم لإرجاع عدد الحقول في الجدول ويمكن استخدامه بالشكل التالي:

```
SELECT COUNT(expression)
FROM tables
WHERE predicates;
```

طبعاً الـ COUNT يرجع إجمالي العدد لذلك نستطيع أن نضع اسم حقل ليرجع لنا عدد بياناته مثلاً أو نستطيع أن نرجع عدد بيانات جدول وذلك باستخدام النجمة * لذلك يجب توضيح نقطة انه لو طلب منا استرجاع بيانات عدد الإدارات مع اسم الإدارة لذلك يجب أن نستخدم معها GROUP BY.

```
SELECT COUNT(DPT_NO) as "Number of Dept",DPT_NAME FROM
DEPARTMENTS
GROUP BY DPT_NAME
```

لا نستخدم مع COUNT الـ GROUP BY في حالة إذا كنا نسترجع فقط عدد شيء معين.

SUM: وهو إيجاد مجموع عدد من الأرقام:

```
SELECT SUM(expression )
FROM tables
WHERE predicates;
```

يمكن أن يستخدم الـ SUM مع DISTINCT

مثال:

```
SELECT SUM(DISTINCT salary) as "Total Salary"
FROM employees
WHERE salary > 25000;
```

هنا يتم جمع البيانات غير المكررة .

ويمكن عمل عمليات أثناء الجمع كطرح أو ضرب أو قسمة نتيجة على رقم معين من كل حقل كما في المثال التالي:

```
SELECT SUM(sales*0.10) as "Commission"
FROM order details;
```

وتستخدم الـSUM مع BY GROUP

مثال:

```
SELECT department, SUM (sales) as "Total sales"
FROM order_details
GROUP BY department;
```

MAX: وهو إرجاع اكبر قيمة بين مجموعة من البيانات.

```
SELECT MAX (expression)
FROM tables
WHERE predicates;
```

MIN: وهو عكس الـ MAX حيث يرجع اقل قيمة بين مجموعة من البيانات

```
SELECT MIN (expression)
FROM tables
WHERE predicates;
```

مثال:

```
SELECT MIN(salary) as "Lowest salary"
FROM employees;
```

ونلاحظ كيف انه وضعنا الـ DPT_NAME لأنه عبارة عن رمز فردي أما COUNTDPT_NOعبارة عن مجموعة لذلك يستخدم معها الـ GROUP BY

وكذلك الحال بالنسبة الـ MAX,MIN,SUM.

HAVING: ويستخدم لفرز البيانات الناتجة عن GROUP BY حيث نستطيع وضع شرط معين من خلالها وهو استخدام فقط أعمدة الـ GROUP BY أو أي شيء ينتمي إلى مجموعة.

القاعدة :

```
SELECT column1, column2, ... column_n, aggregate_function
(expression)
FROM tables
WHERE predicates
GROUP BY column1, column2, ... column_n
HAVING condition1 ... condition_n;
```

ويستخدم أيضاً مع SUM,MAX,MIN,COUNT

أمثلة:

مثال على SUM:

```
SELECT department, SUM(sales) as "Total sales"
FROM order_details
GROUP BY department
HAVING SUM(sales) > 1000;
```

مثال على [COUNT[/COLOR] :

```
 [COLOR=red]
SELECT department, COUNT(*) as "Number of employees"
FROM employees
WHERE salary > 25000
GROUP BY department
HAVING COUNT(*) > 10;
```

مثال على MIN:

```
SELECT department, MIN(salary) as "Lowest salary"
FROM employees
GROUP BY department
HAVING MIN(salary) = 35000;
```

مثال على [MAX[/COLOR] :

```
[COLOR=red]
SELECT department, MAX(salary) as "Highest salary"
FROM employees
GROUP BY department
HAVING MAX(salary) < 50000;
```

BETWEEN / (من إلى) أو(بين):

وتستعمل لوضع مقارنة في جملة الشرط

```
SELECT columns
FROM tables
WHERE column1 between value1 and value2;
```

مثال:

```
SELECT *
FROM suppliers
WHERE supplier_id between 4000 and 4500;
```

NOT BETWEEN والمقصود بها(ما ليس بين):

وهو عبارة عن إحضار البيانات التي لا تنتمي إلى جملة الشرط باستخدام الـ NOT BETWEEN

```
SELECT *
FROM suppliers
WHERE supplier_id not between 4000 and 4500;
```

الأسئلة:

1 . أضف القيم التالية إلى جدول الإدارات رقم الإدارة =DP04 اسم الإدارة =marketing؟

2 . ما هي أفضل الطرق في عملية الحذف أو التعديل:

باستخدام الـ where condition أو بالطريقة العادية ولماذا ؟

3 . بيّن كيف يمكن تعديل أكثر من قيمة في جدول في جملة تعديل واحدة وباستخدام الشرط؟

4 . ما اسم الدالة التي تقوم بتثبيت البيانات؟

5 . اكتب جملة select تعرض من خلالها اسم الإدارة ورقمها وترتبها حسب الإدارة؟

6 . اكتب جملة select تعرض البيانات غير المكررة لجنسية الموظف من جدول الجنسيات؟

7 . أعطي مثال على إنشاء جدول باستخدام جملة الـ select على أن تكون الإدارة رقم DP01.

8 . أعطي مثال على إضافة وحذف وتعديل الجدول باستخدام الـ statement select؟

9 . أعطي مثال تبين فيه شكل التاريخ 01-2006 حيث إن 01 هو الشهر و 2006 هو السنة من خلال استخدام دالة to_char؟

10 . بين باستخدام الدوال التالية nvl و initcap كيف يمكن استغلالها في جملة select؟

11 . بين بمثال كيف يمكن أن نجمع بيانات غير مكررة باستخدام sum ؟

12 . بين عدد الإدارات واسم كل إدارة وترتيبها بشكل تصاعدي؟

13 . بين عدد واسم اصغر إدارة ؟

14 . بين ما هو الفرق في استخدام between و not between ؟

15 . بين عدد الإدارات واسم كل إدارة بشرط أن يكون العدد أكبر من 5 ؟

TO_DATE: تقوم هذه الدالة بتحويل الـ STRING إلى تاريخ فلو أخذنا هذه على أساس إنها STRING فيمكن أن نحولها إلى DATE .

```
to date( string1, [ format mask ], [ nls_language ] )
```

حيث أن STRING1 هو عبارة عن التاريخ المراد تحويله .

FROMAT_MASK : هو عبارة عن الصيغة المراد تحويل التاريخ إليها وقد استعرضنا فيما سبق استخدامات أو تحويلات التاريخ إلى STRING.

NLS_LANGUAGE : وهو صيغة التاريخ وهي اختيارية حيث نستطيع وضع صيغة التاريخ الهجري مثلاً.

مثال:

```
to_date('2003/07/09', 'yyyy/mm/dd') would return a date value of July 9, 2003.
to_date('070903', 'MMDDYY') would return a date value of July 9, 2003.
to_date('20020315', 'yyyymmdd') would return a date value of Mar 15, 2002.
```

LAST_DAY: وهو إرجاع تاريخ آخر شهر فقط؛؛ فإذا أدخلنا أي تاريخ فهو سيرجع لنا آخر يوم في هذا الشهر.

القاعدة:

```
last day( date )
```

مثال:

```
last day(to_date('2003/03/15' ,'yyyy/mm/dd')) would return Mar 31, 2003
last day(to_date('2003/02/03' ,'yyyy/mm/dd')) would return Feb 28, 2003
last day(to_date('2004/02/03' ,'yyyy/mm/dd')) would return Feb 29, 2004
```

ADD_MONTHS : وهو لإضافة شهر على التاريخ المعطى

القاعدة:

```
add months( date1, n )
```

حيث n هو عدد المراد إضافته وممكن إنقاص الأشهر أو زيادتها

مثال:

```
add_months('01-Aug-03', 3) would return '01-Nov-03'
add_months('01-Aug-03', -3) would return '01-May-03'
add_months('21-Aug-03', -3) would return '21-May-03'
add_months('31-Jan-03', 1) would return '28-Feb-03'
```

ABAHE

NEXT_DAY: وهو إضافة يوم أو عدد من الأيام على التاريخ المعطى.

```
next date(date1,n)
```

حيث الـ n هو عدد الأيام المراد إضافتها أو نستطيع وضع اسم اليوم في هذا الأسبوع فيحضر لنا تاريخه.

```
NEXT_DAY('01-01-2006',5) RETURN '06-01-2006'
NEXT_DAY('15-01-2006',-5) RETURN '10-01-2006'
```

```
next_day('01-Aug-03', 'TUESDAY')
```

TRIM: تستخدم لحذف الأحرف في الكلمة سواء من جهة اليمين أو اليسار أو من الوسط

```
trim( [ leading | trailing | both [ trim character ] ] string1 )
```

حيث إن leading تستخدم لحذف الحروف من أول الكلمة trailing تستخدم لحذف الحروف من آخر الكلمة both تستخدم لحذف الحروف من أي جهة موجود في الكلمة (يمين/يسار/وسط) وإذا لم يتم استخدام هذه المتغيرات مع trim فان الحذف يكون للفراغات فقط.

trim(' tech ') سوف نرجع القيمة بدون فراغات 'tech'

trim(' ' from ' tech ') سوف ترجع الكلمة بدون فراغات ولاحظا أننا استخدمنا from 'tech'

trim(leading '0' from '000123') سوف يتم حذف الأصفار من بداية الجملة'123 '

trim(trailing '1' from 'Tech1') سوف يتم حذف الواحد من الآخر 'Tech'

trim(both '1' from '123Tech111') سوف يتم حذف الرقم واحد من اليمين والشمال'23 Tech'

Ltrim : تستخدم للحذف من جهة اليسار. ومن مميزاتها إنها تحذف الأحرف المراد حذفها سواء كانت مرتبه أو لا.

```
ltrim( string1, [ trim_string ] )
```

أمثلة:

```
ltrim(' tech'); would return 'tech'
ltrim(' tech', ' '); would return 'tech'
ltrim('000123', '0'); would return '123'
ltrim('123123Tech', '123'); would return 'Tech'
ltrim('123123Tech123', '123'); would return 'Tech123'
ltrim('xyxzyyyTech', 'xyz'); would return 'Tech'
ltrim('6372Tech', '0123456789'); would return 'Tech'
```

وهنا نلاحظ كيف نحذف الـ y رغم تكرارها

Rtrim: نفس مبدأ عمل trim ولكنها عكس Ltrim حيث تحذف من جهة اليمين

```
rtrim( string1, [ trim_string ] )
```

أمثلة:

```
rtrim('tech '); would return 'tech'
rtrim('tech ', ' '); would return 'tech'
rtrim('123000', '0'); would return '123'
rtrim('Tech123123', '123'); would return 'Tech'
rtrim('123Tech123', '123'); would return '123Tech'
rtrim('Techxyxzyyy', 'xyz'); would return 'Tech'
rtrim('Tech6372', '0123456789'); would return 'Tech'
```

LPAD: وتستخدم لزيادة عدد من الفراغات أو تكرار الكلمة من جهة اليسار طبعاً سوف يبدأ من جهة اليسار ليزيد.

```
lpad( string1, padded length, [ pad string ] )
```

حيث padded length هو العدد المراد زيادته طبعاً ينقص منه طول الكلمة فلو فرضنا أعطانا الكلمة RRR وأراد أن يزيد عليها + وأعطى العدد 10 طبعاً طول الكلمة هي 3 يعني سوف يزيد 7 نجم فيصبح الشكل الكلمة كما يلي +++++++RRR

أمثلة:

```
lpad('tech', 7); would return ' tech'
lpad('tech', 2); would return 'te'
lpad('tech', 8, '0'); would return '0000tech'
lpad('tech on the net', 15, 'z'); would return 'tech on the net'
lpad('tech on the net', 16, 'z'); would return 'ztech on the net'
```

RPAD: وتستخدم لزيادة عدد من الفراغات أو تكرار الكلمة من جهة اليمين طبعاً سوف يبدأ من جهة اليمين ليزيد .

```
rpad( string1, padded_length, [ pad_string ] )
```

أمثلة :

```
rpad('tech', 7); would return 'tech '
rpad('tech', 2); would return 'te'
rpad('tech', 8, '0'); would return 'tech0000'
rpad('tech on the net', 15, 'z'); would return 'tech on the net'
rpad('tech on the net', 16, 'z'); would return 'tech on the netz'
```

SUBSTR: يستخدم لقطع جملة معينة أو كلمة معينة ويجب أن تحدد بداية القطع وعدد الأحرف أو طول المراد قطعه.

```
substr( string, start position, [ length ] )
```

إذا كان بداية القطع صفر فإن الـ SUBSTR تغير القيمة إلى واحد وإذا كان الرقم بداية القطع هو موجب فانه يبدأ من بداية الكلمة يعني اليسار وإذا كان الرقم بداية القطع هو سالب فانه يبدأ من نهاية الكلمة يعني اليمين.

أمثلة:

```
substr('This is a test', 6, 2) would return 'is'
substr('This is a test', 6) would return 'is a test'
substr('TechOnTheNet', 1, 4) would return 'Tech'
substr('TechOnTheNet', -3, 3) would return 'Net'
substr('TechOnTheNet', -6, 3) would return 'The'
substr('TechOnTheNet', -8, 2) would return 'On'
```

INS: يستخدم لإرجاع موقع الحرف في الكلمة وكذلك يستخدم للبحث عن حرف في كلمة فإذا وجده يرجع لموقعه وإلا يرجع القيمة صفر ويمكن أن يستغل في معالجة بعض البيانات.

```
instr( string1, string2, [ start_position ], [ nth_appearance ] )
```

أمثلة:

```
instr('Tech on the net', 'e') would return 2; the first occurrence of 'e'
instr('Tech on the net', 'e', 1, 1) would return 2; the first occurrence of 'e'
instr('Tech on the net', 'e', 1, 2) would return 11; the second occurrence of 'e'
instr('Tech on the net', 'e', 1, 3) would return 14; the third occurrence of 'e'
instr('Tech on the net', 'e', -3, 2) would return 2.
```

LENGTH: يستخدم لإرجاع طول الكلمة أو الجملة.

```
length( string1 )
```

مثال:

```
length(NULL) would return NULL.
length('') would return NULL.
length('Tech on the Net') would return 15.
length('Tech on the Net ') would return 16.
```

TRANSLATE: يستخدم لاستبدال أحرف معينه نحن نحددها بأخرى حيث يبدل كل حرف مقابله حرف يعني لو حددنا الأحرف RTY من الكلمة RTYYRT ونريد إبدالها بـ QW3 نلاحظ انه بدل الـ Y سوف يبدلها بـ 3 وبدل T سوف يبدلها بـ W وبدل الـ R سوف يبدلها بـ Q لتصبح الكلمة QW33QW

القاعدة:

```
translate( string1, string_to_replace, replacement_string
)
```

حيث string_to_replace هو الأحرف المراد إبدالها بالكلمة replacement_string الأحرف الجديدة المراد إبدالها بالأحرف القديمة.

ABAHE

أمثلة:

```
translate('1tech23', '123', '456); would return '4tech56'
translate('222tech, '2ec', '3it'); would return '333tith'
```

REPLACE: وتستخدم لحذف أو استبدال الكلمات replace(string1,

```
[ string_to_replace, [ replacement_string )
```

حيث تبين هنا أن الكلمة المراد استبدالها وهي تشبه الـ TRANSLATE تقريباً والفرق هو إن الـ TRANSLATE تبدل الأحرف فقط أما الـ REPLACE تعمل على الحذف والاستبدال إذا لم نحدد لها الأحرف المراد الاستبدال بها فيما لو استخدمت فقط بالشكل التالي:

```
replace('123123tech', '123'); would return 'tech'
replace('123tech123', '123'); would return 'tech'
```

أما إذا استخدمت بالشكل التالي فإنها تشبه عمل TRANSLATE

```
replace('222tech, '2', '3'); would return '333tech'
replace('0000123', '0'); would return '123'
replace('0000123', '0', ' '); would return ' 123'
```

ABAHE

CONCAT: يستخدم لدمج كلمتين أو أكثر مع بعض.

```
concat( string1, string2 )
```

ويمكن استبدالها بالرمز ||

```
concat('Tech on', ' the Net'); would return 'Tech on the Net'.
concat('a', 'b') would return 'ab'.
'Tech on'|| ' the Net' would return 'Tech on the Net'.
'a'|| 'b' would return 'ab'.
```

الأسئلة:

لدينا الجدول التالي يحتوي على اسم الموظف ورقم الموظف والبيانات بالشكل التالي:

emp_no	emp_name	birth_date
10	ali ahmed ali	20-01-1970
20	Admin05 05 05	20-02-1976
30	ahmed moheme	20-01-1960
40	50dfgdr50	20-10-1950

1–اعرض أسماء الموظفين وتاريخ ميلادهم بزيادة 5 شهور ؟

2–اعرض أسماء الموظفين وتاريخ ميلادهم على أن يكون تاريخ الميلاد زيادة 5 أيام؟

3–احذف أول 3 أحرف من كل اسم ؟

4–استخدم الـ concat لدمج اسم الموظف ورقمه ؟

5–استبدل الـ الحرف a ب e من كل اسم ؟

6–بين موقع الحرف m من كل اسم ؟

7–بين طول كل اسم من كل حرف ؟

8–استبدل آخر 3 أحرف من كل اسم بحرف v واعرضها ؟

9–أضف 9 نجم من جهة اليمين لكل اسم مرة وكن جهة اليسار مرة أخرى ؟

10–ابدأ القطع لكل اسم من الموقع الثالث ؟

أشكال الـ(Subqueries):

1- WHERE CONDITION:

```
select * from all_tables tabs
where tabs.table_name in (select cols.table_name
 from all_tab_columns cols
 where cols.column_name = 'SUPPLIER_ID');
```

نلاحظ أن محور الحدث كله حول WHERE واستخدمنا الـ IN لتوضيح مفهوم هل هذا البيان موجود ضمن مجموعة من البيانات في جدول آخر وكأننا نقول استعرض لنا جميع البيانات من الجدول all_tables على أن يكون مثلاً اسم الجدول موجود ضمن جملة select أخرى بحيث ترجع جميع أسماء الجداول.

```
where cols.column_name in (select * from table_name)
```

مثال توضيحي:

استعرض اسم الموظف ورقمه على أن يكون الموظفين في الإدارة العامة؟

```
SELECT EMP_NAME,EMP_NO FROM EMPLOYEES
WHERE DPT_NO IN (SELECT DPT_NO FROM
DEPARTMENTS
WHERE DPT_NAME='الإدارة العامة'
```

هنا سوف يعرض أسماء الموظفين الذين ينتمون إلى الإدارة رقم 10 مثلا على أن تكون هذه الإدارة ضمن جملة الـ SELECT الثانية وتكون اسمه (الإدارة العامة).

2- UNION Query: ويقصد بها دمج استعلامين لحقلين متشابهين أو أكثر في جدولين مختلفين. ويستفاد منها بعدم إظهار البيانات المتكررة على أن يكون عدد الحقول في الجدول الأول مساوي لعدد الحقول للجدول الثاني.

```
select field1, field2, . field_n
from tables
UNION
select field1, field2, . field_n
from tables;
```

مثال:

```
select supplier_id
from suppliers
UNION
select supplier_id
from orders;
```

استخدام With ORDER BY Clause مع union:

توضع في آخر الـ unio وممكن أن ترمز لاسم الحقل الأول برقم 1وهكذا.....

```
select supplier_id, supplier_name
from suppliers
where supplier_id > 2000
UNION
select company_id, company_name
from companies
where company_id > 1000
ORDER BY 2;
```

رقم 2 هو رمز supplier_name وكذلك company_name ولايعني الرقم عدد الحقول ولكن تعني الترتيب فلو وحدنا الأسماء لكتبنا الأسماء بعد Order By

```
"CODE]select supplier_id, supplier_name as "name
suppliers from
2000 < where supplier_id
```

```
UNION
"company_name as "name ,select company_id
from companies
1000 < where company_id
[BY name;[/CODE ORDER
```

استخدام Query UNION ALL:

هي نفس مبدأ عمل الـ union ولكن تعرض جميع البيانات في الجدولين مع التكرار

```
select field1, field2, . field_n
from tables
UNION ALL
select field1, field2, . field_n
from tables;
```

مثال:

```
select supplier_id
from suppliers
UNION ALL
select supplier_id
from orders;
```

-y3- INTERSECT Quer :

ومبـدأ عملـه يـشبه الـ union وهـو دمـج البيانـات مـع فـارق انـه يعـرض البيانـات الموجودة في الجدول الأول والتي غير موجودة في الجدول الثاني يعني يعرض الغير مكرر فقط select field1, field2, . . field_n

```
from tables
INTERSECT
select field1, field2, . field_n
from tables;
```

مثال:

```
select supplier_id
from suppliers
INTERSECT
select supplier_id
from orders;
```

MINUS Query: هي عملية إرجاع البيانات في الاستعلام الأول والغير موجود في الاستعلام الثاني

```
select field1, field2, . field_n
from tables
MINUS
select field1, field2, . field_n
from tables;
```

مثال:

```
select supplier_id
from suppliers
MINUS
select supplier_id
from orders;
```

ويجب أن يكون عدد الحقول متشابه والنوع كذلك. ومعنى المثال السابق (استعرض البيانات في جملة الاستعلام الأولى والغير موجودة بالاستعلام التالي) كأن تقول نريد أن نعرض أسماء الموردين الذين لم يتم الطلب على منتجاتهم.

ALTER TABLE: هي عبارة عن تعليمة أو أمر يتم من خلاله التعديل على محتويات الجدول وهي على أنواع:

Adding column(s) to a table: وهو إضافة حقل أو عدة حقول على جدول موجود على قاعدة البيانات.

القاعدة:

```
ALTER TABLE table_name
ADD column_name column-definition;
```

حيث نضع ALTER TABLE ثم اسم الجدول ونضع ADD ثم اسم الحقل وبعدها نضع نوع الحقل.

مثال:

```
ALTER TABLE supplier
ADD supplier_name varchar2(50);
```

لإضافة مجموعة من الحقول.

```
ALTER TABLE table_name
ADD ( column_1 column-definition,
column_2 column-definition,
...
column_n column_definition );
```

Modifying column(s) in a table: وهو التعديل على الجدول بحيث نعدل نوع الحقل مع الأخذ بعين الاعتبار أننا لا نستطيع تغيير حالة الحقل من رقم إلى حروف إلا إذا كان فارغاً ونستطيع تحويل الأرقام إلى حروف .

القاعدة:

```
ALTER TABLE table_name
MODIFY column_name column_type;
```

هنا نستخدم MODIFY للتغير وإشعار الـ SQL بالقيام بعملية التغيير.

مثال:

```
ALTER TABLE supplier
MODIFY supplier_name varchar2(100) not null;
```

قاعدة التعديل على مجموعة من الحقول:

```
ALTER TABLE table_name
MODIFY ( column_1 column_type,
column_2 column_type,
...
column_n column_type );
```

Drop-column(s) in a table: وهو حذف حقل من الجدول

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

مثال:

```
ALTER TABLE supplier
DROP COLUMN supplier_name;
```

- **Rename column(s) in a table** :(2 NEW in Oracle 9i Release)

وهو تغير اسم الحقل بالجدول وهي فقط في نسخة أوراكل i R29

```
ALTER TABLE table_name
RENAME COLUMN old_name to new_name;
```

مثال:

```
ALTER TABLE supplier
RENAME COLUMN supplier_name to sname;
```

TABLE DROP:

القاعدة:

```
DROP TABLE table_name;
```

مثال:

```
DROP TABLE supplier;
```

لو كان الجدول مرتبط بعلاقات مع جدول آخر فقط نضيف CASCADE CONSTRAINTS

مثال:

```
DROP TABLE supplier CASCADE CONSTRAINTS;
```

الأسئلة:

1 . بين استخدام Union في جملة Select ؟

2 . بين استخدام الـ count في union ؟

3 . ما الفرق بين INTERSECT Query و Query Minus ؟

4 . ما الأمر الذي من خلاله تستطيع تغيير نوع حقل بجدول ؟

5 . ما الأمر الذي من خلاله تستطيع حذف حقل بجدول ؟

6 . ما الأمر الذي من خلاله تستطيع تغيير اسم حقل بجدول ؟

7 . ما الأمر الذي من خلاله تستطيع حذف جدول ؟

التعامل مع أنواع أخرى من المحددات وكيفية استعراض هذه المحددات وكيفية حذفها وتفعيلها وعدم تفعيلها:

-1- unique constraint :

هو عبارة عن حقل وحيد أو مجموعة حقول وحيدة بالجدول لا تتكرر ويعرفان استثنائياً في سجل وممكن أن يحتوي على قيم فارغة ولكن هذه القيم تكون فريدة أي لا تتكرر أي قيمة واحدة فريدة.

الفرق بين الـ Primary Key و UNIQUE؟

لا يوجد فرق كبير فكلاهما لا يقبل قيم مكررة ولكن الذي يميز الـ UNIQUE عن PRIMARY KEY انه يقبل قيمة فارغة NULL VALUE ولكن غير مكررة.

لا تسمح لنا الأوراكل بتعيين Primary key و unique لنفس العمود ولكن لنفرض أننا قمنا بتعيين رقم الموظف ورقم هاتفه ورقم منزله على أنه key Primary ولا نريد أن يتكرر رقم الموظف فنلاحظ انه لو ورضعنا رقم الموظف E01 ورقم هاتفه 2222 ورقم منزله w01 ففي هذه الحالة لو أضفنا سجل آخر برقم الموظف E01 ورقم هاتفه 3333 ورقم منزله w02 هل سوف يقبل أوراكل؟ الجواب نعم لأن البيان هنا لا يعتبره أوراكل مكرراً لأننا حددنا بالبداية أن الـKey primary يتكون من ثلاثة حقول وإذا أردنا ألا يتكرر رقم الموظف نعينه unique لهذا الحقل ولكن البعض سوف يقول إننا قلنا إن الأوراكل لا تقبل أن يعين الـkey primary و unique لنفس الحقول ونقول نعم ولكن قلنا لنفس الحقل وليست لمجموعة حقول فلو عيننا رقم الموظف على انه key primary لوحده بهذه الحالة لا يمكننا تعينه unique.

قاعدة إنشاء الـ unique: هناك حالتين لإنشاء الـ unique وهي إما إنشائه قبل إنشاء الجدول أو بعد إنشاء الجدول.

```
CREATE TABLE table_name
(column1 datatype null/not null,
column2 datatype null/not null,
...
CONSTRAINT constraint_name UNIQUE (column1, column2,
. column_n)
);
```

مثال:

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
supplier_name varchar2(50) not null,
contact_name varchar2(50),
CONSTRAINT supplier_unique UNIQUE (supplier_id)
);
```

تعيين أكثر من unique واحد:

مثال:

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
supplier_name varchar2(50) not null,
contact_name varchar2(50),
CONSTRAINT supplier_unique UNIQUE (supplier_id,
supplier_name)
);
```

أما إنشاءه بعد الجدول فبهذه الطريقة:

```
ALTER TABLE table_name
add CONSTRAINT constraint_name UNIQUE (column1,
column2, ... column_n);
```

مثال:

```
ALTER TABLE supplier
add CONSTRAINT supplier_unique UNIQUE (supplier_id);
```

تعيين أكثر من unique واحد:

مثال:

```
ALTER TABLE supplier
add CONSTRAINT supplier_unique UNIQUE (supplier_id,
supplier_name);
```

Constraints Check:يسمح لنا بوضع شرط على كل حقل بالجدول يعني لا يمكن الإضافة إلا بعد التأكد هل هو ضمن الصلاحيات أم لا. مع مراعاة:

- أن(check) لا يمكن إنشائه بالـ(View).
- أن(check) يجب أن ينتمي أو يرجع لحقول ضمن الجدول وليست أن تكون ضمن جداول أخرى.
- أن(check) لا يمكن إنشائه في (Sub Query) أو الاستعلام المتداخل.

قاعدة إنشائه:

أثناء إنشاء الجدول:

```
CREATE TABLE table_name
(column1 datatype null/not null,
column2 datatype null/not null,
...
CONSTRAINT constraint_name CHECK (column_name
condition) [DISABLE]
);
```

مثال:

```
CREATE TABLE suppliers
( supplier_id numeric(4),
supplier_name varchar2(50),
CONSTRAINT check_supplier_id
```

```
CHECK (supplier_id BETWEEN 100 and 9999)
);
```

مثال آخر:

```
CREATE TABLE suppliers
( supplier_id numeric(4),
supplier_name varchar2(50),
CONSTRAINT check_supplier_name
CHECK (supplier_name = upper(supplier_name))
);
```

وفي هذا المثال حددنا أن اسم المورد يجب أن يساوي الاسم الـ capital أو حروفه تكون Capital.

قاعدة إنشائه بطريقة أخرى وهي استخدام الـ ALTER:

```
ALTER TABLE table_name
add CONSTRAINT constraint_name CHECK (column_name
condition)
```

مثال على ذلك:

```
ALTER TABLE suppliers
add CONSTRAINT check_supplier_name
 CHECK (supplier_name IN ('IBM', 'Microsoft', 'Nvidia'));
```

حذف الـ CONSTRAINTS:

```
ALTER TABLE table_name
drop CONSTRAINT constraint_name;
```

مثال:

```
ALTER TABLE suppliers
drop CONSTRAINT check_supplier_id;
```

تفعيل الـ CONSTRAINT:

```
ALTER TABLE table_name
enable CONSTRAINT constraint_name;
```

مثال:

```
ALTER TABLE suppliers
enable CONSTRAINT check_supplier_id;
```

إيقاف الـ CONSTRAINT:

```
ALTER TABLE table_name
disable CONSTRAINT constraint_name;
```

مثال:

```
ALTER TABLE suppliers
disable CONSTRAINT check_supplier_id;
```

الجدول المعتمد من أوراكل والذي يخزن فيه الـ CONSTRAINT يسمى بجدول USER_CONSTRAINTS :

لمعرفة أسماء الـ CONSTRAINT التابعة لجدول معين فقط

```
CONNECT SYSTEM/UR_PASSWORD
SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='SUPLLIERS';
```

نستطيع إثناء إنشاء المحددات CONSTRAINT على جداول تحتوي على بيانات نستطيع إنشائها والتحكم إما بغض النظر عن البيانات القديمة ويبدأ من لحظة إنشاء الـCONSTRAINT نستخدم الـ NOVALIDATE أما إذا أردنا أن نشبك على البيانات القديمة فنستخدم معه VALIDATE.

مثال على NOVALIDATE:

```
ALTER TABLE uclass MODIFY CONSTRAINT pk_uclass
DISABLE VALIDATE;
```

مثال على VALIDATE:

```
ALTER TABLE uclass ENABLE NOVALIDATE PRIMARY KEY;
```

الأسئلة:

1 . ما هو الفرق بين الـ PRIMARY KEY و UNIQUE

2 . لنفرض أن لدينا جدول توجد به بيانات مكررة وأردنا أن ننشئ PRIMARY KEY على هذا العمود الذي يحتوي على البيانات المكررة ما هي الخطوات اللازمة لتطبيق ذلك.

3 . كيف نقوم بتفعيل الـ CONSTRAINT

قاعدة إنشاء الـ Function:

```
CREATE [OR REPLACE] FUNCTION function_name
[ (parameter [,parameter]) ]
RETURN return_datatype
IS | AS
[declaration_section]
BEGIN
executable_section
[EXCEPTION
exception_section]
END [function_name];
```

function_name CREATE [OR REPLACE] FUNCTION

هنا نقوم بعمل أو تبديل الدوال يعني نكتب REPLACE CREATE OR ثم نكتب FUNCTION ثم نكتب اسم الدوال وفائدة الـ REPLACE هنا لتبديل الدوال القديم بالدوال الجديد الذي نريد إنشائه [(parameter [,parameter])]

هذه الجزئية هي عبارة البارامیتر الداخل أو الخارج من الدوال حيث عند استخدام البارامیتر وتعريف نوعه نستخدم ثلاث حالات وهي:

IN وتعني قيمة البارامیتر الداخلة والغير مرتجعه بها القيمة.

OUT وتعني قيمة البارامیتر الخارجة أو الناتجة من الدوال

IN OUT وهي القيم الداخلة والخارجة حيث يدخل البارامیتر بقيمة ويرجع بقيمة أخرى. ونحن نعرف أن الدوال ترجع القيمة باسم الدوال نفسه

RETURN return_datatype:

هنا نوع القيمة المراد إرجاعها في الدوال وتعني نوع الدوال هل هو رقم أو حرف أو غير ذلك.

AS | IS وتعني التهيئة ونستخدم عادة الـ IS:

[declaration_section]

هنا نعرف البارامیتر المختلفة والتي نستطيع استخدامها.

:BEGIN

executable_section

هنا نهيئ الدوال لنبدأ بعملية الـ SELECT أو القائم بجمع أو طرح أو غير ذلك

:EXCEPTION]

[exception_section

تستخدم في حال لم ينفذ الشرط

;[function_name] END

وهنا ننهي الدوال مع كتابة اسم الدوال ويجب أن يكون مطابق لاسم الدوال

مثال:

```
CREATE OR REPLACE Function FindCourse
 ( name_in IN varchar2 )
 RETURN number
IS
 cnumber number; cursor c1 is
 select course_number
 from courses_tbl
 where course_name = name_in;
BEGIN
open c1;
fetch c1 into cnumber;
if c1%notfound then
 cnumber := 9999;
end if;
close c1;
RETURN cnumber;
EXCEPTION
WHEN OTHERS THEN
 raise_application_error(-20001,'An error was encountered -
'||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

قاعدة إنشاء الـ (Procedure) الإجراء:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [ (parameter [,parameter]) ]
IS
 [declaration_section]
BEGIN
 executable_section
[EXCEPTION
 exception_section]
END [procedure_name];
```

procedure_name CREATE [OR REPLACE] PROCEDURE

هنا نقوم بعمل أو تبديل الـ Procedure يعني نكتب OR CREATE

REPLACE ثم نكتب PROCEDURE ثم نكتب اسم الـ PROCEDURE

وفائدة الـ REPLACE هنا لتبديل الـ PROCEDURE القديم

بالـPROCEDURE الجديد الذي نريد إنشائه

[([parameter [,parameter)]

هذه الجزئية هي عبارة البارامیتر الداخل أو الخارج من الإجراء حيث عند استخدام البارامیتر وتعريف نوعه نستخدم ثلاث حالات وهي:

IN وتعني قيمة البارامیتر الداخلة والغير مرتجعه بها القيمة .

OUT وتعني قيمة البارامیتر الخارجة أو الناتجة.

IN OUT وهي القيم الداخلة والخارجة حيث يدخل البارامیتر بقيمة ويرجع بقيمة أخرى.

IS /وتعني التهيئة:

[declaration_section]

هنا نعرف البارامیتر المختلفة والتي نستطيع استخدامها للـ Procedure.

:BEGIN

executable section

هنا نهيئ الـ Procedure لنبدأ بعملية الـ SELECT أو القائم بجمع أو طرح أو غير ذلك

:EXCEPTION

[exception section]

تستخدم في حال لم ينفذ الشرط

END [PROCEDURE_name];

وهنا ننهي الـ Procedure مع كتابة اسمه ويجب أن يكون الاسم مطابق.

ABAHE

مثال:

```
CREATE OR REPLACE Procedure UpdateCourse
 ( name_in IN varchar2 )
IS
 cnumber number; cursor c1 is
 select course_number
 from courses_tbl
 where course_name = name_in;
BEGIN
open c1;
fetch c1 into cnumber;
if c1%notfound then
 cnumber := 9999;
end if;
insert into student_courses
( course_name,
 course_number)
values ( name_in,
 cnumber );
commit;
close c1;
EXCEPTION
WHEN OTHERS THEN
 raise_application_error(-20001,'An error was encountered -
'||SQLCODE||' -ERROR- '||SQLERRM);
END;
```

هذا الإجراء يأخذ رقم الكورس ويضيفه إلى جدول الطلاب ويمكن استدعائه عن طريق التريجر مثلاً.

قاعدة إنشاء الـ TRIGGER:

تعرف الـ TRIGGER على أنه الحدث الذي ينفذ أثناء حدوث تغيير على جدول معين بقاعدة البيانات لتنفيذ مجموعة من التعليمات أو لأخذ معلومات وإضافتها بجدول آخر أو استدعاء PROCEDURE أو حتى استدعاء FUNCTION تم إنشائها على قاعدة البيانات

```
CREATE or REPLACE TRIGGER trigger_name
TRIGGER TYPE
 ON table_name
 [ FOR EACH ROW ]
DECLARE
 -- variable declarations
BEGIN
 -- trigger code
EXCEPTION
 WHEN ...
 -- exception handling
END;
```

trigger_name CREATE or REPLACE TRIGGER

هنا نقوم بعمل أو تبديل الـ Trigger كتب CREATE OR REPLACE ثم نكتب TRIGGER نكتب اسم الـ TRIGGER وفائدة الـ REPLACE هنا لتبديل الـ TRIGGERJ القديم بالـ TRIGGER الجديد الذي نريد إنشائه

:TYPE TRIGGER

وهي أنواع Insert Triggers و Update Trigger و Trigger Delete

وسوف نستعرض هذه الأنواع:

ON table_name: وهنا تضع اسم الجدول

[FOR EACH ROW]

ومعناها لكل سطر يضاف

مع ملاحظة أننا:

1 . لا نستطيع إنشاء هذه الأنواع على views

2 . يمكن التعديل على new داخل التريجر

3 . لا يمكن التعديل على old داخل التريجر

والباقي هو عبارة عن إضافة المتغيرات أو التحكم بجزئية الجداول والسيطرة على البيانات المدخلة وفي هذه الجزئية ممكن أن تستدعي بروسيجر أو فانكشن.

وأول الأنواع هو(Triggers Insert):

:INSERT Trigger BEFOR

```
CREATE or REPLACE TRIGGER trigger_name
BEFORE INSERT
 ON table_name
 [ FOR EACH ROW ]
DECLARE
 -- variable declarations
BEGIN
 -- trigger code
EXCEPTION
 WHEN ...
 -- exception handling
END;
```

مثال:

ننشئ الجدول:

```
CREATE TABLE orders
( order_id number(5),
quantity number(4),
cost_per_item number(6,2),
total_cost number(8,2),
create_date date,
created_by varchar2(10)
);
```

ثم ننشئ هذا التريجر:

```
CREATE OR REPLACE TRIGGER orders_before_insert
BEFORE INSERT
 ON orders
 FOR EACH ROW
DECLARE
 v_username varchar2(10);
BEGIN -- اسم المستخدم إيجاد
 SELECT user INTO v_username
        FROM dual; -- تعديل تاريخ التعديل حيث new ترمز إلى قيمة جديدة
    :new.create_date := sysdate; -- هنا نضيف اسم المستخدم الجديد حيث
                                 باراميتر الجديدة وهو عبارة لنضع القيمة new استخدمنا
INSERT
 :new.created_by := v_username;
END;
```

:AFTER INSERT

```
CREATE or REPLACE TRIGGER trigger_name
AFTER INSERT
 ON table_name
 [ FOR EACH ROW ]
DECLARE
 -- variable declarations
BEGIN
 -- trigger code
EXCEPTION
 WHEN ...
 -- exception handling
END;
```

مثال:

ننشئ الجدول:

```
CREATE TABLE orders
( order_id number(5),
quantity number(4),
cost_per_item number(6,2),
total_cost number(8,2)
);
```

ثم ننشئ هذا التريجر:

```
CREATE OR REPLACE TRIGGER orders_after_insert
AFTER INSERT
 ON orders
 FOR EACH ROW
DECLARE
 v_username varchar2(10);
BEGIN
 -- Find username of person performing the INSERT into the
table
 SELECT user INTO v_username
```

```
FROM dual; -- Insert record into audit table
INSERT INTO orders_audit
( order_id,
quantity,
cost_per_item,
total_cost,
username )
VALUES
( :new.order_id,
:new.quantity,
:new.cost_per_item,
:new.total_cost,
v_username );
END
```

أما الأنواع الباقية فهي نفس ما سبق

حذف التريجر / Drop a Trigger:

```
DROP TRIGGER trigger_name;
```

مثال:

```
DROP TRIGGER orders_before_insert;
```

إيقاف عمل التريجر DISABLE:

```
ALTER TRIGGER trigger_name DISABLE;
```

مثال:

```
ALTER TRIGGER orders_before_insert DISABLE;
```

إيقاف عمل جميع التريجرات على الجدول:

```
ALTER TABLE table_name DISABLE ALL TRIGGERS;
```

مثال:

```
ALTER TABLE orders DISABLE ALL TRIGGERS;
```

تفعيل عمل التريجر:

```
ALTER TRIGGER trigger_name ENABLE;
```

مثال:

```
ALTER TRIGGER orders_before_insert ENABLE;
```

تفعيل عمل جميع التريجرات على الجدول:

```
ALTER TABLE table_name ENABLE ALL TRIGGERS;
```

مثال:

```
ALTER TABLE orders ENABLE ALL TRIGGERS;
```

الأسئلة:

1 . ما هي الفائدة من الدوال؟

2 . كيف نستطيع تحديد نوع البارامیتر إذا كان داخل أو خارج؟

3 . قم بإنشاء فانكشن يقوم بجمع رقمين وإرجاعهما في جملة SELECT؟

4 . ما هي الفائدة من البروسيجر؟

5 . كيف نستطيع تحديد نوع البارامیتر إذا كان داخل أو خارج؟

6 . قم بإنشاء بروسيجر يقوم بجمع رقمين وإرجاعهما في جملة SELECT.

إنشاء الـ Views والتعامل معها

الـ views هي عبارة عن استعلام نستطيع وضعه على جدول أو اكثر بحيث لا نستطيع الإضافة أو الحذف أو التعديل من خلاله إذاً هو عرض ويساعد في إنشاء استعلامات ثابتة على الجداول.

قاعدة إنشاء الـ View:

```
CREATE VIEW view_name AS
SELECT columns
FROM table
WHERE predicates;
```

مثال:

```
CREATE VIEW sup_orders AS
SELECT supplier.supplier_id, orders.quantity, orders.price
FROM supplier, orders
WHERE supplier.supplier_id = orders.supplier_id
and supplier.supplier_name = 'IBM';
```

عرض الـ view وعرض محتوياتها هي:

```
SELECT *
FROM sup_orders;
```

يمكن التحديث على الـ view بدون حذفها وذلك حسب القاعدة التالية:

```
CREATE OR REPLACE VIEW view_name AS
SELECT columns
FROM table
WHERE predicates;
```

مثال:

```
CREATE or REPLACE VIEW sup_orders AS
SELECT supplier.supplier_id, orders.quantity, orders.price
FROM supplier, orders
WHERE supplier.supplier_id = orders.supplier_id
and supplier.supplier_name = 'Microsoft';
```

حذف الـ view يتم حسب القاعدة التالية:

```
DROP VIEW view_name;
```

مثال:

```
DROP VIEW sup_orders;
```

الـ Synonyms وما هي وكيفية إنشائها وحذفها:

الـ Synonyms هو عبارة عن بديل لمحتويات قاعدة البيانات مثل الـ tables, views, sequences, stored procedures, وغيرها من المحتويات وفائدتها أن المستخدم يستطيع أن يجلب أو يعدل.

قاعدة الإنشاء:

```
create [or replace] [public] synonym [schema .] synonym_name
for [schema .] object_name [@ dblink];
```

replace or وهذه نستخدمها عندما نريد التعديل على Synonyms موجود أصلاً public نكتبها إذا كنا نريد جميع المستخدمين على قاعدة البيانات الاستفادة منها schema.

```
table package
view materialized view
sequence java class schema object
stored procedure user-defined object
function synonym
```

مثال:

```
create public synonym suppliers
for app.suppliers;
```

كما نرى لقد أنشأنا synonym على app على جدول suppliers ونوعه puplic يعني يستطيع أي مستخدم أن يستعلم عليها وفائدة الـ synonym أنك تستطيع من خلالها القيام بقراءات متعددة.

```
[COLOR=blue]select * from suppliers;[/COLOR]
```

التعديل على synonym نكتب التالي:

```
create or replace public synonym suppliers
for app.suppliers;
```

حذف الـ synonym:

```
drop [public] synonym [schema .] synonym_name [force];
```

حيث public تستخدم في حالة كان الـ synonym أصلا puplic تستخدم لحذف الـ synonym حتى لو كان لها تبعيات فإنها تجبر الأوراكل على حذفها ولا يستحب استخدامها لأنها قد تعطل عمل الـ oject.

مثال:

```
drop public synonym suppliers;
```

القواعد ROLES وكيفية إنشائها

وهي عبارة عن مجموعة من الشروط يتم إنشائها ليتم إلزام المستخدم بها وهي privileges Role system يجب إنشاء الـ Role لإنشاء Creating a Role وليتم ذلك نتبع الخطوات التالية:

```
CREATE ROLE role_name
[ NOT IDENTIFIED |
IDENTIFIED {BY password | USING [schema.] package |
EXTERNALLY | GLOBALLY };
```

حيث role_name هو اسم الـ role

NOT IDENTIFIED : توضع في حالـ تعريف الـ role حيث تفعل الـrole ولا يحتاج لكلمة سر لتفعيل الـ role.

IDENTIFIED: هنا يجب أن تقوم بتعريف خصائص الـ role قبل تفعيلها

BY password: هنا تعني أن المستخدم يجب أن يضع كلمة سر قبل تفعيل الـ role

USING package: هنا تكون قد أنشأت مجموعة من role ويجب تطبيقها من خلال البرنامج.

EXTERNALLY : وهذا يعني انك تحتاج إلى تعريف خصائص EXTERNALLY قبل تشغيل الـ role.

GLOBALLY : يعني أن المستخدم يستطيع تعريف أو الاطلاع على اليوزر عن طريق مجلدات الـ enterprise.

مع ملاحظة إننا إذا لم نضع NOT IDENTIFIED أو نضع IDENTIFIED سوف يتم إنشاء الـ role على أساس أنها NOT IDENTIFIED.

مثال:

ننشئ role ونسميها test_role

```
CREATE ROLE test_role;
```

ننشئ role ولكن نستخدم الباسوورد حيث يطلب الباسوورد عند تشغيلها

```
CREATE ROLE test_role
IDENTIFIED BY test123;
```

Roles Grant Privileges (on Tables) to

إعطاء الصلاحيات على الجداول لـ role باستخدام privilege

وهناك أنواع من الصلاحيات على الجداول هي:

Select لإعطاء صلاحية الاستعلام باستخدام جملة select

Insert هو إعطاء صلاحية إضافة سجل جديد على الجداول

Update إعطاء صلاحية التحديث على سجل في الجدول

Delete إعطاء صلاحية الحذف لسجل من الجدول

References إعطاء صلاحية لإنشاء الـ Constraints

Alter إعطاء صلاحية التعديل على الجدول

Index إعطاء صلاحية لإنشاء الـ index على الجدول

القاعدة:

```
grant privileges on object to role_name
```

مثال على إعطاء بعض الصلاحيات لـ Roles:

```
grant select, insert, update, delete on suppliers to test_role;
```

مثال على إعطاء الصلاحيات لجميع الـ Roles:

```
grant all on suppliers to test_role;
```

إلغاء الصلاحيات عن Roles:

Revoke Privileges (on Tables) to Roles

هذه الخاصية هي كيفية إلغاء الصلاحية عن Roles

```
revoke privileges on object from role_name;
```

مثال إلغاء على مجموعة من الصلاحيات:

```
revoke delete on suppliers from test_role;
```

مثال على إلغاء جميع الصلاحيات:

```
revoke all on suppliers from test_role;
```

إلغاء الصلاحية عن الـ roles:

```
revoke execute on object from role_name;
```

مثال:

```
revoke execute on Find_Value from test_role;
```

إعطاء صلاحيات الـ Roles إلى المستخدم:

Granting the Role to a User

وهي تمكن من السيطرة على صلاحيات المستخدم للنظام

القاعدة:

```
GRANT role_name TO user_name;
```

مثال:

```
GRANT test_role to smithj;
```

استخدام الـ The SET ROLE statement:

هذه الخاصية التي تدعى الـ Set Role لتفعيل أو عدم تفعيل الـ Roles قاعدة الاستخدام:

```
SET ROLE
( role_name [ IDENTIFIED BY password ]
| ALL [EXCEPT role1, role2, ... ]
| NONE );
```

معلومات تلخيصية

امتيازات النظام System Privileges.

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

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

امتيازات العناصر Object Privileges

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

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

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

تمرين حر:

- أنشئ مستخدم جديد باسم ABAHE وكلمة المرور DHID مع الصلاحيات المناسبة له للعمل.
- صمم جداول للطلاب ونتائجهم ولك حرية اختيار العمليات والنتائج التي تراها مناسبة لذلك مستفيداً بما سبق ومر معنا ومن وجهة نظرك كمبرمج...
- احفظ كل الأكواد مع الشرح وأرسلها لأستاذك المشرف.

مع تمنياتنا لكم بالتوفيق والنجاح الدائمين

ABAHE