

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (١)

ما هو مفهوم قواعد البيانات؟

القاعدة بمفهومها المجرد عبارة عن مكان أو وعاء تجتمع فيها عناصر مختلفة للخصائص. فالقاعدة العسكرية مثلاً عبارة عن قطعة أرض تحتوي عناصر مختلفة مثل الجنود و مختلف خصائصهم كالاسم والرتبة وال عمر... كما تحتوي المبني التي يستعملها الجنود كمهاجم النوم والمطبخ والملاجئ والخنادق وساحات التدريب.... كما تحتوي بيانات عن الأسلحة والآليات وغير ذلك من مرافق...

ولا يختلف الحديث كثيراً عن قاعدة البيانات بمفهوم علوم الكمبيوتر Computer Science فقاعدة بيانات الطلاب مثلاً تشمل خصائص Attributes الطلاب مثل أسمائهم وأعمارهم و تخصصاتهم وأرقامهم الجامعية... والخصوصيات لها خصائص مثل رمز التخصص واسم التخصص وغير ذلك.

ولكن بيانات الجامعة تشمل على عدة قواعد بيانات مثل قاعدة بيانات الجامعة (اسمها وشهرتها وعنوانها الدرجات التي تمنحها...) وقاعدة بيانات طلابها سالفه الذكر وقاعدة بيانات المدرسين ودرجاتهم العلمية و تخصصاتهم... وتشمل قاعدة بيانات أصول الجامعة من مبانٍ وملاعب وسيارات ومرافق أخرى وهكذا كلما كبر كيان المنشأة كبرت قواعد بياناتها وتعددت.

وكيف يتم تخزين هذه القواعد وكيف يتم الرجوع إليها؟ لا شك أن أول ما يخطر في بالنا هو الكمبيوتر بحكم أننا نعيش ثورة في مجال الكمبيوتر أخذت أسماء متعددة ونادراً ما يخطر في بالنا وسائل سابقة لتخزين قواعد البيانات. ففي التاريخ القديم كانت عقول الناس هي التي تخزن قواعد البيانات بتفاصيلها كثرة أو قلة وكانت الكتابات على وسائل مختلفة إحدى الوسائل لتخزين قواعد البيانات كالحجارة العريضة النحيفة أو عظام بعض الحيوانات الكبيرة أو على أوراق الشجر كورق نبات البردي المشهورة في مصر وغيرها...

ومع التقدم الحضاري والمدني للبشرية تطورت وسائل تخزين قواعد البيانات ووصلت ذروتها في احتراق الورق الذي تحسن حتى وصل إلى ما نعرفه حالياً. وصارت بيانات الطلاب مثلاً تكتب في أوراق على شكل جدول ورقي فيه اسم الطالب والتخصص والجدول الدراسي دون ترتيب معين؛ وتترفق أحياناً بعض الوثائق كشهادة الميلاد أو كشف العلامات....

وهنا برزت مشكلة الرجوع إلى هذا الجداول الورقية لاستخراج بيانات طالب معين أو مجموعة طلبة ومشكلة فرز الطلاب حسب التخصص أو السنة الدراسية... الخ وبذلت الحلول لهذه المشاكل تأخذ مكانها في تطور متلاحق كما سنرى في المقالات التالية.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلا (٢)

إن تخزين البيانات على الوسائل المختلفة ليس معضلة كبيرة ولكن استرجاع البيانات شكل معضلة كما أسلفنا في المقال السابق. فلو أردت تقريرا عن طالب معين فيجب الرجوع إلى القاعدة الورقية التي تحتوي بيانات الطلاب؛ والبحث عن طالب واحد بين عشرات الطلاب مجده ويأخذ وقتا طويلا؛ فيما بالك بالبحث في مئات أوآلاف الطلاب للبحث عن طالب؛ وتكرر المشكلة مع زيادة عدد الطلاب المطلوب تقرير عنهم وتعقد الأمور أكثر مع تعداد أنواع التقارير المطلوبة. لذلك وجد الناس أنه لا بد من تسهيل عملية البحث واستخراج البيانات المختلفة؛ فقاموا بتحصيص مخزن(ملف) خاص بكل طالب مثلا يمثل سجل الطالب ويكتب على وجه الملف اسم الطالب مثلا ورقم الجامعي مما يسهل عملية البحث. وقاموا بتحصيص خزانات (دواليب) لوضع تلك الملفات فيها مرتبة مثلا حسب التخصص والتخصص مرتب حسب سنة الانضمام للجامعة وهكذا...

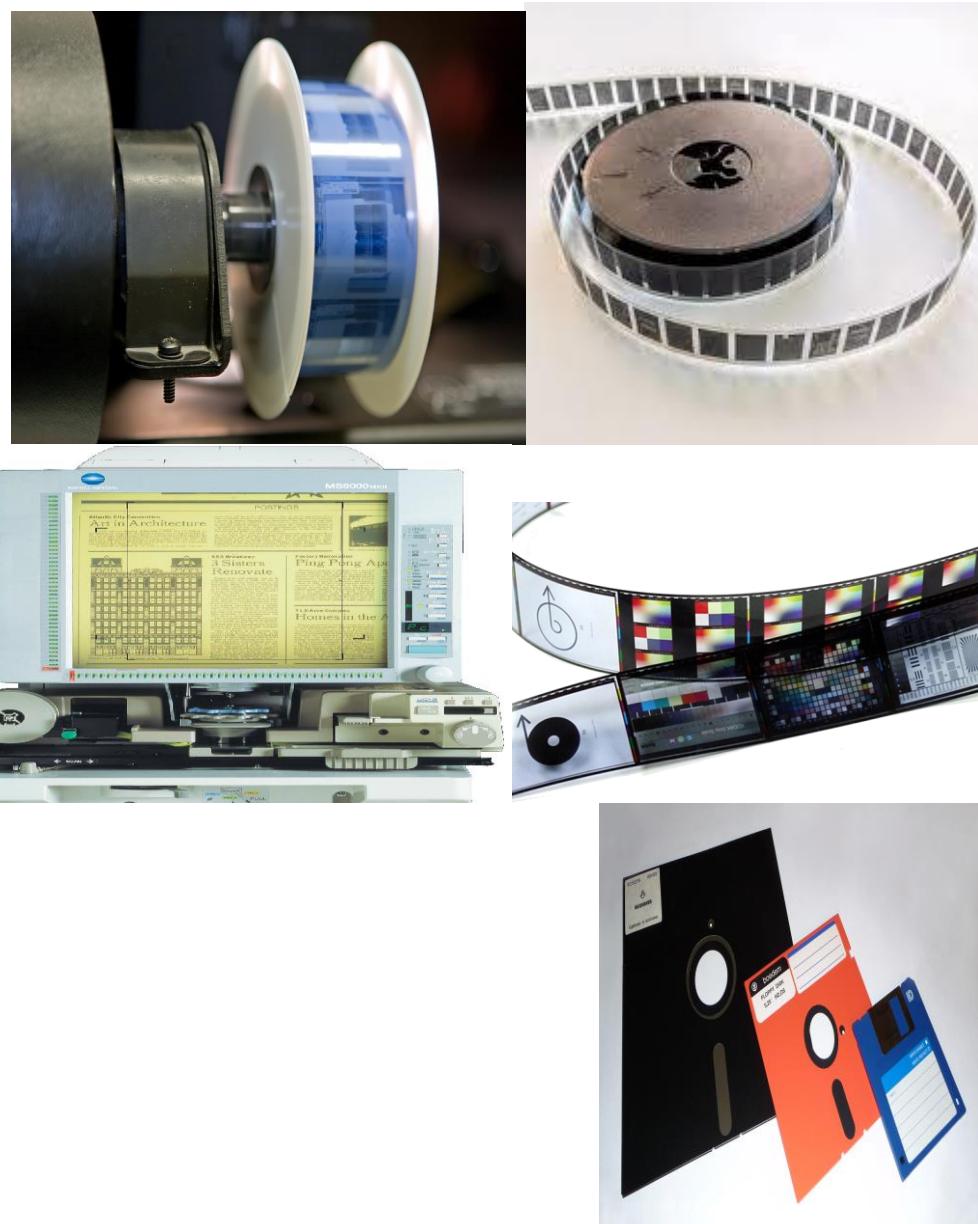
وتطورت وسائل التخزين من الورقية إلى الأفلام المصغرة MicroFilms ووُجِدَت أجهزة لتخزين البيانات والبحث عنها وكانت تشكل حللا ممتازا تلك الأيام وتتوفر الوقت والجهد في البحث فضلا عن سرية وحفظ البيانات من الضياع ومساحة تخزين أصغر مقارنة بالورق. ومع ذلك بقي الورق أكثر استعمالا لرخصه من جهة وعدم الحاجة إلى مهارات خاصة وأجهزة لتخزين والبحث.

ومع تزايد استعمال أجهزة الكمبيوتر وتطورها وتطور لغات البرمجة صار تخزين البيانات أسهل واسترجاعها أسرع وأكثر فعالية. وبرز هنا مفهومان هما: الوسط الفيزيائي لتخزين البيانات وهو الأدوات الفيزيائية المستعملة لتخزين البيانات مثل الأقراص الصلبة مثلا؛ والمفهوم الثاني هو الوسط المنطقي لتخزين البيانات وهو طريقة تخزين البيانات من خلال هيكلية البيانات Data Structure على الوسائل الفيزيائية؛ مثل الملفات المتتالية Sequential Files .

وكانت الوسائل الفيزيائية متعددة والجيل الحالي لم يستعملها بل لم يسمع بها؛ مثل الأشرطة المغناطيسية Magmatic Tapes والطبلول Drums (طبعا شكلها مختلف عن الطبل الموسيقي) والاسطوانات المغناطيسية Magnetic Cylinders والبطاقات المثبتة Punched Cards والأقراص المرنة Floppy Disks وكانت بأحجام وسعات مختلفة مثل ٨٠.٥ ٥٠.٢٥ إنش و ٣٠.٥ إنش وهذا الأخير كان ثورة في عالم التخزين. أما ساعتها فكانت أرقاما بسيطة لما عليه الحال الآن مثل ١٢٨ ك.ب. و ٣٦٠ ك.ب. ووصلت إلى ٧٢٠ ك.ب. وكانت مساحات كبيرة.

أما الأقراص الصلبة فكانت كبيرة الحجم فيزيائيا ومساحات التخزين كانت بميغا تبدأ من ٥ ميغا وقد تصل إلى ٣٠٠ ميغا. وأنذكر أن أحد المهندسين في مركز الكمبيوتر في الجامعة والذي كان عبارة عن ثلاثة كبيرة جدا تحتوي جهاز الكمبيوتر الضخم Mainframe من نوع آي بي إم. ذكر لي هذا المهندس بكل فخر ان القرص الصلب مساحته ٣٠٠ ميغا بايت وكل بيانات الجامعة مخزنة عليه.

وهذه صور بعض الوسائل القديمة



ولكن العلماء استمروا في تطوير وصناعة أقراص صلبة ووصلت الساعات إلى مئات الميجا بايت ومن ثم الجيجا حتى وصلنا إلى زيتابايت ... وبقيت بعض الشركات تستعمل الوسائل القديمة إضافة إلى الوسائل الحديثة حسب حجم البيانات التي تعامل بها.

وماذا عن تخزين البيانات بالوسائل المنطقية؟ أي طريقة ترتيب البيانات في الملفات المنطقية؟ في المقالة التالية نلقي الضوء عليها.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٣)

نأتي الآن إلى ما يهمنا في هذه السلسلة ألا وهو طرق تخزين البيانات كقواعد بيانات واسترجاعها. في المقالة السابقة قلنا إن البيانات يتم تخزينها فيزيائياً ومنطقياً، وألقينا الضوء على التخزين الفيزيائي؛ والآن نلقي الضوء على التخزين المنطقي.

ومقصود بالتخزين المنطقي هو كيفية ترتيب البيانات في وحدات غير فيزيائية باستعمال هيكلة البيانات Data Structure. وأول ما يتبادر إلى الذهن هو التخزين في ملفات على أوساط مغناطيسية مثل الأشرطة المغناطيسية والأقراص بأنواعها وغيرها.

والملفات أسهل شيء وهي كانت الوسيط الأساسي في تخزين البيانات في لغات البرمجة العليا High Level Programming مثل فورتران Fortran وباسكال Pascal وبيسك Basic ولغة الأعمال الأشهر كوبول Cobol وغيرها من اللغات. وكانت هذه الملفات على نوعين؛ أحدهما هو الملفات المتتالية Sequential Files والملفات العشوائية Random Access Files.

في الملفات المتتالية يتم تخزين البيانات على هيئة سجلات منطقية متتالية على شكل شريط طويل؛ ويتم تعريف الملف من خلال أوامر وجمل لغة البرمجة. وتخزين أي سجل جديد؛ يضاف إلى آخر الشريط. وللحث عن أي سجل يجب أن يمر البرنامج على جميع السجلات ابتداءً من أولها إلى أن يصل إلى السجل المنشود. وطبعاً هذه عملية طويلة وملة؛ فكيف إذا أردنا تعديل أو حذف سجل فالمسألة أصعب. وعند حذف سجل يصير هناك فراغات في الملف. وعملية البحث هي الأهم لأن الأهم هو استخراج البيانات بعد تخزينها لاستخراج التقارير اللازمة. ولتعجيل عملية البحث اخترع العلماء أساليب لتعجيل عملية البحث مثل البحث الثنائي Binary Search والتخزين الشجري Hierarchical Trees.

أما في حالة الملفات العشوائية Random Access Files فيتم تخزين السجلات في أي مكان على وسيلة التخزين من غير تقييد بموقعه على الملف؛ فقد يكون آخر سجل في موقع يلي السجل الأول مثلاً. وهذه الطريقة يكون التخزين والبحث أسرع لأنه لا يحتاج أن يمر على جميع السجلات ليصل إلى آخر سجل ليتم التخزين بعده كما في الملفات المتتالية.

وفي كل الأحوال يقوم نظام التشغيل أو لغة البرمجة بتوفير مؤشرات Pointers في الذاكرة وعلى الديسک لتشير إلى السجلات في أنواع الملفات سالف الذكر. فكل سجل يحمل مؤشراً يشير إلى السجل الذي يليه. وطبعاً هذه العملية فيها بطء واستنزاف لموارد الجهاز فضلاً عن جهود المبرمجين لتسريع عمل تلك المؤشرات ومحوها من الذاكرة حين تتحقق الغاية منها. ولغة باسكال في نسخة تيربو باسكال Turbo Pascal مثلاً وفرت ميزات ممتازة للتحكم بالمؤشرات. ولكن يبقى البطء هو سيد الموقف لطبيعة وسائل التخزين الفيزيائية من جهة وطريقة تخزين البيانات منطقياً من جهة أخرى. وإذا أضفنا إلى ذلك تضخم حجم البيانات وتشتتها على وسائل التخزين الفيزيائية -أي أن السجلات ليست متتالية على القرص مثلاً بالرغم من أن التركيب المنطقي لها هو الملف المتتالي- وهذان العاملان الآخرين زاداً من مصاعب المبرمجين في كتابة البرامج التي تحقق الحد الأعلى الممكن من كفاءة استرجاع البيانات واستخراج التقارير.

أوراكل مثلاً (٤)

في أواسط ستينيات القرن العشرين بزرت فكرة بنوك البيانات على أثر تضخم البيانات (فكرة البيانات الضخمة ليست جديدة) مما حدا بالعلماء إلى اختراع آليات جديدة لتخزين البيانات واسترجاعها. وكان من أبرزهم الدكتور إدجار فرانك كود (توفي عام ٢٠٠٣) وهو عالم كومبيوتر بريطاني اشتغل في مختبرات آي بي إم في أمريكا. اقترح الدكتور كود النموذج العلائقى في ورقة بحث أسمها اسم طويلاً هو "النموذج العلائقى للبيانات في مستودعات البيانات الضخمة المشتركة A Relational Model of Data for Large Shared Data Banks" وذلك سنة ١٩٧٠.

في هذا النموذج اقترح الدكتور كود تخزين البيانات على شكل علاقات؛ وهي نفس الفكرة التي درسناها في المرحلة الثانوية والجامعية في مادة الرياضيات-والعلاقة هي مجموعة الأزواج المرتبة- والعلاقة يمكن تمثيلها على شكل جدول يتكون من صفوف وأعمدة؛ إذن هي ليست بسبب وجود علاقات بين الجداول أخذت هذا الاسم ولكن بسبب بنائهما على فكرة العلاقات في علم الرياضيات Relations. وصمم الدكتور كود لغة للتعامل مع هذا النموذج سماها لغة الاستعلام الميكيلية Structured Query Language-SQL المشهورة. ولكن شركة آي بي إم لم تقم كثيراً بمقترنه فأصابه الإحباط. ولكن شركات أخرى تبنت الفكرة وكانت أول شركة هي شركة أوراكل - كان اسمها "مختبرات تطوير البرامج" - أول من تبني هذا النموذج كمنتج تجاري يعني تبنت النموذج العلائقى ولغة الاستعلام الميكيلية عام ١٩٧٩م ووظفتها لبناء التطبيقات والأنظمة التجارية. ثم كرت السبحة فتابعتها آي بي إم وبروجرس Progress وانحرز Ashton-Tate وآشتون تيت Ingres صاحبة الـ DBaseII و الـ DBaseIII و الـ Watcom واتكوم DBaseVI وميكروسوف特 حديثاً وكثير غيرها.

طبعاً هذا النموذج حتى يتم استخدامه يحتاج إلى طريقة كي يتعامل معه الكمبيوتر ويستوعب هذه اللغة الجديدة غير التقليدية- سأشرح خصائصها-. فكان لزاماً على هذه الشركات إنشاء برامج تعامل مع لغة الاستعلام الميكيلية وتوظيف النموذج العلائقى في تخزين البيانات. مجموعة البرامج هذه اسمها نظام إدارة قواعد البيانات العلائقية أو Relational Database Management System اختصاراً RDBMS وهذا هو محور التنافس بين شركات قواعد البيانات لتقديم أفضل نظام لإدارة قواعد البيانات من حيث سرعة استخراج البيانات واستيعاب كميات هائلة من البيانات المتنوعة وصار هذا التنافس محموماً مع تضخم البيانات وتعدد أنواعها وتوزعها على نطاق جغرافي واسع، ترافق ذلك مع ازدياد قدرات أجهزة الكمبيوتر ووسائل التخزين من حيث السرعة والاستيعاب. وبما أني اشتغل على قواعد البيانات أوراكل منذ ١٩٩٢م فإني وجدت القوة والاعتمادية فيها ورأيي ليس ملزماً طبعاً. وفي المقالة التالية شرح للنموذج العلائقى وخصائص SQL.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥)

في الملفات العاديّة (السطحية) مثل الملفات المتتالية والعشوائيّة يتم تخزين مكونات قاعدة البيانات على ملفات منفصلة على وسائل التخزين. فمثلاً بيانات الطّلاب لها ملف منفصل عن ملف الجداول الدراسية وملف آخر عن نتائج الاختبارات... وبعكّتنا استعراض هذه الملفات باستعمال أوامر نظام التشغيل بصورة عاديّة كأي ملف.

بينما في مفهوم قواعد البيانات العلائقية؛ فإننا نستعمل مفهوم العلاقة أو الجداول وهذه العلاقات أو الجداول تكون محتواة في وعاء منطقي يسمى مساحة الجداول Table Space مرتبط بملف فيزيائي على وسائل التخزين وتتعدد هذه المساحات بتنوع قواعد البيانات وتعدد الأنظمة... وتنظيم قواعد البيانات ميدان آخر تتفاوت في شركات قواعد البيانات. وبهذه الطريقة لا يبذل المبرمج أي جهد في كيفية تخزين البيانات على الديسک ولا يشغل باله في ذلك بل هذه مشكلة نظام إدارة قواعد البيانات. ونبأ بخصائص النموذج العلائقى.

خصائص النموذج العلائقى. في هذا النموذج يستطيع المستعمل العادي -فضلاً عن المبرمج- التعامل مع البيانات من خلال جداول (علاقات) ثنائية الأبعاد بحيث يكون لدينا أربعة مفاهيم فقط تحتاجها في هذا النموذج وهي:

١. الجداول Tables

٢. الأعمدة Columns

٣. الصفوف Rows

٤. الحقول Fields

يذكرنا النموذج العلائقى هذا بأحد فروع علم الرياضيات وهو الجبر العلائقى أو جبر العلاقات والذي يتضمن:

□ تجمع من الأشياء يطلق عليها اسم علاقات Relations

□ مجموعة من العوامل Operators التي تعمل على هذه العلاقات والتي تنتج علاقات أخرى.

□ مجموعة من القيود من أجل تكاملية Integrity وتوافقية Consistency ودقة البيانات Data Accuracy.

ويتم تعريف قاعدة البيانات العلائقية Relational database بأنها تجمع من العلاقات Relations أو من الجداول ثنائية الأبعاد.

* العمليات على العلاقات تفينا في فهم جداول البيانات التي هي علاقات.

توجد العمليات التالية على العلاقات:

Restriction التقييد

وهي عملية استخراج البيانات من العلاقة بحيث يمكن عرض بعض الصفوف من العلاقة أو جميعها اعتماداً على شرط أو شروط معينة. يُسمى التقييد أحياناً بأنه "مجموعة جزئية أفقية" Horizontal subset .

الإسقاط

وهي العملية التي تستخرج البيانات من خلال عرض أعمدة معينة من العلاقة ولذلك تسمى هذه العملية أحياناً بـ "مجموعة جزئية رئيسية". "Vertical subset".

الناتج (حاصل الضرب)

وهي العملية التي تعرض جميع الصيغ من علاقتين أو أكثر على صورة حاصل الضرب -ليس الضرب الاعتيادي- ولكن يقترب كل صيغ في إحدى العلاقتين مع جميع صيغ العلاقة الأخرى. والناتج يكون مجموعة جديدة ضخمة غالباً. تسمى عملية الضرب بين العلاقات بالضرب الديكارتي أو الجداء الديكارتي نسبة إلى عالم الرياضيات المشهور ديكارت وبالإنجليزية **Cartesian Product**

الربط

وهي العمليات التي تعرض الصيغ المشتركة من علاقتين أو أكثر بناءً على شرط أو شروط معينة على صورة حاصل الضرب ولكن بشروط. تنتج مجموعة أصغر من أي من العلاقتين عادةً. والربط يمكن من خلال الأ عدة المشتركة.

الاتحاد

وهي العملية التي تستخرج البيانات من علاقتين أو أكثر من خلال عرض جميع الصيغ في العلاقات. والفرق بين الاتحاد **Union** والناتج **Product** هو أنه في عملية الاتحاد يتمأخذ جميع صيغ العلاقة صفاً صفاً. أما في الناتج (حاصل الضرب) ففي مقابل كل صيغ من علاقة يتمأخذ جميع صيغ العلاقة الأخرى بما فيها صيغ العلاقة نفسها وهكذا.

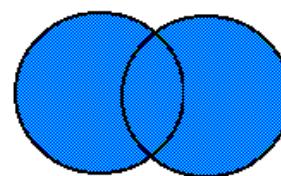
التقاطع

وهي العملية التي تستخرج البيانات من خلال عرض الصيغ المشتركة من علاقتين أو أكثر.

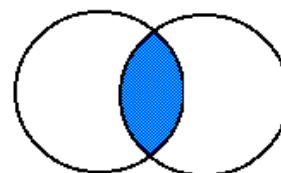
الفرق

وهي العملية التي تستخرج البيانات من خلال عرض صيغ علاقه ليست في علاقة أخرى.

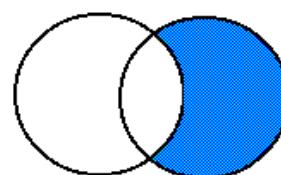
والشكل التالي يوضح العمليات الثلاثة الأخيرة



الاتحاد-Union



النقطة Intersection



الفرق Difference

وتاليا خصائص لغة الاستعلام الهيكلية.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦)

خصائص قاعدة البيانات الالعائقيه

١. تظهر قاعدة البيانات الالعائقيه للمستعمل -أيا كان - كتجمع من العلاقات (الجدوال).
٢. إن الشكل صف/عمود Column/Row هو شكل مألوف وسهل لعرض البيانات.
٣. هناك مجموعة من العمليات على العلاقات لتقسيم علاقة ما أو الربط بين أكثر من علاقة؛ مثل الاتحاد والتقاطع... الخ.
٤. لا حاجة لاستعمال مؤشرات Pointers صريحة حيث يتم التعامل مع البيانات مباشرة. أي لا يحتاج المستعمل (المبرمج أو سواه) لاستعمال المؤشرات للبحث عن البيانات أو التعامل معها.
٥. إن اللغة المستعملة للتتعامل مع البيانات واسترجاعها من العلاقات(الجدوال) هي لغة غير إجرائية Non-Procedural هي لغة غير إجرائية Procedural فلا تحتاج إلى جمل شرطية أو حلقات تكرار. كما أنها شبيهة باللغة المتداولة العاديه مثل اللغة الإنجليزية.
٦. لا يحتاج المستعمل لمعرفة طرق الوصول إلى البيانات كما لا يهمه معرفة كيفية تخزين البيانات فيزيائيا.
٧. جميع الأوامر المستعملة لتحديث البيانات والوصول إليها موجودة في لغة واحدة هي SQL.
٨. لا يوجد اعتماد على نوع طريقة الوصول أو التطبيق الذي يستعمل البيانات؛ حيث لا تتغير البيانات المخزنة بغض النظر عن نوع الوصول أو التطبيق. وهذا يُسمى "استقلالية" البيانات التامة . Full Data Independence

خصائص العلاقات الجداولية

يتميز الجدول الذي يمثل العلاقة بالخصائص التالية:

١. لا توجد صفات متكررة.
٢. لا توجد أعمدة بنفس الاسم.
٣. ترتيب الصفوف غير مهم.
٤. ترتيب الأعمدة غير مهم.
٥. القيم فردية Atomic أي لا تتجزأ إلى أعمدة أو حقول أخرى.

قارن ذلك بطرق أخرى لتمثيل وتخزين البيانات كالملفات السطحية Flat Files سابقة الذكر؛ حيث هناك أهمية لترتيب الحقول في السجل؛ وأهمية لترتيب السجلات في الملف؛ كما أن هناك حقول مقسمة إلى حقول فرعية، و يبدو ذلك أنه يعطي مرونة في البرمجة ولكن يحتاج إلى مهارات برمجية أخرى متعددة - وإن قامت بعض الشركات بتضمين هذه الفكرة في منتجاتها مثل أوراكل ولكن المحافظة على النموذج الالعائقي. وحيث أن هذه السلسلة لا تهدف إلى المقارنة بين تلك الطرق وقاعدة البيانات الالعائقيه ولكن هذه المقارنة البسيطة مجرد مرشد بسيط للقارئ الكريم كي يختار التوجه الذي يراه منتجاً أكثر.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧)

مبادئ لغة سكول SQL

السطور التالية عبارة عن تعريف بمبادئ لغة سكول SQL وهي اللغة المستعملة للوصول إلى قاعدة البيانات أوراكل - وغير أوراكل - والتي تبني عليها جميع منتجات أوراكل. حيث سيتم بحث المواضيع التالية على سبيل المثال:

- تعريف عام بلغة سكول وشكل الأوامر فيها وأنواعها
- استرجاع البيانات query
- العمليات الحسابية Arithmetic
- معالجة الحقول التي ليس لها قيمة Null Values
- بدائل أسماء الأعمدة Column Aliases
- الأعمدة الموصولة Concatenated Columns
- ترتيب الصفوف (النتائج) Ordering
- وضع الشروط في عمليات الاستعلام Restriction

ما هي لغة سكول SQL؟

يحتاج أي نظام إدارة قاعدة بيانات للتعامل مع البيانات المخزنة فيه إلى لغة استعلام للوصول إلى هذه البيانات. هذه اللغة يجب أن تكون بسيطة قدر الإمكان.

وبالنسبة لنظام إدارة قواعد البيانات العلاقية الذي سبقت مناقشته في الرسائل السابقة؛ توجد لغة استعلام اسمها "لغة الاستعلام الميكيلية Structured Query Language" وختصر إلى SQL وتنطق SEQUEL أو سكول باللغة العربية وهو ما سوف نستعمله هنا؛ حيث تُستعمل هذه اللغة في معظم نظم إدارة قواعد البيانات ذات التوجه العلائقى وأوراكل منها.

خصائص لغة سكول

تتميز سكول بمجموعة من الخصائص التي يجعل منها أداة سهلة وفعالة للتعامل مع البيانات ومن هذه الخصائص:

١. تشبه لغة سكول اللغة الإنجليزية العادية حيث تستعمل كلمات من مثل SELECT و UPDATE و ...الخ، وهذا يجعلها في متناول المستعمل العادي فضلاً عن المبرمج.
٢. لغة سكول ليست إجرائية non-procedural فلا تحتاج جمل شرطية أو حلقات تكرار أو غير ذلك، ما عليك سوى تحديد ما تريده وليس كيفية تنفيذ ما تريده.
٣. تعامل سكول مع مجموعة من الصنفوف (السجلات)-صنف أو أكثر - في نفس الوقت بدلاً من التعامل مع سجل واحد فقط.

٤. يستطيع نطاق واسع من المستعملين استخدام سكول وهذا يشمل مدير قواعد البيانات DBA والمبرمجين والمدراء والمستخدمين العاديين وغيرهم.
٥. تقدم سكول مجموعة أوامر تقوم بأعمال كثيرة منها:
 - ١) أوامر الاستعلام عن البيانات.
 - ٢) معالجة البيانات من إضافة وحذف وتعديل البيانات.
 - ٣) إنشاء وتغيير وإلغاء مكونات قاعدة البيانات مثل الجداول Tables والفهارس Indexes.
 - ٤) التحكم بكيفية الوصول إلى قاعدة البيانات مثل نظم الحماية.
 - ٥) ضمان التوافقية في البيانات .Data Consistency

هذا وإن لغة سكول معتمدة من المعهد الوطني الأمريكي للمقاييس ANSI ومنظمة المقاييس العالمية ISO

أوراكل مثلاً (٨)

 مجموعة أوامر سكول

تنقسم أوامر سكول إلى المجموعات الرئيسية التالية:

١. أوامر معالجة البيانات وهي:

SELECT

INSERT

UPDATE

DELETE

MERGE

وستعمل هذه الأوامر لاستخراج وإضافة صفوف وحذف صفوف وتعديل بيانات صفوف في الجدول وهي تُعرف باسم "لغة معالجة البيانات DML".

٢. أوامر معالجة هياكل البيانات وهي:

CREATE

ALTER

DROP

RENAME

TRUNCATE

COMMENT

وستعمل لإنشاء وتغيير وإلغاء مكونات قاعدة البيانات وتُعرف باسم "لغة تعريف البيانات DDL".

٣. أوامر التحكم في الوصول إلى البيانات (الصلاحيات Privileges) وهي:

GRANT

REVOKE

وتعمل باسم "لغة التحكم بالبيانات DCL" ولغرض منها تحديد صلاحيات المستعملين في الوصول إلى كل من البيانات الموجودة في قاعدة البيانات وهيأكل مكونات قاعدة البيانات ونزع هذه الصلاحيات.

٢ - لغة مراقبة العمليات Transactions Control وهي:

COMMIT

ROLLBACK

SAVEPOINT

والمهدف منها السيطرة على عمليات معالجة البيانات من إدخال وتعديل...أي العمليات التي تستعمل فيها لغة معالجة البيانات
وأمر SAVEPOINT خاص بأوراكل وليس قياسيًا DML.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٩)

□ الشكل العام لجمل سكول

لغة سكول تتكون من جمل Statements وليس من أوامر. والجملة مكونة من عبارات Clauses وكتب هذه الجمل حسب القواعد التالية:

١. يوجد فراغ واحد على الأقل بين كل كلمة من كلمات جملة سكول.
٢. يمكن كتابة جمل سكول على أكثر من سطر.
٣. لا يمكن كتابة كلمة سكول على أكثر من سطر.
٤. تنتهي جملة سكول بالفاصلة المدقونة (,).
٥. لا يهم نع الحرف كبيراً كان أو صغيراً.

أمثلة

الجمل التالية صحيحة:

SELECT * FROM employees;

SELECT

*

FROM

employees

;

SELECT *

FROM employees;

الجمل التالية غير صحيحة:

SEL

ECT * FROM EMPLOYEES;

SELECT * FROMEMPLOYEES;

وقبل البدء بشرح جمل SQL يجدر بنا تركيب قاعدة البيانات أوراكل على الجهاز الذي ستطبق عليه الأمثلة.

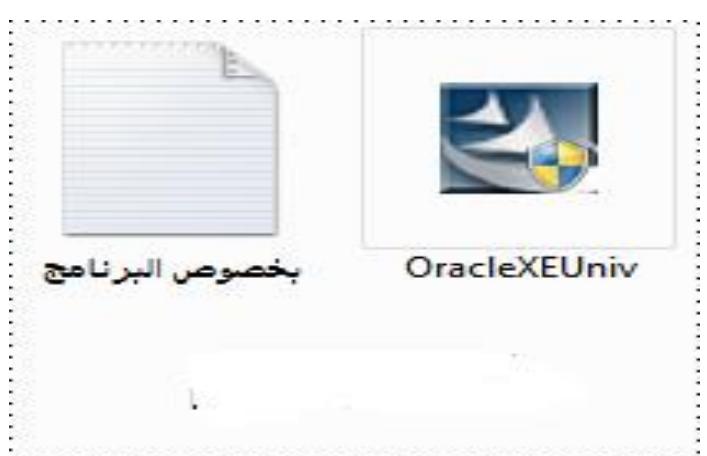
يوجد من أوراكل عدة نسخ إحداها اسمها أوراكل إكسبريس Oracle XE هدفها التعلم ويمكن استعمالها في الأنظمة التي لا يزيد حجم البيانات فيها عن 11 جيجابايت بما فيها بيانات أوراكل نفسها وهي نسخة مجانية. ونسخة أخرى تسمى نسخة الأعمال Enterprise Edition وحجمها يالتي بايت وأكبر من ذلك وهناك النسخة الشخصية Personal Edition والنسخ الأخيرة ليست مجانية. وكلها نسخة 10 هي أو أعلى. ويمكنك تركيب نسخة أوراكل أكبر من 10 هي.

قبل كل ذلك يجب تنزيل نسخة الأوراكل المرغوبة من موقع أوراكل وهذا يحتاج إلى اشتراك في موقع أوراكل والاشتراك في الموقع مجاني.

وسارق صور تركيب نسخة الأوراكل الصغيرة Express في ملف لصعوبة ترتيبها في الواتس أب.

بعد تحميل الملف المضغوط من موقع أوراكل؛ فك الضغط من ملف zip الذي تم تنزيله من موقع أوراكل. افتح المجلد الذي حصلت عليه واضغط بالنقر المزدوج على الملف setup لتشغيل برنامج التركيب. وتتابع التعليمات حسب الصور المرفقة.

بعد تحميل الملف المضغوط من موقع أوراكل؛ فك الضغط من ملف zip الذي تنزيله من موقع أوراكل. افتح المجلد الذي حصلت عليه واضغط بالنقر المزدوج على الملف setup لتشغيل برنامج التنصيب. وتتابع التعليمات حسب الصور المرفقة.





Welcome to the InstallShield Wizard for Oracle Database 10g Express Edition

The InstallShield® Wizard will install Oracle Database 10g Express Edition on your computer. To continue, click Next.



< Back **Next >** Cancel

تظهر الشاشة التالية. اضغط **Next**



License Agreement

Please read the following license agreement carefully.



ORACLE DATABASE 10g EXPRESS EDITION LICENSE AGREEMENT

To use this license, you must agree to all of the following terms (by either clicking the accept button or installing and using the program):

ELIGIBILITY EXPORT RESTRICTIONS

- I accept the terms in the license agreement 1
- I do not accept the terms in the license agreement 2

InstallShield -

< Back **Next >** Cancel

اضغط **Next** حتى تظهر لك الشاشة التالية

Oracle Database 10g Express Edition - Install Wizard

Specify Database Passwords

ORACLE
DATABASE
EXPRESS EDITION

Enter and confirm passwords for the database. This password will be used for both the SYS and the SYSTEM database accounts.

Enter Password

ضع كلمة سر هنا

Confirm Password

أعد كتابة كلمة السر هنا

Note: You should use the SYSTEM user along with the password you enter here to log in to the Database Home Page after the install is complete.

InstallShield

< Back

Next >

Cancel

أدخل كلمة السر لمدير قاعدة البيانات أوراكل system ولا تنساها لأن نسيانها مشكلة!

ثم اضغط next ليظهر لك زر install. اضغط عليه فيهر لك تقدم عملية التنصيب مشابهة للشاشة التالية

Oracle Database 10g Express Edition - Install Wizard

Setup Status

ORACLE
DATABASE
EXPRESS EDITION

The InstallShield® Wizard is installing Oracle Database 10g Express Edition

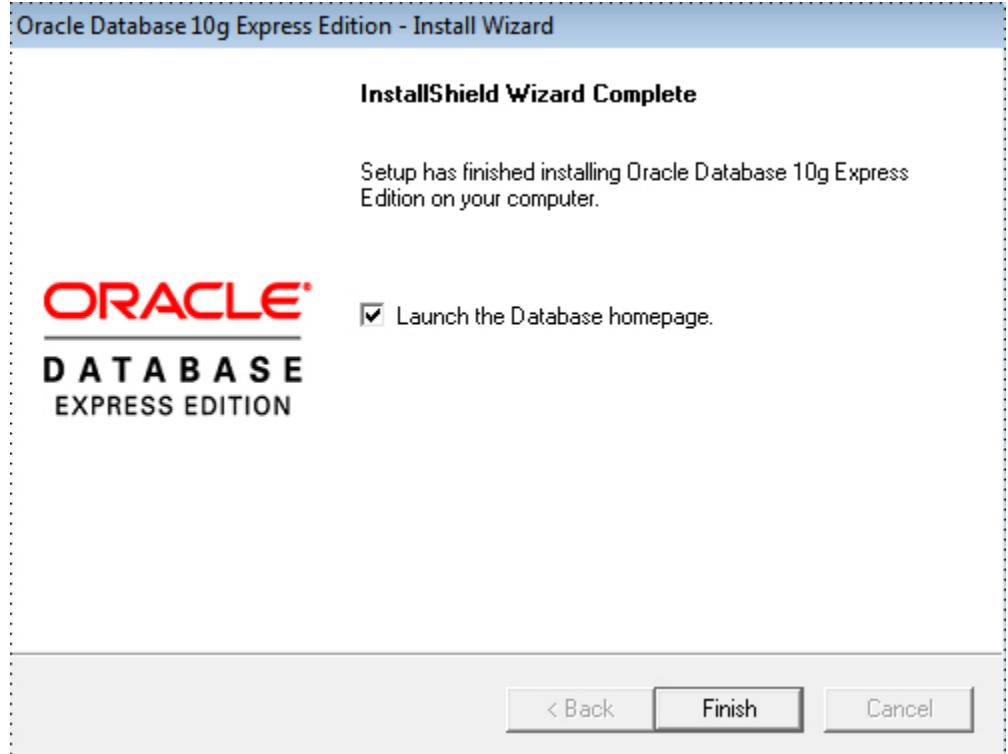
Removing system registry values



InstallShield

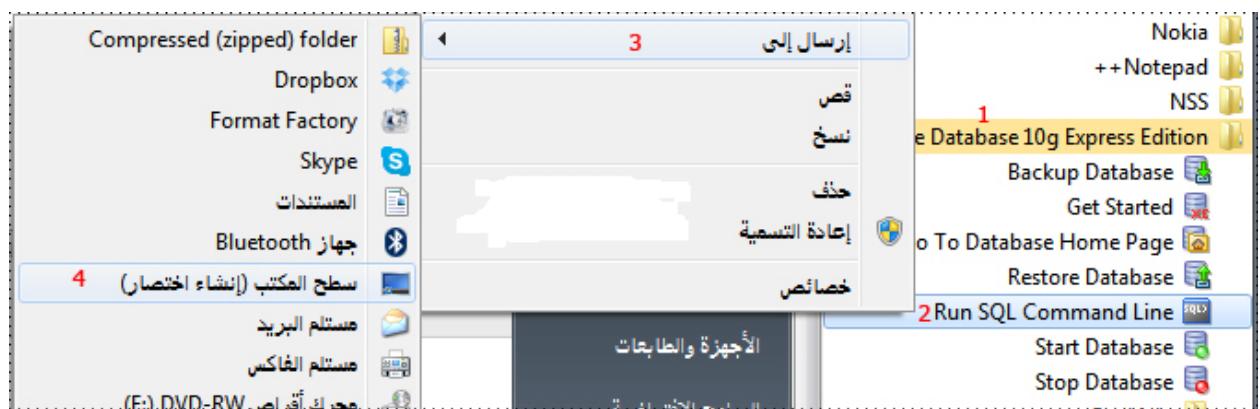
Cancel

بعد الانتهاء تظهر الشاشة التالية:



ألغى علامة الصح من المربع أعلاه ثم اضغط Finish

وللدخول إلى أوراكل اذهب إلى قائمة البرامج ثم أوراكل إكسبريس ثم sql كما في الشكل التالي



هنا تم إنشاء اختصار على سطح المكتب لشاشة الدخول إلى أوراكل. انقر مررتين على الاختصار فتظهر الشاشة التالية.

```
Run SQL Command Line
SQL*Plus: Release 10.2.0.1.0 - Production on
Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> connect system/fahad
Connected. ← تم الإتصال بنجاح
SQL>
```

أكتب الجملتين التاليتين واضغط enter بعد كل جملة:

alter user hr account unlock;

alter user hr identified by hr;

وسوف نتعلم لاحقاً عن هاتين الجملتين

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلا (١٠)

و قبل البدء في شرح جمل سكول أشير إلى أن أوراكل توفر مثلا تدريبيا في قاعدة بيانات لها من خلال خطط قاعدة بيانات - سنتعرف على هذا المفهوم . يحتوي المخطط على مجموعة من الجداول نستعملها لنتعلم جمل سكول . وهذه الجداول هي employees و departments و salegrade . أية جداول أخرى سيتم ذكرها في وقتها . وفيما يلي وصف أعمدة هذه الجداول

جدول الوظيفين Employees

وصف العمود	اسم العمود
رقم الموظف(وحيد)	EMPLOYEE_ID
اسم الموظف الأول	FIRST_NAME
اسم الموظف الأخير	LAST_NAME
البريد الإلكتروني	EMAIL
رقم التلفون	PHONE_NUMBER
تاريخ التعيين	HIRE_DATE
رمز الدالة	JOB_ID
الراتب	SALARY
نسبة العمولة	COMMISSION_PCT
رمز المدير	MANAGER_ID
رمز القسم	DEPARTMENT_ID

DEPARTMENTS جدول الأقسام

الوصف	اسم العمود
رقم القسم(وحيد)	DEPARTMENT_ID
اسم القسم	DEPARTMENT_NAME
رمز مدير القسم	MANAGER_ID
رمز الموقع	LOCATION_ID

salegrade جدول درجات الرواتب

الوصف	اسم العمود
الدرجة	GRADE_LEVEL
حد الراتب الأدنى للدرجة	LOWEST_SAL
حد الراتب الأعلى للدرجة	HIGHEST_SAL

و سنبدأ بشرح عملية الاستعلام query من جداول قاعدة البيانات . إن عملية الاستعلام هي الأساس في البحث عن البيانات واستخراج التقارير. ودوال هذه الجملة أساسا:

١. للاستعلام عن البيانات المخزنة في قاعدة البيانات وهذه العملية تسمى استعلاما QUER Y .
٢. القيام بالعمليات الحسابية.
٣. الاستعلام عن متغيرات بيئه العمل مثل التاريخ والوقت المستعمل... الخ.
- ٤ - استعمال جميع العمليات الجبرية على العلاقات في جملة SELECT .

وستبقى معنا جملة الاستعلام هذه من الآن فصاعدا حتى نترك البرمجة بلغة سكول أو نترك مجال الكمبيوتر كله! وتكون جملة الاستعلام أساسا مما يلي:

١. الكلمة SELECT .
 ٢. أسماء الأعمدة المراد عرضها (الإسقاط PROJECTION).
 ٣. الكلمة FROM .
 ٤. اسم الجدول table الذي سيتم استخراج بياناته.
- والشكل العام لجملة الاستعلام هو:

select col1,col2,... from table;

عرض أرقام الأقسام والاسم الأخير للموظفين ورواتبهم من جدول الموظفين EMPLOYEES نكتب ما يلي :

SELECT department_id, last_name, salary FROM employees;

تظهر لنا نتائج مشابهة للجدول أدناه

DEPARTMENT_ID	LAST_NAME	SALARY
50	OConnell	2600
50	Grant	2600
10	Whalen	4400
20	Hartstein	13000
20	Fay	6000
40	Mavris	6500
70	Baer	10000
110	Higgins	12000
110	Gietz	8300
90	King	24000
90	Kochhar	17000
90	De Haan	17000
60	Hunold	9000
60	Ernst	6000

لعلكم لاحظتم أننا لم نفتتح عن مؤشرات في الذاكرة ولا عن تقنيات البحث ولا أشغلنا أنفسنا بمكان وجود السجلات ولا ترتيبها على الديسك وغير ذلك من التفاصيل. فقط كتبنا ما نريد عرضه من جدول معين.

ولعرض جميع محتويات جدول ما؛ إما أن نكتب أسماء جميع الأعمدة أو نستعمل النجمة (*)؛ فلعرض جميع الأعمدة في جدول الأقسام DEPARTMENTS نكتب ما يلي:

SELECT * FROM departments;

تظهر نتائج مشابهة للجدول أدناه:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700

إن المثالين السابقين يعتبران مثلاً على ما يُسمى **شكل الاستعلام الأساسي Block Basic Query**.

ملاحظات

- ١ - عناوين الأعمدة Column Headings تظهر بصورة حرف كبير Uppercase
- ٢ - محاذة أو ترصيف العناوين للأعمدة العددية إلى اليمين Right Alignment
- ٣ - محاذة أو ترصيف العناوين للأعمدة Character والتاريخية Date تكون إلى الشمال Left Alignment
- ٤ - الأمثلة مستخرجة من بيئة آي إس كيو إل بلس iSQLPlus وهي بيئة تفاعلية خاصة بشركة أوراكل تعمل من خلال المستعرض على نسخة أوراكل ١٠ جي واستعاضت عنها أوراكل بيئة SQLDeveloper

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلا (١١)

لتحريك خطوة أخرى إلى الأمام.

إضافات يمكن استعمالها في جملة الاستعلام *SELECT*

يمكن استعمال إضافات في جملة الاستعلام *SELECT* تساعده في إيجاد صورة أكثر فهما للنتائج. وهذه الإضافات تشمل:

١. التعبيرات الحسابية Arithmetic Expressions

٢. الأسماء البديلة للأعمدة Column Aliases

٣. الأعمدة المتصلة Concatenated Columns

٤. إضافات معنوية أو حرفية Literals

٥. الدوال(الدوال) Functions وسنفرد لها فصلا خاصا

وفيما يلي أمثلة لكل نوع.

العبارات الحسابية Arithmetic Expressions

يمكن استعمال العمليات الحسابية المعروفة وهي الجمع والطرح والضرب والقسمة بحسب أهميتها الحسابية المعروفة؛ الضرب والقسمة أولاً-أيهمما على اليسار-ثم الجمع والطرح-أيهمما على اليسار-أما إذا تضمن التعبير الحسابي أقواسا فللعمليات التي بداخل الأقواس الأولوية. وتستعمل العمليات الحسابية مع البيانات من نوع عدد Number وتاريخ Date مع الأخذ بعين الاعتبار عدم استعمال الضرب والقسمة مع البيانات من نوع تاريخ Date.

ولأجل العمليات الحسابية نستعمل العوامل الحسابية المعروفة Operators التالية والظاهره في الجدول أدناه

الوصف Description	العامل Operator
الضرب Multiply	*
القسمة Divide	/
الطرح Subtract	-
الجمع Add	+

مثال

١. حساب راتب الموظف مضافا إليه مبلغ ٣٠٠ نكتب الجملة التالية:

SELECT last_name, salary, salary + 300 FROM employees;

LAST_NAME	SALARY	SALARY+300
OConnell	2600	2900
Grant	2600	2900
Whalen	4400	4700
Hartstein	13000	13300
Fay	6000	6300
Mavris	6500	6800
Baer	10000	10300
Higgins	12000	12300
Gietz	8300	8600
King	24000	24300

أولويات العوامل Operators Precedence

لندرس المثالين التاليين ثم نعلم عليةما

المثال الأول

SELECT last_name, salary, 12*salary+100 FROM employees;

LAST_NAME	SALARY	12*SALARY+100
OConnell	2600	31300
Grant	2600	31300
Whalen	4400	52900
Hartstein	13000	156100
Fay	6000	72100

Mavris	6500	78100
Baer	10000	120100
Higgins	12000	144100
Gietz	8300	99700
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100

المثال الثاني

SELECT last_name, salary, 12*(salary+100) FROM employees;

LAST_NAME	SALARY	12*(SALARY+100)
OConnell	2600	32400
Grant	2600	32400
Whalen	4400	54000
Hartstein	13000	157200
Fay	6000	73200
Mavris	6500	79200
Baer	10000	121200
Higgins	12000	145200
Gietz	8300	100800
King	24000	289200
Kochhar	17000	205200

De Haan	17000	205200
Hunold	9000	109200
Ernst	6000	73200
Austin		

في المثال الأول تمت عملية ضرب الراتب بـ ١٢ ثم جمع ١٠٠ على النتيجة بينما في المثال الثاني استعملنا قوسين لإحاطة عملية الجمع ثم تمت عملية ضرب الناتج بـ ١٢؛ ولذلك لاحظت الفرق في النتيجة. وعليه تكون الأوليات كما يلي؛ الأقواس أولاً ثم عملية الضرب والقسمة أيهما على شمال التعبير يكون قبل الثاني ثم الجمع والطرح أيهما على الشمال يكون قبل الثاني.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (١٢)

٢- الأسماء البديلة Column Aliases

الأسماء البديلة هي أسماء أخرى من وضع المستعمل للأعمدة والتعبيرات بحيث تعطي إمكانية أفضل للقراءة من أسماء الأعمدة التي تستعملها أوراكل من خلال تعريف الجدول؛ فيمكن استعمال كلمة ANNUALSAL بدلاً من SALARY*12 كما في المثال الأول في بند العمليات الحسابية. وخصائص الأسماء البديلة:

١. أنها تظهر بحروف كبيرة دائماً Upper Case
٢. لا يجوز أن تحتوي على فراغات
٣. يوجد فراغ واحد على الأقل بين العمود واسم البديل
٤. لا توجد فاصلة بين العمود واسم البديل

ويمكن احتواء الشرطين الأول والثاني بتضمين الاسم البديل ضمن زوج من الحاصلتين العلويتين ""- Double Quotation

أمثلة:

١. في هذا المثال سوف نستعيض عن اسم العمود commission_pct باسم comm بكلمة commission_name؛ والعمود last_name

:comm

SELECT last_name AS name, commission_pct comm FROM employees;

NAME	COMM
OConnell	
Grant	
Whalen	
Hartstein	
Fay	
Mavris	
Baer	
Higgins	

Gietz

King

لاحظ كيف ظهر الاسم البديل بالحروف الكبيرة بالرغم أنا كتبناه بالحروف الصغيرة. أيضا لاحظ العمود comm والذي كتبناه بالحروف الصغيرة تحول إلى الحروف الكبيرة. وهكذا أوراكل دائما تحول إلى الحروف الكبيرة ما لم يكن هناك قيد عليها.

لاحظ أيضا أنها استعملنا AS مرة ولم نستعملها مرة أخرى وهذا يعني أن AS اختيارية وليس إجبارية

٢. في هذا المثال سنلغي مفعول الشرطين الأول والثاني للأسماء البديلة كما يلي:

SELECT last_name "Name" , salary*12 "الراتب السنوي" FROM employees;

Name	الراتب السنوي
OConnell	31200
Grant	31200
Whalen	52800
Hartstein	156000
Fay	72000
Mavris	78000
Baer	120000
Higgins	144000
Gietz	99600

لاحظ كيفية القفز على ذيئن الشرطين في هذا المثال وظهر الاسم البديل كما نريده.

إن الأسماء البديلة مفيدة في حالة التقارير السريعة؛ ولو كان نظامك يدعم اللغة العربية بإمكانك استعمال اللغة العربية لذلك كما في المثال أعلاه المطبق في بيئة النوافذ windows.

الأعمدة الموصولة Concatenated Columns

الأعمدة الموصولة تعني أنه يمكن أن تظهر مجموعة أعمدة (أو تعبيرات expressions) وكأنها عمود واحد. فيمكن أن نعرض رقم الموظف واسمه كعمود مثلا.

ولمذه الغاية نستعمل العامل " | " كما يلي:

```
SELECT last_name || job_id AS "Employees" FROM employees;
```

Employees
OConnellSH_CLERK
GrantSH_CLERK
WhalenAD_ASST
HartsteinMK_MAN
FayMK_REP
MavrisHR_REP
BaerPR_REP
HigginsAC_MGR
GietzAC_ACCOUNT
KingAD_PRES

الإضافات المعنوية Literals

وهي أي رمز أو مجموعة رموز أو تعبيرات أو أرقام على أن تكون التعبيرات من نوع رمز أو تاريخ محصورة بين حاصلتين علويتين (" ") وهي ليست اسم عمود وليس اسم بديلا؛ فهي ليست من تركيبة الجدول نفسه.

مثال: هنا نعرض عبارة a is كإضافة معنوية.

```
SELECT last_name || ' is a ' || job_id AS "Employee Details"
```

```
FROM employees;
```

Employee Details
OConnell is a SH_CLERK

Grant is a SH_CLERK
Whalen is a AD_ASST
Hartstein is a MK_MAN
Fay is a MK_REP
Mavris is a HR_REP
Baer is a PR_REP
Higgins is a AC_MGR
Gietz is a AC_ACCOUNT
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP

مثال آخر :

في هذا المثال سيتم عرض تاريخ النظام ضمن جدول الموظفين علما بأن تاريخ النظام ليس من تركيبة جدول Structure الموظفين.

SELECT last_name || ': 1 Month salary = '|| salary Monthly, SYSDATE FROM employees

King: 1 Month salary = 24000	09-JUL-20
Kochhar: 1 Month salary = 17000	09-JUL-20
De Haan: 1 Month salary = 17000	09-JUL-20
Hunold: 1 Month salary = 9000	09-JUL-20
Ernst: 1 Month salary = 6000	09-JUL-20
Austin: 1 Month salary = 4800	09-JUL-20

Pataballa: 1 Month salary = 4800	09-JUL-20
Lorentz: 1 Month salary = 4200	09-JUL-20
Greenberg: 1 Month salary = 12008	09-JUL-20
Faviet: 1 Month salary = 9000	09-JUL-20
Chen: 1 Month salary = 8200	09-JUL-20
Sciarra: 1 Month salary = 7700	09-JUL-20
Urman: 1 Month salary = 7800	09-JUL-20

أوراكل مثلاً (١٣)

 معالجة الأعمدة التي ليس لها قيمة *NULL Values*

ستتناول هنا معالجة الأعمدة التي لا تحتوي شيئاً أي ليس فيها أية قيمة؛ ونحب الملاحظة هنا أن الفراغات **Spaces** ولو واحداً فقط هو قيمة كما أن الصفر أيضاً قيمة، أما اللاشيء **Null** فلا قيمة له. فإن رقم الصفر في حدول الآسكنى هو ٤٨ بينما رقم الآسكنى للفراغ هو ٣٢ بينما رقم الآسكنى للاشيء هو الصفر. وسوف أشرح مفهوم اللاشيء **Null** واستعماله في دروس قادمة إن شاء الله.

إذا لم يكن هناك بيانات في العمود فإن أوراكل لا تفترض أية قيمة مسبقة فلا تفترض الصفر للأعمدة العددية ولا تفترض الفراغ **Space** للأعمدة الرمزية-سيتم بحث أنواع البيانات لاحقاً وهذا يوفر مساحة على وسائل التخزين.

تعالج أوراكل اللاشيء **NULL** بحيث إذا تضمنت أية عملية حسابية اللاشيء **NULL** تكون نتيجتها أيضاً لاشيء؛ فعلى سبيل المثال لو أردنا تقريراً عن دخل الموظفين السنوي(الراتب السنوي+العمولات) فإننا نكتب ما يلي:

```
SELECT last_name, 12*salary*commission_pct FROM employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
OConnell	
Grant	
Whalen	
Hartstein	
Fay	
Mavris	
Baer	
Higgins	
Gietz	
King	
Partners	48600

Errazuriz	43200
Cambrault	39600
Zlotkey	25200

لاحظ النتيجة غير المتوقعة أن الذين ليس لهم عمولات كان دخلهم السنوي لاشيء! وهذا غير صحيح طبعا. ولمعالجة هذا الوضع علينا أن نقوم بتحويل العمولات التي قيمها اللاشيء إلى قيمة أخرى-صفر مثلا-حيث سنتعمل دالة Function لهذا الغرض هي NVL كما يلي من المثال السابق:-سيتم بحث الدوال أو الدوال في دروس منفصلة

```
SELECT last_name, 12*salary*NVL(commission_pct,0)+12*salary
```

```
FROM employees;
```

LAST_NAME	12*SALARY*NVL(COMMISSION_PCT,0)+12*SALARY
OConnell	31200
Grant	31200
Whalen	52800
Hartstein	156000
Fay	72000
Tucker	156000
Bernstein	142500
Hall	135000
Olsen	115200
Cambrault	108000

: Argument NVL لها عاملان وبصورة عامة فإن الدالة NVL

١. التعبير Expression والمقصود به أي تعبير مثل اسم العمود أو تعبير حسابي أو تاريخ...الخ.
٢. قيمة غير اللاشيء وهذه القيمة قد لا تكون صفراء بل يمكن أن تكون حسب متطلبات العمل.

ويمكن أن تكون القيمة أي شيء؛ ولكنها على كل حال تعتمد على نوع البيانات الموجودة ضمن التعبير في حالة استعمالها؛ فإن كان التعبير عديا مثل العمولة commission_pct فيجب أن تكون القيمة عددية مثل صفر أو ٢٠٠... الخ. أما إن كان التعبير من نوع تاريخ Date Type فيجب أن تكون القيمة كذلك ونفس الأمر ينطبق على التعبير النصية(الرمزية) Character.

وفيما يلي أمثلة:

١. NVL(numericexpression,100)

٢. NVL(dateexpression,'01-JAN-96')

٣. NVL(characterexpression,'Any String')

وبذلك تتم معالجة التعبيرات التي لا تحتوي شيئاً NULL Values

معالجة تكرار الصيغ

عادة تقوم جملة الاستعلام بعرض جميع الصيغ المكررة إلا إذا رغبت عدم عرض الصيغ المكررة(الصيغ المكررة تعني تكرار قيم بعض الأعمدة وليس جميعها-مثل رقم القسم). فمثلاً لعرض أرقام الأقسام من جدول الموظفين نكتب: EMPLOYEES

```
SELECT department_id FROM employees;
```

DEPARTMENT_ID
50
50
10
20
20
40
70

ولتجنب التكرار نضيف كلمة DISTINCT بعد الكلمة SELECT مباشرة فنمنع التكرار كما في المثال السابق بعد إضافة DISTINCT.

```
SELECT DISTINCT department_id FROM employees;
```

DEPARTMENT_ID

100
30
20
70
90
110
50

شروط استعمال *DISTINCT*

١. أن تتبع الكلمة **SELECT** مباشرة.
٢. أن تكتب مرة واحدة فقط في جملة **SELECT**.

أوراكل مثلاً (١٤)

 **معالجة ترتيب الصفوف** 

نتذكر أننا لم نختتم كيف يتم تخزين البيانات في قاعدة البيانات؛ فهذه مهمة نظام إدارة قواعد البيانات. واهتمامنا استخراج البيانات بالطريقة التي نريد. وأعطتنا SQL أساليب بسيطة لترتيب المخرجات دون التعقيدات أو الخوارزميات المعماول بها في اللغات التقليدية. لذلك لإظهار الصفوف مرتبة حسب العمود المرغوب الترتيب عليه نستعمل عبارة ORDER BY متبوعة باسم العمود؛ بحيث تكون عبارة ORDER BY هي آخر عبارة في جملة SELECT Clause. في المثال التالي سيتم ترتيب الصفوف حسب تاريخ التعيين(الأقدم فالأحدث):

```
SELECT last_name, job_id, department_id, hire_date FROM employees  
ORDER BY hire_date ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91
De Haan	AD_VP	90	13-JAN-93
Mavris	HR_REP	40	07-JUN-94

يتم ترتيب الصفوف تصاعدياً بصورة تلقائية Default حسب نوع البيانات كما يلي:

١. يتم ترتيب الأعداد الأصغر أولاً ثم الأكبر.
٢. يتم ترتيب التواريخ بحيث يكون التاريخ الأقدم أولاً.
٣. يتم ترتيب النصوص حسب الترتيب الأبجدي.

 **عكس الترتيب التلقائي للصفوف** 

يمكن عكس الترتيب التلقائي للصفوف باستعمال الكلمة DESC بعد اسم العمود. فمثلاً لعرض بيانات الموظفين الأحدث تعينينا(مرتبة حسب تاريخ التعيين HIRE_DATE بصورة عكسية) نكتب :

```

SELECT last_name, job_id, department_id, hire_date FROM employees
ORDER BY hire_date DESC ;

```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Banda	SA_REP	80	21-APR-00
Kumar	SA_REP	80	21-APR-00
Ande	SA_REP	80	24-MAR-00
Markle	ST_CLERK	50	08-MAR-00
Lee	SA_REP	80	23-FEB-00
Philtanker	ST_CLERK	50	06-FEB-00
Geoni	SH_CLERK	50	03-FEB-00
Zlotkey	SA_MAN	80	29-JAN-00

الترتيب على أكثر من عمود

يمكن استعمال أكثر من عمود في عبارة ORDER BY بحيث يكون تأثير الأعمدة ابتداءً من العمود الذي على أقصى الشمال. يتم ذكر الأعمدة المرغوب الترتيب عليها بوجود فاصلة بين كل عمودين؛ وللترتيب العكسي نضيف كلمة DESC كما سبق بعد آخر عمود أو بعد العمود المراد عكس الترتيب عليه.

لمثال التالي يتم عرض البيانات مرتبة حسب رقم القسم DEPARTMENT_ID الأصغر فالأكبر وحسب الراتب SALARY الأكبر فالأصغر.

```

SELECT last_name, job_id, department_id, salary FROM employees
ORDER BY department_id,salary DESC ;

```

LAST_NAME	JOB_ID	DEPARTMENT_ID	SALARY
Whalen	AD_ASST	10	4400
Hartstein	MK_MAN	20	13000
Fay	MK_REP	20	6000
Raphaely	PU_MAN	30	11000

Khoo	PU_CLERK	30	3100
Baida	PU_CLERK	30	2900
Tobias	PU_CLERK	30	2800
Himuro	PU_CLERK	30	2600
Colmenares	PU_CLERK	30	2500
Mavris	HR_REP	40	6500

الترتيب حسب قيمة اللاتيجة Null Values

إذا لم تكن هناك بيانات Null في عمود ما في أحد الصفوف، فالترتيب ممكن أيضاً؛ حيث يتم إظهار تلك الصفوف آخرًا في حالة الترتيب التصاعدي(اللتلقائي) وتظهر أولاً في حالة الترتيب التنازلي كما في المثالين التاليين.

```
SELECT last_name, job_id, commission_pct FROM employees
ORDER BY commission_pct ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	COMMISSION_PCT
Abel	SA_REP	80	.3
Smith	SA_REP	80	.3
Partners	SA_MAN	80	.3
Errazuriz	SA_MAN	80	.3
Tucker	SA_REP	80	.3
Cambrault	SA_MAN	80	.3
Doran	SA_REP	80	.3
King	SA_REP	80	.35
Sully	SA_REP	80	.35
McEwen	SA_REP	80	.35
Russell	SA_MAN	80	.4
OConnell	SH_CLERK	50	
Grant	SH_CLERK	50	

Whalen	AD_ASST	10	
Hartstein	MK_MAN	20	
Fay	MK_REP	20	

```
SELECT last_name, job_id, commission_pct FROM employees
ORDER BY commission_pct DESC;
```

LAST_NAME	JOB_ID	COMMISSION_PCT
OConnell	SH_CLERK	
Grant	SH_CLERK	
Whalen	AD_ASST	
Hartstein	MK_MAN	
Fay	MK_REP	
Russell	SA_MAN	.4
McEwen	SA_REP	.35
Sully	SA_REP	.35
King	SA_REP	.35
Doran	SA_REP	.3

{ملاحظة: الترتيب لا يؤثر على التمثيل الفيزيائي للبيانات}

الترتيب حسب رقم العمود

تعطي سُكُون رقم # ID لكل عمود مذكور في جملة الاستعلام يدل على ترتيبه فيها؛ ويمكن استعمال هذا الرقم في جملة الاستعلام بدلاً من ذكر اسم العمود كما في المثال التالي:

```
SELECT job_id, last_name, commission_pct FROM employees
ORDER BY 1, 2;
```

JOB_ID	LAST_NAME	COMMISSION_PCT
AC_ACCOUNT	Gietz	
AC_MGR	Higgins	

AD_ASST	Whalen	
AD_PRES	King	
AD_VP	De Haan	
AD_VP	Kochhar	
FI_ACCOUNT	Chen	
FI_ACCOUNT	Faviet	
FI_ACCOUNT	Popp	
FI_ACCOUNT	Sciarra	
FI_ACCOUNT	Urman	
FI_MGR	Greenberg	

لاحظ الترتيب على العمودين job_id ورقمه last_name هو ١ و ورقمه هو ٢ .

أوراكل مثلاً (١٥)

استخراج البيانات بشرط Restriction

في الأمثلة السابقة كان يتم عرض جميع الصفوف في الجدول. ولكن نحتاج في كثير من الأحيان إلى عرض مجموعة من الصفوف فقط لذلك نحدد شرطاً أو شروطاً معينة؛ يتم تضمينها في جملة SELECT وهذا يُسمى التقييد Restriction في جبر العلاقات Relational Algebra. ولتنفيذ التقييد في لغة SQL نستعمل الكلمة WHERE مت preceding the condition بشرط أو شروط بعد ذكر اسم الجدول. والشكل العام لجملة SELECT سوف يكون كما يلي:

```
SELECT    column(s)
FROM      table
WHERE     condition(s)
ORDER BY  column(s)
```

لاحظ أن العبارة المكونة من `.FROM table` والشرط الذي يتبعها يجب أن تتبعد عبارة WHERE

ت تكون عبارة WHERE مما يلي:

١. كلمة WHERE

٢. اسم عمود Column Name

٣. عامل مقارنة Comparison Operator

٤. اسم عمود أو ثابت Constant أو مجموعة قيم

عوامل المقارنة تنقسم إلى مجموعتين رئيسيتين هما العوامل المنطقية Logical Operators وعوامل SQL Operators.

العوامل المنطقية Logical Operators

في الجدول التالي ملخص العوامل المنطقية

العامل المنطقي	الدالة(المعنى)
المساواة Equal	=
أكبر من Greater Than	>
أقل من Less Than	<
أكبر أو يساوي	>=

الشروط التي تتضمن حقوقاً نصية أو حقوقاً تاريخية

إن أنواع البيانات - بصورة أساسية - في أوراكيل هي إما رقمية Number أو تاريجية Date وسوف نعالج ظهور هذه الأنواع في عبارة الشرط WHERE. فالنسبة للحقول العددية لا توجد مشكلة؛ أما الحقول النصية والتاريجية فيجب أن تكون ضمن فاصلتين علويتين " ". وبالنسبة للحقول النصية فمن المهم مراعاة حالة الحروف كبيرة Upper أو صغيرة Lower كي تظهر النتائج المطلوبة.

ملاحظة: من الممكن استعمال الدوال (الدوال) Functions ضمن عبارة الشرط.

أمثلة:

١. لعرض الاسم والرقم ورمز المهنة ورقم القسم لجميع الموظفين في القسم رقم ٩٠ نكتب:

```
SELECT employee_id, last_name, job_id, department_id
```

```
FROM employees
```

```
WHERE department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

٢. لعرض الاسم والرقم ورمز المهنة وتاريخ التعيين لجميع الموظفين الذين تواريخ تعينهم قبل الأول من كانون الثاني (يناير) عام

١٩٩٤ نكتب:

```
SELECT employee_id, last_name, job_id, hire_date
```

```
FROM employees
```

```
WHERE hire_date < '01-JAN-94';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE
200	Whalen	AD_ASST	17-SEP-87

100	King	AD_PRES	17-JUN-87
101	Kochhar	AD_VP	21-SEP-89
102	De Haan	AD_VP	13-JAN-93
103	Hunold	IT_PROG	03-JAN-90
104	Ernst	IT_PROG	21-MAY-91

عوامل سِكُّول SQL Operators

الجدول التالي يحتوي على عوامل سِكُّول والتي تعمل مع جميع أنواع البيانات أيضاً.

العامل	الدالة(المعنى)
BETWEEN...AND...	بين قيمتين معينتين(مشمولتان)
IN(<i>List</i>)	لتطابقة قيمة معينة من مجموعة قيم
LIKE	لتطابقة جزء من نص Character Pattern
IS NULL	للبحث عن الالاشيء Null

وسنبدأ بشرح هذه العوامل:

. ١ BETWEEN .

عرض جميع القيم التي تقع ضمن قيمتين دنيا وعليها حيث يتم شمول تلك القيمتين إن وجدتا ضمن بيانات الجدول.

لنفرض أننا نريد تقريراً بالموظفين الذي تتراوح رواتبهم بين ٢٥٠٠ و ٣٠٠٠ فنكتب:

SELECT last_name, salary

FROM employees

WHERE salary BETWEEN 2500 AND 3500 ;

LAST_NAME	SALARY
OConnell	2600
Grant	2600

Khoo	3100
Baida	2900
Tobias	2800
Himuro	2600
Colmenares	2500
Nayer	3200
Mikkilineni	2700
Bissot	3300

لاحظ أن القيمة ٢٥٠٠ (القيمة الدنيا) ضمن الشرط وضمن البيانات بينما القيمة ٣٥٠٠ (القيمة العليا) ضمن الشرط وليس ضمن البيانات لأنها ليست موجودة في الجدول.

IN .٢

لاختبار قيمة ضمن مجموعة .set

عرض بيانات الموظفين الذين مدیرهم واحد من المدراء الذي أرقامهم ١٠٠ أو ١٠١ أو ٢٠١ نكتب:

SELECT employee_id, last_name, salary, manager_id

FROM employees

WHERE manager_id IN (100, 101, 201) ;

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
201	Hartstein	13000	100
101	Kochhar	17000	100
102	De Haan	17000	100
114	Raphaely	11000	100
120	Weiss	8000	100

	121	Fripp	8200	100
	122	Kaufling	7900	100
	123	Vollman	6500	100
	124	Mourgos	5800	100
	145	Russell	14000	100
	146	Partners	13500	100
	147	Errazuriz	12000	100
	148	Cambrault	11000	100
	149	Zlotkey	10500	100
	200	Whalen	4400	101
	203	Mavris	6500	101
	204	Baer	10000	101
	205	Higgins	12000	101
	108	Greenberg	12000	101
	202	Fay	6000	201

يحدث كثيراً أنك لا تعرف بالضبط القيمة التي تبحث عنها، ولكن باستعمال العامل LIKE يمكنك تحديد جزء من القيمة التي تبحث عنها وبالتالي تحصل على تلك القيمة(القيم).

يوجد رمزان يستعملان مع العامل LIKE وهما:

١ . % (علامة النسبة المئوية) وتمثل أي سلسلة من الرموز(قد لا يوجد أي رمز يطابق الشرط).

٢ . _ (الشرطه السفلية) وتمثل أي رمز مفرد.(رمزا واحدا فقط)

فلعرض الموظفين الذي تبدأ أسماؤهم بحرف S مثلا نكتب :

```
SELECT first_name FROM employees WHERE first_name LIKE 'S%' ;
```

FIRST_NAME
Sundar
Shelli
Sarah
Shelley
Steven
Sundita
Steven
Susan
Samuel
Sarath
Stephen
Sigal
Shanta

ولعرض الموظفين الذي تنتهي أسماؤهم بحرف r (الصغير) نكتب :

SELECT ENAME FROM EMP WHERE ENAME LIKE '%r';

FIRST_NAME
Sundar
Jennifer
Tayler
Peter
Alexander
Alexander
Christopher
Peter
Oliver
Peter
Jennifer

ولعرض الموظفين الذي تحتوي أسماؤهم على حرف L نكتب :

SELECT first_name

FROM employees

WHERE first_name LIKE '%L%' ;

FIRST_NAME
Laura
Lex

Louise

Lisa

Luis

Lindsey

لعرض الموظفين الذي تتكون أسماؤهم الأخيرة من أربعة حروف فقط نكتب : (أربع شرطات سفلية underscore)

SELECT last_name FROM employees WHERE last_name LIKE'____'

LAST_NAME

Abel

Ande

Baer

Bell

Bull

Chen

Hall

Khoo

King

King

Ozer

Popp

Rajs

13 rows selected.

لاحظ أن الشرطة السفلية ظهرت متصلة كأنها شرطة واحدة طويلة. ويمكن استعمال كلا من `%` و `_` سوية وبأي تركيب للحصول على النتائج المطلوبة.

٤ . IS NULL

للبحث عن الصفوف التي تحتوي أعمدة قيمها الالاشيء `NULL`. فلعرض أسماء الموظفين الذي ليس لهم مدير نكتب:

```
SELECT last_name FROM employees WHERE manager_id IS NULL
```

LAST_NAME
King

جرب أن تكتب الجملة السابقة كما يلي:

```
SELECT last_name FROM employees WHERE manager_id = NULL;
```

```
SELECT last_name FROM employees WHERE manager_id != NULL;
```

راقب النتيجة وأخبرنا عنها

أوراكل مثلاً (١٦)

الاستعلام باستعمال أكثر من شرط

لا بد أن الحاجة موجودة للاستعلام من خلال عدة شروط؛ ولهذا الغرض نستعمل العاملين AND و OR حيث أن تتطلب أن تكون جميع الشروط صحيحة TRUE بينما OR تكتفي بأي شرط يكون صحيحاً مع العلم أن لها أولوية على OR . في المثالين التاليين سوف نستخدم AND و OR لاستخراج نفس البيانات ونلاحظ كيف اختلفت النتائج.

أولاً: AND

في هذا المثال نريد عرض بيانات الموظفين الذين ١) تراوح رواتبهم بين ١٠٠٠٠ و ٢٠٠٠٠ و ٢) وظيفتهم مثل %MA%. (نريد تحقيق الشرطين معاً)

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary BETWEEN 10000 AND 20000
AND job_id LIKE '%MA%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
201	Hartstein	MK_MAN	13000
114	Raphaely	PU_MAN	11000
145	Russell	SA_MAN	14000
146	Partners	SA_MAN	13500

جدول الحقيقة Truth Table التالي يبين عمل AND مع القيم TRUE و FALSE و NULL

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE

NULL	NULL	FALSE	NULL
------	------	-------	------

ثانياً: OR

في المثال التالي نريد عرض نفس البيانات ولكن بتحقق أحد الشرطين فقط؛ الذين يعملون بدالة تحتوي على 'MA%' أو الذين تتراوح رواتبهم بين 10000 و 20000.

```

SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary BETWEEN 10000 AND 20000
OR job_id LIKE '%MA%';

```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
201	Hartstein	MK_MAN	13000
204	Baer	PR_REP	10000
205	Higgins	AC_MGR	12000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
108	Greenberg	FI_MGR	12000
114	Raphaely	PU_MAN	11000
120	Weiss	ST_MAN	8000
121	Fripp	ST_MAN	8200
122	Kaufling	ST_MAN	7900
123	Vollman	ST_MAN	6500
124	Mourgos	ST_MAN	5800
145	Russell	SA_MAN	14000

146	Partners	SA_MAN	13500
147	Errazuriz	SA_MAN	12000
148	Cambrault	SA_MAN	11000
149	Zlotkey	SA_MAN	10500

لاحظ اختلاف النتائج بصورة دراماتيكية!

وجدول الحقيقة Truth Table التالي يبين عمل OR مع القيم FALSE و TRUE و NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

الجدول التالي يلخص قواعد الأولويات عموماً

الترتيب	العوامل والشروط
١	العامل الحسابية Arithmetic operators
٢	عامل الوصل Concatenation operator
٣	عوامل المقارنة Comparison conditions
٤	IS [NOT] NULL, LIKE, [NOT] IN
٥	[NOT] BETWEEN
٦	عدم التساوي Not equal to
٧	NOT
٨	AND
٩	OR

ويمكن بخاوز القواعد السابقة باستعمال الأقواس التي لها الأولوية دائمًا

في المثال التالي سيتم عرض الموظفين الذين ١ - يعملون بدالة مثل ممثل مبيعات SA_REP و ٢ - الذين هم مشرفون AD_REPS ورواتبهم أعلى من ١٥٠٠٠.

```
SELECT last_name, job_id, salary  
FROM employees  
WHERE job_id = 'SA_REP'  
OR job_id = 'AD_PRES'  
AND salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Tucker	SA_REP	10000
Bernstein	SA_REP	9500
Hall	SA_REP	9000
Olsen	SA_REP	8000
Cambrault	SA_REP	7500
Tuvault	SA_REP	7000
King	SA_REP	10000
Sully	SA_REP	9500
McEwen	SA_REP	9000
Smith	SA_REP	8000
Doran	SA_REP	7500
Sewall	SA_REP	7000
Vishney	SA_REP	10500

نص الجملة السابقة هو: اعرض بيانات الموظفين الذين هم مدراء و رواتبهم أعلى من ١٥٠٠٠، أو الذين هم ممثلو مبيعات

أما إذا أردنا بيانات الذين يعملون بدالة مدير MANAGER أو مثل مبيعات SALESMAN الذين رواتبهم أكبر من 15000 - كما في المثال التالي - فإننا نضع شرط الدالة بين قوسين؛ حيث أن الأولوية دائماً للأقواس فيتم تنفيذ شرط OR أولاً ثم شرط AND أخيراً.

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

. يتم تقييم الشرط OR أولاً بحيث يصبح شيئاً واحداً ثم يتم ربطه مع الشرط الثاني salary > 15000 بواسطة AND

ونقرأ الجملة كالتالي: اعرض بيانات الموظفين الذين هم (ممثلو مبيعات أو مدراء) و رواتبهم أعلى من 15000

ملخص جملة الاستعلام

من المناقشات السابقة نستطيع كتابة الشكل العام لجملة الاستعلام SELECT كما يلي:

```
SELECT * | [DISTINCT]{expr | column[alias],...}
```

```
FROM table
```

```
WHERE condition(s)
```

```
ORDER BY {column | expr,...} [ASC | DESC];
```

والجدول التالي ملخص لعناصر جملة الاستعلام (الرموز مثل | و [] و {} ... ليست من جملة الاستعلام وإنما وضعت لشرح دلالتها فقط)

العنصر	المعنى
	العمود يحتم علينا اختيار أحد الطرفين؛ إما الأيسر فقط أو الأيمن فقط
[]	ما داخل الأقواس المربعة اختياري
{}	يمكن اختيار بين العناصر التي داخل الأقواس المعقولة {} بحد أدنى

عنصر واحد	
أمر الاستعلام	SELECT
الاسم البديل	Alias
جميع الأعمدة	*
منع التكرار	DISTINCT
اسم الجدول المرغوب استخراج البيانات منه	FROM table
عبارة الشرط	WHERE
لربط أكثر من شرط معا	AND/OR
تستعمل لتغيير الأولويات	الأقواس ()
تستعمل لترتيب البيانات المعروضة وتظهر في آخر جملة الاستعلام	ORDER BY
ترتيب تصاعدي وهو الترتيب المفترض	ASC
ترتيب تنازلي	DESC

يمكن وضع العبارات Clauses في سطور منفصلة لغرض الوضوح.

أوراكل مثلا (١٧)

استعمال المتغيرات لتنفيذ جمل الاستعلام

قلنا في البداية أن لغة سكول ليست إجرائية Non-Procedural؛ لذلك لا تحتوي على تركيبات شروط مثل IF..THEN.. وتعريف متغيرات DECLARE Variables وحلقات تكرار FOR...LOOP ستتعرف على أساليب تمكينا من استعمال بعض تلك الخصائص. تقدم أوراكل وغيرها من الشركات بيئات لكتابة جمل سكول فأوراكل تقدم مثلا بيئه سكول + iSQLPlus وبيئة آي سكول + SQL Plus وغيرها. سوف نتعلم من الآن فصاعدا محاكاً لهذه الخصائص.

يمكنا استعمال المتغيرات variables لتنفيذ جمل الاستعلام؛ حيث لا تقييد بقيمة معينة للشرط أو الشروط التي نضعها لاستخراج وعرض البيانات. وهناك نوعان من المتغيرات هما:

١ - المتغيرات المرجعية Bind Variables وليس هنا مجالها.

٢ - المتغيرات التعويضية Substitution Variables .

وهي استعمال المتغيرات التعويضية لتخزين جزء من عناصر جملة SQL والتي يمكن مراجعتها؛ وهذه العناصر هي:

□ عبارة WHERE

□ عبارة ORDER BY

□ أسماء الأعمدة

□ أسماء الجداول

□ عموم جملة SELECT

وإليك شرح هذا النوع من المتغيرات.

علامة التعويض المفردة & Single Ampersand

يتم استعمال متغير ما مسبوقا بالرمز & بحيث تطلب منا آي سكول +(أو أي بيئه أخرى مثل سكول+ iSQLPlus) إدخال القيمة المرغوبة؛ كما في المثال التالي:

```
SELECT employee_id, last_name, salary, department_id  
FROM employees  
WHERE employee_id = &employee_num ;
```

 **Input Required**
Cancel**Continue**

Enter value for employee_num:

old 3: WHERE employee_id = &employee_num

new 3: WHERE employee_id = 101

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
101	Kochhar	17000	90

لاحظ كيف يُطلب إدخال القيمة المرغوب استعمالها كشرط وبعد إدخالها؛ تعرض آي سكّول + القيمة القديمة ثم الجديدة.
إن جملة الاستعلام السابقة تكافأ جملة الاستعلام التالية:

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = 101 ;
```

لاحظ أن القيمة القديمة تُفقد في كل مرة يتم التنفيذ فيها.

بالنسبة للبيانات من نوع نص Character أو من نوع تاريخ Date يجب وضع القيم داخل حاصلين علويتين ". وبدلاً من ذلك نضع المتغير مع علامة التعویض داخل الحاصلين كما في المثال التالي:

```
SELECT last_name, department_id, salary*12
FROM employees
WHERE job_id = '&job_title' ;
```

 **Input Required**
Cancel**Continue**

Enter value for job_title: IT_PROG

LAST_NAME	DEPARTMENT_ID	SALARY*12
Hunold	60	108000
Ernst	60	72000
Lorentz	60	50400

ويمكن استعمال المتغيرات التعويضية لتمثيل ليس قيمة مفردة فقط؛ بل لتمثل تعبيراً ما أو اسم عمود وحتى اسم جدول؛ وفي المثال التالي تمثل المتغيرات التعويضية عموداً وشرطًا وتعبيرًا للترتيب عليه:

```

SELECT employee_id, last_name, job_id, &column_name
FROM employees
WHERE &condition
ORDER BY &order_column;

```

The image shows three separate input dialog boxes, each with a 'Cancel' and 'Continue' button. The first box asks 'Enter value for column_name:' with the input 'salary'. The second box asks 'Enter value for condition:' with the input 'salary > 15000'. The third box asks 'Enter value for order_column:' with the input 'last_name'.

علامة التعويض المزدوجة &&

إذا استعملنا علامة التعويض المزدوجة && مع المتغير فإن سُكُّون سوف تطلب قيمة المتغير مرة واحدة فقط وتظل محفوظة بما حتى الخروج من سُكُّون كما في المثال التالي:

```

SELECT employee_id, last_name, job_id, &&column_name
FROM employees
ORDER BY &column_name ;

```

The image shows an input dialog box asking 'Enter value for column_name:' with the input 'department_id'. Below it is a table from the employees database:

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
200	Whalen	AD_ASST	10
201	Hartstein	MK_MAN	20

في المثال، المطلوب إدخال اسم العمود المتضمن في كل من عبارة SELECT وعبارة ORDER BY وقد قدمناه مرة واحدة وتم التنفيذ بناءً على ذلك

استعمال أمر التعريف DEFINE

يعKen استعمال الأمر DEFINE لتخصيص قيمة معينة لمتغير يتم استعماله في جمل الاستعلام. في المثال التالي سوف نعرف المتغير TTL_INCOME بـ تخصيص التعبير SALARY*12+NVL(COMMISSION_PCT,0) ونقوم باستعماله في جملة SELECT كالتالي:

```
DEF TTL_INCOME = SALARY*12+NVL(COMMISSION_PCT, 0)
```

للتأكد من المتغير الذي قمنا بتعريفه نكتب الأمر DEF بدون إضافات فنحصل على ما يلي:

```
DEFINE TTL_INCOME="" SALARY*12+NVL(COMMISSION_PCT, 0)"  
(CHAR)
```

و الآن سنكتب الجملة التالية:

```
SELECT LAST_NAME, &TTL_INCOME FROM EMPLOYEES;
```

```
old 1: SELECT LAST_NAME, &TTL_INCOME FROM EMPLOYEES
```

```
new 1: SELECT LAST_NAME, SALARY*12+NVL(COMMISSION_PCT, 0) FROM  
EMPLOYEES
```

LAST_NAME	SALARY*12+NVL(COMMISSION_PCT,0)
OConnell	31200
Grant	31200
Whalen	52800
Hartstein	156000
Fay	72000
Mavris	78000
Baer	120000
Higgins	144000
Gietz	99600

ودائماً يكون نوع المتغير نصا CHAR بغض النظر عن نوع محتوياته.

استعمال أمر الإدخال ACCEPT

يمكن استعمال الأمر ACCEPT لإفساح المجال للمستعمل لإدخال قيمة معينة لمتغير ما؛ والذي يستعمل في جمل سُكُولْ.
وهذا الأمر يستعمل غالباً في ملفات الأوامر. وهناك فوائد لاستخدام الأمر ACCEPT منها:

١. تحديد نوع البيانات المدخلة
٢. يمكن استخدام رسالة حث PROMPT للتوضيح

٣. يمكن إخفاء القيمة المدخلة
والصيغة العامة للأمر ACCEPT هي:

ACC[EPT] variable [NUMBER/CHAR] [PROMPT/NOPROPT 'text' [HIDE]

حيث:

- ACC[EPT] يمكن استعمال أول ثلاثة حروف كما في الأمر DEF[INE]. لطلب إدخال قيمة
- Variable اسم متغير.
- NUMBER/CHAR تحديد نوع المتغير. وفي حالة الخطأ هناك رسالة خطأ.
- PROMPT 'text' إظهار نص معين قد يحتوي على توجيه أو رسالة ما.
- NOPROMPT ينتقل المؤشر إلى السطر التالي بانتظار المدخلات.
- HIDE لإخفاء المدخلات مثل كلمة السر.

أمثلة

ACCEPT SALARY NUMBER PROMPT 'Enter Salary:'

Enter Salary:2500

ACCEPT PASSWORD CHAR PROMPT 'ENTER PASSWORD :' HIDE

ENTER PASSWORD :***

ACCEPT SALARY NUMBER NOPROMPT

1200

من الممكن ذكر عبارة NOPROMPT عند الرغبة في عدم إظهار أية رسالة للمستعمل.

ولإظهار آخر قيم التي أدخلناها نستعمل الأمر DEFINE متبوعاً باسم المتغير فنحصل على القيمة التي أدخلناها ونوع المتغير كما يلي:

DEFINE SALARY = 1200 (NUMBER)

DEFINE PASSWORD = "ABC" (CHAR)

DEFINE COMM = 3000 (NUMBER)

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (١٨)

الدوال Functions

للذكر: الدالة أو ال Function هي عبارة عن قطعة برمجية Piece of code تقوم بأداء مهمة-وظيفة معينة؛ وهذه المهمة يتكرر استعمالها في البرامج التي نكتبها. فبدلاً من تكرار كتابة الكود البرمجي لهذه المهمة؛ نكتب الكود البرمجي مرة واحدة ونقوم باستدعائه Call باسمه في كل مرة تحتاجه فيها.

.one and only one value Returns بقيمة واحدة فقط وفقط قيمة واحدة . والوظيفة أو الدالة يجب أن تعود فإن عادت بدون قيمة أو بأكثر من قيمة فإن لغة البرمجة تعطي رسالة خطأ.

هناك عدد كبير من الدوال في لغة سكول ولها استعمالات عدّة وهذه بعض الاستعمالات:

١. إجراء العمليات الحسابية Arithmetic.

٢. تغيير شكل البيانات.

٣. معالجة شكل النتائج لمجموعات من الصنوف.

٤. تغيير شكل البيانات التي من نوع تاريخ لأغراض العرض .

٥. التحويل من نوع بيانات إلى نوع بيانات آخر.

أما أنواع الدوال فهي:

١. دوال تتعامل مع النصوص Character Functions

٢. دوال تتعامل مع الأعداد Number Functions

٣. دوال تتعامل مع التاريخ Date Functions

٤. دوال التحويل من نوع بيانات إلى آخر Conversion Functions

٥. دوال الجموعات Group Functions

٦. دوال تقبل أي نوع كمدخلات لها .

• شرح رموز مدخلات الدوال

الجدول التالي يشرح الرموز المستعملة في الدوال كمدخلات بصورة عامة

المدخلات	الشرح
col.	أي اسم عمود في أي جدول
Value	أي قيمة من أي نوع(نص أو عددي أو تاريخ)
n	تمثل أي عدد صحيح

يمثل سلسلة رموز	string ”نص”
يمثل عددا محددا من الرموز	chars رموز
يمثل متغير أو عمود من النوع تاريخ Date Col./Value	تاريخ date

وسوف نستعمل الحروف اللاتينية كرموز للمدخلات للسهولة كما سوف نستعمل التعبير expression ليدل على أي من المدخلات السابقة.

وفيما يلي شرح لهذه الدوال

دوال النصوص Character Functions

١- الدالة LOWER

- * عملها : تحويل النصوص من حالة الحرف الكبير Upper Case إلى حالة الحرف الصغير Lower Case .
- الشكل العام : LOWER(expression) حيث expression اسم عمود أو اسم متغير أو ثابت ... الخ.
- مثال :

```
SELECT department_name, LOWER(department_name), LOWER('SQL Course') FROM departments;
```

DEPARTMENT_NAME	LOWER(DEPARTMENT_NAME)	LOWER('SQLCOURSE')
Administration	administration	sql course
Marketing	marketing	sql course
Purchasing	purchasing	sql course
Human Resources	human resources	sql course
Shipping	shipping	sql course
IT	it	sql course
Public Relations	public relations	sql course
Sales	sales	sql course
Executive	executive	sql course
Finance	finance	sql course

Accounting	accounting	sql course
------------	------------	------------

٢ - الدالة UPPER

* عملها : تحويل النصوص من حالة الحرف الصغير Lower Case إلى حالة الحرف الكبير Upper Case.

* الشكل العام : UPPER(expression) حيث expression اسم عمود أو اسم متغير أو ثابت ... الخ.

مثال :

```
SELECT last_name,UPPER(last_name),UPPER('sql')
```

```
FROM employees;
```

LAST_NAME	UPPER(LAST_NAME)	UPPER('SQ
Abel	ABEL	SQL
Ande	ANDE	SQL
Atkinson	ATKINSON	SQL
Austin	AUSTIN	SQL
Baer	BAER	SQL
Baida	BAIDA	SQL
Banda	BANDA	SQL
Bates	BATES	SQL
Bell	BELL	SQL
Bernstein	BERNSTEIN	SQL
Bissot	BISSOT	SQL
Bloom	BLOOM	SQL
Bull	BULL	SQL

* عملها : تحويل أول حرف من النص من حالة الحرف الصغير Lower Case إلى حالة الحرف الكبير Upper Case وإلى حالة الحرف الصغير وتحويل باقي النص إلى حالة الحرف الصغير.

* الشكل العام : INITCAP(expression) حيث expression اسم عمود أو اسم متغير أو ثابت ... الخ.

مثال:(سنبدأ باستعمال جدول وهمي في قاعدة البيانات أوراكل اسمه DUAL فنستطيع أن نختار منه أي شيء)

```
SELECT INITCAP('sql course')
```

```
FROM DUAL;
```

INITCAP('SQLCOURSE')

Sql Course

٤ - الدالة INSTR

* عملها : البحث عن نص داخل نص آخر. والنتيجة عبارة عن عدد يمثل موقع النص المبحوث عنه في النص المبحوث فيه للمرة n . وبمعنى آخر؛ قد يتكرر النص المبحوث عنه مرة أو أكثر أو لا يوجد داخل النص المبحوث فيه، ونحن هنا نحدد للدالة رقم التكرار(n) كي تعطينا موقع النص لهذا التكرار. إذا لم يكن النص المبحوث عنه في النص المبحوث فيه في التكرار المطلوب فالنتيجة صفر.

* الشكل العام : INSTR(expression,expression1,pos,n) حيث

١ - expression اسم عمود أو اسم متغير أو ثابت ... الخ-النص المبحوث فيه.

٢ - expression1 اسم عمود أو اسم متغير أو ثابت ... الخ-النص المبحوث عنه.

٣ - pos الموقع المراد البحث ابتداءً منه في النص المبحوث فيه.

٤ - n رقم تكرار النص المبحوث عنه داخل النص المبحوث فيه.

مثال :

```
SELECT Department_name, INSTR(Department_name, 'C', 1, 1)
FROM Department_name;
```

DEPARTMENT_NAME

INSTR(DEPARTMENT_NAME,'C',1,1)

Administration	0
Marketing	0
Purchasing	0
Human Resources	0
Shipping	0
IT	0
Public Relations	0
Sales	0
Executive	0
Finance	0
Accounting	0
Treasury	0
Corporate Tax	1
Control And Credit	1
Shareholder Services	0
Benefits	0
Manufacturing	0
Construction	1
Contracting	1
Operations	0
IT Support	0

NOC		3
IT Helpdesk		0

هنا طلبنا البحث عن حرف C في أسماء الأقسام بتحديد بداية البحث من أول النص ولمرة واحدة وعرض موقعه

٥- الدالة LPAD

* عملها : إضافة عدد من الرموز إلى أول النص(من الشمال) بحيث يصبح مجموع الرموز بما فيها الرموز المكونة للنص مساويا للعدد الذي يتم تحديده ضمن مدخلات الدالة.

* الشكل العام : LPAD(expression,n,char) حيث :

١- اسم عمود أو اسم متغير أو ثابت ...إلخ

٢- العدد الكلي للرموز بما فيها رموز التعبير.

٣- char الرمز المراد إضافته إلى أول النص من الشمال.

مثال :

```
SELECT LPAD(DEPARTMENT_NAME,20,'*')
```

```
FROM DEPARTMENTS
```

```
LPAD(DEPARTMENT_NAME,20,'*')
```

```
*****Administration
```

```
*****Marketing
```

```
*****Purchasing
```

```
*****Human Resources
```

```
*****Shipping
```

```
*****IT
```

```
****Public Relations
```

*****Sales

*****Executive

*****Finance

*****Accounting

*****Treasury

٥ - الدالة RPAD

* عملها : إضافة عدد من الرموز إلى آخر النص(من اليمين) بحيث يصبح مجموع الرموز بما فيها الرموز المكونة للنص مساويا للعدد الذي يتم تحديده ضمن مدخلات الدالة.

* الشكل العام : RPAD(expression,n,char) حيث :

-١ expression اسم عمود أو اسم متغير أو ثابت ... الخ.

-٢ n العدد الكلي للرموز بما فيها رموز التعبير.

-٣ char الرمز المراد إضافته إلى آخر النص أي من اليمين.

مثال :

```
SELECT RPAD(department_name,20,'*') FROM departments;
```

RPAD(DEPARTMENT_NAME,20,'*')

Administration*****

Marketing*****

Purchasing*****

Human Resources*****

Shipping*****

IT*****

Public Relations****

Sales*****

Executive*****

Finance*****

Accounting*****

Treasury*****

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (١٩)

الدوال Functions

استكمال دوال النصوص

٦ - الدالة SUBSTR

* عملها : استخلاص نص محدد بعدد من الرموز من نص آخر اعتباراً من موقع معين.

* الشكل العام : $\text{SUBSTR}(\text{expression}, \text{pos}, \text{n})$ حيث :

١ - اسم عمود أو اسم متغير أو ثابت ... الخ

٢ - عدد الرموز المستخلصية

٣ - pos الموقع المراد استخلاص الرموز ابتداءً منه. إذا كان pos سالباً فإن الاستخلاص يبدأ من اليمين

مثال :

```
SELECT SUBSTR('COUNTER',1,5) FROM DUAL;
```

SUBSTR('COUNTER'
COUNT

٧ - الدالة LTRIM

* عملها : إزالة رمز أو رموز ابتداءً من أول النص (من الشمال). إذا لم يتم تحديد الرمز أو الرموز المرغوب إزالتها فإن هذه الدالة تزيل جميع الفراغات من شمال النص.

* الشكل العام : $\text{LTRIM}(\text{expression}, \text{char(s)})$ حيث :

١ - اسم عمود أو اسم متغير أو ثابت ... الخ.

٢ - الرمز أو الرموز المرغوب إزالتها.

مثال :

```
SELECT department_name,LTRIM(department_name,'A')
```

FROM departments;

DEPARTMENT_NAME	LTRIM(DEPARTMENT_NAME,'A')
Administration	dministration
Marketing	Marketing
Purchasing	Purchasing
Human Resources	Human Resources
Shipping	Shipping
IT	IT
Public Relations	Public Relations
Sales	Sales
Executive	Executive
Finance	Finance
Accounting	ccounting

٧- الدالة RTRIM

* عملها : إزالة رمز أو رموز ابتداء من آخر النص (من اليمين). إذا لم يتم تحديد الرمز أو الرموز المرغوب إزالتها فإن هذه الدالة تزيل جميع الفراغات من يمين النص.

* الشكل العام : $RTRIM(expression, char(s))$ حيث :

١ - $expression$ اسم عمود أو اسم متغير أو ثابت ...الخ-النص المبحث فيه.

٢ - $char(s)$ الرمز أو الرموز المرغوب إزالتها.

مثال :

SELECT department_name,RTRIM(department_name,'s') FROM departments;

DEPARTMENT_NAME	RTRIM(DEPARTMENT_NAME,'S')
Administration	Administration
Marketing	Marketing
Purchasing	Purchasing
Human Resources	Human Resource
Shipping	Shipping
IT	IT
Public Relations	Public Relation
Sales	Sale

٩ - الدالة LENGTH

* عملها : استخلاص طول النص أي عدد الرموز المكونة له.

* الشكل العام : LENGTH(expression) حيث :

١ - expression اسم عمود أو اسم متغير أو ثابت ... الخ.

مثال :

SELECT last_name,LENGTH(last_name) FROM employees;

LAST_NAME	LENGTH(LAST_NAME)
Abel	4
Ande	4
Atkinson	8
Austin	6

Baer		4
Baida		5
Banda		5
Bates		5
Bell		

١٠ - الدالة TRANSLATE

* عملها : تبديل رموز النص الأصلي برموز من مجموعة رموز بما يقابلها من رموز تلك المجموعة. وبمعنى آخر ترجمة النص اعتماداً على مجموعة رموز - "لغتين" - تماماً كما نريد أن نترجم نصاً من الإنجليزية إلى العربية فإننا نترجم الحرف الإنجليزي إلى الحرف العربي المقابل له؛ مع الفارق في عمل هذه دالة TRANSLATE وعملية الترجمة من الإنجليزية إلى العربية.

* الشكل العام : TRANSLATE(expression,fromset,toset) حيث :

١ - expression اسم عمود أو اسم متغير أو ثابت ...الخ-النص المراد ترجمته.

٢ - fromset مجموعة الرموز التي سوف تتم الترجمة منها-أي تبديلها.

٣ - toset مجموعة الرموز التي سيتم إحلالها محل الرموز في المجموعة .fromset

تكون مجموعتنا الحروف عادة متساوية في عدد الرموز، ولكن لو كان عدد الرموز في المجموعة fromset أكبر من عددها في فلا يتم تبديل "ترجمة" الرموز التي في fromset والتي ليس لها مقابل في toset. ولكن لو كانت هذه الرموز ضمن النص المترجم فيتم في هذه الحالة حذفها من النص المترجم. ولكن لا يجوز استعمال الفراغات spaces لإلغاء الرموز من النص المترجم.

مثال : سيتم في هذا المثال استبدل-ترجمة-جميع الأرقام إلى الرقم 9 وجميع الحروف إلى الحرف X. لاحظ تساوي عدد رموز المجموعتين.

```
SELECT    TRANSLATE ('2KRW229',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'9999999999XXXXXXXXXXXXXXXXXXXX'),
FROM DUAL;
```

```
TRANSLATE('2KRW229','
```

```
9XXX999
```

في المثال التالي سيتم حذف جميع الحروف. لاحظ عدم تساوي عدد الحروف في المجموعتين.

```
SELECT TRANSLATE('2KRW229',
```

```
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ',
```

```
'0123456789')
```

```
FROM DUAL;
```

```
TRANSLATE('2
```

```
2229
```

١١ - الدالة REPLACE

- عملها : تبديل نص بآخر مبحث عنه في نص. وهذا تعميم للدالة السابقة TRANSLATE والتي تعامل مع الرموز رمز مقابل رمز أي علاقة ١-١ أما الدالة REPLACE فتسمح بأكثر من نوع من علاقة مثل متعدد-١ أو ١-متعدد.

* الشكل العام : REPLACE(expression,expr1,[expr2) حيث :

١ - expression اسم عمود أو اسم متغير أو ثابت ...إلخ-النص المراد تبديل نص منه.

٢ - expr1 النص المراد تبديله.

٣ - expr2 النص المراد إحلاله محل النص المراد تبديله وهو اختياري بحيث لو لم يذكر فيتم حذف رموز النص المراد تبديله.

مثال : سيتم في هذا المثال تبديل الحرف J -النص المراد تبديله- بالحرفين BL -النص الذي سيتم إحلاله محل النص المراد تبديله.

```
SELECT REPLACE('JACK and JUE','J','BL') FROM DUAL;
```

```
REPLACE('JACKANDJUE','J','BL')
```

```
BLACK and BLUE
```

* عملها : استخلاص الرمز المقابل للعدد n من مجموعة رموز قاعدة البيانات. قد يختلف هذا الرمز حسب نوع مجموعة الرموز واللغة المستعملة مثل الإنجليزية أو الفرنسية.

* الشكل العام : $\text{CHR}(n)$ حيث :

١ - n العدد الذي يمثل رمزاً من مجموعة الرموز.

`SELECT CHR(75) FROM DUAL;`

CHR
K

* عملها : إعطاء العدد المقابل للرمز $char$ من مجموعة رموز قاعدة البيانات. قد يختلف هذا الرمز حسب نوع مجموعة الرموز واللغة المستعملة مثل الإنجليزية أو الفرنسية.

* الشكل العام : $\text{ASCII}(char)$ حيث :

١ - $char$ الرمز الذي نريد العدد مقابلة.

مثال:

`SELECT ASCII('L') FROM DUAL;`

ASCII('L')
76

* عملها : ربط نصين لإنتاج نص جديد.

* الشكل العام : $\text{CONCAT(expr1,expr2)}$ حيث :

١ - $expr1$ النص الأول.

٢ - $expr2$ النص الثاني.

مثال:

```
SELECT CONCAT(CONCAT(last_name,' is a '),job_id) "Emp and Job" FROM employees WHERE employee_id=102;
```

Emp and Job

De Haan is a AD_VP

١٥ - الدالة TRIM

* عملها : حذف رمز(رموز) من أول أو آخر نص ما.

* الشكل العام : TRIM(expr1 FROM expr2) حيث :

١ - expr1 النص الأول(المراد حذفه).

٢ - expr2 النص الثاني(المحذوف منه).

مثال:

```
SELECT TRIM('H' FROM 'HelloWorld') FROM DUAL;
```

TRIM('H'FROM'HELLOWORLD')

elloWorld

مثال عام

```
SELECT employee_id, CONCAT(first_name, last_name) NAME, LENGTH(last_name), INSTR(last_name, 'a') "Contains 'a'?" FROM employees WHERE SUBSTR(last_name, -1, 1) = 'n';
```

EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
200	JenniferWhalen	6	3
201	MichaelHartstein	9	2

102	LexDe Haan	7	5
105	DavidAustin	6	0
110	JohnChen	4	0
112	Jose ManuelUrman	5	4
123	ShantaVollman	7	6
130	MozheAtkinson	8	0
132	TJOlson	5	0

هذه هي أهم دوال النصوص strings التي تهم المبتدأين والمتقدمين على حد سواء. وهناك دوال أخرى تهم المتقدمين يمكنهم الرجوع إليها من خلال أدلة الاستعمال.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٢٠)

الدوال Functions

دوال الأعداد Number Functions

١ - الدالة ABS

* عملها : إعطاء القيمة المطلقة للعدد

* الشكل العام: $ABS(n)$ حيث :

١ - n العدد المراد إيجاد قيمته المطلقة.

مثال:

```
SELECT ABS(-90) "Absolute" FROM DUAL;
```

Absolute

90

٢ - الدالة CEIL

* عملها : إعطاء أصغر عدد صحيح أكبر أو يساوي n

* الشكل العام: $CEIL(n)$ حيث :

١ - n العدد المراد إيجاد أصغر عدد صحيح أكبر منه أو يساويه.

مثال:

```
SELECT CEIL(13.4) "Ceiling" FROM DUAL
```

Ceiling

14

٣ - الدالة FLOOR

* عملها : إعطاء أكبر عدد صحيح أصغر أو يساوي n .

* الشكل العام: $\text{FLOOR}(n)$ حيث:

١ - n العدد المراد إيجاد أكبر عدد صحيح أصغر منه أو يساويه.

مثال:

```
SELECT FLOOR(13.4) "Floor" FROM DUAL;
```

Floor

13

٤ - الدالة LN

* عملها : إعطاء اللوغاریتم الطبيعي للعدد n حيث n أكبر من صفر.

* الشكل العام: $\text{LN}(n)$ حيث:

١ - n العدد المراد إيجاد اللوغاریتم الطبيعي له.

مثال:

```
SELECT LN(100) "Natural Log of 100" FROM DUAL;
```

Natural Log of 100

4.60517019

٥ - الدالة LOG

* عملها : إعطاء اللوغاریتم للعدد n على الأساس m .

* الشكل العام: $\text{LOG}(m,n)$ حيث:

١ - n العدد المراد إيجاد اللوغاریتم الذي يخصه.

٢ - الأساس m حيث m ليس ضمن المجموعة $\{1, 0\}$.

مثال:

SELECT LOG(10,100) "Log 100 base 10 of 100" FROM DUAL;

Log 100 base 10 of 100

2

٦ - الدالة EXP

* عملها : إعطاء نتيجة رفع العدد e إلى القوة n.

* الشكل العام: EXP(n) حيث:

- ١ n العدد الذي سيتم رفع e إليه.

- ٢ e وتساوي ٢.٧١٨٢٨١٨٢٨٤٥٩

مثال:

SELECT EXP(4) "e to the power 4" FROM DUAL;

e to the power 4

54.59815

٧ - الدالة POWER

* عملها : إعطاء نتيجة رفع العدد m إلى القوة n.

* الشكل العام: POWER(m,n) حيث:

- ١ m العدد المراد رفعه إلى القوة n.

- ٢ n أي عدد. في حالة أن m عدد سالب فيجب أن تكون n عدداً صحيحاً.

مثال:

SELECT POWER(3,2) "3 to power 2" FROM DUAL;

3 to power 2

٨- الدالة SQRT

* عملها : إعطاء الجذر التربيعي للعدد n .

* الشكل العام: $SQRT(n)$ حيث:

١- n يجب أن تكون عدداً أكبر من أو يساوي الصفر

* مثال:

```
SELECT SQRT(100)"Square Root" FROM DUAL;
```

Square Root

10

٩- الدالة ROUND

* عملها : تقريب العدد n إلى m خانة عشرية يمين الفاصلة.

* الشكل العام: $ROUND(n,[m])$ حيث:

-١ n العدد المراد تقريبه.

-٢ m عدد صحيح يمثل عدد الخانات العشرية.

مثال: سيقوم هذا المثال بتقريب العدد إلى خانة عشرية واحدة.

```
SELECT ROUND(15.193,1)"Round 15.193"
```

```
FROM DUAL;
```

Round 15.193

15.2

إذا كانت m عدداً سالباً فإن التقريب يتم من اليمين حيث يتم إهمال الأرقام التي على يمين الفاصلة العشرية وتقريب الأرقام الصحيحة ابتداءً من اليمين كما في المثال التالي:

```
SELECT ROUND(15.193,-1) "Round" FROM DUAL;
```

Round

20

تم إهمال الأرقام على يمين الفاصلة وتقريب العدد ١٥ إلى ٢٠ ، ولو كان العدد هو ١٤ بدلًا من ١٥ وكانت نتيجة التقريب هي ١٠ أي يتم جبر العدد إلى ١٠ إذا كان أكبر أو يساوي ٥ وإلى الصفر إذا كان أقل من ٥ وهذا ما يحدث تماماً بالنسبة للأرقام التي على يمين الفاصلة العشرية في حالة كون m عدداً موجباً.

١٠ - الدالة TRUNC

* عملها : حذف الأرقام التي على يمين الفاصلة العشرية حسب القيمة m .

* الشكل العام: $\text{TRUNC}(n,[m])$ حيث:

- ١ n العدد المراد تخلصه من الأرقام العشرية.
- ٢ m عدد الخانات المرغوب التخلص منها.

وهنا لا يتم أي تقريب كما في الدالة ROUND بل يتم حذف عدد من الأرقام من يمين الفاصلة حسب m . أما إن كانت m عدداً سالباً فإنه يتم التخلص من جميع الأرقام التي على يمين الفاصلة العشرية وتحويل عدد من الأرقام التي على شمال الفاصلة العشرية إلى صفر ابتداءً من أول رقم من اليمين وحسب m .

مثال:

SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;

Truncate

15.7

هنا النتيجة تحتوي على خانة عشرية واحدة لأن $m=1$.

في المثال التالي تم تحويل الرقم ٥ إلى صفر لأن m عدد سالب:

SELECT TRUNC(15.791,-1) "Truncate" FROM DUAL

Truncate

10

١١ - الدالة MOD

* عملها : إعطاء باقي قسمة m على n.

* الشكل العام: $\text{MOD}(m,n)$ حيث:

m العدد المقسم.

n العدد المقسم عليه.

مثال:

```
SELECT MOD(11,3) "Modulus" FROM DUAL;
```

Modulus

2

ملاحظة:

هذه الدالة تختلف في عملها و نتيجتها عن مثيلتها في علم الرياضيات في حالة كون m عددا سالبا؛ ففي لغة SQL لا يتغير تعريفها بينما في الرياضيات تُحسب نتيجة هذه الدالة كما يلي:

$$m - n * \text{FLOOR}(m/n)$$

.FLOOR بُرجى مراجعة الدالة

المثال التالي محاكاة لنتيجة الدالة في كل من الرياضيات و SQL.

```
SELECT MOD(-11,3)"SQL MOD", -11-3*FLOOR(-11/3)"MATH. MOD"
```

```
FROM DUAL
```

MATH. MOD

-2

1

١٢ - الدالة SIGN

* عملها : إذا كانت n عددا سالبا و صفر إذا كانت n صفراء و 1 عندما تكون n عددا موجبا.

* الشكل العام: $\text{SIGN}(n)$

مثال:

```
SELECT SIGN(-15)"Sign" FROM DUAL;
```

هذه هي أهم دوال الأعداد في أوراكل وهناك دوال أخرى غير متوفرة في نسخة أوراكل أقل من النسخة 7 مثل دوال حسابات المثلثات Trigonometric Functions مثل SIN و COS و TAN ... الخ. لذلك يرجى الرجوع إلى دليل الاستعمال للغة SQL من موقع أوراكل.

أوراكل مثلاً (٢١)

دوال Functions

دوال التاريخ Date Functions

* أولاً: استعمال العمليات الحسابية مع التواريخ

عندما نستعمل العمليات الحسابية مع التواريخ؛ فإن أوراكل تنظر إليها وتعالجها كأعداد حقيقية ولذلك فإن النتيجة قد تكون تاريخاً أو عدداً حقيقياً. والعمليات الحسابية المستعملة في التاريج هي الجمع والطرح فقط - أما إن كانت النتيجة عدداً حقيقياً فيمكن استعمال العمليات الحسابية بصورة عادلة عليها - حسب الترتيب التالي:

- أ- إيجاد الفرق بين تاريخين والنتيجة عدد حقيقي حيث يمثل العدد العشري أجزاء اليوم أي الوقت.
- ب- جمع عدد إلى تاريخ والنتيجة تاريخ جديد يساوي التاريخ القديم زائداً الأيام حسب العدد المضاف.
- ج- طرح عدد من تاريخ والنتيجة تاريخ جديد يساوي التاريخ القديم ناقصاً الأيام حسب العدد المطروح.

ملاحظة: جميع دوال التاريخ نتيجتها تاريخ ما عدا طرح تاريخ من تاريخ فيعطي عدداً كنتيجة

ملاحظة: للتعامل مع الساعات باضافتها إلى تاريخ معين نقسم عدد الساعات على ٢٤ ومن ثم إجراء العمليات الحسابية على التاريج

أمثلة:

```
SELECT SYSDATE "System Date", SYSDATE+5 "After 5 Days"
```

```
FROM DUAL;
```

System Date	After 5 Days
01-MAY-07	06-MAY-07

```
SELECT SYSDATE "System Date", SYSDATE-30 "Before One Month"
```

```
FROM DUAL;
```

System Date	Before One Month
-------------	------------------

01-MAY-07

01-APR-07

```

SELECT SYSDATE "System Date",SYSDATE-HIRE_DATE "Difference"
FROM EMPLOYEES
WHERE EMPLOYEE_ID=101;

```

System Date	Difference
01-MAY-07	6431.5585

لاحظ أن الأعداد التي نضيفها أو الناتجة هي بالأيام وأجزائها-الوقت-والتي يمكنك تطبيق دوال الأعداد عليها لتعطي نتائج ذات شكل أفضل. وستعرض بعض أشكال نتائج العمليات والدوال على التوالي.

*ثانياً: الدوال

١ - الدالة: ADD_MONTHS:

* الشكل العام: ADD_MONTHS(d,m)

عملها : إعطاء تاريخ جديد بإضافة m من الشهر إلى التاريخ d .
إذا كانت m عدد سالباً فتكون النتيجة هي التاريخ d ناقصاً عدد الشهور m .

مثال:

```

SELECT SYSDATE "System Date", ADD_MONTHS(SYSDATE,1) "Next Month"
FROM DUAL;

```

System Date	Next Month
01-MAY-07	01-JUN-07

٢ - الدالة: MONTHS_BETWEEN:

* الشكل العام: MONTHS_BETWEEN(d1,d2)

عملها : إعطاء عدد الشهور بين التاریخین d1 و d2

مثال ١:

```
SELECT HIRE_DATE,MONTHS_BETWEEN(SYSDATE,HIRe_DATE)
FROM EMPLOYEES
WHERE EMPLOYEE_ID=101;
```

HIRE_DATE	MONTHS_BETWEEN(SYSDATE,HIRe_DATE)
21-SEP-89	211.372962

ملاحظات:

١ - إذا كان التاريخ d_2 أكبر من التاريخ d_1 فإن النتيجة عدد سالب

مثال ٢:

```
SELECT HIRe_DATE,SYSDATE,
MONTHS_BETWEEN(HIRe_DATE,SYSDATE)"-Ve"
FROM EMPLOYEES
WHERE EMPLOYEE_ID=101;
```

HIRE_DATE	SYSDATE	-Ve
21-SEP-89	01-MAY-07	-211.37309

٢ - إذا كان كل من التاریخین d_1 و d_2 آخر يومین من الشهور فإن النتيجة عدد صحيح سالب أو موجب.

مثال ٣:

```
SELECT MONTHS_BETWEEN(TO_DATE('31-OCT-96'),
TO_DATE('31-JAN-96')) "Integer"
FROM DUAL;
```

Integer

٣- إذا لم تتحقق الملاحظتان السابقتان فإن أوراكل تحسب الفرق وتضيف الكسور كما تأخذ الوقت بعين الاعتبار كما في المثال رقم ١

٣- الدالة: NEXT_DAY:

* الشكل العام: NEXT_DAY(date,char/n)

* عملها : إعطاء تاريخ اليوم التالي المحدد بـ n أو char حيث تمثل n رقم اليوم أما char فتمثل اسم اليوم.

مثال ١ : هذا المثال يعطينا تاريخ يوم الاثنين التالي للتاريخ ١-كانون الثاني يناير-٩٧

```
SELECT NEXT_DAY('01-JAN-97','MONDAY')
FROM DUAL;
```

NEXT_DAY('01-JAN-9

06-JAN-97

مثال ٢ : نفس المثال السابق ولكن باستعمال رقم اليوم وليس اسمه.

```
SELECT NEXT_DAY('01-JAN-97',2) FROM DUAL;
```

NEXT_DAY('01-JAN-9

06-JAN-97

أرقام الأيام هي ١ يوم الأحد، ٢ يوم الاثنين، ٣ يوم الثلاثاء، ٤ يوم الأربعاء، ٥ يوم الخميس، ٦ يوم الجمعة و ٧ يوم السبت.

٤- الدالة: LAST_DAY:

* الشكل العام: LAST_DAY(date)

* عملها : إعطاء تاريخ آخر يوم في الشهر الموجود في التاريخ date

مثال ١ : هذا المثال يعطينا تاريخ آخر يوم في شهر كانون الثاني-يناير عام ٩٧ كما يعطينا عدد الأيام الباقية من الشهر.

```
SELECT LAST_DAY(SYSDATE) "Last",LAST_DAY(SYSDATE)-SYSDATE "Left"
FROM DUAL;
```

Last	Left
31-MAY-07	30

٥ - الدالة: ROUND

* الشكل العام: ROUND(date,[fmt])

* عملها : إعطاء أقرب تاريخ إلى التاريخ date وأكبر منه حسب وحدة التاريخ-يوم، شهر، سنة...الخ-المحددة في الشكل Conversion Function. انظر الجدول في نهاية هذا الفصل وتفاصيل أخرى في بند "دوال التحويل".

إذا تم حذف الشكل fmt فالتقريب يتم إلى أقرب يوم.

الشكل المفترض "DD" هو Default Format

مثال ١ : هذا المثال يعطينا بداية سنة ٩٣ للتاريخ ٢٧ نوفمبر-تشرين الثاني عام ١٩٩٢ م وذلك باستعمال الشكل ."YEAR"

```
SELECT ROUND(TO_DATE('27-NOV-92'),'YEAR') "Next Year"
FROM DUAL
```

Next Year
01-JAN-93

٦ - الدالة: TRUNC

* الشكل العام: TRUNC(date,[fmt])

* عملها : إعطاء أقرب تاريخ إلى التاريخ date وأصغر منه حسب وحدة التاريخ-يوم، شهر، سنة...الخ-المحددة في الشكل Conversion Function. انظر الجدول في نهاية هذا الفصل وتفاصيل أخرى في فصل "دوال التحويل".

إذا تم حذف الشكل fmt فالتقريب يتم إلى أقرب يوم.

الشكل المفترض "DD" هو Default Format

مثال ١ : هذا المثال يعطينا بداية سنة ٩٢ للتاريخ ٢٧ نوفمبر-تشرين الثاني عام ١٩٩٢ م وذلك باستعمال الشكل ."YEAR"

```
SELECT TRUNC(TO_DATE('27-NOV-92'),'YEAR')"First Of Year"
FROM DUAL;
```

First Of Year

01-JAN-92

أما هذا المثال فيعطيانا بداية الشهر لتاريخ النظام

```
SELECT TRUNC(SYSDATE,'MM') FROM DUAL;
```

TRUNC(SYSDATE,'MM'

01-MAY-07

الجدول التالي لقائمة الأشكال المستعملة في الوظيفتين ROUND و TRUNC .

الشكل Format	وحدة التقريب Unit Of ROUND or TRUNC
CC,SCC	Century القرن
YYYY, YYYY, YEAR, SYEA R, YYYY, YY, Y	السنة
Q	ربع السنة حيث يتم التقريب إلى أعلى اعتبارا من اليوم السادس عشر للشهر من الشهر الثاني للربع
MONTH,MON,MM,RM	الشهر حيث يتم التقريب إلى أعلى اعتبارا من اليوم السادس عشر من الشهر
WW	رقم الأسبوع لنفس السنة
W	تاريخ بداية الأسبوع لنفس الشهر
DDD,DD,J	اليوم
DAY,DY,D	أول يوم في الأسبوع
HH,HH12,HH24	الساعات

الدقائق	MI
الشوازي	SS
صباحاً/مساءً AM/PM	AM

أوراكل مثلاً (٢٢)

الدوال Functions

دوال التحويل

وهذه الدوال تقوم بتحويل نوع بيانات إلى نوع آخر؛ فهي تقبل أي نوع كمدخلات. وهذه الدوال هي:

١- الدالة: TO_CHAR و لها استعمالان أساسيان هما:

أ- التحويل من تاريخ إلى نص.

ب- التحويل من عدد إلى نص.

أ- التحويل من تاريخ إلى نص.

* الشكل العام: TO_CHAR(date,[fmt])

* عملها : تحويل تاريخ إلى نص حسب الشكل المحدد بالنص fmt وهو اختياري.

إذا تم حذف الشكل fmt فالشكل الناتج للتاريخ يكون هو الشكل الافتراضي DD-MON-) Default Format .VARCHAR2(YY) ويكون نوع النص

يرجى الرجوع إلى الجدول في نهاية الدرس لمزيد من المعلومات.

: مثال ١

```
SELECT TO_CHAR(SYSDATE,'DD-MM-YYYY') "Normal Format"  
FROM DUAL
```

Normal Format

01-05-2007

: مثال ٢

```
SELECT TO_CHAR(SYSDATE,'DAY HH:MI:SS AM') "Day and Time"  
FROM DUAL;
```

Day and Time

TUESDAY 01:53:58 PM

ويمكنك الربط بين مختلف الأشكال لجعل شكل المخرجات مفهوماً أكثر وذا معنى. كما يمكن استعمال الأشكال المختلفة في عمليات الاختبار Validation على مدخلات معينة مثلاً.

بـ- التحويل من عدد إلى نص.

* الشكل العام: TO_CHAR(n,[fmt])

* عملها : تحويل العدد n إلى نص حسب الشكل المحدد بالنص fmt وهو اختياري.

إذا تم حذف الشكل fmt فيتم التحويل إلى النوع VARCHAR2 بحيث يستوعب جميع أرقام العدد والفاصلة العشرية أيضاً إن وجدت.

مثال ١ :

```
SELECT TO_CHAR(SALALRY, '$9,999.99') "Formatted Salary"  
FROM EMPLOYEES  
WHERE SALALRY <=1100
```

Formatted Salary
\$2,200.00
\$2,100.00
\$2,200.00

ـ الدالة TO_DATE:

* الشكل العام: TO_DATE(char,[fmt])

* عملها : تحويل نص إلى تاريخ حسب الشكل المحدد بالنص fmt وهو اختياري.

إذا تم حذف الشكل fmt فالشكل الناتج للتاريخ يكون هو الشكل الافتراضي Default Format ويكون نوع النص VARCHAR2 ويجب أن يكون النص على الشكل DD-MON-YY وهو الافتراضي حيث DD تمثل اليوم و MON تمثل أول ٣ حروف من اسم الشهر YY تمثل السنة.

عند إدخال بيانات التاريخ إلى جدول ما فإن أوراكل أيضاً تأخذ الوقت بعين الاعتبار ويكون الوقت جزءاً من البيانات ولهذا يجب الانتباه لهذه النقطة عند مقارنة تاريخ باخر في عبارة WHERE .

يُرجى الرجوع إلى الجدول في نهاية الفصل لمزيد من المعلومات.

مثال ١ :

```
SELECT TO_DATE('10-01-1997 10:30:20 PM', 'DD-MM-YYYY  
HH:MI:SS AM') "Long Format"  
FROM DUAL;
```

Long Format
10-JAN-97

الدالة TO_NUMBER : ٢

* الشكل العام: TO_NUMBER(char,[fmt])

* عملها : تحويل نص إلى عدد حسب الشكل المحدد بالنص fmt وهو اختياري.

يُرجى الرجوع إلى الجدول في نهاية الفصل لمزيد من المعلومات.

مثال ١ :

```
SELECT TO_NUMBER('3,648.23','9,999,999.99') "New Number"  
FROM DUAL;
```

New Number
3648.23

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٢٣)

الدوال Functions

دوال عامة تقبل أي نوع من المدخلات

هذه الدوال تقبل أي نوع كمدخلات لها ونواتجها تعتمد على ما تقوم أنت بتحديده وهذه الدوال هي في الجدول التالي علماً بأنها متعلقة باستعمال اللاشيء NULL

تحويل اللا شيء إلى قيمة حقيقية	NVL(expr1,expr2)
تعتمد على expr1 بحيث إذا كان ذا قيمة يعني ليس اللا شيء فالنتيجة expr2 أما إذا كان expr1 له القيمة اللا شيء فالنتيجة هي expr3	NVL2(expr1,expr2,expr3)
يقارن بين التعبيرين expr1 و expr2 فإن كانوا متساوين فالنتيجة هي اللا شيء ولكن تكون النتيجة التعبير الأول في حالة اختلاف قيمتي التعبيرتين	NULLIF(expr1,expr2)
تعطي أول تعبير ذي قيمة من القائمة	COALESCE(expr1,expr2,expr3 ...exprn)

وها هي الأمثلة التوضيحية

NVL - ١

```
SELECT
LAST_NAME, SALARY, NVL(COMMISSION_PCT, 0), (SALARY*12) + (SALARY*
12*NVL(COMMISSION_PCT, 0)) ANNUAL_SALARY
FROM EMPLOYEES;
```

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	ANNUAL_SALARY
Hutton	8800	.25	132000
Taylor	8600	.2	123840
Livingston	8400	.2	120960
Grant	7000	.15	96600
Johnson	6200	.1	81840

Taylor	3200	0	38400
Fleur	3100	0	37200
Sullivan	2500	0	30000

لاحظ أن الموظفين الذي ليس لديهم عمولات تحسب رواتبهم السنوية من حاصل ضرب الراتب الشهري بـ ١٢ أما الذين لديهم عمولات فتتم إضافتها إلى الراتب السنوي. لاحظ نتيجة الدالة NVL إذ أن الذين ليس لديهم عمولات تحولت عمولاتهم إلى صفر لأننا أخبرنا NVL بذلك ويمكن أن نضع قيمة أخرى غير الصفر. حاول المثال بدون NVL

NVL2 -٢

```
SELECT LAST_NAME, SALARY, COMMISSION_PCT,
NVL2(COMMISSION_PCT, 'SAL+COMM', 'SAL') INOMCE
FROM EMPLOYEES WHERE DEPARTMENT_ID IN(50, 60);
```

LAST_NAME	SALARY	COMMISSION_PCT	INOMCE
Weiss	8000		SAL
Fripp	8200		SAL
Kaufling	7900		SAL
Vollman	6500		SAL
Mourgos	5800		SAL
Vargas	2500		SAL
Russell	14000	.4	SAL+COMM
Partners	13500	.3	SAL+COMM
Errazuriz	12000	.3	SAL+COMM

لاحظ العمود الأخير كيف تتغير قيمته اعتماداً على قيمة العمود COMMISSION_PCT فمرة يأخذ القيمة SAL إذا كانت قيمة العمود COMMISSION_PCT اللاثيء والقيمة SAL+COMM إذا كانت قيمة العمود COMMISSION_PCT غير اللاثيء. وتلاحظ أيضاً أن نوع بيانات العمود الأخير لا يعتمد على نوع العمود .COMMISSION_PCT

NULLIF-٣

```

SELECT first_name, LENGTH(first_name) "expr1", last_name, LENGTH(last_name) "expr2",
NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM employees;

```

FIRST_NAME	expr1	LAST_NAME	expr2	Result
Mozhe	5	Atkinson	8	5
David	5	Austin	6	5
Sarah	5	Bell	4	5
Laura	5	Bissot	6	5
Alexis	6	Bull	4	6
Anthony	7	Cabrio	6	7
Kelly	5	Chung	5	
Curtis	6	Davies	6	
Julia	5	Dellinger	9	5

لاحظ أن قيمة العمود الأخير (Result) يأخذ قيمة expr1 في حالة اختلاف قيمته عن expr2 أما في حالة التساوي فيأخذ القيمة اللاشيء NULL

COALESCE - ٤

هذه الدالة تعطي أول تعبير له قيمة أي لا يأخذ اللاشيء NULL كقيمة من قائمة التعبارات، والشكل العام هو
COALESCE(expr1,expr2,expr3,...,exprn)

مثال:

```

SELECT      LAST_NAME, COALESCE (MANAGER_ID, COMMISSION_PCT, -1)
COMM FROM EMPLOYEES ORDER BY COMMISSION_PCT

```

LAST_NAME	COMM
Partners	100
Errazuriz	100
Doran	146
Abel	149

Smith	146
Cambrault	100
Tucker	145
King	146
McEwen	146
Sully	146
Russell	100
King	-1
Kochhar	100
De Haan	100
Hunold	102
Ernst	103

لاحظ أن الموظف King ليس له مدير وليس له عمولة لذلك أعطت الدالة نتيجة أو تعبير لا يساوي اللاشيء وهو في هذه الحالة (-1) ويمكننا استبداله بشيء آخر

* التعبير الشرطية Conditional Expressions

وهي محاكاة لاستعمال جملة الشرط IF...THEN...ELSIF...THEN...ELSE...
CASE وتعبير هو DECODE

CASE تعبير - ١

وعمله محاكاة جملة ... ELSE... شكلها العام هو IF...THEN...

```
CASE expr WHEN compare_expr1 THEN return_expr1
           [WHEN compare_expr2 THEN return_expr2
           WHEN compare_exprn THEN return_exprn
           ELSE else_expr]
```

END

في هذه الصيغة يتم تقييم التعبير `expr` بحيث إذا حقق الشرط `compare_expr1` فالنتيجة هي `return_expr1` وإلا يتم تقييم التعبير التالي وهكذا حتى نصل إلى عبارة ELSE حيث تكون النتيجة هي القيمة `else_expr`. لاحظ أن ما بين القوسين المربعين اختياري.

مثال:

تريد الشركة زيادة رواتب موظفيها الذين حسب وظائفهم، وظيفة مبرمج IT_PROG بنسبة ١٠٪ ووظيفة كاتب مستودع ST_CLRCK بنسبة ١٥٪ أما مثل المبيعات SA_REP فيحصل على نسبة ٢٠٪. باقي الموظفين لا يحصلون على شيء! لذلك تمت كتابة الجملة التالية:

```
SELECT last_name, job_id, salary,
CASE job_id
WHEN 'IT_PROG'      THEN 1.10*salary
WHEN 'ST_CLRCK'    THEN 1.15*salary
WHEN 'SA_REP'       THEN 1.20*salary
ELSE salary
END "Revised Salary"
```

FROM employees

LAST_NAME	JOB_ID	SALARY	Revised Salary
King	AD_PRES	24000	24000
Lorentz	IT_PROG	4200	4620
Greenberg	FI_MGR	12000	12000
Faviet	FI_ACCOUNT	9000	9000
Baida	PU_CLERK	2900	2900
Tobias	PU_CLERK	2800	2800
Vargas	ST_CLERK	2500	2500
Russell	SA_MAN	14000	14000
Partners	SA_MAN	13500	13500

* الشكل العام:

DECODE(col/expr,s1,r1,s2,r2,...,default)

* عملها : تقييم col أو expr حسب شرط البحث s1 أو s2...الخ وتعطي القيمة r1 أو r2...الخ حسب شرط البحث s1 أو s2...الخ.

إذا لم تتعثر أوراكل على أي من شروط البحث فإن أوراكل تعطي القيمة default. أما إذا لم توجد القيمة default فإن أوراكل لا تعطي شيئاً وبمعنى آخر تعطي اللا شيء NULL.

مثال: نفس المثال السابق في دالة CASE ولكن باستعمال DECODE

```
SELECT last_name, job_id, salary,  
DECODE (job_id, 'IT_PROG', 1.10*salary,  
        'ST_CLRCK', 1.15*salary,  
        'SA_REP', 1.20*salary,  
        Salary)  
END "Revised Salary"
```

```
FROM employees
```

LAST_NAME	JOB_ID	SALARY	Revised Salary
King	AD_PRES	24000	24000
Lorentz	IT_PROG	4200	4620
Greenberg	FI_MGR	12000	12000
Faviet	FI_ACCOUNT	9000	9000
Baida	PU_CLERK	2900	2900
Tobias	PU_CLERK	2800	2800
Vargas	ST_CLERK	2500	2500
Russell	SA_MAN	14000	14000
Partners	SA_MAN	13500	13500

- الدالة GREATEST:

* الشكل العام: GREATEST(col1/expr1,col2/expr2,...etc)

* عملها : إعطاء أكبر قيمة من مجموعة قيم.

مثال ١ :

```
SELECT GREATEST(-100,100,90),  
GREATEST('HARRY','HARRIOT','HAROLD')  
FROM DUAL;
```

GREATEST(-100,100,90)	GREATEST('HARRY')
100	HARRY

٤ - الدالة LEAST:

* الشكل العام: LEAST(col1/expr1,col2/expr2,...etc)

* عملها : إعطاء أقل قيمة من مجموعة قيم.

مثال ١ :

```
SELECT LEAST(-100,100), LEAST('HARRY','HARRIOT','HAROLD')  
FROM DUAL
```

LEAST(-100,100)	LEAST('HARRY','HAR')
-100	HAROLD

٥ - الدالة UID:

* الشكل العام: UID

* عملها : إعطاء رقم صحيح وحيد يمثل رقم المستخدم الحالي Current User والذي تحفظ به أوراكل في قاموس .Data Dictionary بياناتها.

مثال ١: انظر USER

٦ - الدالة USER:

* الشكل العام: USER

* عملها : إعطاء اسم المستعمل الحالي Current User

مثال ١ :

```
SELECT USER,UID FROM DUAL;
```

USER	UID
HR	58

ملاحظة مهمة: المقصود بالمستخدم USER هنا هو مستخدم قاعدة البيانات database user وليس المستخدم الذي ينشأه المبرمج من خلال كتابة برمجه.

- الدالة : USERENV:

* الشكل العام: USERENV(option)

* عملها : إعطاء متغيرات بيئية المستعمل الحالي. وهذه المتغيرات هي :

١ - 'LABEL' هي رقم الجلسة SESSION الحالية. متوفّر في TRUSTED ORACLE .

٢ - 'LANGUAGE' وهي اسم اللغة والمنطقة وجموعة الرموز المستعملة في الجلسة الحالية.

٣ - 'SESSION' وهي اسم الجلسة الحالية.

٤ - 'SESSIONID' وهي رقم الجلسة الحالية.

٥ - 'TERMINAL' وهي اسم نوع الكمبيوتر الذي تستعمله في المهمة الحالية.

مثال ١ :

```
SELECT USERENV('SESSIONID') FROM DUAL;
```

USERENV('SESSIONID')
364904

```
SELECT USERENV('LANGUAGE') FROM DUAL;
```

USERENV('LANGUAGE')
AMERICAN_AMERICA.AR8MSWIN1256

```
SELECT USERENV('TERMINAL') FROM DUAL;
```

USERENV('TERMINAL')
KAMAL-NOTEBOOK

- الدالة VSIZE:

* الشكل العام: VSIZE(expr)

* عملها : إعطاء الطول الحقيقي بالبايتات للتعبير expr أي عدد البايتات التي يتكون منها التعبير. فعلى سبيل المثال في اللغات الشرقية يتكون الحرف الواحد أو الرمز من ٢ بايت أو ٣ بايت؛ فهذه الدالة تعطي عدد البايتات وليس عدد الرموز

: مثال ١ :

```
SELECT LAST_NAME, VSIZE(LAST_NAME)
FROM EMPLOYEES
WHERE EMPLOYEE_ID=101;
```

LAST_NAME	VSIZE(LAST_NAME)
Kochhar	7

مثال ٢ : حرف(لا) في اللغة العربية رمز واحد ولكنه بطول ٢ بايت

```
select vsize('لا') from dual
```

```
VSIZE('لا')
```

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٢٤)

الدوال Functions

دوال المجموعات Group Functions

هذه الدوال تعامل مع مجموعات من الصنف وليس كما في الدوال السابقة والتي تسمى دوال الصنف الواحد Single Row functions أما هذه فتأثيرها يكون على مجموعة من الصنف دفعه واحدة قد تشمل جميع صنف الجدول. وهذا تسمى Multi-Row functions مثل الجداول. ويمكن استعمال الخيارات التاليين معها:

-١ - **DISTINCT** لعدم اعتبار القيم المتكررة.

-٢ - **ALL** لأخذ جميع القيم متكررة وغير متكررة وهو الوضع الافتراضي.

فمثلاً المتوسط الحسابي لـ $1, 1, 1, 3, 10, 10, 10, 10, 10, 10$ هو 10.5 عند استعمال **DISTINCT** بينما هو 2 عند استعمال **ALL**.

دوال المجموعات عادة تحمل اللاثيء **NULL** - أي لا تدعها - ماعدا الدالة **COUNT(*)** وفيما يلي شرح لهذه الدوال:

.**AVG:** الدالة .

* الشكل العام: **AVG([DISTINCT | ALL]n)**

* عملها : إعطاء المتوسط الحسابي لعدد (n) من القيم

: مثال ١

```
SELECT AVG(COMMISSION_PCT) FROM EMPLOYEES
```

AVG(COMMISSION_PCT)

.222857143

```
SELECT AVG(NVL(COMMISSION_PCT, 0)) FROM EMPLOYEES
```

AVG(NVL(COMMISSION_PCT,0))

.072897196

- الدالة **COUNT:** ٢

* الشكل العام: COUNT({*| |[DISTINCT | ALL]expr})

* عملها : إعطاء عدد الصفوف في الاستعلام بحيث:

أ- استعمال * يؤدي إلى عد جميع الصفوف بما فيها الالاشيء NULL والقيم المكررة.

ب- استعمال expr يؤدي إلى عد جميع الصفوف ما عدا تلك التي لا تحمل أية قيمة أي قيمتها الالاشيء NULL.

ج- استعمال DISTINCT يؤدي إلى استثناء القيم المكررة.

أمثلة:

```
SELECT COUNT(COMMISSION_PCT) FROM EMPLOYEES
```

COUNT(COMMISSION_PCT)
35

المثال أعلاه أعطانا عدد الموظفين الذين يأخذون عمولا فقط(حاول أن تحصل على النتيجة السابقة بطريقة مختلفة!)

```
SELECT COUNT(NVL(COMMISSION_PCT, 0)) FROM EMPLOYEES
```

COUNT(NVL(COMMISSION_PCT,0))
107

المثال أعلاه أعطانا عدد كل الموظفين سواءً يأخذون عمولات أم لا

```
SELECT COUNT(JOB_ID) FROM EMPLOYEES
```

COUNT(JOB_ID)
107

```
SELECT COUNT(DISTINCT JOB_ID) FROM EMPLOYEES
```

COUNT(DISTINCTJOB_ID)
19

لاحظ أن المثال الأول أعطانا عدد جميع الموظفين بينما المثال الثاني أعطانا عدد الوظائف فقط

٣- الدالة: MAX

* الشكل العام: MAX([DISTINCT | ALL]expr)

* عملها : إعطاء أكبر قيمة.

مثال:

```
SELECT MAX(SALARY), MAX(COMMISSION_PCT) FROM EMPLOYEES
```

MAX(SALARY)	MAX(COMMISSION_PCT)
24000	.4

٣ - الدالة MIN:

* الشكل العام: MIN([DISTINCT | ALL]expr)

* عملها : إعطاء أصغر قيمة

مثال:

```
SELECT MIN(HIRE_DATE) FROM EMPLOYEES
```

MIN(HIRE_DATE)
17-JUN-87

٤ - الدالة SUM:

* الشكل العام: .SUM([DISTINCT | ALL] expr n)

* عملها : إعطاء مجموع .expr

أمثلة:

```
SELECT SUM(COMMISSION_PCT) FROM EMPLOYEES
```

SUM(COMMISSION_PCT)
7.8

```
SELECT SUM(NVL(COMMISSION_PCT, 0)) FROM
```

SUM(NVL(COMMISSION_PCT,0))
7.8

لاحظ أن NVL لا تؤثر في الدالة SUM

٥- الدالة .VARIANCE:

* الشكل العام: .VARIANCE([DISTINCT | ALL]x)

* عملها : إعطاء معامل التغاير للقائمة العددية X.

أمثلة:

```
SELECT VARIANCE (SALARY) FROM EMPLOYEES
```

VARIANCE(SALARY)
15283140.5

أيضا NVL لا تؤثر في الدالة VARIANCE

٦- الدالة .STDDEV:

* الشكل العام: .STDDEV([DISTINCT | ALL]x)

* عملها : إعطاء الانحراف المعياري للقائمة العددية X.

أمثلة:

```
SELECT STDDEV (SALARY) FROM EMPLOYEES
```

STDDEV(SALARY)
3909.36575

أيضا NVL لا تؤثر في الدالة STDDEV

أوراكل مثلاً (٢٥)

الدوال Functions

استعمال عبارة GROUP BY

تستعمل عبارة GROUP BY مع دوال المجموعات group functions للحصول على ملخصات إحصائية مثل مجموع الرواتب حسب القسم أو معدله حسب الدالة... الخ.

مثال: المثال التالي يستخرج مجموع الرواتب حسب القسم (SALARY)

```
SELECT DEPARTMENT_ID, SUM
      FROM EMPLOYEES
     GROUP BY DEPARTMENT_ID
```

DEPARTMENT_ID	SUM(SALARY)
10	4400
20	19000
30	24900
40	6500
50	156400
60	28800
70	10000
80	304500
90	58000
100	51600
110	20300

	7000
--	------

• استعمال عبارة GROUP BY مع عبارة WHERE

يمكنا استعمال WHERE وعبارة BY في نفس جملة الاستعلام لتحديد الحصول على النتائج المرغوبة. فمثلاً لمعرفة مجموع الرواتب حسب الوظيفة JOB_ID ماعدا وظيفة مثل مبيعات SA_REP فإننا نكتب الاستعلام التالي:

```
SELECT JOB_ID, SUM(SALARY)
```

```
FROM EMPLOYEES
```

```
WHERE JOB_ID != 'SA_REP'
```

```
GROUP BY JOB_ID
```

JOB_ID	SUM(SALARY)
AC_ACCOUNT	8300
AC_MGR	12000
AD_ASST	4400
AD_PRES	24000
AD_VP	34000
FI_ACCOUNT	39600
FI_MGR	12000
HR REP	6500
IT_PROG	28800
MK_MAN	13000
MK_REP	6000
PR_REP	10000

	PU_CLERK	13900
	PU_MAN	11000
	SA_MAN	61000
	SH_CLERK	64300
	ST_CLERK	55700
	ST_MAN	36400

لاحظ أن عبارة WHERE سبقت عبارة GROUP BY وهذا ينطبق على جميع العبارات الأخرى.

* ويمكن استعمال عبارة GROUP BY للتجميع على أكثر من عمود. فلو أردنا مثلاً إحصائية عن مجموع الرواتب حسب القسم وحسب الوظيفة في القسم فإننا نكتب التالي:

```
SELECT DEPARTMENT_ID, JOB_ID, SUM(SALARY)
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID, JOB_ID
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
30	PU_MAN	11000
30	PU_CLERK	13900
40	HR_REP	6500
50	ST_MAN	36400
50	SH_CLERK	64300
50	ST_CLERK	55700

60	IT_PROG	28800
70	PR_REP	10000
80	SA_MAN	61000
80	SA_REP	243500
90	AD_VP	34000
90	AD_PRES	24000
100	FI_MGR	12000
100	FI_ACCOUNT	39600
110	AC_MGR	12000
110	AC_ACCOUNT	8300

لاحظ كيف اختلفت مجاميع الرواتب حسب القسم للوظيفة الواحدة.

ملاحظات مهمة

- ١- يجب ذكر اسم العمود الذي نريد التجميع عليه ضمن الأعمدة المستعملة عنها في عبارة SELECT وإنما سوف يكون هناك خطأ مفاده أن هذه ليست دالة مجموعة بالرغم من ذكر دالة المجموعة؛ فوجب الانتباه.
- ٢- عند استعمال عبارة GROUP BY يجب استعمال دالة مجموعة .group function

* ويمكن استعمال عبارة GROUP BY للحصول على نتائج فردية مثل أكبر راتب حسب الدالة كما يلي:

```
SELECT JOB_ID, MAX(SALARY)
FROM EMPLOYEES
GROUP BY JOB_ID
```

JOB_ID	MAX(SALARY)
AC_ACCOUNT	8300
AC_MGR	12000

AD_ASST	4400
AD_PRES	24000
AD_VP	17000
FI_ACCOUNT	9000
FI_MGR	12000
HR REP	6500
IT_PROG	9000
MK_MAN	13000
MK REP	6000
PR REP	10000
PU_CLERK	3100
PU_MAN	11000

استعمال عبارة HAVING

عندما بحثنا استعمال عبارة WHERE كا نطبقها على الصنوف بصورة فردية؛ فالصنف الذي يحقق الشرط يتم استخراجه. وهنا نستطيع تطبيق نفس الشيء على مجموعات من الصنوف ولكن باستعمال عبارة HAVING. على سبيل المثال لو أردنا معرفة معدلات الرواتب للأقسام التي يوجد فيها أكثر من 5 موظفين؛ فإننا نكتب جملة الاستعلام

التالية:

```

SELECT DEPARTMENT_ID, AVG(SALARY)
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
HAVING COUNT(*) > 5;

```

DEPARTMENT_ID	AVG(SALARY)

30	4150
50	3475.55556
80	8955.88235
100	8600

وها هو الشكل العام لجملة الاستعلام بعد الإضافات والتحسينات السابقة.

ملخص جملة الاستعلام SELECT Summary

من المناقشات السابقة ومع استمرار إضافة معلومات جديدة لدينا تتطور جملة الاستعلام فنستطيع الآن كتابة الشكل العام لجملة الاستعلام SELECT كما يلي:

```

SELECT      */ {[DISTINCT]column / expression [alias],...}

FROM        table

[WHERE    condition(s)]

[HAVING   group condition(s)]

[GROUP BY column(s)]

ORDER BY    {column / expr,...} {[ASC/DESC]};


```

والجدول التالي ملخص لعناصر جملة الاستعلام

العنصر	المعنى
[]	الأقواس المربعة اختيارية
	اختيار أحد العنصرين فقط
{}	يمكن الاختيار بين العناصر التي داخل الأقواس المعقولة {} بحد أدنى عنصر واحد
SELECT	أمر الاستعلام
Alias	الاسم البديل
*	جميع الأعمدة

منع التكرار	DISTINCT
اسم الجدول المرغوب استخراج البيانات منه	FROM table
عبارة الشرط	WHERE
لربط أكثر من شرط معاً	AND/OR
تستعمل لتغيير الأولويات	(الأقواس)
عبارة الشرط لمجموعات من الصنوف	HAVING
تستعمل لتجمیع النتائج في مجموعات	GROUP BY
تستعمل لترتيب البيانات المعروضة وتظهر في آخر جملة الاستعلام	ORDER BY
ترتيب تصاعدي وهو الترتيب المفترض	ASC
ترتيب تنازلي	DESC

يمكن وضع العبارات Clauses في سطور منفصلة لغرض الوضوح

نماذج التشكيل Format Models

نرغب كثيراً في تجاوز الشكل التقليدي (الافتراضي) Default Format للبيانات العددية أو التاريخية؛ ويوجد في أوراقك عدد كبير من هذه النماذج التي تتيح المرونة اللازمة التي تفي بهذا الغرض. وتُستعمل هذه النماذج مع الوظيفتين TO_CHAR و TO_DATE السابقتين شرحهما فيما يلي جدولان لتشكيل البيانات الأول للبيانات العددية Numeric Data والثاني للبيانات التاريخية Date Data مع الأمثلة:

جدول رقم ١ نماذج الأعداد

المودج	مثال على المودج	الشرح
9	9999	يمثل عدد التسعات عدد الأرقام المرغوب إظهارها. إذا كان العدد صفرًا فالنتيجة صفرًا مهما كان عدد التسعات
0	0999	إضافة الصفر على شمال المخرجات
\$	\$9999	إضافة علامة الدولار قبل العدد مهما كان موقع علامة الدولار في

	النموذج		
إلغاء الصفر الزائد-على الشمال-من المخرجات حتى لو تضمن العدد أصفاراً من الشمال طبعاً		B9999	B
إضافة علامة(-) للعدد السالب و فراغ للعدد الموجب على يمين العدد		9999MI	MI
إضافة علامتي السالب والموجب		S999	S
تحديد موقع الفاصلة العشرية		99D99	D
تحديد موقع الفاصلة العادية		9,999	-، الفاصلة
تحديد موقع المجموعة(آحاد، عشرات.. إلخ)		9G999	G
يعطي نوع العملة		C999	C
يعطي رمز العملة المحلية		L999	L
تحديد موقع النقطة العشرية		99.99	(النقطة).
ضرب العدد ب 10^n حيث n تمثل عدد التسعات بعد حرف V		99V99	V
عرض العدد على الصورة العلمية		9.999EEEE	EEEE
عرض العدد حسب الرموز الرومانية (كبيرة/صغرى)		RN	RN/rn

جدول رقم ٢ هو لعناصر تشكيل التاريخ المستعملة في كل من الوظيفتين TO_CHAR و TO_DATE

العنصر	يمثل
القرن حيث S تضيف علامة السالب(-) للتاريخ التي قبل الميلاد	SCC/CC
السنة كأربعة أرقام حيث S تضيف علامة السالب(-) للسنوات قبل الميلاد	YYYY/SYYY Y
السنة كثلاثة أرقام أو رقمين أو رقم واحد	YY/YY/Y
الفاصلة في الموضع المحدد	Y,YYY

السنة كحروف وإضافة (-) في حالة التاريخ قبل الميلاد	SYEAR/YEAR
علامة قبل وبعد الميلاد متبوعة بنقطة أو غير متبوعة بنقطة	BC/AD,BC./A D.
رقم ربع السنة	Q
رقم الشهر	MM
اسم الشهر مضافاً إليه فراغات بطول أقصى ٩ رموز كطول كلي	MONTH
اختصار الشهر إلى ثلاثة رموز	MON
رقم الأسبوع	WW/W
رقم اليوم في السنة/الشهر/الأسبوع	DDD/DD/D
اسم اليوم مضافاً إليه فراغات بطول أقصى ٩ رموز كطول كلي	DAY
اسم اليوم مختصرًا إلى ٣ رموز	DY
التاريخ الجولياني—أي عدد الأيام اعتباراً من ١٢/٣١/٤٧١٣ ق.م.	J
مؤشر انتصاف النهار(صباحاً/مساءً) بنقطة أو بدونها	AM/PM,AM./PM.
ساعات اليوم من ١٢-١	HH/HH12
ساعات اليوم من ٢٣-٠	HH24
الدقائق	MI
الثوانی	SS
عدد الثوانی اعتباراً من منتصف الليل(٠٠-٨٦٣٩٩)	SSSSS

ملاحظات:

- يمكنك كتابة الدوال بصورة حلقات يلف بعضها بعضًا Nested مع الأخذ بعين الاعتبار أن الدالة التي تكون أعمق ما يكون most inner يتم تقييمها أولاً ثم التي تليها وهكذا حتى أول دالة.

٢. عند إضافة صفوف تحتوي حقولاً تاريخية إلى جدول؛ فإن استعمال نموذج التشكيل ممكن باستعمال الدالة TO_DATE وبالتالي يتم تجاوز الشكل التلقائي الذي تحدده أوراكل.

وهكذا أنهينا المرحلة الأولى والتي تعرفنا فيها على جملة الاستعلام واستعمالاتها باستخراج البيانات من جدول واحد فقط. في المرحلة التالية نفس جملة الاستعلام سترافقنا ولكن من خلال استخراج البيانات من أكثر من جدول-جدولين فأكثر

أوراكل مثلا (٢٦)

استخراج البيانات من أكثر من جدول

الربط Joining

إن الأرقام تكون عادة مطلقة وبجريدة؛ فنحن-مثلا- لا نستفيد كثيراً من ذكر رقم القسم فقط عند استخراج البيانات الإحصائية حيث يظل رقم القسم غامضاً ولا نستطيع تذكره دائماً-بل لا داعي لذكره-فإن مدير المنشأة يريد أن يعرف تكلفة القسم الفلاحي وليس القسم رقم كذا مثلاً.

ولكن رقم القسم ورقم الموظف مثلاً مهمان للمبرمج ومصمم قاعدة البيانات وغيرهما من الفنيين. أما المستعمل العادي فالمهم عنده أسماء الأشياء وليس أرقامها.

والطريق الطبيعي لتخزين البيانات المختلفة يتم في جداول مختلفة، في بيانات الأقسام تخزن في جدول الأقسام وبيانات الموظفين تخزن في جدول الموظفين وهكذا. ويتم الربط بين هذه الجداول بواسطة مفتاح معين مثل رقم القسم في جدول الموظفين. وعليه لعرض بيانات الموظفين مع أسماء الأقسام لابد من استعمال الجدولين معاً وذلك من خلال المفتاح الذي يربط بينهما. وهنا تبرز أهمية تصميم الجداول قبل كتابة البرامج.

وهناك نوعان من الربط *Join* بين الجداول:

١- الربط المتساوي *equi-joins* وهي علاقة واحدة بواحد one-one. وهنا تتساوى قيمة عمودين في جدولين مختلفين فيتم أخذ الصورتين التي تتحقق فيها المساواة

٢- الربط غير المتساوي *non-equijoins* وهي علاقة متعددة بمتعدد many-many أو واحد بمتعدد one-to-many

أما كيفية الربط فهناك عدة أساليب:

١- استعمال الربط الطبيعي Natural joins

٢- استعمال عبارة USING

٣- استعمال التقاطع Cross joins

٤- الربط الكامل (أو الربط من الجهتين أو الربط الخارجي Full Outer Join) (two-sided)

٥- شروط اختيارية للربط الخارجي Arbitrary conditions for outer joined

• الربط المتساوي equi-joins

و هنا نبحث عن مساواة عمود في جدول بعمود في الجدول الآخر - علاقـة واحد بواحد . و نستعمل الأـساليـب

التالية :

أولاً : الربط الطبيعي NATURAL JOIN وله الخصائص التالية :

- أ . مبني على جميع الأعمدة المشتركة والتي لها نفس الاسم ونفس النوع في الجداول ذات العلاقة
ب . يتم عرض جميع الصفوف التي تكون فيها قيمة الأعمدة المشتركة متساوية
ت . إذا كانت الأعمدة المشتركة تحمل نفس الاسم ولكن تختلف في النوع ، فإن أوراكل تظهر رسالة خطأ
ث . يتم ذكر نوع الأعمدة المشتركة أثناء تصميم الجداول بحيث يكون أحد الأعمدة هو العمود الرئيسي (الأب) في الجدول الرئيسي (الأب) والعمود الثاني (الابن) في الجدول ابن . لا حقاً سنعرف على تصميم الجداول في قاعدة البيانات .

في المثال التالي سوف يتم عرض رقم باسم القسم ورمز الموقع (المدينة) واسم المدينة .

```
SELECT DEPARTMENT_ID, DEPARTMENT_NAME, LOCATION_ID, CITY  
FROM DEPARTMENTS  
NATURAL JOIN LOCATIONS
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
30	Purchasing	1700	Seattle
90	Executive	1700	Seattle
100	Finance	1700	Seattle
110	Accounting	1700	Seattle
120	Treasury	1700	Seattle
130	Corporate Tax	1700	Seattle

لاحظ كيفية الربط حيث ذكرنا اسم جدول الأقسام DEPARTMENTS في عبارة FROM وذكرنا اسم جدول المواقع LOCATIONS في عبارة NATURAL JOIN ولم نذكر اسم العمود المشترك LOCATION_ID في عبارة الربط الطبيعي ولكن تم ذكره في جملة SELECT للتوضيح فقط.

تقوم أوراكل بالبحث في قاموسها Data Dictionary عن العمود المشترك بين الجداولين وتعرض الصنوف منهما بناءً على هذا العمود، لذلك سميت العملية بالربط الطبيعي أي طبيعياً أن يتم ربط جدولين أو أكثر ربطاً متساوياً من خلال الأعمدة المشتركة.

يمكننا كتابة الجملة السابقة بطريقة أخرى هي

```
SELECT DEPARTMENT_ID, DEPARTMENT_NAME,  
LOCATIONS.LOCATION_ID, CITY  
FROM DEPARTMENTS, LOCATIONS  
WHERE DEPARTMENTS.LOCATION_ID = LOCATIONS.LOCATION_ID
```

تلحظ بعض الفروق بين الجملتين، فقد ذكرنا اسم جدول المواقع LOCATIONS لتحديد العمود LOCATION_ID المشترك لأن أوراكل لا تعرف من أين تستخرج العمود المشترك. إضافة إلى ذلك كتبنا في عبارة WHERE اسم العمود المشترك وأسمى الجداولين مفصولين بالنقطة(.) يمكنك الاختيار بين الصيغتين.

طبعاً في الطريقة الأولى(الربط الطبيعي) يمكننا إضافة عبارة WHERE لتقييد الصنوف الناتجة. فنزيد عبارة WHERE فتصبح كما يلي:

```
SELECT DEPARTMENT_ID,DEPARTMENT_NAME,LOCATION_ID, CITY  
FROM DEPARTMENTS  
  
NATURAL JOIN LOCATIONS  
  
WHERE LOCATION_ID IN(20,30)
```

لاحظ أن عبارة WHERE تأتي بعد عبارة NATURAL JOIN

ثانياً: الربط باستخدام عبارة (استعمال USING) وله الخصائص التالية:

- أ- يربط بين جداولين باستعمال عمودين بنفس الاسم في كلا الجداولين ولكنهما مختلفان في النوع
- ب- إذا كان هناك أكثر من عمود تتطابق في الاسم والنوع، ولكننا نريد الربط بين الجداولين باستعمال عمود واحد فقط
- ج- لا نستعمل أسماء الجداول للتعريف بالأعمدة

د- الربط الطبيعي USING NATURAL JOIN و USING نفس الأسلوب إذا كان العمودان لهما نفس النوع والاسم

مثال: نريد عرض تقرير يتضمن رقم الموظف واسميه الأخير ورمز موقع القسم الذي يعمل فيه ورمز القسم الذي يعمل فيه.

SELECT employees.employee_id, employees.last_name,

departments.location_id, department_id

FROM employees JOIN departments

USING (department_id);

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
198	OConnell	1500	50
199	Grant	1500	50
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
203	Mavris	2400	40
204	Baer	2700	70
205	Higgins	1700	110
206	Gietz	1700	110
100	King	1700	90
101	Kochhar	1700	90
102	De Haan	1700	90

الأسماء البديلة للجداول Table Aliases

مر معنا استعمال الأسماء البديلة للأعمدة ونعرفنا على فوائدها. وهنا نستعمل الأسماء البديلة للجداول أيضا ولسببين:

١. سهولة التعامل معها وذلك بعدم تكرار ذكر اسم الجداول كل مرة وفي كل جزء من جملة الاستعلام وخاصة إذا كان الربط بين عدة جداول وبعده شروط خاصة إذا كانت أسماء الجداول طويلة
٢. زيادة سرعة عملية البحث في قاموس البيانات لسرعة الوصول إلى البيانات تأخذ الأسماء البديلة نفس قاعدة أسماء الجداول في أنها يجب أن تبدأ بحرف ولا يزيد طولها عن ٣٠ رمزاً أو حرف. سنعيد كتابة المثال السابق باستعمال الأسماء البديلة.

```

SELECT e.employee_id, e.last_name, d.location_id, department_id
FROM employees e JOIN departments d
USING (department_id);

```

إرشادات

- كما ذكرنا لا يجوز أن يتجاوز طول الاسم البديل ٣٠ رمزاً
 - إذا ذُكر الاسم البديل للجدول في عبارة FROM فيجب ذكر هذا الاسم في جملة SELECT
 - الأفضل أن تكون الأسماء البديلة ذات معنى
 - الاسم البديل مقبول وذو معنى في جملة SELECT الحالية أي التي نعمل عليها وليس جملة أخرى
 - لا يجوز تأهيل qualify العمود المستعمل في الربط باستعمال USING بواسطة الاسم البديل.
- مثلا، المثال السابق يصبح غير صحيح إذا كتبناه كالتالي:

```

SELECT e.employee_id, e.last_name, d.location_id, d.department_id
FROM employees e JOIN departments d
USING (department_id);

```

فالخطأ هو تأهيل qualify العمود department_id بواسطة الاسم البديل للجدول departments وهو d.

ثالثا: الربط باستخدام عبارة (على ON) وله الخصائص التالية:

- أ- يربط بين جدولين باستعمال عمودين بنفس الاسم في كلا الجدولين ولكنهما مختلفان في النوع
- ب- إذا كان هناك أكثر من عمود يتطابقان في الاسم والنوع، ولكننا نريد الربط بين الجدولين باستعمال عمود واحد فقط
- ج- يجب أن نستعمل أسماء الجداول لتأهيل qualify للأعمدة
- د- نستعمل إشارة المساواة بين العمودين المشتركين وبتحديد العمود المشترك qualify باستعمال الأسماء البديلة للجدول. وهي الصيغة العامة لاستعمال ON
- هـ- كما يمكن استعمال ON مع أية أعمدة للربط بين الجداول

```

SELECT e.employee_id, e.last_name, e.department_id,
d.department_id, d.location_id
FROM employees e JOIN departments d
ON (e.department_id = d.department_id);

```

ربط جدول بنفسه Self-Joins

نحتاج أحياناً لربط جدول بنفسه بحيث نستغني عن كتابة برنامج أو جمل معقدة لاستخراج نتائج معينة. فلو فرضنا أننا نريد أسماء الموظفين وأسماء مدرائهم؛ فإننا نكتب جملة الاستعلام التالية باستعمال الأسماء البديلة بنوع من الذكاء.

```

SELECT e.last_name EMP, m.last_name mgr
FROM employees e JOIN employees m
ON (e.manager_id = m.employee_id);

```

EMP	MGR
OConnell	Mourgos
Grant	Mourgos
Whalen	Kochhar
Hartstein	King
Fay	Hartstein
Mavris	Kochhar
Baer	Kochhar
Higgins	Kochhar
Gietz	Higgins
Kochhar	

...

هنا اعتبرنا الجدول employees كأنه جدولان؛ أحدهما جدول المدراء m والآخر جدول الموظفين e يربط بينهما العمودان employee_id و manager_id له قيمة في العمود employee_id و manager_id مع الأخذ بالاعتبار أن العمود manager_id يربط بينهما العمودان

هل أعجبك هذا الأسلوب؟

الربط غير المتساوي non-equi-joins

لا نبحث في هذا النوع من الربط عن علاقة واحد بواحد -كما في الربط المتساوي بل نبحث عن علاقة واحد بمتعدد-- one-to-many - وإن ظهرت بعض النتائج على شكل واحد بواحد فإن هذه النتيجة تتغير بتغيير البيانات بينما النتائج في علاقة واحد بواحد لا تتغير بتغيير البيانات.

مثال:

في جدول درجات الرواتب job_grades يوجد حدود للراتب تبدأ بحد أدنى وتنتهي بحد أعلى للراتب وفي جدول الموظفين فإن راتب كل موظف يأخذ درجة معينة بحيث يقع الراتب في الحدود.

وما أن هذا الجدول ليس موجودا عند تركب أوراكل فإليك جملة إنشائه (سيتم بحث كيفية إنشاء الجداول لاحقا):

```
CREATE TABLE job_grades (
grade_level VARCHAR2(3),
lowest_sal NUMBER(8),
highest_sal NUMBER(8));
```

ولإدخال البيانات عليه هذه هي جملة الإضافة:

```
INSERT INTO job_grades
VALUES('A',1000,2999);
```

```
INSERT INTO job_grades
VALUES('B',3000,5999);
```

```
INSERT INTO job_grades
VALUES('C',6000,9999);
```

```
INSERT INTO job_grades
VALUES('D',10000,14999);
```

```
INSERT INTO job_grades
```

```
VALUES('E',15000,24999);
```

```
INSERT INTO job_grades
```

```
VALUES('F',25000,40000);
```

```
COMMIT;
```

ادرس جملتي الاستعلام التاليتين :

```
SELECT * FROM JOB_GRADES
```

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

```
SELECT e.last_name, e.salary, j.grade_level
```

```
FROM employees e JOIN job_grades j
```

```
ON e.salary
```

```
BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRADE_LEVEL
Olson	2100	A
Philtanker	2200	A
Markle	2200	A
Landry	2400	A

Gee	2400	A
Perkins	2500	A
Colmenares	2500	A
Patel	2500	A
Vargas	2500	A
Sullivan	2500	A
Marlow		

في الشكل السابق عينة من البيانات.

في جملة الاستعلام الأولى حصلنا على سلم الرواتب أما في الجملة الثانية فحصلنا على درجة كل راتب من رواتب الموظفين في جدول الرواتب. لاحظ علاقة واحد بمتعدد

ملخص الشكل العام لجملة الاستعلام

تحسينات أخرى في جملة الاستعلام فتكون الشكل العام التالي لجملة الاستعلام

```

SELECT      [DISTINCT] { [table].*/expr{alias},... }

FROM        table(s) [alias(es)]

WHERE       [join condition]

AND         [row(join) condition]

OR          [more conditions]

GROUP BY {exp/col.}

HAVING     {group condition}

ORDER BY {expr/col.} [AS/DESC];

```

أوراكل مثلاً (٢٧)

استخراج البيانات من أكثر من جدول

الربط المتساوي الخارجي Outer-Joins

إضافة إلى الأقسام الواردة في مثال استخراج اسم القسم الذي ينتمي إليه الموظف؛ هناك قسم لا يوجد فيه موظفون، ومع ذلك فإن الإدارة العليا تريد إظهار أي اسم قسم –سواء يوجد فيه موظفون أم لا يوجد فيه موظفون– في نفس التقرير الناتج عن جملة الاستعلام السابق شرحها.

نستعمل هنا طريقة الربط الخارجي؛ وتعني ربط صفت في جدول لا تقابلها قيمة في جدول آخر فيتم إظهار قيمة هذا العمود في ذلك الصفت بصورة إجبارية.

فعلى سبيل المثال يوجد قسم في جدول الأقسام departments لا يوجد فيه موظفون وهو القسم رقم ١٩٠ كما أن الموظف Grant لا ينتمي لأي قسم.

إن استعمال أساليب الربط السابقة يؤدي إلى استخراج الجداول ذات القيم المتطابقة فقط؛ ولكننا نريد استخراج الصنوف من الجدول الذي يقع على شمال JOIN حتى لو لم تحتوي على قيم مطابقة من الجدول الذي على شمال JOIN وهذا يسمى الربط الخارجي من الشمال LEFT OUTER JOIN ونفس الأمر نريده من الجدول الذي على يمين JOIN وهذا يسمى الربط الخارجي من اليمين RIGHT OUTER JOIN. وإذا أردنا استخراج الصنوف غير المتطابقة من الجداولين معاً فهذا يسمى الربط الخارجي الكامل FULL OUTER JOIN.

مثال الربط الخارجي من الشمال LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
```

```
FROM employees e LEFT OUTER JOIN departments d
```

```
ON (e.department_id = d.department_id);
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Greene	80	Sales
Marvins	80	Sales
Lee	80	Sales
Abel	80	Sales

Hutton		80	Sales
Taylor		80	Sales
Livingston		80	Sales
Grant			
Johnson		80	Sales
Taylor		50	Shipping

لاحظ في المثال التالي جدول الموظفين يوجد على شمال JOIN لذلك استخرجنا الصف الذي لا يوجد رقم قسم للموظف .Grant

مثال الربط الخارجي من اليمين RIGHT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
De Haan	90	Executive
Chen	100	Finance
Greenberg	100	Finance
Sciarra	100	Finance
Urman	100	Finance
Popp	100	Finance
Faviet	100	Finance

Gietz	110	Accounting
Higgins	110	Accounting
		Treasury
		Corporate Tax
		Control And Credit
		Shareholder Services
		Benefits
		Manufacturing
		Construction

مثال الربط الخارجي الكامل FULL OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e FULL OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

وهذه هي النتيجة

DEPARTMENT_NAME	DEPARTMENT_ID	LAST_NAME
Executive	90	King
Executive	90	Kochhar
Executive	90	De Haan
IT	60	Hunold
IT	60	Ernst
IT	60	Austin
IT	60	Pataballa

IT	60	Lorentz
Finance	100	Greenberg
Finance	100	Faviet
Finance	100	Chen
Finance	100	Sciarra
Finance	100	Urman
Finance	100	Popp
Purchasing	30	Raphaely
Purchasing	30	Khoo
Purchasing	30	Baida
Purchasing	30	Tobias
Purchasing	30	Himuro
Purchasing	30	Colmenares
Shipping	50	Weiss
Shipping	50	Fripp
Shipping	50	Kaufling
Shipping	50	Vollman
Shipping	50	Mourgos
Shipping	50	Nayer
Shipping	50	Mikkilineni
Shipping	50	Landry
Shipping	50	Markle
Shipping	50	Bissot
Shipping	50	Atkinson

Shipping	50	Marlow
Shipping	50	Olson
Shipping	50	Mallin
Shipping	50	Rogers
Shipping	50	Gee
Shipping	50	Philtanker
Shipping	50	Ladwig
Shipping	50	Stiles
Shipping	50	Seo
Shipping	50	Patel
Shipping	50	Rajs
Shipping	50	Davies
Shipping	50	Matos
Shipping	50	Vargas
Sales	80	Russell
Sales	80	Partners
Sales	80	Errazuriz
Sales	80	Cambrault
Sales	80	Zlotkey
Sales	80	Tucker
Sales	80	Bernstein
Sales	80	Hall
Sales	80	Olsen
Sales	80	Cambrault

Sales	80	Tuvault
Sales	80	King
Sales	80	Sully
Sales	80	McEwen
Sales	80	Smith
Sales	80	Doran
Sales	80	Sewall
Sales	80	Vishney
Sales	80	Greene
Sales	80	Marvins
Sales	80	Lee
Sales	80	Ande
Sales	80	Banda
Sales	80	Ozer
Sales	80	Bloom
Sales	80	Fox
Sales	80	Smith
Sales	80	Bates
Sales	80	Kumar
Sales	80	Abel
Sales	80	Hutton
Sales	80	Taylor
Sales	80	Livingston
		Grant

Sales	80	Johnson
Shipping	50	Taylor
Shipping	50	Fleur
Shipping	50	Sullivan
Shipping	50	Geoni
Shipping	50	Sarchand
Shipping	50	Bull
Shipping	50	Dellinger
Shipping	50	Cabrio
Shipping	50	Chung
Shipping	50	Dilly
Shipping	50	Gates
Shipping	50	Perkins
Shipping	50	Bell
Shipping	50	Everett
Shipping	50	McCain
Shipping	50	Jones
Shipping	50	Walsh
Shipping	50	Feeney
Shipping	50	OConnell
Shipping	50	Grant
Administration	10	Whalen
Marketing	20	Hartstein
Marketing	20	Fay

Human Resources	40	Mavris
Public Relations	70	Baer
Accounting	110	Higgins
Accounting	110	Gietz
NOC		
Manufacturing		
Government Sales		
IT Support		
Benefits		
Shareholder Services		
Retail Sales		
Control And Credit		
Recruiting		
Operations		
Treasury		
Payroll		
Corporate Tax		
Construction		
Contracting		
IT Helpdesk		

لعلكم لاحظتم ظهور الموظفين الذي ليس لهم أرقام أقسام والأقسام التي ليس فيها موظفون

ما هو الضرب حاصل الضرب الديكارتي؟ هو ربط كل عنصر في مجموع جميع عناصر المجموعة المقابلة؛ بحيث يكون عدد العناصر الناتج من هذا الرابط هو حاصل ضرب عدد عناصر المجموعتين. فمثلا لنفرض أن المجموعة $A = \{1, 2, 3\}$ والمجموعة $B = \{5, 4\}$ فإن المجموعة $A \times B = \{(1, 5), (1, 4), (2, 5), (2, 4), (3, 5), (3, 4)\}$ أي 6 أزواج مرتبة وهو 3×2 .

وفي بحثنا فإن حاصل الضرب الديكارتي بين جدولين مذكورين في جملة الاستعلام يُنتج عدداً من الصفوف يساوي حاصل ضرب عدد صفوف الجدول الأول \times عدد صفوف الجدول الثاني. وعادة فإن مثل هذا الناتج غير مرغوب فيه!

أما الأحوال التي ينتج عنها حاصل الضرب الديكارتي فهي:

- عدم وجود أي ربط بين الجدولين في جملة الاستعلام أي حذف عبارة JOIN وما شاكلها
- شرط الرابط غير صحيح

ولتجنب حاصل الضرب الديكارتي يجب ذكر شروط ربط صحيحة في جملة الاستعلام.

استعمال عبارة CROSS JOIN لإنتاج حاصل الضرب الديكارتي

في المثال التالي نحصل على حاصل الضرب الديكارتي باستعمال عبارة CROSS JOIN من الجدولين employees وdepartments عدد صفوفه هو $107 \times 27 = 2889$ صفا. قد تختلف النتيجة عندك حسب البيانات التي لديك للأقسام).

```
SELECT last_name, department_name
```

```
FROM employees
```

```
CROSS JOIN departments ;
```

والسؤال؛ أنه ما دام حاصل الضرب الديكارتي غير مرغوب فيه، فلماذا وضعت أوراق طريقة لعمله؟! والجواب أننا نحتاج أحياناً إلى تكوين جداول بأحجام كبيرة بغرض اختبار أداء النظام وكيفية إدارته للمساحات على وسائل التخزين مثل الأقراص الممعنطة ومثل فحص أداء النظام عند حصول استعلام ضخم.

ولعلك تدرك أنها القارئ الكريم أن حاصل الضرب الديكارتي يمكن أن ينتج من جدولين فأكثر.

ننتقل إلى مرحلة جديدة.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

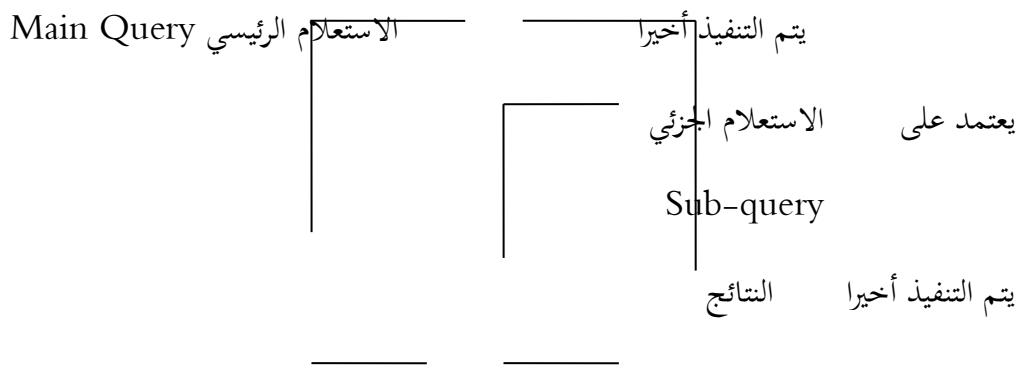
أوراكل مثلاً (٢٨)

الاستعلامات الجزئية والاستعلامات الجزئية المتداخلة

Sub-queries and

Nested Sub-queries

الاستعلام الجزئي Sub-queries هو استعلام عادي داخل استعلام آخر يُسمى الاستعلام الرئيسي Main Query تعتمد نتائجه على نتائج الاستعلام الجزئي. بمعنى آخر الاستعلام الجزئي لا يعطينا نتائج بل إن نتائجه هي أساس الاستعلام الرئيسي التابع له.



في موضوع الاستعلام الجزئي هناك ما يلي:

- ١- يتم تنفيذ الاستعلام الجزئي Sub-Query أولاً.
 - ٢- تعتمد نتائج الاستعلام الرئيسي Main Query على نتائج الاستعلام الجزئي.
 - ٣- لا يتم عرض نتائج الاستعلام الجزئي.
- والشكل العام للاستعلام الجزئي هو

SELECT select_list

FROM table

WHERE exproperator

(SELECT select_list

FROM table);

حيث يمكن استعمال الاستعلام الجزئي في العبارات التالية:

WHERE -١
HAVING -٢
FROM -٣

أما العامل operator يمكن أن يكون (<, =, >, <=, >=) وهي خاصة بالاستعلام الجزئي الذي ينتج صفا واحدا فقط. أما تلك التي تنتج أكثر من صفات فسيستعمل معها فهي (IN, ANY, ALL). وسوف تتولى الأمثلة على ما ذكر أعلاه.

مثال :

- نحن لا نعلم أقل راتب في الشركة ولكننا نستطيع مع ذلك معرفة من هو الموظف الذي يتضمن ذلك الراتب كما يلي:
 ١. نبحث عن أقل راتب كما في جملة الاستعلام التالية

SELECT MIN(salary) FROM employees;

٢. نبحث بعد ذلك عن الموظف الذي يتضمن ذلك الراتب والذي لا نعرفه كما يلي:

SELECT last_name

FROM employees

WHERE salary = lowest salary which is ‘unknown’;

سنربط جملتي الاستعلام السابقتين بغض النظر عن معرفة نتائج جملة الاستعلام الفرعية-أي أن جملة الاستعلام عن أقل راتب في مثالنا سوف تكون هي جملة الاستعلام الفرعية- كما يلي:

SELECT last_name FROM employees جملة الاستعلام الرئيسية

WHERE salary=

(SELECT MIN(salary)FROM employees); جملة الاستعلام الفرعية

LAST_NAME
Olson

ونحن لا نعلم مثلاً مسمى وظيفة الموظف Kochhar ومع ذلك يمكننا معرفة الموظفين الذين يشغلون نفس وظيفة Kochhar بين فيهم Kochhar نفسه كما يلي:

```
SELECT last_name,job_id FROM employees
```

```
WHERE job_id=
```

```
(SELECT job_id FROM employees WHERE  
last_name='Kochhar')
```

LAST_NAME	JOB_ID
Kochhar	AD_VP
De Haan	AD_VP

وجود مسمى الوظيفة في الجملة الرئيسية للتوضيح فقط .

إرشادات لاستعمال الاستعلام الجزئي

- 1 الاستعلام الجزئي يجب أن يظهر على يمين الشرط وبداخل أقواس.
- 2 لا داعٍ لاستعمال ORDER BY في الاستعلام الجزئي.
- 3 تظهر عبارة ORDER BY في آخر جملة الاستعلام الرئيسي.
- 4 يجب أن تكون الأعمدة المتاظرة في كل من الاستعلام الرئيسي والجزئي متطابقة في نوع البيانات.
- 5 يتم تنفيذ الاستعلامات الجزئية المتداخلة اعتباراً من أعمق استعلام most inner إلى أول استعلام-ماعدا الاستعلامات المترابطة correlated-سيتم شرحها لاحقاً.
- 6 يمكن استعمال العوامل المنطقية فيها.
- 7 في حالة الاستعلام الجزئي الذي ينتج صفاً واحداً يجب استعمال العوامل التي تستعمل مع الصفر الواحد وفي حالة تعدد الصنوف يجب استعمال العوامل ذات الصنوف المتعددة

إمكانيات الاستعلام الجزئي

- 1 تنتج صفاً أو أكثر.
- 2 تنتج عموداً أو أكثر.
- 3 يمكنها استعمال دوال المجموعات معها group functions.
- 4 يمكن ربطها باستعمال AND أو OR.
- 5 من الممكن أن تربط جدولين فأكثر.
- 6 استخراج البيانات من جدول أو جداول غير تلك التي في المستوى الأعلى منها أو تلك التي في الاستعلام الرئيسي.
- 7 يمكن استعمالها في جمل الأوامر التالية:
 - أ. أمر الاستعلام SELECT
 - ب. أوامر لغة معالجة البيانات (DML) (INSERT, UPDATE, DELETE)

ت. أمر إنشاء جدول CREATE TABLE.

-٨ يمكن أن تتلازم مع استعلام جزئي ذي مستوى أعلى.

-٩ من الممكن أن تُستعمل العمليات على المجموعات فيها.

ويعنى آخر أن الاستعلام الجزئي هو حملة الاستعلام التي مرت معنا بمختلف إمكانياتها مع بعض الشروط السهلة

أمثلة:

١- سوف نعرض لأرقام الأقسام التي فيها أقل راتب أكبر من أقل راتب في القسم رقم ٥٠ وذلك باستعمال HAVING

```
SELECT      department_id, MIN(salary)
FROM        employees
GROUP BY    department_id
HAVING      MIN(salary) >
(SELECT    MIN(salary)
FROM      employees
WHERE     department_id = 50);
```

DEPARTMENT_ID	MIN(SALARY)
100	6900
30	2500
	7000
20	6000
70	10000
90	17000
110	8300
40	6500
80	6100
10	4400
60	4200

٢- نريد عرض الوظائف التي متوسط رواتبها أقل شي

```

SELECT job_id, AVG(salary)
FROM employees
GROUP BY job_id
HAVING AVG(salary) =
    (SELECT MIN(AVG(salary))
     FROM employees
     GROUP BY job_id);

```

JOB_ID	AVG(SALARY)
PU_CLERK	2780

بعض مشاكل الاستعلام الجزئي

من مشاكل الاستعلام الجزئي:

- أن الاستعلام الرئيسي لا ينتج صفوف والسبب في ذلك أن الاستعلام الجزئي لم ينتج أية صفوف وليس لعدم انطباق الشرط. فمثلاً:

```

SELECT last_name, job_id
FROM employees
WHERE job_id =
    (SELECT job_id
     FROM employees
     WHERE last_name = 'Haas');

```

فالاستعلام الجزئي لم ينتج أية صفوف لأنه لا يوجد اسم Haas وعليه لم ينتج من الاستعلام الرئيسي أي صف.

- الاستعلام الجزئي ينتج أكثر من صف، وهي ليست مشكلة حقيقة لأنه يمكن استعمال العوامل المتعلقة بالصفوف المتعددة MultiRows Operators ولكن إذا استعملنا عاماً من التي تتعلق بالصف الواحد ينتج خطأ؛ فمثلاً:

```

SELECT employee_id, last_name
FROM employees
WHERE salary =
    (SELECT MIN(salary)

```

```

FROM      employees
GROUP BY department_id);

```

ERROR at line 4:

ORA-01427: single-row subquery returns more than one row

الاستعلامات ذات الصنفوف المتعددة

ويتتج عنها عدة صنفوف نستعمل فيها العوامل الخاصة بالصنفوف المتعددة وهي في الجدول التالي:

العامل	معناه أو وظيفته
IN	يساوي أي قيمة من مجموعة
ANY	يقارن مع كل قيمة من مجموعة القيم وبذلك يتحقق الشرط مع أي قيمة
ALL	المقارنة مع كل قيمة من المجموعة بحيث يتحقق الشرط مع كل القيم

- ١ استعمال IN

نريد تقريراً بالموظفين الذين يتتقاضون الرواتب الأقل في أقسامهم؛ فنكتب التالي:

```

SELECT last_name, salary, department_id
FROM      employees
WHERE    salary IN (SELECT      MIN(salary)
                    FROM      employees
                    GROUP BY department_id);

```

يقوم الاستعلام الجزئي بجلب القيم أولاً ثم يقوم الاستعلام الرئيسي بقراءة هذه النتائج ثم عرض الصنفوف التي تطابق أي قيمة منها. أما خادم أوراكل Oracle Server فتبذل له جملة الاستعلام السابقة كما يلي:

```

SELECT last_name, salary, department_id
FROM      employees
WHERE    salary IN (2500, 4200, 4400, 6000, 7000, 8300, 8600, 17000);

```

- ٢ استعمال ANY

نريد عرض بيانات الموظفين الذين يتتقاضون رواتب أقل من أي موظف في قسم تكنولوجيا المعلومات IT

```

SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
       (SELECT salary
        FROM   employees
        WHERE  job_id = 'IT_PROG')
        AND   job_id <> 'IT_PROG';

```

عينة من النتائج

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
132	Olson	ST_CLERK	2100
136	Philtanker	ST_CLERK	2200
128	Markle	ST_CLERK	2200
135	Gee	ST_CLERK	2400
127	Landry	ST_CLERK	2400
119	Colmenares	PU_CLERK	2500
191	Perkins	SH_CLERK	2500
182	Sullivan	SH_CLERK	2500
144	Vargas	ST_CLERK	2500
140	Patel	ST_CLERK	2500
131	Marlow	ST_CLERK	2500
143	Matos	ST_CLERK	2600

. ANY هو SOME له نفس عمل

لاحظ أن ANY < ANY> تكافئ أصغر من أعلى قيمة (٩٠٠٠) أما ALL فتعني أكبر من أصغر قيمة أما IN فتعني

نريد عرض رواتب الموظفين الذين رواتبهم أقل من رواتب جميع موظفي قسم IT

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ALL
      (SELECT salary
       FROM employees
       WHERE job_id = 'IT_PROG')
      AND job_id <> 'IT_PROG';
```

عينة من النتائج

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
198	OConnell	SH_CLERK	2600
199	Grant	SH_CLERK	2600
115	Khoo	PU_CLERK	3100
116	Baida	PU_CLERK	2900
117	Tobias	PU_CLERK	2800
118	Himuro	PU_CLERK	2600
119	Colmenares	PU_CLERK	2500
125	Nayer	ST_CLERK	3200
126	Mikkilineni	ST_CLERK	2700

لاحظ أن هذا الاستعلام يعني رواتب الموظفين الذين رواتبهم أقل من أقل راتب في قسم IT.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلا (٢٩)

التعامل مع عوامل المجموعات

Working with sets operators

في هذا الدرس سوف نتعرف على كيفية التعامل مع المجموعات من خلال جمل الاستعلام في سكول. وستكون كنوع من المماثلة مع الرياضيات.

وحين نتكلم عن المجموعات في سكول فإننا نتكلم عن مجموعة الصفوف الناتجة عن استعلام ما Query .

هناك العمليات التالية على المجموعات:

- ١ الاتحاد UNION وهو اتحاد من مجموعتين فأكثر لتشكيل مجموعة أخرى تحتوي على جميع عناصر المجموعات بدون تكرار العناصر المشابهة
- ٢ اتحاد الكل UNION ALL نفس التعريف السابق مع تكرار العناصر المشابهة
- ٣ التقاطع INTERSECT مجموعة العناصر المشتركة
- ٤ الطرح MINUS العناصر المتممة إلى مجموعة ولا تنتمي إلى المجموعة الأخرى

سنستعمل الجدولين التاليين في هذا الدرس، وهما:

- ١ جدول الموظفين employees وفيه بيانات الموظفين
 - ٢ جدول التاريخ الوظيفي job_history، وفيه يتم تسجيل تنقلات الموظف من وظيفة إلى أخرى ومن قسم إلى آخر وتاريخ تلك التنقلات
- بنية جدول الموظفين Structure employees

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER (6)
FIRST_NAME		VARCHAR2 (20)
LAST_NAME	NOT NULL	VARCHAR2 (25)
EMAIL	NOT NULL	VARCHAR2 (25)
PHONE_NUMBER		VARCHAR2 (20)
HIRE_DATE	NOT NULL	DATE

JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

بنية جدول التاريخ الوظيفي job_history Structure

EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

يمكنك الاستعلام عن محتويات الجدولين.

عوامل المجموعات Sets Operators

الاتحاد UNION

الاتحاد يعطينا جميع عناصر المجموعتين في مجموعة واحدة بدون تكرار العناصر المتشابهة.

مثال: نريد عرض بيانات الموظفين الحالية والسابقة وبدون تكرار. فنكتب ما يلي:

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

عينة من النتائج

EMPLOYEE_ID	JOB_ID
-------------	--------

100	AD_PRES
101	AC_ACCOUNT
101	AC_MGR
101	AD_VP
102	AD_VP
102	IT_PROG

117 ROWS

لاحظ أن الموظف رقم ١٠١ تكرر ٣ مرات والموظف ١٠٢ تكرر مرتين. وهذا لا يعني أن هناك تكرارا للعناصر المتشابهة(العناصر هي الصنوف). فنجد أن الموظف ١٠١ تنقل بين عدة وظائف مما يجعل كل صف مختلف عن الثاني، وكذلك يقال في حق الموظف رقم ١٠٢.

❖ إرشادات عامة

- ١ - عدد الأعمدة ونوع البيانات يجب أن يكون متطابقا في جميع عبارة SELECT في جملة الاستعلام(جملة الاستعلام تتكون من عدة عبارات SELECT مرتبطة بأحد عوامل المجموعات). أسماء الأعمدة لا يشترط أن تكون متشابهة
- ٢ - العامل UNION يعمل على جميع الأعمدة المذكورة في عبارة SELECT
- ٣ - قيم اللاشيء NULL VALUES لا تحمل إنما تؤخذ بعين الاعتبار
- ٤ - العامل IN له الأولوية على UNION
- ٥ - الترتيب الافتراضي Default يكون حسب أول عمود في أول عبارة SELECT ويكون الترتيب تصاعديا Ascending
- ٦ - عناوين الأعمدة في الصنوف الناتجة هي أسماء أو عناوين الأعمدة المذكورة في أول SELECT بغض النظر عن أسماء الأعمدة المذكورة في باقي عبارات SELECT التي تكون جملة الاستعلام.

❖ اتحاد الكل UNION ALL

نريد عرض الأقسام التي عمل فيها الموظفون سابقا ويعملون فيها حاليا

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
100	AD_PRES	90
101	AD_VP	90
101	AC_ACCOUNT	110
101	AC_MGR	110
102	AD_VP	90
102	IT_PROG	60
103	IT_PROG	60
104	IT_PROG	60
105	IT_PROG	60

لو أعدنا نفس المثال باستعمال UNION فقط سنجد التالي:

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
100	AD_PRES	90
101	AC_ACCOUNT	110
101	AC_MGR	110
101	AD_VP	90
102	AD_VP	90
102	IT_PROG	60
103	IT_PROG	60
104	IT_PROG	60
105	IT_PROG	60

106	IT_PROG	60
107	IT_PROG	60
108	FI_MGR	100
109	FI_ACCOUNT	100
110	FI_ACCOUNT	100
111	FI_ACCOUNT	100

التقاطع INTERSECT

عملية التقاطع هي أخذ الصنوف المشتركة الناتجة عن عبارتي SELECT المكونة بجملة الاستعلام. فلو أردنا عرض الصنوف للموظفين الذين تغيرت وظائفهم ثم رجعوا إلى وظائفهم الأصلية، وهذا يعني أن لهم سجلات في الجدول JOB_HISTRY والمجدول EMPLOYEES لذلك نكتب التالي:

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

فينتج لدينا الصنوف التالية:

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

نشاط: عمل الاستعلام السابق بإضافة العمود department_id

إرشادات حول استعمال التقاطع :INTERSECT

- ١- يجب أن يكون عدد ونوع الأعمدة في جميع عبارات SELECT التي تكون الاستعلام متطابقاً. ولكن أسماء الأعمدة فليس شرطاً تطابقها.
- ٢- تبديل موقع الجداول في الاستعلام لا يؤثر في النتائج

٣- عملية التقاطع تأخذ بعين الاعتبار اللاشيء NULL values

تساؤل: ما الفرق بين الربط join وعملية التقاطع INTERSECT

• الطرح MINUS

تقتضي عملية الطرح استثناء الصنوف المشتركة بين عبارات SELECT التي تكون الاستعلام. أي عرض الصنوف في عبارة SELECT معينة واستثناء الصنوف المشتركة الناتجة من عبارة SELECT أخرى.

لو أردنا عرض أرقام الموظفين ووظائفهم الذين لم يغيروا وظائفهم—أي الذين ليس سجلات في الجدول JOB_HISTORY نكتب ما يلي:

```
SELECT employee_id, job_id  
FROM employees  
MINUS  
SELECT employee_id, job_id  
FROM job_history;
```

عينة من النتائج:

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AD_VP
102	AD_VP
103	IT_PROG
104	IT_PROG
105	IT_PROG
106	IT_PROG

إرشادات عامة لاستعمال عوامل المجموعات

- يجب أن تكون عدد الأعمدة أو التعبير expressions متطابقة في العدد ونوع البيانات
- يمكن استعمال الأقواس للتحكم في ترتيب تنفيذ الاستعلام
- يمكن استعمال عبارة الترتيب ORDER BY كما يلي :
- يجب أن تظهر عبارة الترتيب في آخر جملة الاستعلام فقط وليس مع أي SELECT

- تقبل عبارة الترتيب أسماء الأعمدة Column Names أو الأسماء البديلة Aliases أو موقع العمود(Column Name) SELECT من أول Positional Notation فقط.

كيف يتعامل خادم أوراكل Oracle Server مع عوامل المجموعات؟

- يقوم خادم أوراكل بحذف الصنوف المكررة آلياً ما عدا حالة UNION ALL
- أسماء الأعمدة من أول SELECT هي التي تظهر كعناوين في النتائج
- يتم ترتيب النتائج تصاعدياً بصورة آلية عدا في حالة استعمال UNION ALL
- إذا أنتجت عبارات SELECT المكونة للاستعلام جميعها نوع البيانات رموز عادية CHAR فالنتائج جميعها من نفس النوع
- أما إذا كان بعضها رموز ممتدة VARCHAR2 وبعضها رموز CHAR فالنتائج تكون من نوع رموز ممتدة VARCHAR2

التطابق في عبارات SELECT المكونة للاستعلام

قلنا إنه يجب أن يكون عدد ونوع الأعمدة أو التعبيرات في جميع عبارات SELECT متطابقين. لنأخذ التالي؛ وهو عرض رمز القسم department_id والموقع location_id وتاريخ التعيين لجميع الموظفين مع الأخذ بعين الاعتبار أن هذه الأعمدة موجودة في الجداول departments و employees مع العلم أن الجدول departments لا يحتوي العمود hire_date بينما لا يحتوي الجدول employees على العمود location_id. يبدو لأول وهلة صعوبة الأمر؛ ولكن بقليل من التصرف نستطيع كتابة الاستعلام المطلوب كما يلي:

```
SELECT department_id, TO_NUMBER(null) location, hire_date
FROM   employees
UNION
SELECT department_id, location_id,   TO_DATE(null)
FROM   departments;
```

عينة من النتائج

DEPARTMENT_ID	LOCATION	HIRE_DATE
10	1700	
10		17-SEP-87
20	1800	
20		17-FEB-96

	20		17-AUG-97
	30	1700	
	30		07-DEC-94
	30		18-MAY-95
	30		24-JUL-97
	30		24-DEC-97
	30		15-NOV-98

لاحظ استعمال الوظيفتين TO_DATE و TO_NUMBER مع الالاشيء NULL كما وبالمثل نعرض رقم الموظف ووظيفته وراتبه لجميع الموظفين من الجدولين EMPLOYEES و JOB_HISTORY كما يلي:

```
SELECT employee_id, job_id, salary
FROM employees
UNION
SELECT employee_id, job_id, 0
FROM job_history;
```

عينة من النتائج

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AC_ACCOUNT	0
101	AC_MGR	0
101	AD_VP	17000
102	AD_VP	17000
102	IT_PROG	0
103	IT_PROG	9000
104	IT_PROG	6000

105	IT_PROG	4800
106	IT_PROG	4800
107	IT_PROG	4200
108	FI_MGR	12000

لاحظ استعمال الإضافة Literal (الصفر - ٠) في عبارة SELECT الثانية!

بهذا نكون أخذنا فكرة واسعة وعميقة عن جملة الاستعلام وسننتقل في الدروس القادمة إلى مرحلة جديدة هي مرحلة معالجة البيانات دون نسيان جملة الاستعلام.

أوراكل مثلاً (٣٠)

معالجة البيانات في أوراكل Data Manipulation

معالجة البيانات تكون بإضافة صفوف إلى الجدول أو تعديل بيانات أو حذف بيانات من جدول. ولهذا الغرض هناك مجموعة جزئية من مجموعة أوامر سُكّول تسمى لغة معالجة البيانات (DML) وهي:

١. الأمر INSERT لإضافة صف أو صفوف إلى جدول.
٢. الأمر UPDATE لتعديل بيانات عمود أو أعمدة في صف أو صفوف.
٣. الأمر DELETE لحذف صف أو صفوف من جدول.

وها هي هذه الأوامر بالتفصيل.

١. أمر الإضافة INSERT لإضافة صف أو صفوف إلى جدول. وله عدة حالات نتناولها بالتفصيل:
أ- إضافة صف إلى جدول بتحديد بيانات لكل عمود ويأخذ إحدى الصيغتين التاليتين:

أولاً: ذكر جميع أسماء الأعمدة كما يلي:

```
INSERT INTO table_name(col1,col2,...coln)
          VALUES(val1,val2,...valn);
```

مثال: لإضافة صف جديد إلى جدول الأقسام departments نكتب الجملة التالي:

```
INSERT INTO departments(department_id,
                        department_name, manager_id, location_id)
VALUES (700, 'Public Relations', 100, 1700);
```

1 row created.

لاحظ أن البيانات من نوع رموز CHAR أو VARCHAR2 يجب أن تُحصر بين حاصلتين علويتين مفردتين Single DATE وكذلك عامل البيانات من نوع تاريخ Quotation

ثانياً: عدم ذكر أي اسم من أسماء الأعمدة كما يلي:

```
INSERT INTO table_name
          VALUES(val1,val2,...valn);
```

مثال: لإضافة صف جديد آخر إلى جدول الأقسام *departments* نكتب الجملة التالية:

```
INSERT INTO departments VALUES(12,'Reception',102,1800);
```

1 row created.

وفي هذه الحالة يجب مراعاة ترتيب الأعمدة في الجدول من حيث:

١- أنواع البيانات في كل عمود؛ بحيث لا ندخل في عمود بيانات تختلف عنه في النوع

٢- حجم كل عمود؛ فلا نضيف بيانات أكبر من حجم العمود

٣- البيانات المرغوبة لكل عمود؛ فلا نبدل بيانات بأخرى، فمثلاً كل من العמודين *manager_id* و *location_id* لهما نفس نوع البيانات فلو بدلنا رمز الموقع *location_id* مع رمز المدير *manager_id* فإن أوراكل لا تصدر أية رسالة خطأ إطلاقاً؛ ولكننا هنا خلطنا بين القيمتين مما يؤدي إلى أخطاء عملية أو منطقية!

ب- إضافة صف إلى جدول بتحديد بيانات بعض الأعمدة وترك البعض الآخر يأخذ قيمة اللاشيه *NULL* ويأخذ إحدى الصيغتين التاليتين:

أولاً: ذكر أسماء الأعمدة المراد إعطاء قيم لها مع مراعاة أن لا تكون هذه الأعمدة إجبارية أي ذات الخاصية *NOT NULL*- وكمثال على ذلك نريد إضافة صف جديد بإضافة رقم واسم قسم جديد ولكن بدون تحديد الموقع ولا المدير:

```
INSERT INTO departments(department_no,department_id)
```

```
VALUES(13,'Documentation')
```

1 row created.

وهذا الأسلوب يُسمى الأسلوب الضمني Implicit Method

ثانياً: ذكر جميع أسماء الأعمدة كما ورد في بند (أ) مع إعطاء قيمة اللاشيه *NULL* للعمود المراد إهماله. سنكرر المثال السابق بالصيغة الجديدة كما يلي:

```
INSERT INTO departments(department_no,department_name,
```

```
manager_id,location_id)
```

```
VALUES(14,'IT',NULL,NULL);
```

1 row created.

وهذا يُسمى الأسلوب الصريح Explicit Method

 أخطاء شائعة أثناء إضافة بيانات جدول:

- ١- فقدان قيم إجبارية لأعمدة إجبارية مثل رقم الموظف employee_id
- ٢- تكرار لقيم يفترض أن تكون وحيدة وذلك عند عمل الفهرس الوحيد Unique Index أو قيد الفردية Uniqueness Constraint
- ٣- تجاوز قيد المفتاح الأجنبي Foreign Key Constraint وذلك بإضافة موظف ورقم قسم له ليس موجوداً في جدول الأقسام مثلاً
- ٤- تجاوز قيد التحقق CHECK Constraint مثل أن يكون هناك قيد على عمود الراتب salary بحيث يكون محصوراً بين ١٥٠٠ و ٢٠٠٠٠، فهنا يتم تجاوز هذا القيد إذا كانت قيمة الراتب المدخلة لا تقع في هذا المدى
- ٥- اختلاف أنواع البيانات بين القيم المدخلة ونوع الأعمدة، مثل إدخال قيمة رمزية Character value على عمود Numeric
- ٦- القيمة المدخلة أكبر من حجم العمود، كأن ندخل اسم قسم مثلاً عدد الرموز فيه أكبر من عرض(حجم) عمود الاسم department_name.

سوف يأتي شرح هذه القيود لدى بحث هيكل البيانات في أوراقل

معالجة إدخال التاريخ والوقت

عندما تتم إضافة صف يحتوي على قيمة تاريخ date فإنه التاريخ يأخذ الشكل DD-MON-RR والمعرف مسبقاً. يمثل السنة والشهر واليوم والساعة والدقيقة والثانية. الوقت عادة هو منتصف الليل حيث تكون الساعة بنظام الـ ٢٤ هي (٠٠:٠٠:٠٠). سنتعمل في هذا المثال الدالة SYSDATE المار ذكرها؛ كما في المثال التالي:

```
INSERT INTO employees (employee_id,first_name,last_name,email,
phone_number,hire_date,job_id,salary,
commission_pct,manager_id,department_id)

VALUES(113,'Louis','Popp','LPOPP',
'515.124.4567',SYSDATE,'AC_ACCOUNT',6900,
NULL, 205, 100);
```

1 row created.

استعملنا الدالة SYSDATE كقيمة للعمود .hire_date

استعمال المتغيرات في الأمر INSERT

يمكّنا استعمال المتغيرات بدلاً من قيم ثابتة للأعمدة لإضافة صفوف إلى الجداول بنفس الأسلوب الذي ورد في بحث الاستعلام وعلى سبيل المثال يمكننا كتابة الأمر التالي لإضافة صف إلى جدول الأقسام العتيق departments :

```
INSERT INTO departments
```

```
VALUES(&dno,'&dbname',&dmngr,&loc);
```

سوف يطلب أوراكل إدخال قيم المتغيرات كما مر شرحه في الاستعلام، حيث سيظهر ما يلي:

Enter value for dno:	16
Enter value for dbname:	Labors
Enter value for loc:	1700
Enter value for dmngr:	102

1 row created.

لاحظ أن المتغيرات التي تمثل البيانات الرمزية Character قد تم وضعها بين حاصلتين، بنفس الأمر ينطبق على البيانات من نوع تاريخ Date.

ويعكينا أن نحفظ الجملة السابقة في ملف وتشغيله في أي وقت نريده.

إضافة صفوف من جدول آخر

يمكننا أن نضيف صفوفا إلى جدول ما بأخذها من جدول آخر باستعمال أساليب الاستعلام المار ذكرها سابقا. والشكل العام لذلك هو:

```
INSERT INTO table_name[(col1,col2,...coln)]
```

```
query;
```

حيث تشير الأقواس المربعة إلى أن ذكر أسماء الأعمدة ليس إجباريا. ومثال على ذلك:

```
INSERT INTO sales_reps(id,name,salary,commission_pct)
```

```
SELECT employee_id,last_name,salary,commission_pct
```

```
FROM employees
```

```
WHERE job_id LIKE "%REP;%"
```

4 rows created

ولا ننسى هنا أن الجدول sales_reps يجب أن يكون سبق إنشاؤه.

ملاحظات:

١ - لا نستعمل عبارة VALUES هنا.

٢ - يجب أن يتطابق عدد ونوع الأعمدة المختارة من الجدول المصدر مع عدد ونوع الأعمدة المختارة من الجدول الهدف يمكننا اختيار جميع الأعمدة من الجدول المصدر بالإضافة صفوف إلى المصدر الهدف؛ وفي هذه الحالة يجب أن يكون الجدول الهدف نسخة من الجدول الأصل من حيث عدد وأنواع الأعمدة-ليس شرطاً أسماء الأعمدة.

مثال على ذلك:

```
INSERT INTO copy_emp
```

```
SELECT *
```

```
FROM employees;
```

قم بنسخ جميع الصفوف بجميع الأعمدة من الجدول المصدر(الأصل) employees إلى الجدول الهدف copy_emp للتأكيد مرة أخرى؛ جميع ما تعلمناه وما سوف نتعلم في جمل الاستعلام يمكننا استعماله هنا.

\

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٣١)

معالجة البيانات في أوراكل Data Manipulation

٢- أمر التعديل UPDATE

ويستعمل لتعديل بيانات صف أو صفوف في جداول وصورته العامة هي:

```
UPDATE table_name/alias
```

```
SET      col1 = newvalue1,  
        [col2 = newvalue2|...coln=newvaluen]
```

```
[WHERE condition(s)];
```

حيث *newvalue* تمثل القيمة الجديدة المراد إعطاؤها للعمود، وقد تكون القيمة الجديدة *newvalue* إما:

١. تعبيراً *expression* (قيمة ثابتة، عملية حسابية، اسم عمود... الخ.).
٢. استعلاماً جزئياً *sub-query*.

مع ملاحظة أن عدم استعمال عبارة WHERE يؤدي إلى تعديل جميع الصفوف.

مثال ١ :

في هذا المثال سوف نغير قسم الموظف رقم ١١٣ إلى القسم رقم ٧٠ بغض النظر عن قسمه الأصلي كما يلي:

```
UPDATE employees
```

```
SET    department_id = 70
```

```
WHERE employee_id = 113;
```

```
1 row updated.
```

مثال ٢ :

في هذا المثال سوف يتم تغيير رقم القسم لجميع الموظفين في الجدول copy_emps إلى القسم رقم ١١٠ بغض النظر عن أقسامهم السابقة كما يلي:

```
UPDATE copy_emp
```

```
SET    department_id = 110;
```

22 rows updated.

على فرض أن الجدولين employees و copy_emps متطابقان في عدد الصفوف(قد تختلف البيانات عمما هو موجود لديك).

مثال ٣: سوف نستخدم الاستعلام الجزئي لتعديل بيانات كل من الأعمدة salary و commission_pct و hire_date في جدول الموظفين employees كما يلي:

UPDATE employees E

SET (salary,commission_pct)=

(SELECT AVG(salary)*1.1,AVG(commission_pct)

FROM employees WHERE department_id = E.department_id,

hire_date='11-JUN-86';

107 rows updated.

لاحظ أنه لا يمكننا كتابة الشكل التالي لتعديل بيانات كل من العمودين salary و commission_pct :

SET (salary,commission_pct) = AVG(salary*1.1),AVG(commission_pct)

كما أن الصيغة التالية أيضا خطأ:

SET (salary,commission_pct) = salary*1.1,commission_pct + 100

بل يجب استعمال الاستعلام الجزئي أو ذكر كل عمود مفصولا عن الآخر بفاصله كما عالجنا عمود تاريخ التعيين .hire_date

كما يمكننا استعمال الصيغة التالية لتعديل أكثر من عمود بواسطة الاستعلام الجزئي:

UPDATE table

SET column = (SELECT column

FROM table WHERE condition)

[,

column =

(SELECT column FROM table WHERE condition)

[WHERE condition];

ومثال على ذلك نريد تعديل راتب وظيفة الموظف رقم ١١٤ ليكون نفس وظيفة وراتب الموظف رقم ٢٠٥؛ فنكتب ما يلي:

```
UPDATE employees  
SET job_id = (SELECT job_id  
FROM employees  
WHERE employee_id = 205),  
salary = (SELECT salary  
FROM employees  
WHERE employee_id = 205)  
WHERE employee_id = 114;  
1 row updated.
```

ولا يقتصر التعديل باستخدام التعديل الجزئي على قيم من نفس الجدول بل قد تكون القيم من جدول أو جداول أخرى كما في المثال التالي:

```
UPDATE copy_emp  
SET department_id = (SELECT department_id  
FROM employees WHERE employee_id = 100)  
WHERE job_id = (SELECT job_id FROM employees  
WHERE employee_id = 200)  
1 row updated.
```

بسم الله الرحمن الرحيم

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٣٢)

معالجة البيانات في أوراكل **Data Manipulation**

٣. أمر الحذف DELETE

ويستعمل لحذف صف أو صفوف من جدول وصوريته العامة هي:

```
DELETE [FROM] table_name
```

```
[WHERE condition(s)];
```

حيث تشير الأقواس المربعة إلى أن عبارتي WHERE و FROM اختياريتان؛ علماً بأن حذف عبارة WHERE يؤدي إلى حذف جميع الصفوف من الجدول.

ملاحظة: إذا لم يتم حذف صفوف، فلا رسالة خطأ ولكن رسالة تفيد بحذف عدد صفر صف تم حذفها “0” deleted

في المثال التالي سيتم حذف قسم أول Finance من جدول الأقسام :departments

```
DELETE FROM departments
```

```
WHERE department_name = 'Finance';
```

```
1 row deleted.
```

أما في هذا المثال فسوف يتم حذف جميع الصفوف من الجدول :copy_emps

```
DELETE FROM copy_emp;
```

```
22 rows deleted.
```

حذف صفوف من جدول اعتماداً على جدول آخر 

من الممكن حذف صفوف من جدول معين اعتماداً على قيمة من جدول آخر. ولنفرض أننا نريد حذف بيانات الموظفين من قسم أول Public، فنكتب الجملة التالية:

```
DELETE FROM employees
```

```
WHERE department_id =
```

```
(SELECT department_id
```

```
FROM departments
```

```
WHERE department_name
```

```
LIKE '%Public%');
```

```
1 row deleted
```

أمر البتر TRUNCATE 

يستخدم أمر البتر TRUNCATE لحذف جميع البيانات من الجدول؛ بدلاً من استعمال الأمر DELETE وهو أسرع من الأمر DELETE ولا يحتاج إلى عمليات تخزين مؤقتة بل إنه يمسح البيانات وينظف الذاكرة وبسرعة. وصيغته العامة هي:

```
TRUNCATE TABLE table_name [REUSE STORAGE];
```

لاحظ الكلمة TABLE في الصيغة وذلك لأن هذا الأمر من لغة تعريف البيانات DDL. أما الخيار REUSE STORANGE فيعني أن يحتفظ الجدول بالمساحة على القرص التي كانت مستعملة لخزن البيانات الممسوحة. أما إن حذف ذلك الخيار فإن تلك المساحة تحرر لاستعمال جداول أخرى وهو الخيار الافتراضي default.

ويجب الحذر أيضا عند استعمال هذا الأمر لأنه لا سبييل إلى استرجاع البيانات الملغاة ما لم يكن هناك نسخة احتياطية لآخر تعديل قبل إلغاء تلك البيانات.

إذا كان الجدول أبا parent فيجب تعطيل قيود التكامل disable integrity constraints قبل عملية البت

أوراكل مثلا (٣٣)

معالجة البيانات في أوراكل Data Manipulation

TRANSACTION معالجة الحركات

الموضوع طويل وصعب إلى حد ما ولكنها مهم جدا. ويتفق معنا عندما نكتب برامج واجهة المستخدم باستعمال نماذج أوراكل وتقاريرها وال BI وغيرها وهي مبنية على لغة SQL و لغة أوراكل الإجرائية PL/SQL والتي سوف أتطرق إلى شرح معظمها بعد الانتهاء من ال SQL إن شاء الله.

الحركة transaction هي عملية على قاعدة البيانات تتضمن مجموعة من التغييرات على جدول أو أكثر. وهناك أنواع من هذه الحركات هي:

- ما يتعلق بمعالجة البيانات DML وتكون الحركة فيه من أي عدد من أوامر لغة معالجة البيانات DML مثل INSERT,UPDATE,...
- ما يتعلق بتعريف مكونات قاعدة البيانات DDL وتكون فيه الحركة من أمر واحد من لغة تعريف البيانات مثل CREATE,ALTER,...
- ما يتعلق بالتحكم بالبيانات DCL مثل GRANT,REVOKE وقد يحدث أثناء تنفيذ الحركة أن بعض الأوامر المكونة للحركة تتم على قاعدة البيانات بينما البعض الآخر لا يتم. أي أن بعض الأوامر تؤدي إلى تغييرات في قاعدة البيانات بينما الأوامر الأخرى لا تتم أصلا. وهذا ما يسمى بحالة منتصف الطريق. وعلى كل حال فإن ميكانيكية الحركة تعني أن يتم ثبيت جميع التغييرات أو لا يتم منها شيء (تحمل جميعها).

مثلا عند تسجيل القيد في الأستاذ العام يجب أن يكون الجانب المدين مساوايا للجانب الدائن، وعليه فإن جميع الأطراف وبمبالغها يجب أن تسجل جميعها دفعه واحدة أو تلغى جميعها كي لا يتم تسجيل مبالغ غير متوازنة.

تبدأ الحركة مع أول أمر من لغة معالجة البيانات أو لغة تعريف البيانات وتنتهي(نحاها أو فشلا) في الحالات التالية:

- وجود أحد الأمرين COMMIT أو ROLLBACK.
- تفريد أمر من لغة تعريف البيانات DDL أو التحكم بالبيانات DCL.
- وجود أخطاء مثل حالة إغلاق الصف أو الجدول لعملية ما ولم تتم العملية بسبب أو لآخر فلم يمكن إعادة فتح الصف أو الجدول.
- الخروج من SQL*Plus أو غيرها من بيئات التطوير بالأمر EXIT مثلا.
- تعطل الكمبيوتر الذي يحتوي قاعدة البيانات.

إن أوامر لغة تعريف البيانات أو لغة التحكم بالبيانات DCL تؤكّد الحركة آليا committed وبنها تنتهي الحركة ضممتها. وعندما تنتهي حركة ما فإن جملة سُكُون التالية تبدأ تنفيذ الحركة التالية.

لجعل التغييرات دائمة على قاعدة البيانات لا بد من تأكيد هذه التغييرات وذلك باستخدام الأمر COMMIT. ولكن الأمر ROLLBACK يلغى تلك التغييرات قبل إصدار الأمر COMMIT. أي أنه بعد إصدار أمر مثل INSERT أو UPDATE أو DELETE فإنه يتم تخزين الصنفون التي جرت عليها التغييرات الناتجة من أحد تلك الأوامر مؤقتاً في مكان ما في قاعدة البيانات بانتظار صدور أحد الأمرين COMMIT لجعلها دائمة في الجدول أو صدور الأمر ROLLBACK الذي يلغيها. ولا تكون التغييرات متاحة visible للمستخدمين الآخرين - أي تعتبر غير موجودة بالنسبة لهم - حتى يتم تأكيد التغييرات.

كيفية إلغاء التغييرات غير المرغوبة

إن التغييرات التي لم يتم تأكيدها يمكن إلغاؤها باستخدام الأمر ROLLBACK. هذا الأمر يلغى كل التغييرات التي تمت بعد آخر عملية تأكيد COMMIT. أي أن التغييرات التي جرى لها عملية تأكيد سوف تغير البيانات وتكون متاحة للآخرين، أما التغييرات التي تتم بعد آخر عملية تأكيد ولكن لم تؤكّد فيمكن إهمالها باستعمال الأمر ROLLBACK. وعليه فإن للأمرين COMMIT و ROLLBACK الفوائد التالية:

١- التتحقق من توافقية البيانات Data Consistency

٢- مراجعة البيانات قبل تأكيدها وجعلها دائمة

٣- تجميع العمليات ذات العلاقة في مجموعات، مثل عمليات البيع في مجموعة وعمليات تحديث بيانات المستودع في مجموعة

آخر...

• فشل النظام System Failure

يتم إلغاء الحركة transaction آلياً عندما ينقطع تنفيذ العمليات التي يحتويها بسبب جدي أي خارج نطاق التحكم. فتقوم أوراكل آلياً بإلغاء أي حركة بصورة كاملة. وهذا يعني الحفاظ على سلامة البيانات من أية تغييرات غير مرغوبة والحافظة على تكاملية البيانات بين الجداول مما يقي قاعدة البيانات عموماً سليمة من المشاكل.

والذي يسبب إلغاء الآلي للحركات هو فشل النظام مثل نقصان التيار الكهربائي أو انقطاعه عن النظام أو قيام شخص ما بإطفاء جهاز الكمبيوتر وتشغيله مرة أخرى. أما الأخطاء الناتجة عن عدم إدخال الفواصل بين الأوامر أو الأخطاء الإملائية فإنها لا تسبب إلغاء الحركة آلياً، لأنها لم يتم تنفيذ الحمل أصلاً، حيث يتم تدقيقها أولاً ثم تنفيذها.

أهمية الحركة

إن التغييرات التي نرغب بإجرائها على البيانات يجب أن تكون متوافقة consistent بمعنى أنها تحقق التكاملية. فعلى سبيل المثال؛ عند إدخال قيد اليومية Journal Voucher لا يجوز أن تحتوي الحركة على أمر حفظ الجانب المدين في جدول القيد قبل أمر حفظ الجانب الدائن، بل يجب أن تحتوي الحركة على الأوامر التي تحتوي الجانبين معاً بحيث يتم تأكيدهما معاً committed أو إلغاؤهما معاً rolled back. ومثل إصدار فاتورة؛ فيجب أن تتضمن الحركة التغييرات التي تتم على جميع الجداول ذات العلاقة مثل جدول الأصناف والفوائير والعملاء وقيود اليومية بحيث يتم ضمان تأكيد التغييرات على جميع تلك الجداول مما يجعل بياناتنا سليمة.

ومتكاملة-متواقة. وفي حالة حدوث مشكلة في النظام كانقطاع التيار الكهربائي كما سبق؛ فإن أوراكل تلغى الحركة بكمالها-أي جميع التغييرات التي تحتويها- مما يرجع البيانات إلى حالتها الأصلية من التوافقية والتكمالية. وبهذا تتم السيطرة على البيانات وبنفس الوقت توجد مرونة كبيرة في عمليات صيانة البيانات.

ملاحظة: كما أسلفت سابقاً فيمكننا تجميع التغييرات على قاعدة البيانات في حركات منفصلة فمثلاً تخزين بيانات الفاتورة بتفاصيلها يمكن أن يتم من خلال حركة واحدة؛ بينما التأثير على جداول المستودع مثلاً أو العملاء يمكن جمعها في حركة واحدة لكل منها فتصدر الفاتورة مثلاً ويتم تأجيل التأثير على الجداول الأخرى من خلال برنامج صيانة خاصة.

❖ خصائص الحركات

يمكن تلخيص خصائص الحركات فيما يلي:

- يمكنك-كمستعمل أوراكل الحالي current user أن تشاهدها من خلال الاستعلام.
- لا يمكن للمستخدمين الآخرين مشاهدتها أو عرضها.
- يمكنك إلغاؤها.

في سُكُول يمكنك تأكيد COMMIT التغييرات آلياً باستعمال الخيار AUTOCOMMIT يجعله عاملاً ON أو إلغاء التغييرات بجعل الخيار لاغياً OFF وذلك بالأمر SET AUTOCOMMIT ON/OFF. في هذه الحالة يتم تأكيد التعديلات آلياً أو إلغاؤها. وفي حالة أن التأكيد آلياً يمكنك إلغاء التعديلات ما لم تترك سُكُول SQL. أما إذا خرحت من سُكُول SQL فمعنى ذلك أن التغييرات أصبحت مؤكدة-دائمة.

❖ التحكم في الحركات Control TRANSACTION

يتم التحكم في الحركات من خلال أوامر سُكُول التالية:

- أمر التأكيد COMMIT[WORK]
 - أمر نقطة الحفظ SAVEPOINT savepoint_name
 - أمر الإلغاء ROLLBACK[WORK]/ROLLBACK TO savepoint_name
- وهذا شرح مفصل لكل منها.

أمر التأكيد COMMIT[WORK]

والشكل العام له هو:

COMMIT[WORK];

حيث يقوم بما يلي:

1. جعل التغييرات في الحركة دائمة.

٢. إلغاء جميع نقاط الحفظ savepoints في الحركة- يأتي شرحها بعد قليل.
٣. إنتهاء الحركة.
٤. تحرير أية الجداول أو الصنوف من أية عمليات إغلاق ناجمة عن حركة ما.

ملاحظات:

١. الكلمة WORK اختيارية وتنكتب للمطابقة مع سُكّون القياسية standard SQL.
 ٢. يجب عليك تأكيد COMMIT أو إلغاء ROLLBACK التغييرات يدويا في البرنامج المكتوب بإحدى أدوات أوراكل مثل نماذج أوراكل OracleForms.
 ٣. يتم التأكيد آلياً في الحالات التالي:
 ٤. بعد تنفيذ أمر من أوامر لغة تعريف البيانات DDL.
 ٥. بعد تنفيذ أمر من أوامر لغة التحكم في البيانات DCL.
- دائما تقوم جمل تعريف البيانات بتأكيد التغييرات غير المؤكدة قبل تنفيذها منهية الحركة، ثم يتم تأكيدها هي نفسها بعد تنفيذها. أي أن هذه الأوامر توّكّد نفسها بنفسها.

أمر نقطة الحفظ SAVEPOINT

إذا كان لديك حركة طويلة أو معقدة-تحتوي عددا كبيراً من أوامر معالجة البيانات DML فمن الممكن تجزئه الحركة إلى عدة أجزاء بحيث ينجح تنفيذ تلك الأجزاء تباعاً أو إلغاء بعضها عند حدوث خطأ.

فمثلاً يمكن تقسيم حركة إصدار فاتورة إلى عدة أجزاء مثل حفظ بيانات الفاتورة في الجدول الرئيسي master table والجدول التفصيلي detail table؛ وجزءاً آخر لتعديل بيانات جدول الأصناف...الخ.

ولهذا المدف يمكن إنشاء عدة نقاط حفظ save points يتم تنفيذ كل جزء من الحركة على حدة وإلغاء أجزاء معينة عند نقطة حفظ معينة.

وإذا أنشأت نقطة حفظ معينة باسم ما، فلا تستعمل نفس الاسم لإنشاء نقطة أخرى لأن ذلك يؤدي إلى إلغاء الأخرى. ويمكن إنشاء حتى خمس نقاط حفظ.

أمر الإلغاء ROLLBACK

يستعمل أمر الإلغاء كما سبق لإلغاء جزء من الحركة إلى نقطة حفظ معينة أو إلغاء كل الحركة. وصيغته العامة كما يلي:

ROLLBACK [WORK] TO [SAVEPOINT *savepoint_name*];

حيث أن كلاً من WORK و SAVEPOINT اختياريان. ويتلخص عمل ROLLBACK بدون خيار نقطة الحفظ في:

١. إلغاء التغييرات في الحركة موضع التنفيذ.
٢. إنهاء الحركة.
٣. مسح جميع نقاط الحفظ.
٤. تحرير الجداول والصفوف من الإغلاق الناتج عن الحركة.

وفي حالة استعمال الخيار :SAVEPOINT

١. إلغاء جزء من الحركة حتى نقطة الحفظ.
٢. إلغاء جميع نقاط الحفظ بعد أمر الإلغاء والمحافظة على نقاط الحفظ السابقة لأمر الإلغاء.
٣. تحرير الجداول والصفوف من الإغلاق الناتج عن الحركة.

الإلغاء الضمني

تقوم أوراكل ضمنيا بإلغاء التغييرات الناشئة عن الحركة في حالة وجود قطع للحركة بصورة غير عادية abnormal مثل انقطاع التيار أو توقف أوراكل فجأة أو تعطل جهاز الكمبيوتر وغير ذلك.

فيما يلي مثال يوضح استعمال أمر الإلغاء مع نقاط الحفظ.

```
INSERT INTO departments
```

```
VALUES(50,'TESTING',103,1400);
```

```
SAVEPOINT insert_ok;
```

```
UPDATE departments
```

```
SET department_name = 'MARKETING'
```

```
WHERE...;
```

```
ROLLBACK TO insert_ok;
```

يتم إلغاء عمل أمر التعديل UPDATE

```
UPDATE departments
```

```
SET department_name = 'MARKETING'
```

```
WHERE department_name = 'SALES';
```

COMMIT;

خيار التأكيد الآلي : **AUTOCOMMIT**

في سُكُول يمكنك تأكيد COMMIT التغييرات آلياً باستعمال الخيار AUTOCOMMIT يجعله عاملاً ON أو يدوياً بجعل الخيار لاغياً OFF وذلك بالأمر SET AUTOCOMMIT ON/OFF. في هذه الحالة يتم تأكيد التعديلات آلياً أو يدوياً باستعمال الأمر COMMIT. وفي حالة أن التأكيد آلياً يمكنك إلغاء التعديلات ما لم تترك سُكُول SQL. أما إذا خرحت من سُكُول SQL فمعنى ذلك أن التغييرات أصبحت مؤكدة دائمة. وفي حالة أن الخيار لاغ OFF فيمكنك استعمال أمر التأكيد أو أمر الإلغاء يدوياً.

يجب أن نذكر أن الأمر SET AUTOCOMMIT ON/OFF هو من أوامر سُكُول + و sqlplus .

ويعمل من خلال الإعدادات preferences من sqlplus تفعيل أو إلغاء هذا الخيار

حالة البيانات قبل تأكيد **COMMIT** أو إلغاء **ROLLBACK** التغييرات

- يمكن استرجاع الحالة السابقة للبيانات
- يمكن للمستعمل الحالي الذي قام بتغيير البيانات مراجعتها بواسطة SELECT قبل تأكيدتها أو إلغائها
- لا يمكن للمستعملين الآخرين أن يشاهدو التغييرات التي أحدثتها مستعمل آخر باستعمال DML
- يتم إغلاق LOCK الصنوف التي خضعت للتغيير بواسطة مستعمل ما ولا يمكن للمستعملين الآخرين إحداث أي تغييرات عليها

فمثلاً لو أن المستعمل HR قام بإضافة صف على جدول الأقسام وتعديل بعض صنوف في جدول الموظفين وحذف صف من جدول آخر، فإن المستعمل HR هو الوحيدة قادر على مراجعة واسترجاع هذه التغييرات، أما المستعملون الآخرون فلا يرون هذه التغييرات. أضاف إلى ذلك أن المستعمل HR قد أغلق Locked الصنوف التي جرت عليها التغييرات؛ وعليه فإن المستعملين الآخرين لا يستطيعون إحداث أية تغييرات على الصنوف ذات العلاقة، حتى يقوم المستعمل HR بتحرير Release تلك الصنوف بتأكيد COMMIT أو إلغاء ROLLBACK التغييرات التي أحدثها.

وهنا يحق لنا التساؤل، ماذا لو قام مستعمل ما بتعديل وإجراء تغييرات على كامل الجدول؛ وتركه دون تأكيد أو إلغاء، وبقي الجدول مغلقاً بسبب إغلاق جميع الصنوف. فإن الآخرين سوف يتعطّلون فلا بد هنا من تدخل مدير قاعدة البيانات DBA حل مثل هذه المشاكل.

حالة البيانات بعد تأكيد COMMIT التغييرات

- ١- تصبح التغييرات دائمة
- ٢- الحالة السابقة للبيانات تنتهي بمعنى أن النسخة القديمة من الجداول لا تعود موجودة
- ٣- يستطيع جميع المستعملين(الذين لهم صلاحية) عرض جميع التغييرات التي تمت
- ٤- يتم تحرير release جميع الصفوف التي كانت مغلقة وبالتالي تكون متاحة لتعديلات أخرى بواسطة جميع المستعملين
- ٥- يتم مسح erase (إلغاء) جميع نقاط الحفظ savepoints

حالة البيانات بعد إلغاء ROLLBACK التغييرات

- ١- يتم إلغاء التغييرات
- ٢- يتم استرجاع الحالة السابقة للبيانات
- ٣- يتم تحرير الصفوف ذات العلاقة

مثال ذلك:

```
DELETE FROM copy_emp;
```

```
22 rows deleted.
```

```
ROLLBACK;
```

```
Rollback complete.
```

في المثال التالي ارتكبنا خطأ بمسح جميع الصفوف من الجدول test ولكن لم نؤكّد عملية المسح باستعمال COMMIT؛ فأمامنا فرصة للتراجع ROLLBACK وسوف نمسح صفا واحدا فقط هو المطلوب مسحه وتأكّد COMMIT المسح!

```
DELETE FROM test;
```

```
rows deleted. 25,000
```

```
ROLLBACK;
```

```
Rollback complete.
```

```
DELETE FROM test WHERE id = 100;
```

```
1 row deleted.
```

```
SELECT * FROM test WHERE id = 100;
```

```
No rows selected.
```

COMMIT;

Commit complete.

الإلغاء **ROLLBACK** على مستوى الجملة

سبق وان قلنا أن الحركة ممكن أن تكون سلسلة من جمل لغة معالجة البيانات DML. وقد تكون هذه السلسلة أو المجموعة في ملف SQL تقوم بتنفيذها دفعه واحدة كما مرة سابقا. ولكن قد تفشل بعض الجمل فيه لسبب أو آخر كما مر معنا في موضوع إضافة أو إلغاء أو تعدي صفوف في جدول.

ما يحدث هو أن الجملة التي فشلت لا تؤثر على الجمل التالية لها وبديهي على السابقة عنها. أي يتم تنفيذ الجملة التي لا تختلف أي قاعدة من قواعد سُكُولْ والتي تخالف قواعد معالجة البيانات يتم إلغائها Rolled Back آلياً بواسطة أوراكل. ومسؤوليتها أن تؤكد أو تلغى يدوياً-التغييرات التي أحدثتها الجمل الصحيحة وذلك بـ COMMIT أو ROLLBACK.

توافقية البيانات **Data Consistency**

- نذكر أن المستعملين يستطيعون الوصول إلى قاعدة البيانات بطريقتين
أ. بقراءة البيانات بواسطة جملة الاستعلام SELECT
- ب. بكتابة البيانات على الجداول بواسطة جمل INSERT, UPDATE, DELETE
- زيد ضمان توافقية البيانات بحيث:
 - أ. كل من قارئ Reader وكاتب Writer قاعدة البيانات Data Base يريدون ضمان نسخة متوافقة من البيانات
 - ب. القارئون Readers لا يستطيعون مشاهدة البيانات التي تحت التغيير، بل يرون النسخة القديمة منها
 - ت. الكاتبون Writer يريدون التأكد من أن التغييرات التي أحدثوها تمت بصورة متوافقة
 - ث. التغييرات التي يحدثها أحد المستعملين لا تتعارض مع تلك التي يحدثها المستعملون الآخرون
 - ج. القارئون لا ينتظرون الكاتبين والعكس صحيح
- ح. لا يوجد تداخل بين كاتب وكاتب؛ فلو بدأ مستعمل بتغيير البيانات في صف فإن أي مستعمل آخر لا يستطيع ذلك على نفس الصف حتى يتم تحرير الصف.

إن الهدف من توافقية البيانات هو التأكد من أن المستعمل يستطيع مشاهدة آخر نسخة للبيانات قد تم تأكيدها بواسطة آخر COMMIT وقبل إجراء أي تغيير عليها

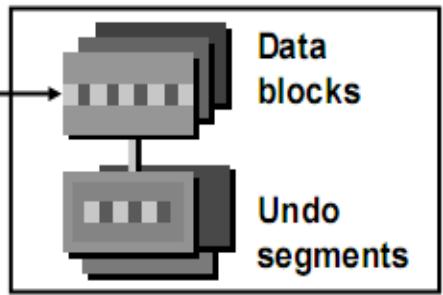
إن النسخة القديمة للصفوف التي يجري عليها التغيير أو تلك التي تتم إضافتها، تخزن في مكان في قاعدة البيانات اسمه القطعة المؤقتة أو قطعة التراجع Undo Segment. وعليه فإن المستعملين الآخرين يرون النسخة القديمة من الصدف من هذه المساحة.

في الشكل التالي، نرى أن المستعمل A يعدل راتب للموظف Grant. تقوم أوراكل بتحزين النسخة القديمة من الراتب في القطعة المؤقتة، وعليه فإن المستعمل B لا يستطيع أن يشاهد التغيير الذي أحدثه المستعمل A. بل يشاهد النسخة القديمة الموجودة في القطعة المؤقتة

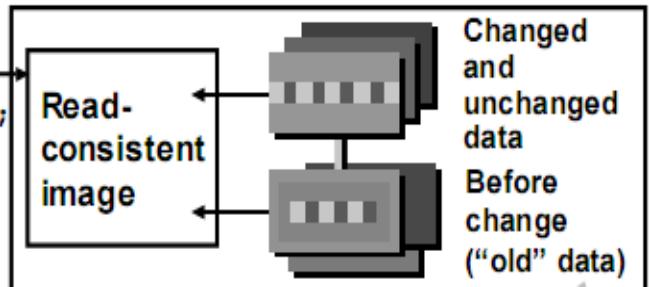
User A



```
UPDATE employees  
SET salary = 7000  
WHERE last_name = 'Grant';
```



```
SELECT *  
FROM userA.employees;
```



User B

هيكل البيانات في أوراكل ORACLE Data Structure

لغة هيكل البيانات Data Definition Language

لغة هيكل البيانات في أوراكل هي مجموعة جزئية من أوامر سُكُّول تسمى لغة تعريف البيانات Data Definition Language(DDL) وهذه الأوامر هي:

- ❖ أمر الإنشاء **CREATE** ووظيفته إنشاء أي مكون Object من مكونات قاعدة البيانات مثل الجداول Tables والفهارس Indexes.
- ❖ أمر التغيير **ALTER** وظيفته تغيير تعريف أحد مكونات قاعدة البيانات مثل إضافة عمود لجدول أو تغيير طول عمود في جدول.
- ❖ أمر الإلغاء أو الescاط **DROP** وظيفته إلغاء نهائياً لأحد مكونات قاعدة البيانات مثل إلغاء فهرس مثلا.
- ❖ أمر إعادة التسمية **RENAME** وظيفته إعادة تسمية جدول ما.
- ❖ أمر البت **TRUNCATE** لحذف بيانات جدول ما.

وستعرض لهذه الأوامر بشيء من التفصيل بعد استعراض بعض أنواع البيانات المهمة في أوراكل.

أنواع البيانات في أوراكل Data Types

ستعرض لأهم أنواع البيانات في أوراكل والتي تُستخدم غالباً وهذه الأنواع هي:

١ - النوع رمز: أي سلسلة من الرموز (حروف وأرقام وأية رموز أخرى) وشكلها هو **CHAR(w)** حيث **w** تمثل عدداً ثابتاً من الرموز بحد أقصى ٢٠٠٠ رمز. وهذا يعني أنه لو كانت **w = ٥** فإن عدد الرموز الذي سيتم تخزينه فعلاً هو ٥ بغض النظر عن العدد الحقيقي للرموز المدخلة وهذا يعني أن **w** تمثل عدداً ثابتاً من الرموز. والقيمة الافتراضية **Default** لـ **w** هي ١ بحيث يمكن إعطاؤها ١ أو تركها بدون قيمة فتؤول إلى ١

٢ - النوع رمز متغير(**w**): **VARCHAR2(w)** حيث تمثل **w** عدداً متغيراً بحد أقصى ٤٠٠٠ رمز. وهذا يعني أنه يتم تخزين العدد الحقيقي من الرموز؛ فلو كانت **w = ٥** مثلاً وعدد الرموز المدخلة فعلاً هو ٢٠ فإنه يتم تخزين أول ٢٠ رمزاً فقط. يجب تحديد قيمة **w** أي لا يوجد قيمة افتراضية.

ملاحظة مهمة: لا يوجد في أوراكل ٦ وما قبلها؛ النوع **CHAR** وتم الحفاظ على النوع **VARCHAR2** لأغراض التوافق مع النسخ القديمة.

٣- النوع عددي: **NUMBER(p,s)** هذا النوع هو أحد الأشكال التي يتم بواسطتها تخزين البيانات العددية في قاعدة بيانات أوراكل. حيث تمثل **p** (**Precision**) العدد الكلي للأرقام بحد أقصى ٣٨ رقما بينما تمثل **s** عدد الأرقام على يمين الفاصلة العشرية بحدود تتراوح بين -٨٤ و ١٢٧؛ حيث يعني العدد السالب تقريب العدد إلى عدد الخانات العشرية الذي يمثله العدد السالب؛ فمثلا **NUMBER(-3,10)** يعني تقريب العدد إلى خانة الآلاف.

٤. النوع تاريخ **DATE**: ويستعمل لتخزين التواريخ؛ ويتراوح بين ١٢/٣١/٩٩٩٩ و ٤٧١٢/١/١ ق.م. إلى ٤٦٧١٢/١/١ ق.م.

٥. النوع طويل **LONG**: هذا النوع يستعمل لتخزين النصوص الطويلة بحد أقصى ٢ جيجابايت من الرموز. وهذا النوع يحد من استعماله إمكانيات الكمبيوتر المستعمل مثل الذاكرة والقرص الصلب.

٦. النوع **CLOB** كائن رمزي ضخم **Character Large Object** ويشبه النوع طويل **LONG** من حيث أن يستعمل لتخزين النصوص الكبيرة ولكنه بحد أقصى ٤ جيجابايت، وهناك فروق أخرى (انظر الملاحظات)

٧. النوع خام **RAW(w)**: يستعمل لتخزين البيانات على الشكل الثنائي **Binary** (بيانات خام) مثل الصور والرسوم البيانية والأصوات وخلافها وبحد أقصى ٢٠٠٠ بايت. يجب إعطاء قيمة لـ **w**.

٨. النوع خام طويل **LONG RAW**: مشابه لنوع خام **RAW** السابق غير أن حجمه يصل إلى ٢ جيجابايت كحد أقصى.

٩. النوع كائن خام ضخم **BLOB-Binary Large Object**: وله نفس خصائص النوع خام طويل **LONG RAW** غير أن حجمه يصل إلى ٤ جيجابايت كحد أقصى وفروق أخرى (انظر الملاحظات)

١٠. النوع ملف ثانئي **BFILE – Binary File**: أيضا يستخدم لتخزين البيانات الثنائية **Binary Data** ولكن على شكل ملف خارجي أي خارج قاعدة البيانات. وسيتم بحث هذه النقطة.

١١. النوع هوية الصف **RWOID**: وهذا النوع يمثل عنوان الصف في الجدول، وليس هنا مجال البحث فيه.

هذه هي أهم أنواع البيانات في أوراكل؛ وهناك أنواع أخرى ليس هنا مجال بحث عنها.

ملاحظات

١. عند إنشاء جدول بواسطة الاستعلام لا يتمأخذ العمود من نوع طويل **LONG** سواءً أكان نصيا أم ثنائيا **binary**

٢. لا نستطيع استعمال النوع طويل **LONG** ضمن عبارتي **ORDER BY** أو **GROUP BY**

٣. لا يجوز تعريف أكثر من عمود من نوع طويل **LONG** في الجدول الواحد-أي في نفس الجدول.

٤. لا يمكن إنشاء قيود **constraints** على العمود من نوع طويل **LONG**

٥. يستحسن استعمال أعمدة من نوع **LOB** كائن ضخم مثل **BLOB** و **CLOB** بدلاً من النوع طويل **LONG** لتفادي بعض الملاحظات السابقة مثل الملاحظة رقم ٣.

أنواع أخرى متقدمة للبيانات تم تقديمها اعتباراً من نسخة أوراكل 9i آي Oracle9i. وهي:

١. النوع؛ دمجة الوقت **TIMESTAMP**: وهذا النوع يشتمل على أجزاء الثانية، ولا يقتصر على الثانية فقط كما في النوع تاريخ **DATE**.
 ٢. النوع؛ الفترة سنة شهر **INTERVAL YEAR TO MONTH** هكذا تماماً وهنا يتم تخزين بيانات التاريخ على شكل سنوات وأشهر فقط بدلاً من تضمين الأيام وال ساعات والثواني وأجزاء الثانية.
 ٣. النوع؛ يوم ثانية **INTERVAL DAY TO SECOND** هكذا تماماً، وفيه يتم تخزين التاريخ على شكل أيام و ساعات و دقائق و ثواني وأجزاء الثانية.
- إن النوع دمجة الوقت **TIMESTAMP** جديد و متطور كما أسلفنا، ويستعمل لتخزين السنة والشهر واليوم والساعة والدقيقة والثانية وأجزاء الثانية.

وهذا النوع يمكن إضافة المنطقة الزمنية إليه **Time Zone** بحيث يتم تخزين البيانات حسب المنطقة الزمنية حتى لو لم نكن فيها. لو كنتم في السعودية يمكن تخزين بيانات التاريخ والوقت حسب منطقة اليابان مثلاً. وأمثلة على ذلك:

TIMESTAMP[fractional_seconds_precision]

نحدد هنا دقة أجزاء الثانية؛ مثلاً ٧ خانات لجزء الثانية وبحد أقصى ٩ خانات. وعدم التحديد يعني القيمة الافتراضية وهي ٦ خانات من أجزاء الثانية وبحد أدنى صفر خانة.

للتوسيع: ٠٠٠٩٨٧٦٥٤٣٢١٢٣٤٥٦٧٨٩ جزء من الثانية كحد أعلى لعدد الخانات يعين الفاصلة ٩ خانات.

مثال آخر يتضمن المنطقة الزمنية:

TIMESTAMP[fractional_seconds_precision] WITH TIME ZONE

TIMESTAMP[fractional_seconds_precision] WITH LOCAL TIME ZONE

وسوف يتم تفصيل ذلك مع الأمثلة عند شرح كيفية إنشاء الجداول.

هيكل البيانات في أوراكل ORACLE Data Structure

قواعد تسمية المكونات في أوراكل Objects Naming Rules

ت تكون قاعدة البيانات في أوراكل من مجموعة مكونات يطلق عليها اسم أشياء **Objects** مثل الجداول **Tables** والفهارس **Indexes** والأعمدة **Columns** ... الخ. ولهذه المكونات أسماء تخضع للقواعد التالية:

١. يجب أن يبدأ الاسم بحرف.
٢. لا يزيد طول الاسم عن ٣٠ رمزا.
٣. لا يوجد فراغات بين رموز الاسم.
٤. لا يجوز استعمال الكلمات المحفوظة لأوراكل Reserved Words مثل كلمة **SELECT**.
٥. لا يجوز استعمال الاسم أكثر من مرة في نفس قاعدة البيانات لنفس المستعمل - مخطط البيانات schema.
٦. يجوز استعمال الرمزين \$, # في الأسماء.

مثال: أسماء مقبولة

mytable, your_index, s123, s_\$2a

مثال: أسماء غير مقبولة

_mytable, \$yorindex, 1se, emp names, table

والآن باستطاعتنا التعامل مع لغة تعريف هيكل البيانات بعد التعرف على أنواع البيانات وقواعد التسمية؛ ولنبدأ باسم الله بجملة إنشاء الجداول **CREATE TABLE Statement**

أمر الإنشاء CREATE

كما أسلفنا فإن هذا الأمر يستعمل لإنشاء مكونات قاعدة البيانات مثل الجداول.

أولاً: إنشاء جدول بالتعريف المباشر والصيغة العامة لذلك هي:

CREATE TABLE [schema.]table_name

(column datatype [DEFAULT expr][,...]);

حيث :

اسم المستعمل الذي يملك الجدول - سيتم التطرق إلى مفهوم المخطط *schema* لاحقا

أي اسم جدول يخضع لقواعد التسمية السابق ذكرها.

اسم أول عمود في الجدول أيضاً يخضع لقواعد التسمية السابق ذكرها.

نوع العمود وحجمه.

DEFALT تخصيص قيمة افتراضية (expr) للعمود باستعمال عبارة **DEFALT expr**

لاحظ أن الأمر يبدأ بقوس وينتهي بقوس مقابل.

والمستعمل الذي يريد إنشاء جداول عليه أن يحصل على صلاحية إنشاء الجداول ومساحة مخصصة له Quota كافية لهذا الغرض. (سبم ذكر ذلك لاحقا)

مثال:

لإنشاء الجدول dept الذي يحتوي بعض بيانات الأقسام في شركة ما نكتب ما يلي:

```
CREATE TABLE dept(  
    dept# NUMBER(3)      NOT NULL,  
    dept_name  VARCHAR2(20) NOT NULL,  
    dept_loc  VARCHAR2(30));
```

Table created.

لاحظ أن كل من رقم القسم dept# واسمه dept_name إجباريان بينما الموقع dept_loc غير إجباري.

ولعرض تعريف الجدول (هيكل البيانات) dept نستعمل الأمر DESCRIBE والذى يمكننا استعمال أول أربعة حروف منه DESC كما يلى:

```
DESC dept
```

Name	Null?	Type
------	-------	------

```
-----
```

DEPT#	NOT NULL	NUMBER(3)
-------	----------	-----------

DEPT_NAME	NOT NULL	VARCHAR2(20)
-----------	----------	--------------

DEPT_LOC	VARCCHAR2(30)
----------	---------------

ويمكننا استعمال الأمر DESC لعرض هيكل البيانات لأي جدول. والأمر DESCRIBE هو من أوامر سكول+ SQLPlus وليس من أوامر سكول

الخيار افتراضي *DEFAULT Option*

يمكن إضافة الخيار DEFAULT إلى تعريف العمود في الجدول متبعاً بقيمة معينة تخصيصاً لذل العمود في حالة عدم تحديد قيمة معينة له.

مثال:

Hiredate DATE DEFAULT SYSDATE,

sal NUMBER(7,2) DEFAULT 0

في هذا المثال تم تعين تاريخ النظام SYSTDATE كقيمة افتراضية للعمود hiredate والتي يمكن تغييرها حسب رغبة المستعمل. ونفس الشيء ينطبق على العمود sal حيث عينا الصفر كقيمة افتراضية له.

أما القيم التي يمكن استعمالها كقيم افتراضياً مع الخيار *DEFAULT* فهي كما يلي:

١. قيم ثابتة Literal values (راجع بند الإضافات Expressions)
٢. تعبيرات SQL Functions
٣. دوال سكول SQL Functions

أما القيم التي لا يمكن استعمالها كقيم افتراضية مع الخيار *DEFAULT* فهي كما يلي:

١. عمود آخر في الجدول
 ٢. الأعمدة الوهمية ROWNUM مثل pseudo columns
- يجب أن يتطابق نوع البيانات المعطاة في *DEFAULT* مع نوع بيانات العمود المرغوب تحديده قيمة افتراضية له

* إنشاء جداول تعتمد على نوع البيانات دمجة الوقت TIMESTAMP

سنقوم بإنشاء جداول تحتوي على النوع دمجة الوقت كما يلي:

CREATE TABLE new_employees

(employee_id NUMBER,

first_name VARCHAR2(15),

last_name VARCHAR2(15),

```
start_date TIMESTAMP(7),
```

```
...);
```

في هذه المثال لدينا جدول يحتوي على العمود start_date من نوع دمجة الوقت بدقة ٧ خانات لأجزاء الثانية، وعليه يمكننا استخراج بيانات الجدول كما يلي:

```
SELECT start_date
```

```
FROM new_employees;
```

```
17-JUN-03 12.00.00.0000000 AM
```

```
21-SEP-03 12.00.00.0000000 AM
```

بافتراض أن البيانات تم إدخالها الساعة ١٢ ليلا بالضبط!

وعكن ضمن المنطقة الزمنية TIME ZONE والتي تحدد فرق الساعات والدقائق بين التوقيت المحلي والإحداثيات الدولية Greenwich Mean Time (GMT) المعروفة اصطلاحاً بتوقيت غرينتش Universal Time Coordinate (UTC) فمثلاً

```
TIMESTAMP '2003-04-15 8:00:00 -8:00'
```

-٨:٠٠ يعني بعد غرينتش بـ ٨ ساعات. أما:

```
TIMESTAMP '2003-04-15 8:00:00 US/Pacific'
```

تعني التوقيت الشرقي للولايات المتحدة وهذا:

```
TIMESTAMP '2003-04-15 8:00:00 +3:00'
```

يعني توقيت المملكة العربية السعودية والدول الذي يبدأ قبل غرينتش بـ ٣ ساعات.

و الخيار آخر هو تضمين المنطقة الزمنية المحلية في تعريف البيانات فمثلاً

```
CREATE TABLE time_example
```

```
(order_date TIMESTAMP WITH LOCAL TIME ZONE);
```

فهنا يتم تخزين الوقت حسب التوقيت المحلي وهو المأخوذ من نظام التشغيل (لا ننسى أن التوقيت في نظام التشغيل قد يختلف عن التوقيت الحقيقي؛ فقد نركب نظام التشغيل ونحن في المملكة العربية السعودية على أنه توقيت أي بلد آخر وهنا LOCAL TIME ZONE سيكون للبلد الآخر وليس للمملكة العربية السعودية). ونطبق المثال السابق كما يلي:

```
INSERT INTO time_example
```

```
VALUES('15-JAN-04 09:34:28 AM');
```

```
SELECT * FROM time_example;
```

```
ORDER_DATE
```

```
-----  
15-JAN-04 09.34.28.000000 AM
```

إن فكرة دمجة الوقت تمكّنا من التعامل مع البيانات في مناطق زمنية مختلفة وهذا يفيد المؤسسات التي تنتشر في مناطق زمنية مختلفة

 النوع سنة إلى شهر YEAR TO MONTH والشكل العام له هو:

INTERVAL YEAR [(year_precision)] TO MONTH

حيث year_prcedion اختياري يمثل عدد الأرقام الصحيحة والعدد الافتراضي هو ٢ . وكمثال على ذلك:

- INTERVAL '123-2' YEAR(3) TO MONTH

يعني ١٢٣ سنة وشهرين.

- INTERVAL '123' YEAR(3)

يعني ١٢٣ سنة فقط(لا يوجد أي شهر).

- INTERVAL '300' MONTH(3)

يعني ٣٠٠ شهر

- INTERVAL '123' YEAR

سوف يؤدي إلى خطأ لأن عدد الأرقام الافتراضي هو رقمان فقط والمعطى فترة تتكون من ٣ أرقام.

```
CREATE TABLE time_example2
```

```
(loan_duration INTERVAL YEAR (3) TO MONTH);
```

```
INSERT INTO time_example2(loan_duration)
```

```
VALUES(INTERVAL '120' MONTH(3));
```

```
SELECT
```

```
TO_CHAR(sysdate+loan_duration,'dd-mon-yyyy')
```

```
FROM time_example2;
```

TO_CHAR(SYSDATE+LOAN_DURATION,'DD
15-jul-2020

نوع يوم إلى ثانية DAY TO SECOND والشكل العام له هو:

INTERVAL DAY(day_precision) TO SECOND(fractional_seconds_precision)

حيث أن day_precision يمثل عدد الأرقام التي تشمل الأيام من ٠ إلى ٩ خانات وبعدد افتراضي رقمان.

وأما fractional_seconds_precision فيمثل عدد الخانات العشرية من جزء من الثانية من ٠ إلى ٩ خانات عشرية وعدد افتراضي ٦ خانات عشرية. وكأمثلة:

INTERVAL '4 5:12:10.216' DAY TO SECOND(3)

يعني ٤ أيام و ٥ ساعات و ١٢ دقيقة و ١٠ ثوان و مائتين و سته عشر جزءاً من الألف من الثانية.

INTERVAL '180' DAY(3)

يعني ١٨٠ يوماً.

INTERVAL '4 5:12' DAY TO MINUTE

يعني ٤ أيام و ٥ ساعات و ١٢ دقيقة.

INTERVAL '400 5' DAY(3) TO HOUR

يعني ٤٠٠ يوم و ٥ ساعات.

INTERVAL '11:12:10.2615378' HOUR TO SECOND(7)

يعني ١١ ساعة و ١٢ دقيقة و ١٠ ثوان و ٢٠٦١٥.٣٧٨ من ١٠،٠٠٠،٠٠٠ من الثانية.

مثال على جدول باستعمال النوع يوم إلى ثانية.

CREATE TABLE time_example3

(day_duration INTERVAL DAY (3) TO SECOND);

INSERT INTO time_example3 (day_duration)

```
VALUES (INTERVAL '180' DAY(3));
```

```
SELECT sysdate + day_duration "Half Year"
```

```
FROM time_example3;
```

٢٠١٠ يوم ١٥ يوليو تاريخ

Half Year
11-JAN-11

هيكل البيانات في أوراكل ORACLE Data Structure

قيود السلامة (التكامل) Integrity Constraints

المقصود بقيود السلامة(التكامل) هو ما يضعه المستعمل من قواعد أثناء إنشاء الجدول بغرض الحفاظة على سلامية البيانات وضمان إدخالها أو تعديلها أو حذفها بصورة صحيحة وتأكيد صحة العلاقات بين الجداول. وقد مر معنا من هذه القيود أن يكون العمود إجبارياً NOT NULL على سبيل المثال.

وستعمل قيود السلامة (التكامل) كما يلي:

- + يقوم خادم أوراكل Oracle Server باستعمال قيود السلامة بغرض تحقيق القواعد التي تم وضعها من خلال هذه القيود على مستوى الجدول أثناء عمليات إدخال وتعديل وحذف الصفوف.
- + منع عملية إلغاء الجدول أو حذف صفوف منه في حالة وجود جداول أخرى تعتمد عليه.
- + يتم استعمال قيود السلامة بواسطة أدوات أوراكل الأخرى مثل نماذج أوراكل Oracle Forms لإنشاء قطع من البرامج لتحقيق القواعد الناتجة عن تلك القيود مما يوفر جهداً ووقتاً كبيرين على المبرمج.

(وكقاعدة عامة؛ كلما كانت القيود واضحة وكافية أثناء تصميم وإنشاء مكونات قاعدة البيانات عموماً والجداول خصوصاً؛ فإنه يقصر وقت إنجاز البرنامج؛ لأن أوراكل تتكفل بكتابة الأجزاء الضرورية من البرنامج التي تحقق تلك القواعد. وهذا ما سنراه عند بحث نماذج أوراكل كمثال إذا تيسر ذلك إن شاء الله).

وعن تقسيم قيود السلامة(التكامل) إلى صفين هما:

- القيود على مستوى الجدول وتعامل مع عمود أو أكثر. وتوضع منفصلة عن تعريف الأعمدة-أي في آخر تعريف الجدول.

هذا ويمكن إضافة القيود على الجدول بعد إنشائه؛ ويمكن تعطيلها مؤقتاً DISABLE وتنشيطها ENABLE كما سيأتي عند بحث أمر تغيير تعريف الجدول ALTER TABLE.

- القيود على مستوى العمود: وهذا يتم تعريف القيد على عمود واحد فقط أثناء تحديد مواصفاته.

تقوم أوراكل بإعطاء أي قيد اسمه يبدأ SYS_Cn حيث تمثل n رقمًا وحيدًا؛ ويتم تخزين جميع تفاصيل القيد في قاموس بيانات Oracle Data Dictionary. ومن الأفضل لك أن تقوم بتسمية القيد بأسماء مناسبة ذات معنى ويسهل استذكارها باستعمال الكلمة CONSTAINT كم في المثال التالي:

```
deptno NUMBER CONSTRAINT not_null NOT NULL;
```

حيث not_null اسم القيد و NOT NULL نوع القيد

أنواع قيود السلامة

تنقسم قيود السلامة إلى ما يلي:

- ١- الإجبارية .NOT NULL
- ٢- الوحدة .UNIQUNESS
- ٣- المفتاح الأساسي .PRIMARY KEY
- ٤- المفتاح الأجنبي FOREIGN KEY أو مرجعية التكامل Referential Integrity
- ٥- التدقيق .CHECK

و سنبحث هذه القيود الآن بالتفصيل.

١- الإجبارية NOT NULL

هذا القيد يعني وجوب أن يحتوي العمود على بيانات أثناء عمليات إضافة صفات جديدة على الجدول أو تعديل بيانات على الجدول. تظهر رسالة خطأ إذا لم يقوم المستخدم بإدخال قيمة للعمود المقصود ويقوم خادم أوراكل بإلغاء الحركة Transaction

٢- الوحدة UNIQUNESS

ويعني أن هناك عموداً أو تجتمعاً من أكثر من عمود تمثل مفتاحاً وحيداً UNIQUE KEY بحيث لا يوجد أكثر من صف يحمل قيمة أو قيمة للأعمدة التي تمثل المفتاح. ويتم تعريف هذا القيد على مستوى العمود إن كان المفتاح عموداً واحداً؛ وعلى مستوى الجدول إن كان المفتاح أكثر من عمود. والشكل العام هو:

• على مستوى العمود

```
[CONSTRAINT constraint_name] UNIQUE
```

حيث تشير الأقواس المربعة إلى أن هذا الخيار ليس إجبارياً.

• على مستوى الجدول

```
,CONSTRAINT constraint_name]UIQUE(col1,col2,...)
```

حيث تشير الأقواس المربعة إلى أن هذا الخيار غير إجباري. كما أن الفاصلة تسبق تعريف القيد وكأنه عمود.

مثال: سوف تقوم بإنشاء الجدول dept_loc والذي فيه نضع قياداً على كل من اسم fname وموقع loc القسم بحيث لا يكون اسم القسم موجوداً في نفس الموقع مرتين كما يلي:

```
CREATE TABLE dept_loc
(deptno NUMBER,
fname      VARCHAR2(10),
loc        VARCHAR2(10),
CONSTRAINT unique_dept_loc UNIQUE (fname,loc));
```

لا تسمح أوراكل هنا بتكرار نفس القسم في نفس الموقع. كما أنها تنشأ فهراً وحيداً لمعالجة هذه القاعدة. (سيتم بحث الفهرس لاحقاً).

٣- المفتاح الأساسي PRIMARY KEY

والغرض منه نفس الغرض من المفتاح الوحد **UNIQUE KEY** حيث تقوم أوراكل أيضاً بإنشاء فهرس لهذا الغرض.

والفرق بين المفتاح الأساسي والمفتاح الوحد هو أنه لا يجوز أن يكون هناك أكثر من مفتاح أساسي للجدول؛ كما لا يجوز أن تتحمل الأعمدة المكونة للمفتاح الأساسي قيمة اللاشيء **NULL VALUES**. ويعرف المفتاح الأساسي بأنه المفتاح المحدد الذي يعرف وحدوية الصفوف. كما أن المفتاح الأساسي يستخدم كمفتاح أجنبي **Foreign Key** لجدول **definitive key** آخر.

أما المفتاح الوحد فيمكن أن يكون أكثر من مفتاح وحيد للجدول ويمكن السماح بقيمة اللاشيء.

ملاحظة: لا يجوز استعمال نفس الأعمدة في تعريف المفتاح الأساسي والمفتاح الوحد.

والشكل العام هو:

على مستوى العمود

[CONSTRAINT *constraint_name*] PRIMARY KEY

حيث تشير الأقواس المربعة إلى أن هذا القيد اختياري.

على مستوى الجدول

, [CONSTRAINT *constraint_name*] PRIMARY KEY (col1,col2,...)

حيث تشير الأقواس المربعة إلى أن هذا الخيار غير إجباري. كما أن الفاصلة تسبق تعريف القيد وكأنه عمود.

المثال التالي ينشأ الجدول emp بتعريف مفتاح اساسي على مستوى العمود empno كما يلي:

```
CREATE TABLE emp  
(empno NUMBER(4) CONSTARINT dept_prim PRIMARY KEY,...)
```

٤ - المفتاح الأجنبي FOREIGN KEY

المفتاح الأجنبي يعني أن جدولًا ما يحتوي عموداً يستمد قيمته من جدول آخر.

ويُستعمل المفتاح الأجنبي بين الجداول لتأكيد تكاملية البيانات وعدم السماح بحذف البيانات من جدول تعتمد عليها بيانات في جدول آخر، أو عدم إضافة صفوف إلى جدول ما ليس له مرجع في جدول آخر يعتمد عليه.

ولتوسيع الأمر سوف ندرس كلاً من جدولي الأقسام وجدول الموظفين *deptno*. كلاً الجدولين فيها العمود *deptno*؛ فنقول إن جدول الموظفين-ويُعرف بالابن *child*- أو التفصيل *detail*؛ يحتوي العمود *deptno* كمفتاح أجنبي REFERENCED TABLE FORGIEN KEY بينما جدول الأقسام-ويُعرف بالأب *parent* أو الجدول المرجع *master*- يحتوي العمود *deptno* كمفتاح اساسي- وعلى العموم؛ يجب أن يكون المفتاح الأجنبي مفتاحاً اساسياً في الجدول المرجع REFERENCED TABLE حيث أنه لا يمكن إضافة صف إلى جدول الموظفين لا يحتوي على رقم قسم غير موجود في جدول الأقسام. وبالمقابل لا يمكن حذف صف من جدول الأقسام يحتوي على رقم قسم موجود في جدول الموظفين أي منع حذف قسم يوجد فيه موظفون.

والشكل العام هو كما يلي:

على مستوى العمود

```
[CONSTRAINT constraint_name] FOREIGN KEY REFERENCES table (column)
```

حيث تشير الأقواس المربعة إلى أن هذا الخيار غير إجباري.

على مستوى الجدول

```
,[CONSTRAINT constraint_name] FOREIGN KEY (col1,col2,...) REFERENCES  
foreign_table(col1,col2,...)
```

حيث تشير الأقواس المربعة إلى أن هذا الخيار غير إجباري. كما أن الفاصلة تسبق تعريف القيد وكأنه عمود.

لاحظ أن عبارة FOREIGN KEY لا تُستعمل في تعريف القيد على مستوى العمود إذا كان مكوناً من أكثر من عمود أي إذا كان مركباً Composite.

- لا يتم تخزين قيمة العمود الأجنبي في الجدول الابن وإنما يعرفه خادم أوراكل بواسطة مؤشر pointer وإن كنا نراها في الجدول الابن كبيانات عادية!

المثال التالي ينشأ الجدول emp بتعريف مفتاح أجنبي على مستوى العمود deptno كما يلي:

```
CREATE TABLE emp
```

```
(deptno NUMBER(2)CONSTRAINT foreign_deptno FOREIGN KEY(deptno)
REFERENCES dept(deptno),...)
```

❖ خيارات المفتاح الأجنبي:

- ١ خيار مسح الأصل والتفاصيل معا ON DELETE CASCADE Option

نضطر أحياناً لحذف الصف الأصلي والصفوف الفرعية معاً بدلاً من حذف كل صف فرعى على حدة ثم أخيراً حذف الصف الأصلي. وقد تكون هذه الحاجة في حالة إنشاء فاتورة عرض أسعار مثلاً وأردنا إلغاء هذه الفاتورة؛ فمن الأفضل إلغاء الأصل والتفاصيل معاً. وهذا ما يوفره لنا الخيار ON DELETE CASCADE عند إنشاء القيد في الجدول ابن؛ وذلك كما يلى بإعادة كتابة المثال السابق:

```
CREATE TABLE emp
```

```
(deptno NUMBER(2),
CONSTRAINT foreign_deptno FOREIGN KEY(deptno) REFERENCES dept(deptno)
ON DELETE CASCADE,...)
```

- ٢ خيار مسح الأصل والإبقاء على التفاصيل ON DELETE SET NULL Option

ونضطر أحياناً أخرى لحذف الصف الأصلي والمحافظة على الصحف الفرعية. وفي هذه الحالة لا يجوز أن يكون للمفتاح الأجنبي قيمة في الجدول ابن بسبب حذف المفتاح الأب أي يجب أن نحوله إلى لاشيء NULL. وفي هذه الحالة نختار أن نقوم بذلك باستعمال جملة التعديل UPDATE وستكون هذه العملية كما مر معنا بدلاً من حذف كل صف فرعى على حدة ثم أخيراً حذف لصف الأصلي. وقد تكون مرهقة وطويلة. ولذلك وفرت أوراكل خياراً آخر هو تحويل قيمة العمود الأجنبي إلى لاشيء آلياً. ولذلك نضيف الخيار ON DELETE SET NULL عند إنشاء القيد في الجدول ابن؛ وذلك كما يلى بإعادة كتابة المثال السابق:

```
CREATE TABLE emp
```

```
(deptno NUMBER(2),
CONSTRAINT foreign_deptno FOREIGN KEY(deptno) REFERENCES dept(deptno)
ON DELETE SET NULL,...)
```

• الجدول كمرجع لنفسه Referencing table to itself

هل يمكن أن يكون الجدول مرجعاً لنفسه؟ ويحتوي عموداً أجنبياً فيه مرجعه نفسه؟! نعم.

لتأخذ جدول الموظفين؛ بحد أنه يحتوي على العمود mgr والذي يمثل رقم المدير لمجموعة من الموظفين، ولكن يجب أن يكون هذا الرقم موجوداً أصلاً كرقم موظف empno. وعليه يمكننا القول إن العمود empno هو المفتاح الأجنبي للعمود mgr في الجدول emp. ويمكننا كتابة المثال التالي الذي يتحقق الغرض السابق:

```
CREATE TABLE emp
(empno      NUMBER(4) PRIMARY KEY,
mgr        NUMBER(4),
CONSTRAINT empno_mgr FOREIGN KEY(mgr) REFERENCES emp(empno),...)
```

- **CHECK Constraint**

التدقيق يعني ضمان إدخال بيانات ضمن النطاق المرغوب فيه أو ضمن قيمة معينة مثلاً؛ مثل أن يتم إدخال أسماء الموظفين بحروف كبيرة أو أن لا تتجاوز قيمة الراتب ٥٠٠٠ وغير ذلك. وهذا يعني وضع شروط على إدخال البيانات. والشكل العام لقيد التدقيق هو:

```
[CONSTRAINT constraint_name ] CHECK (condition)
```

مع ملاحظة ما يلي:

- عدم جواز استعمال الاستعلام الجزئي في قيد التدقيق
 - عدم استعمال الأعمدة الزائفة .SYSDATE pseudo column مثل
- لاحظ أن قيد التدقيق يعرف على مستوى العمود.

ومن الممكن استعمال خيار التعطيل DISABLE عند تعريف القيد:

ويستعمل خيار التعطيل مؤقتاً بغض إصلاح البيانات مثلاً أو نقل كميات هائلة من البيانات (مليارات الصفوف مثلاً) عبر الشبكة الواسعة WAN من منطقة إلى أخرى ؛ مع استمرارية إمكانية استعمال القيد مع أدوات أوراكل الأخرى. ويمكن إعادة تكين ENABLE القيد لاحقاً باستعمال أمر التغيير ALTER. وكمثال على ذلك:

```
CREATE TABLE emp
(...,
ename VARCHAR2(10) CONSTRAINT chk_upper_name
CHECK(ename=UPPER(ename)) DISABLE,...)
```

وأخيراً يجب ملاحظة أن القيود سابقة الشرح يجب أن تكون مملوكة لنفس المستخدم الذي يتصرف بها أي أن تكون موجودة في نفس قاعدة البيانات التابعة له وإلا وجب أن يذكر اسم المستخدم قبل اسم الجدول المراد استمداد القيد منه.

وإليك المثال التالي الذي ينشأ جدول الموظفين emp بالقيود التي عليه:

```
CREATE TABLE emp
(
    employee_id    NUMBER(6)
    CONSTRAINT emp_employee_id PRIMARY KEY
    , first_name    VARCHAR2(20)
    , last_name     VARCHAR2(25)
    CONSTRAINT emp_last_name_nn NOT NULL
    , email         VARCHAR2(25)
    CONSTRAINT emp_email_nn    NOT NULL
    CONSTRAINT emp_email_uk    UNIQUE
    , phone_number  VARCHAR2(20)
    , hire_date     DATE
    CONSTRAINT emp_hire_date_nn NOT NULL
    , job_id        VARCHAR2(10)
    CONSTRAINT emp_job_nn      NOT NULL
    , salary        NUMBER(8,2)
    CONSTRAINT emp_salary_ck   CHECK (salary>0)
    , commission_pct NUMBER(2,2)
    , manager_id    NUMBER(6)
    , department_id NUMBER(4)
    CONSTRAINT emp_dept_fk     REFERENCES
        departments (department_id);
```

هيكل البيانات في أوراكل ORACLE Data Structure

ثانياً: إنشاء جدول من خلال استعلام.

في النقاط السابقة ناقشنا كيفية إنشاء الجداول مباشرةً من خلال جملة CREATE TABLE . وفي النقطة التالية سنناقش إنشاء الجداول بصورة غير مباشرةً من خلال الاستعلام الجرئي . بهذه الطريقة يمكننا إنشاء نسخة أخرى من جدول موجود سابقاً في قاعدة البيانات بما في ذلك أي بيانات سابقة فيه؛ والشكل العام هو:

```
CREATE TABLE table_name[(col1,col2,...)]
```

```
AS query;
```

وتشمل ... *col1*,*col2*,... أسماء الأعمدة في الجدول الجديد.

أما *query* فتمثل استعلاماً ما كما سبق شرحه . علماً بأنه يمكن استعمال جميع أنواع الاستعلام وشروطها وأشكالها المختلفة.

مثال: نريد إنشاء جدول يحتوي بيانات الموظفين في القسم ١٠ فقط من جدول الموظفين ونسمي هذا الجدول باسم *emp10* كما يلي:

```
CREATE TABLE emp10
```

```
AS
```

```
(SELECT * FROM employees
```

```
WHERE departmentId=10);
```

Table created.

لاحظ أن هذه النسخة مطابقة في تركيبها Data Structure لجدول الموظفين employees غير أنه يقتصر في بياناته على موظفي القسم رقم ١٠ فقط.

في المثال التالي سنقوم بإنشاء جدول جديد اسمه dept80 بتحديد عدد معين من الأعمدة واستعمال اسم بديل لأحد الأعمدة من خلال الاستعلام التالي:

```
CREATE TABLE dept80
```

```
AS
```

```
SELECT employee_id, last_name, salary*12 ANNSAL, hire_date
```

```

FROM employees
WHERE department_id = 80;

```

Table created.

ولتأكيد ذلك استعمل الأمر DESCRIBE لعرض هيكلاة الجدول الجديد كما يلي :

```
DESCRIBE dept80
```

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

ملاحظات:

- س يتم إنشاء الجدول الجديد بأسماء الأعمدة المحددة في جملة الاستعلام.
- يرث الجدول الجديد قيد الشيء/اللاشيء NULL/NOT NULL فقط من الجدول القديم.
- إذا كانت أسماء الأعمدة في الاستعلام معروفاً جيداً (أي ليست تعبيرات أو خلافها) فيمكن عندئذ عدم ذكر أسماء الأعمدة في عبارة CREATE TABLE.
- إذا تم تحديد أسماء الأعمدة التي تكون الجدول الجديد في عبارة CREATE TABLE فحينئذ يجب أن يتطابق عدد الأعمدة مع نظيره في جملة الاستعلام.
- يتم نسخ الصنوف (البيانات) من الجدول القديم إلى الجدول الجديد إن وجدت -حسب شروط الاستعلام الجزئي- وإلا يتم إنشاء الجدول فقط دون بيانات.

أمر التغيير

ويستعمل أمر التغيير - كما قلنا سابقاً - لتعديل هيكل أحد مكونات قاعدة البيانات أي تغيير تعريف بعض أجزائه. وسنتحدث هنا عن تغيير جزأين من هيكل جدول ما:

- إضافة عمود أو أعمدة إلى جدول والشكل العام لذلك هو:

```
ALTER TABLE table_name ADD
```

```
(col1 datatype1 (size)),
```

```
col2 datatype2 (size)...);
```

مثال : نريد أن نضيف عمود اسم القرین spouse- الزوجة أو الزوج- إلى الجدول emp10 الذي أنشأناه في بند إنشاء جدول كما يلي :

```
ALTER TABLE emp10 ADD (spouse VARCHAR2 (20));
```

Table altered.

٢- إضافة قيد إلى جدول موجود. فلو أردنا إضافة قيد للتدقيق على قيمة الراتب SAL فإننا تكتب التالي :

```
ALTER TABLE emp10  
ADD (CHECK (salary>=5000));
```

٣- تغيير طول عمود بالنقصان أو الزيادة (حسب القواعد التي سوف نذكرها قريبا)؛ حيث يمكننا أن نزيد في طول عمود أو نقص منه حسب حاجتنا. والشكل العام لهذا الأمر هو :

```
ALTER TABLE table_name MODIFY  
(col1 datatype1 (size),  
col2 datatype2 (size)...);
```

في المثال التالي سنقوم بزيادة طول عمود اسم الموظف من ٢٠ رمزا إلى ٢٥ رمزا في الجدول emp10 كما يلي :

```
ALTER TABLE emp10 MODIFY (last_name VARCHAR2 (25));
```

Table altered.

٤- تغيير العمود من NULL إلى NOT NULL .

بعض قواعد تغيير هيكل جدول

أ. العمود الذي يحتوي القيمة "لا شيء" NULL لا يمكن تغييره من NULL إلى NOT NULL. ولكن يمكن تغييره بإعطائه قيمة معينة ثم تغييره إلى إجباري NOT NULL .

ب. لا يمكن إضافة عمود باستعمال عبارة NOT NULL في تعريفه. ولتفادي ذلك نقوم بإضافة العمود أولا بصورة عادية- أي وجود إمكانية أن يحتوي القيمة لا شيء NULL- ثم تغييره إلى NOT NULL بشرط عدم إضافة أي صفات للجدول قبل التغيير.

ت. لا يمكن تقليل حجم الأعمدة التي تحتوي بيانات؛ بل يمكن زراعتها.

استعمال خيار الإسقاط DROP

ويستعمل لإلغاء قيد ما من الجدول؛ حيث يأخذ أمر التغيير ALTER الشكل التالي :

ALTER TABLE *tablename* DROP CONSTRAINT *constraint_name*;

ALTER TABLE *tablename* DROP PRIMARY KEY;

مثال:

ALTER TABLE employees DROP CONSTRAINT empno_mgr;

ALTER TABLE employees DROP PRIMARY KEY;

ويضاف خيار إلغاء الكل CASCADE لإلغاء القيد من الجدول الأساسي parent table إضافة إلى إلغاء قيد المفتاح الأجنبي المرتبط به من الجدول الابن child table كما يلي:

ALTER TABLE employees DROP PRIMARY KEY CASCADE;

استعمال خيارات التعطيل ENABLE والتمكين DISABLE

وبدلاً من إلغاء القيود فإنه يمكن تعطيلها DISABLE مؤقتاً ثم تمكينها ENABLE مرة أخرى. ويضاف الخيار CASCADE إلى أمر تغيير الجدول ليؤثر على القيود الأخرى المرتبطة بالقيد الذي يتم تعطيله أو تمكينه. وهذه هي الصيغة العامة لذلك:

ALTER TABLE *tablename* [DISABLE/ENABLE]

[UNIQUE (*col1, col2, ...*) PRIMARY KEY /

CONSTRAINT *constraint_name*] [CASCADE]

وتشير الأقواس المربعة إلى أن ما بداخلها اختياري.

مثال:

ALTER TABLE departments

DISABLE CONSTRAINT deptno_prim CASCADE;

وهناك خيارات أخرى للأمر DROP سيتم بحثها في موضوعها

أمر الإسقاط DROP

ويستعمل لإلغاء جدول من قاعدة البيانات وصيغته العامة:

DROP TABLE *table_name* [CASCADE CONSTRAINTS];

وتحب ملاحظة أن الحذف النهائي يلغى البيانات تماما من قاعدة البيانات؛ فوجب الحذر لدى التعامل مع مثل هذه الأوامر.
ويفيد الخيار CASCADE في تعطيل أية قيود معتمدة على الجدول.

مثال: سنقوم بإلغاء الجدول dept1 – بافتراض وجوده.

```
DROP TABLE dept1;
```

```
Table dropped.
```

١. يتم إلغاء جميع البيانات تماماً من قاعدة البيانات إضافة لهيكلية الجدول Table Structure.
٢. لا يتم إلغاء أي مرادف SYNONYM أو عرض VIEW ولكنها تصبح غير مقبولة invalid. ويعني آخر لا نستطيع اعتبارها صحيحة وأي محاولة للوصول إليها يعطي أخطاء نحن في غنى عنها. ولذا فمن الأفضل إلغائها بعد إلغاء الجدول المرتبط بها.
٣. الذي له صلاحية إلغاء الجدول هو من أنشأه أو مدير قاعدة البيانات DBA.
٤. يتم إلغاء جميع الفهرس indexes والقيود التي أنشئت على الجدول.

أمر التسمية RENAME

يقوم أمر التسمية بتغيير اسم الجدول في قاموس أوراكل من اسمه القديم إلى الاسم الجديد؛ وصيغته العامة هي:

`RENAME old_name TO new_name;`

مثال: لنغير اسم الجدول `emp10` إلى `emps10` –على فرض أنه موجود– كما يلي:

`RENAME emp10 TO emps10;`

Table renamed.

أمر البتر TRUNCATE

وقد سبقت الإشارة إليه. ويستعمل أمر البتر TRUNCATE لحذف جميع البيانات من الجدول؛ بدلاً من استعمال الأمر DELETE وهو أسرع من الأمر DELETE ولا يحتاج إلى عمليات تخزين مؤقتة بل إنه يمسح البيانات وينظف الذاكرة وبسرعة. وصيغته العامة هي:

`TRUNCATE TABLE table_name [REUSE STORAGE];`

لاحظ كلمة TABLE في الصيغة وذلك لأن هذا الأمر من لغة تعريف البيانات DDL. أما الخيار REUSE STORAGE فيعني أن يحتفظ الجدول بالمساحة على القرص التي كانت مستعملة لخزن البيانات الممسوحة. أما إن حذف ذلك الخيار فإن تلك المساحة تحرر لاستعمال جداول أخرى وهو الخيار الافتراضي default.

ويجب الحذر أيضاً عند استعمال هذا الأمر لأنه لا سبيل إلى استرجاع البيانات الملغاة ما لم يكن هناك نسخة احتياطية لآخر تعديل قبل إلغاء تلك البيانات.

تلك كانت مجموعة أوامر لغة تعريف البيانات؛ ولها الخصائص التالية:

١. يمكن إنشاء وإلغاء وتغيير وإعادة تسمية الجداول في أي وقت دون حاجة لوقف المستعملين عن العمل بل إنهم لا يشعرون بذلك إلا إذا في بعض الحالات مثل تغيير اسم جدول يستعمله برنامج فحينئذ يجب تعديل البرنامج بالاسم الجديد.
٢. يمكن تغيير حجم البيانات –أي الأعمدة– لحظياً ON-LINE.

٣. لا حاجة عادة لتحديد حجم الجدول المراد إنشاؤه. الواقع أن هذا يصبح ضروريا في التطبيقات الكبيرة.
٤. إذا كانت عمليات الإلغاء أو التسمية متعلقة ببعض البرامج فوجب صيانة تلك البرامج فقط بما يتاسب مع التغييرات الحاصلة.
٥. حين استعمال أي جملة من جمل لغة هيكل البيانات تنهي أي حركة Transaction مع تثبيت COMMIT التغييرات على البيانات أي تثبيت تلقائي Auto Commit.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٣٨)

هيكل البيانات في أوراكل ORACLE Data Structure

مكونات أخرى لقاعدة البيانات في أوراكل Other Oracle Database Objects

ناقشتنا في الفصل السابق إنشاء الجداول tables وهي المكون الأساسي لقاعدة البيانات أوراكل. وفيها يتم تخزين البيانات التي نتعامل معها مثل بيانات الموظفين والمخزون والمبيعات... وستتحدث في هذا الفصل عن مكونات أخرى-وليس أخيرة- لقاعدة البيانات أوراكل. وهذه المكونات هي:

١. العرض view

٢. الفهرس index

٣. السلسلة sequence

٤. المرادف synonym

VIEW العرض

والحقيقة احترت في ترجمة كلمة view فهي ذات معانٍ عدة مثل رؤية وصورة ومرأى ومنظر ومعاينة ومشهد... وأوراكل ترجمته مرة إلى منظر وأخرى إلى عرض وثالثة إلى جدول اعتباري. وأقرب كلمتين لتعبير عن الغرض منها هو عرض ومشهد كما هو التعبير عنها بالإنجليزية. وسوف أحذّر الكلمة عرض لقرئها من المقصود.

والعرض VIEW يشبه المنظر العام الذي تراه من نافذتك التي تطل منها على حديقة بيتك أو على أفق واسع أمامك ترى فيه عدة مكونات للطبيعة مثل الجبال والغابات وغيرها. وهنا العرض في أوراكل لا يختلف كثيرا؛ فهو النافذة التي تطل منها على بيانات جدول ما أو عدة جداول دفعه واحدة وبالتالي إمكانية التعامل مع تلك البيانات.

يتم تكوين العرض من جدول أو أكثر؛ ويعرف الجدول أو الجداول التي يبني عليها العرض بالجدول ال قاعدة base tables للعرض.

يتم تخزين العرض كجملة استعلام SELECT فقط. أي أنه لا يوجد حقيقة بل إن وجوده افتراضي. وبمعنى آخر لا يوجد فيزيائياً؛ وإن كان يظهر للمستعمل وكأنه موجود. ولهذا فعند إلغاء جدوله من قاعدة البيانات، فإنه لا يكون متاحاً للمستعمل-أي لا نستطيع استخراج بيانات منه-لأنه لم يعد مقبولاً invalid وإن كان لا يزال موجوداً في قاموس أوراكل.

فوائد استعمال العرض:

١. تقييد الوصول إلى قاعدة البيانات. إذ أن العرض عادة (يحتوي) على جزء من البيانات مقيداً بجملة الاستعلام الناشئ عنها.

٢. السماح للمستعملين باستخراج البيانات باستعمال جمل استعلام بسيطة. مثل ذلك أن المستعمل يمكنه استخراج البيانات من عدة جداول دون معرفة كيفية الربط بينها.

٣. يمكن المستعملين من استخراج البيانات بعدة أشكال كل مستعمل حسب حاجته.

أنواع العروض:

تنقسم العروض إلى نوعين هما البسيط Simple والمعقد Complex.

يعتبر العرض بسيطا في إحدى الحالتين التاليتين:

١- يكون مبنيا على جدول واحد.

٢- لا يحتوي على دوال functions أو مجموعات من البيانات groups أي لا تحتوي جملة الاستعلام التي أنشأته على دوال المجموعات.

ومن خصائصه يمكن استعمال لغة معالجة البيانات عليه DML

وعلى عكس العرض البسيط فإن العرض يكون معقدا في إحدى الحالتين التاليتين:

١- يكون مبنيا على أكثر من جدول واحد.

٢- يحتوي على دوال functions أو مجموعات من البيانات groups أي تحتوي جملة الاستعلام التي أنشأته على دوال المجموعات.

ومن خصائصه أنه لا يمكن استعمال لغة معالجة البيانات عليه بصورة سهلة.

أمر إنشاء العرض

يتم إنشاء العرض بجملة CREATE VIEW كما يلي:

CREATE [OR REPLACE] [FORCE] VIEW view_name

[(Col1,col2,...)]

AS SELECT statement

[WITH CHECK OPTION [CONSTRAINT constraint_name]]

[WITH READ ONLY [CONSTRAINT constraint]];

حيث:

col1,col2,...

أسماء الأعمدة في العرض والمتناظرة مع تلك الموجودة في جملة SELECT.

OR REPLACE

إلغاء العرض المعرف سابقا وإنشاء آخر بنفس الاسم. ويجب الحذر هنا لأن العرض عبارة عن جملة استعلام وقد تكون معقدة وبذل المبرمج جهدا كبيرا في كتابتها؛ فاستعمال **OR REPLACE** يمسح تلك الجملة السابقة وحيثند لا ينفع الندم ما لم يكن منها نسخة احتياطية ومع ذلك ليس من السهولة استرجاع العرض السابق!

FORCE

إنشاء العرض حتى لو لم يكن الجدول المبني عليه موجودا. ويجب ملاحظة أنه يجب إنشاء الجدول قبل أن يكون استعمال العرض ممكنا.

[WITH CHECK OPTION [CONSTRAINT *constraint_name*]]

هذا الخيار يمنع من إجراء إضافة أو تعديل صفات لا تتحقق الشرط المذكور في عبارة WHERE ضمن جملة الاستعلام التي تكون العرض وعبارة CONSTRAINT *constraint_name* اختيارية كما مر سابقا.

[WITH READ ONLY [CONSTRAINT *constraint_name*]]

هذا الخيار يمنع من إجراء أية عمليات تعديل للبيانات في العرض - حتى لو كان بسيطاً؛ أي يمنع استعمال لغة معالجة البيانات DML وبالتالي يكون العرض للقراءة فقط. وعبارة CONSTRAINT *constraint_name* اختيارية كما مر سابقا.

أمثلة:

١- لإنشاء عرض بسيط اسمه d10emp ويحتوي بيانات الموظفين قسم ١٠ نكتب التالي:

```
CREATE VIEW d10emp
```

```
AS
```

```
SELECT employee_id, last_name, salary
```

```
FROM employees
```

```
WHERE department_id = 10;
```

View created.

ويمكننا استعمال العرض كأي جدول كما في المثال التالي:

```
SELECT * FROM d10emp
```

```
ORDER BY last_name;
```

كما يكتننا عرض هيكل البيانات للعرض باستعمال الأمر DESCRIBE

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY		NUMBER(8,2)

-٢ ولإنشاء الجدول salvu50 باستعمال الأعمدة البديلة aliases نكتب التالي:

```
CREATE VIEW salvu50  
AS SELECT employee_id id_number, last_name name,  
Salary*12 ann_salary  
FROM employees  
WHERE department_id = 50;
```

ويمكننا كتابة الجملة بصورة أخرى كما يلي:

```
CREATE OR REPLACE VIEW salvu50 (id_number, name, ann_salary)  
AS SELECT employee_id, last_name, salary*12  
FROM employees  
WHERE department_id = 50;
```

لاحظ كيف تم استعمال الأسماء البديلة aliases مرة في جملة الاستعلام كما في الطريقة الأولى ومرة ضمن تعريف العرض كما في الطريقة الثانية

تغيير العرض alter view

لا توجد جملة تغيير خاصة بالعرض كما في الجدول؛ وإنما يتم إعادة كتابة جملة الاستعلام لتغيير هيكلية العرض. فلتغيير هيكلية العرض d10emp السابق نعيد كتابة جملة الاستعلام لتكون كما في الشكل التالي:

```
CREATE OR REPLACE VIEW d10emp  
(id_number, name, sal, department_id)
```

```

AS SELECT employee_id, first_name || ''
|| last_name, salary, department_id
FROM employees
WHERE department_id = 10;

```

View created.

-٣ لإنشاء العرض المعقد dept_summary والذي يحتوي دوال المجموعات وبيانات من أكثر من جدول نكتب:

```
CREATE OR REPLACE VIEW dept_summary
```

```
(name, minsal, maxsal, avgsal)
```

```
AS SELECT d.department_name, MIN (e.salary),
```

```
MAX (e.salary), AVG (e.salary)
```

```
FROM employees e JOIN departments d
```

```
ON (e.department_id = d.department_id)
```

```
GROUP BY d.department_name;
```

View created

وللاستعلام عن محتويات العرض الجديد نكتب:

```
SELECT * FROM dept_summary;
```

NAME	MINSAL	MAXSAL	AVGSAL
Administration	4400	4400	4400
Accounting	8300	12000	10150
Human Resources	6500	6500	6500
Public Relations	10000	10000	10000
Executive	17000	24000	19333.3333
IT	4200	9000	5760

Purchasing	2500	11000	4150
Shipping	2100	8200	3475.55556
Finance	6900	12000	8600
Sales	6100	14000	8955.88235
Marketing	6000	13000	9500

11 rows selected.

• معالجة البيانات باستعمال العرض

من الممكن معالجة البيانات على العرض بدلاً من الجدول أو الجداول المبني عليها العرض. مما يعطي مرونة عالية وسهولة في الاستعمال؛ مع ملاحظة أنه يمكن تطبيق القيود على العرض كما يمكن تطبيقها على الجدول وذلك بإضافة الخيار CHECK عند إنشاء العرض كما في المثال التالي:

```
CREATE OR REPLACE VIEW empvu20
```

```
AS SELECT*
```

```
FROM employees
```

```
WHERE department_id = 20
```

```
WITH CHECK OPTION CONSTRAINT empvu20_ck;
```

View created.

المثال أعلاه ينشأ العرض empvu20 وبشرط أن تكون بيانات الموظفين فقط من القسم رقم ٢٠ . وعليه فإن أي محاولة لإضافة صفوف جديدة أو تعديل صفوف موجودة لا يكون رقم القسم فيها ٢٠ سيتم رفضها.

في المثال التالي حاولنا تعديل قيمة العمود department_id من ٢٠ إلى ١٠ للموظف رقم ٢٠١ ؛ فكانت النتيجة:

```
UPDATE empvu20
```

```
SET department_id = 10
```

```
WHERE employee_id = 201;
```

*

ERROR at line 1:

ORA-01402: view WITH CHECK OPTION where-clause violation

منع معالجة البيانات على العرض

ولمنع أية تحديثات للبيانات على العرض نستخدم الخيار WITH READ ONLY فيكون العرض للقراءة فقط كما في المثال التالي:

```
CREATE OR REPLACE VIEW empvu10  
(employee_number, employee_name, job_title)  
AS SELECT Employee_id, last_name, job_id  
FROM employees  
WHERE department_id = 10  
WITH READ ONLY;
```

View created.

لن نستطيع إجراء أية تحديثات على بيانات العرض لأنه للقراءة فقط مثل أي ملف على القرص. سنحاول حذف صفوف من العرض كما يلي:

```
DELETE FROM empvu10  
WHERE employee_number = 200;  
DELETE FROM empvu10  
*
```

ERROR at line 1:

ORA-01752: cannot delete from view without exactly one

Key-preserved table

ظهرت الرسالة السابقة. أما في حالة إضافة صف أو تعديل صف فتظهر الرسالة الغربية التالية:

01733: virtual column not allowed here.

قواعد معالجة البيانات باستعمال العرض

- الحذف DELETE منوع في الحالات التالية:
 - وجود شروط الربط بين جدولين فأكثر في تعريف العرض .join conditions

- ٢ وجود دوال المجموعات group functions في تعريف العرض.
- ٣ وجود عبارة GROUP BY في تعريف العرض.
- ٤ وجود عبارة DISTINCT في تعريف العرض.

- التعديل UPDATE متعدد في الحالات التالية:
 - ١ أي حالة من حالات منع الحذف (انظر الحذف أعلاه).
 - ٢ تعديل عمود معرف بواسطة تعبير مثل .COMM/12.

- الإضافة INSERT متعددة في الحالات التالية:
 - ١ أي حالة من حالات منع التعديل (انظر التعديل أعلاه).
 - ٢ تعديل أي عمود إجباري NOT NULL لم يتم إدخاله إلى تعريف العرض مثل .employee_id.

إلغاء العرض من قاعدة البيانات

لإلغاء العرض من قاعدة البيانات نستعمل الأمر DROP كما يلي:

DROP VIEW *view_name*;

ولعلك تدرك أيها القارئ الكريم أن إلغاء العرض لا يؤثر على الجداول الأساس ولا على البيانات فيها، ولكن يؤثر على البرامج التي تتحذى من العرض المُلغى أساساً لها فوجب تعديل تلك البرامج عند الحاجة لها.

- ملاحظة: يمكننا تطبيق لغة معالجة البيانات على العروض المعقّدة باستعمال زنادات قاعدة البيانات database triggers والتي سأطرق لها عن شرح لغة أورالك الإجرائية PL/SQL إن كان هناك نصيّب!

أوراكل مثلاً (٣٩)

هيكل البيانات في أوراكل ORACLE Data Structure

مكونات أخرى لقاعدة البيانات في أوراكل Other Oracle Database Objects

• المرادف SYNONYM

المرادف SYNONYM ما هو إلا اسم آخر لمكونات قاعدة البيانات مثل الجدول أو العرض أو السلسلة SEQUENCE. وعادة عندما نريد استعمال جدول مستخدم آخر فإننا نسبق اسم الجدول باسم المستخدم؛ فلو أراد المستخدم ahmed استعلام الجدول *emp* والذي يخص المستخدم scott فإن عليه أن يكتب الأمر التالي:

```
SELECT * FROM scott.emp;
```

فيتمكن للمستخدم scott أن ينشأ مرادفاً *synonym* للجدول *emp* وعنده صلاحيات استعماله لمستخدمين آخرين. والصيغة العامة لإنشاء المرادف هي كما يلي:

```
CREATE SYNONYM synonym_name FOR user.table;
```

مثال:

```
CREATE SYNONYM scottemp FOR scott.emp;
```

Synonym created.

ويستطيع ahmed أن يستعمل من الجدول *emp* باستعمال المرادف *scottemp* كما يلي:

```
SELECT * FROM scott.scottemp;
```

وتم سبق اسم المرادف باسم المستعمل scott وذلك لأن هذا المرادف خاص بالمستعمل scott. انظر الشرح التالي.
وهنالك نوعان للمرادفات أحدهما خاص بالمستعمل PRIVATE ويمكنه إعطاء صلاحيات استعمال هذا المرادف لآخرين
وصيغته العامة هي:

```
CREATE SYNONYM synonym_name FOR user.table;
```

حيث أن ذكر اسم المستعمل اختياري لأن المرادف المنشأ بواسطة هذا المستعمل يكون خاصاً. وهذا ينطبق على جميع المكونات الأخرى ما لم يتم ذكر خلاف ذلك.

والثاني هو العام PUBLIC والذي يقوم بإنشائه مدير قاعدة البيانات DBA أو من له صلاحية إنشاء مرادف عام. وصيغته العامة هي :

```
CREATE PUBLIC SYNONYM synonym_name
```

FOR /owner./object_name;

وفي هذه الحالة لا داعي لذكر صاحب الجدول الأصلي في جمل SQL لأنه صار متاحاً بشكل عام؛ ولذلك على من أنشأ هذا المرادف أن يحدد الصلاحيات عليه كأن تكون قراءة فقط

ولإلغاء المرادف من قاعدة البيانات نستعمل أمر الإلغاء DROP كما يلي:

DROP [PUBLIC] SYNONYM synonym_name;

حيث تشير الأقواس المربعة إلى أن ما بداخلها اختياري.

والغرض من استعمال المرادفات هو:

١. إخفاء الأسماء الحقيقة لمكونات قاعدة البيانات التي تخص المستعمل وإعطائهما أسماءً أخرى.
٢. استعمال مكونات قاعدة البيانات المستعمل ما دون تحديد اسم المستعمل.

هذا ومن الأفضل عدم استعمال المرادفات كثيراً في التطبيقات كي لا يقل مستوى أداء التطبيق.

أوراكل مثلاً (٤٠)

هيكل البيانات في أوراكل ORACLE Data Structure

مكونات أخرى لقاعدة البيانات في أوراكل Other Oracle Database Objects

الفهرس INDEXES

والفهرس هو أحد مكونات قاعدة البيانات يستعمل لأغراض مثل السرعة في البحث؛ وهو يشبه فهرس الكتاب فللوصول إلى موضوع ما فإنك تبحث عن الكلمة يتضمنها الموضوع أو رقم الصفحة فتصل بسرعة إلى الموضوع.

وللفهرس لها الخصائص التالية:

- ١- إحدى مكونات قاعدة البيانات Database Object.
- ٢- تقوم أوراكل باستعمال الفهرس للبحث السريع في بعض أنواع الاستعلام باستعمال المؤشرات Pointers ولا دخل للمستخدم في الكيفية التي تقوم بها أوراكل لهذا الغرض.
- ٣- يخفف البحث في الجدول. فبدلاً من البحث في جميع الجداول؛ يقوم أوراكل بالبحث في الفهرس حتى يجد القيمة المبحوث عنها ثم يستعمل عنوان الصفت السابق ذكره ROWID ثم يحضر الصفت من الجدول باستعمال عنوان الصفت.
- ٤- الفهرس مستقلة عن الجدول يعني أنه يمكن إنشاؤها بعد إنشاء الجدول.
- ٥- تقوم أوراكل بصيانة الفهرس واستعمالها آلياً.

وفي أوراكل يقوم المستخدم (المبرمج) عادة بإنشاء الفهرس وتنظيمها بحيث يمكن استعمالها في البحث والاستعلام.

ويتم إنشاء الفهرس لتحقيق الهدفين التاليين:

١. سرعة البحث عن الصفوف خلال مفتاح KEY.
٢. ضمان عدم تكرار البيانات نفسها مثل رقم الموظف مثلاً.

وتنقسم الفهرس من حيث العمل إلى نوعين:

١. وحيدة UNIQUE- أي عدم تكرار القيمة الواحدة.
٢. غير وحيدة NON-UNIQUE- أي تسمح بتكرار القيمة الواحدة.

١. الفهرس الوحيد unique index وتقوم أوراكل بإنشاء الفهرس الوحيد بطريقتين:
 - أ. عند إنشاء المفتاح الأولي Primary Key
 - ب. عند إنشاء المفتاح الوحد Unique Key

ففي أرقام الموظفين لا يُسمح بتكرار رقم الموظف بحيث يعطى موظف آخر؛ بينما يُسمح بتكرار الأسماء لأن الأسماء تتكرر، فننشأً فهرساً وحيداً للأرقام وآخر غير وحيد للأسماء.

والمستخدم(المبرمج) لا يحتاج إلى إنشاء الفهارس الوحيدة ولكن يمكنه إنشاء قيد الوحدة unique constraint على العمود أو الأعمدة التي لا يرغب في تكرارها

٢. الفهرس العادي CREATE INDEX انظر non-unique index يتم إنشاؤه باستخدام جملة إنشاء الفهرس

الشرح التالي

والصيغة العامة لإنشاء الفهرس هي:

CREATE [UNIQUE] INDEX *index* ON *table* (*col1, col2, ..., coln*);

ولا يفضل إنشاء فهارس وحيدة باستعمال جملة الإنشاء بل من خلال المفتاح الأساسي أو المفتاح الوحديد.

مثال:

المثال التالي سينشأ فهرساً Index على الاسم الأخير من اسم الموظف.

CREATE INDEX emp_last_name_idx

ON employees(last_name);

Index created.

كيف تقرر أوراكل استعمال الفهرس INDEX

فيما يلي القواعد التي تقرر أوراكل بناءً عليها استعمال الفهرس:

- يجب أن يظهر العمود أو الأعمدة المستعملة في الفهرس في عبارة الشرط WHERE clause . أمثلة:
في المثال التالي لن يتم استعمال أي فهرس لعدم وجود عبارة الشرط WHERE Clause

SELECT last_name, job_id, Salary

FROM employees

بينما المثال التالي سوف تستعمل فيه أوراكل فهرساً على الاسم-في حال تم إنشاؤه مسبقاً-

SELECT * FROM employees

WERE last_name = 'King';

٢- لن يتم استعمال الفهرس إذا كان العمود المنشأ عليه الفهرس جزءاً من تعبير expression كما في المثال التالي الذي لن يستعمل فيه الفهرس على اسم الموظف - رغم وجود ذلك الفهرس على اسم الموظف.

```
SELECT * FROM employees
```

```
WHERE UPPER(last_name) = 'KING';
```

إرشادات حول إنشاء الفهارس INDEXES

يكون إنشاء فهارس مفيداً في الحالات التالية:

١. إذا كان الجدول يحتوي على عمود فيه قيم ذات نطاق واسع. مثلاً يحتوي قيمة تتراوح بين ١ و ١٠٠،٠٠٠،٠٠٠. فإن إنشاء فهرس على هذا العمود له اعتبار.
٢. إذا كان العمود يحتوي على قيمة لا شيء NULL لعدد كبير جداً من الصفوف.
٣. الأعمدة المستعملة غالباً في عبارة WHERE؛ فإن فهرستها يسرع كثيراً من عمليات البحث.
٤. إذا كان عدد الصفوف المراد استرجاعه من الاستعلام لا يزيد عن ٤٪ من عدد الصفوف الكلية في الجدول

ويكون إنشاء فهارس غير مفيد في الحالات التالية:

- ١- إذا كان العمود أو الأعمدة لا تستعمل عادة في عبارة WHERE.
- ٢- إذا كان الجدول صغيراً (بضعة آلاف من الصفوف مثلاً) فلا يفيد إنشاء فهرس عليه.
- ٣- إذا كان عدد الصفوف المراد استرجاعها من الاستعلام يتجاوز ٤٪ من عدد الصفوف الكلية في الجدول
- ٤- إذا كان الجدول يتم تحديثه باستمرار أي إضافة وحذف صفوف في معظم الوقت
- ٥- إذا كانت الأعمدة جزءاً من تعبير expression
- ٦- من الخطأ الإكثار من الفهارس لأن ذلك يشكل عبئاً إضافياً على قاعدة البيانات عموماً ومنها أوراق ل لأنها تقوم بصيانة الفهارس إضافة إلى الجداول.

وإلغاء الفهرس من قاعدة البيانات نستعمل أمر إلغاء DROP كما يلي:

```
DROP INDEX index_name;
```

وإلغاء الفهرس لا يؤثر على الجداول نفسها. ولكن عند إلغاء جدول يتم إلغاء فهارسه آلياً.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٤١)

هيكل البيانات في أوراكل ORACLE Data Structure

مكونات أخرى لقاعدة البيانات في أوراكل Other Oracle Database Objects

السلالس Sequences

من المكونات المفيدة لقاعدة البيانات؛ السلالس التي تفيد في توليد الأرقام المتسلسلة مثل رقم الفاتورة أو رقم قيد اليومية أو أي رقم متسلسل يستعمل في البرامج التطبيقية. وإنشاء سلسلة نكتب الصيغة التالية:

CREATE SEQUENCE [schema.]sequence_name

[INCREMENT BY m]

[START WITH n]

[MAXVALUE x | NOMAXVALUE]

[MINVALUE y | NOMINVALUE]

حيث:

-١ الأقواس المربعة تعني أن ما بداخلها اختياري.

-٢ اسم مخطط البيانات الذي سيتم إنشاء السلسلة عليه. وإذا لم يُذكر فهو المخطط الحالي current schema.

-٣ اسم السلسلة والذي يخضع لشروط الأسماء في أوراكل.

-٤ الرقم m هو عدد صحيح Integer يمثل مقدار الزيادة في السلسلة عن آخر رقم. والقيمة الافتراضية لـ m هي ١. إذا كان مقدار الزيادة عدداً صحيحاً موجباً فإنه يُقال إن السلسلة تصاعدية؛ أما إن مقدار الزيادة عدداً صحيحاً سالباً $-m$ فإنه يُقال إن السلسلة تناظرية Descending. يجب أن تكون القيمة المطلقة للزيادة أقل من الفرق بين MINVALUE و MAXVALUE.

-٥ الرقم n هو الرقم الصحيح الذي تبدأ منه السلسلة. والقيمة الافتراضية هي ١ للسلالس التصاعدية و MAXVALUE - انظر أسفل - للسلالس التناظرية.

-٦ قيمة MINVALUE تحديد أقل قيمة سيتم توليدتها للسلسلة بحيث لا تقل عن القيمة الابتدائية n ولا تزيد عن القيمة العليا MAXVALUE. أما y فهي عبارة عن عدد يتكون من ٢٨ رقمًا كحد أقصى.

-٧ قيمة NOMINVALUE؛ في حالة ذكر هذا الخيار فإنه يحدد ١ كأقل قيمة للسلالس التصاعدية و $(10^{26}) \times 1$ للسلالس التناظرية. الوضع الافتراضي هو NOMINVALUE.

- ٨ n MAXVALUE x؛ يحدد القيمة العليا للسلسلة والذي يجب أن يكون أقل أو يساوي القيمة الابتدائية للسلسلة x وأكبر من القيمة الدنيا y. عبارة عن عدد يتكون من ٢٨ رقمًا كحد أقصى.
- ٩ NOMAXVALUE؛ في حالة ذكر هذا الخيار فإنه يحدد القيمة ٣٧١٠ كحد أعلى للسلسلة التصاعدية و للسلسلة التناظرية. الوضع الافتراضي هو .NOMAXVALUE

مثال:

نريد إنشاء السلسلة dept_ser لعمود رقم القسم department_id؛ نكتب ما يلي:

```
CREATE SEQUENCE dept_ser
```

```
INCREMENT BY 10
```

```
START WITH 10
```

```
MAXVALUE 1000;
```

Sequence created.

• توليد الأرقام من السلسلة

لتوليد الأرقام من السلسلة نستعمل العمود الزائف NEXTVAL مسبوقاً باسم السلسلة فيعطيها الرقم التالي المتولد من السلسلة عند استعماله في جملة الاستعلام كما في المثال التالي:

```
Select dept_ser.NEXTVAL FROM DUAL;
```

```
NEXTVAL
```

```
-----  
10
```

وعند تنفيذ جملة الاستعلام السابقة مرة أخرى ينتج لدينا الرقم التالي في السلسلة كما يلي:

```
select dept_ser.NEXTVAL FROM DUAL;
```

```
NEXTVAL
```

```
-----  
20
```

• استعمال العمود NEXTVAL في لغة معالجة البيانات DML

يمكن استعمال العمود الزائف NEXTVAL في أي أمر من أوامر لغة معالجة البيانات مباشرةً كما في المثال التالي:

INSERT INTO departments VALUES

```
(dept_seq.NEXTVAL, 'SECRATERY', 201, 1800);
```

1 record created

وغمي عن البيان أنه إذا كان رقم القسم موجوداً سابقاً؛ فسوف تظهر رسالة خطأ لأن رقم القسم هو مفتاح أولي Primary Key على جدول الأقسام وبالتالي سوف نضطر لتكرار نفس جملة الإضافة السابقة حتى نصل إلى الرقم الصحيح المتولد من السلسلة. وبدلأً من ذلك يمكننا تعريف السلسلة بحيث يكون أول رقم فيها هو أكبر من آخر رقم قسم من جدول الأقسام. راجع صيغة إنشاء السلسلة سابقة الشرح.

• استعمال العمود CURRVAL

يفيد استعمال العمود الزائف CURRVAL في معرفة آخر رقم تم توليه من السلسلة دون تغيير في السلسلة أي بدون توليد رقم جديد. ويستعمل بنفس أسلوب استعمال العمود NEXTVAL كما في المثال التالي:

```
select dept_ser.CURRVAL FROM DUAL;
```

CURRVAL

20

• قواعد استعمال العمودين NEXTVAL و CURRVAL

هناك مجموعة من القواعد لاستعمال العمودين CURRVAL و NEXTVAL وهي تشمل حالات السماح باستعمالهما وحالات عدم السماح باستعمالهما.

الحالات المسموح فيها استعمال العمودين CURRVAL و NEXTVAL :

- ١ في عبارة SELECT من جملة الاستعلام (ماعدا تلك المتعلقة بالعرض views).
- ٢ في مجموعة القيم لعبارة VALUES من أمر الإضافة INSERT.
- ٣ في عبارة SET من أمر التعديل UPDATE.
- ٤ في أول عبارة SELECT من الاستعلام الجزئي وبمعنى آخر في جملة الاستعلام الرئيسية فقط .main query

أما الحالات الممنوع استعمال العمودين CURRVAL و NEXTVAL فيها فهي:

- ١ في جملة الاستعلام المتعلقة بالعرض .
- ٢ مع عبارة DISTINCT .
- ٣ إذا احتوت جملة الاستعلام العبارات التالية: .ORDER BY
- ٤

- 5 في جملة الاستعلام الفرعية .subquery
- 4 مع العوامل UNION و INERSECT و MINUS .
- 3 .HAVING
- 2 .CONNECT BY
- 1 .GROUP BY

• **Altering a Sequence تغيير السلسلة**

من الممكن تغيير تعريف السلسلة كأي مكون من مكونات قاعدة البيانات أوراكل. ويأخذ أمر التغيير الصيغة التالية:

`ALTER SEQUENCE [usr.]sequence_name`

`[INCREMENT BY n]`

`[MAXVALUE m|NOVALUE]`

`[MINVALUE l|NOVALUE];`

مثال: لتغيير القيمة العليا للسلسلة dept_ser سابقة الشرح نكتب الأمر التالي:

`ALTER SEQUENCE dept_ser`

`MAXVALUE 200000;`

ملاحظات

- 1 لا يمكن تغيير الرقم الابتدائي للسلسلة .START WITH
- 2 يجب أن تكون القيمة العليا الجديدة أكبر من آخر رقم وصلت له السلسلة.
- 3 تتأثر الأرقام الجديدة فقط بتغيير تعريف السلسلة؛ فإذا كان مقدار الزيادة INCREMENT BY في تعريف السلسلة
- 4 مثلاً ثم تغيرت إلى ٢٠ بواسطة أمر التغيير ALTER فإن أول رقم جديد في السلسلة يساوي آخر رقم قدسم + مقدار الزيادة الجديدة أي ٢٠.
- 5 لتغيير القيمة الابتدائية للسلسلة يجب إلغائها أولاً من قاعدة البيانات ثم إنشاؤها مرة أخرى.
- 6 ما بداخل الأقواس المربعة اختياري.

• **Dropping a Sequence إلغاء السلسلة من قاعدة البيانات**

يمكن إلغاء السلسلة من قاعدة البيانات باستعمال الأمر DROP كما يلي:

`DROP SEQUENCE [user.]sequence_name;`

لإلغاء السلسلة dept_ser نكتب الأمر التالي:

`DROP SEQUENCE dept_ser;`

ملاحظات حول إنشاء وتغيير وإلغاء السلسلة

- ١- إذا لم يذكر اسم المستعمل في أمر تعريف السلسلة-أي أمر من لغة DLL-؛ فنفترض أوراكل أن السلسلة مملوكة للمستعمل الحالي `current user`.

- ٢- لتعريف السلسلة يجب أن تكون للمستعمل صلاحية التعامل مع السلسلة أو أن يكون مديرًا لقاعدة البيانات DBA.

- ٣- يستطيع المستعمل المالك للسلسلة منح صلاحية تغيير ALTER أو الاختيار SELECT لمستعملين آخرين.

مشاكل السلسلة

- ١- عند استعمال السلسلة لأكثر من جدول، فإنه تحدث فراغات gaps بين الأعداد المستعملة كقيم في أعمدة الجداول. فلو فرضنا أنها نستعمل سلسلة واحدة بجدول الفواتير وجدول سندات الصرف وسندات القبض؛ وببدأنا بجدول الفواتير برقم الفاتورة ٢٠. ثم استعملنا السلسلة لسندات القبض بالرقم ٢١ ثم وصلنا إلى الرقم ٣٠ مثلاً وببدأنا باستعمال السلسلة لسندات الصرف سنببدأ بالرقم ٣١ ونستمر إلى ٣٥ مثلاً ثم نعود إلى الفواتير سنببدأ بالرقم ٣٦ ونستمر إلى الرقم ٥٥ مثلاً ثم نعود لسندات الصرف فنستأنف عند ٥٦ وثم سندات القبض عند ٥٧ وهكذا سوف نجد أرقام مفقودة من الجداول الثلاث السابقة؛ وهي لا تعني طبعاً أنه هناك سندات مفقودة بل فراغات بين الأرقام وهذا عادة لا يقنع الجهات المسئولة أو مراجعى الحسابات. والحل باستعمال سلسلة لكل جدول على حدة أو نضمن أن الفراغات لا تؤثر على التطبيق الذي نعمل عليه أو تقوم بتوليد الأرقام من ذات الجدول وليس من سلسلة!

- ٢- إن الأعداد تتولد من السلسلة بمجرد استعمالها. وهذا يعني أنها لو استعملنا سلسلة لإنشاء أرقام للفواتير؛ وحدثت مشكلة للنظام أثناء إضافة صف بجدول الفواتير؛ فإننا نفقد هذا الرقم وعند إعادة التشغيل يطلع لدينا رقم جديد غير المفقود مما يسبب فراغات بين الأرقams.

هذه هي أهم مشاكل استعمال السلسلة وهي ليست مشاكل جديدة ويمكن التصرف معها خاصة في الجداول التي لا يهم فيها تسلسل الأرقام والخيار لك!

أوراكل مثلاً (٤٢)

التعامل مع قاموس أوراكل Managing Oracle Data Dictionary

ناقشتنا في الرسائل السابقة ما يتعلق بالبيانات من استخراج ومعالجة وإنشاء مكونات قواعد البيانات...

وفي المواضيع التالية سنناقش قاموس بيانات أوراكل.

إن أوراكل تقوم بتسجيل كل حركة تحدث في قاعدة بياناتها. ومنها التعامل مع مكونات قاعدة البيانات من جداول وفهارس وعروض وسلال... ونستطيع تقسيم قاعدة البيانات أوراكل إلى قسمين:

 قسم يقوم المستعمل بإنشائه مثل الجداول وفيها يتم تخزين البيانات الحقيقية أي بيانات الحياة العادية مثل بيانات الموظفين والمخزون والمبيعات... الخ.

 قسم تقوم أوراكل بإنشائه أثناء تركيبها على الكمبيوتر وهذا القسم مجموعة من الجداول؛ فيها يتم تخزين معلومات هيكل البيانات التي يقوم بإنشائها المستعمل؛ فتقوم أوراكل بتسجيل معلومات الجدول مثل الاسم والحجم وصاحب الجدول... وبيانات الأعمدة مثل الاسم والنوع والحجم... وغير ذلك في جداول خاصة تسمى جداول قاموس بيانات أوراكل Oracle Data Dictionary Tables.

وهذه الجداول تتم صيانتها آلياً بواسطة أوراكل؛ ولا يجوز لأحد آخر عمل ذلك؛ إذ قد يوقف قاعدة البيانات نفسها حتى لا مدیر قاعدة البيانات. وإنما يتم الوصول إليها لاستخراج بياناتها من خلال عروض مبنية عليها والتي ستكون محور البحث في هذا الفصل.

أهمية قاموس البيانات

تبعد أهمية قاموس البيانات من خلال تسجيل جميع معلومات مكونات قاعدة البيانات كما أسلفنا وهذه تفاصيل في حالة أردننا معلومات عن هذه المكونات في حالة عدم وجود توثيق مثلاً.

أنواع عروض قاموس البيانات

١. عرض يستطيع المستعمل أن يقرأ معلومات جميع مكونات قاعدة البيانات التي أنشأها objects database. وهذه تبدأ اسماؤها بـ _USER.

٢. عرض يستطيع المستعمل أن يقرأ معلومات جميع مكونات قاعدة البيانات التي يستطيع الوصول إليها أي التي لها صلاحية معينة عليها. وهذه تبدأ اسماؤها بـ _ALL.

٣. عرض مدير قواعد البيانات Database Administrator وهي تحتوي على معلومات جميع مكونات قاعدة البيانات لجميع المستعملين. وهذه تبدأ اسماؤها بـ _DBA.

٤. عروض ذات علاقة بأداء قاعدة البيانات أوراكل أثناء تشغيلها Performance Views وتبدأ بـ \$V.



نبدأ بعرض اسمه DICTIONARY وهذا يحتوي على جميع أسماء ووصف جميع جداول وعروض قاموس أوراكل وهيكيلية بياناته هي كما يلي

DESCRIBE DICTIONARY

Name	Null?	Type
TABLE_NAME		VARCHAR2(30)
COMMENTS		VARCHAR2(4000)

فمثلا نريد معلومات عن عرض اسمه USERS_OBJECTS نكتب ما يلي:

```
SELECT *
FROM dictionary
WHERE table_name = 'USER_OBJECTS';
```

TABLE_NAME	COMMENTS
USER_OBJECTS	Objects owned by the user

العرضان USER_OBJECTS، ALL_OBJECTS

كما أسلفنا سابقا فإن العرض USER_OBJECTS يحتوي على معلومات جميع مكونات قاعدة البيانات التي تخصك بينما ALL_OBJECTS يحتوي على جميع معلومات المكونات التي يمكنك الوصول إليها.

بعض الحقول التي تكون العرض USER_OBJECTS مثل:

وصف العمود	اسم العمود
اسم مكون قاعدة البيانات مثل employees.	OBJECT_NAME
رقم المكون.	OBJECT_ID
نوع المكون(جدول، عرض،.....).	OBJECT_TYPE
تاريخ ووقت إنشاء المكون.	CREATED

آخر تاريخ ووقت تم تغيير المكون فيه بواسطة ALTER	LAST_DDL_TIME
حالة المكون(مقبول/غير مقبول VALID/INVALID)	STATUS
هل قامت أوراكل بتكوين اسم المكون(نعم/لا Y/N).	GENERATED

وهناك أعمدة كثيرة أخرى تجدها في كتاب مرجع قاعدة البيانات أوراكل Oracle Database Reference وفيه وصف جميع الجداول والعرض وأعمدتها.

ولنأخذ الاستعلام التالي:

```
SELECT object_name, object_type, created, status
FROM user_objects
ORDER BY object_type;
```

OBJECT_NAME	OBJECT_TYPE	CREATED	STATUS
REG_ID_PK	INDEX	01-JUN-10	VALID
JHIST_EMPLOYEE_IX	INDEX	01-JUN-10	VALID
JHIST_JOB_IX	INDEX	01-JUN-10	VALID
JHIST_EMP_ID_ST_DATE_PK	INDEX	01-JUN-10	VALID
EMP_EMP_ID	INDEX	15-JUL-10	VALID
SYS_C006073	INDEX	15-JUL-10	VALID
LOC_CITY_IX	INDEX	01-JUN-10	VALID
LOC_STATE_PROVINCE_IX	INDEX	01-JUN-10	VALID
LOC_COUNTRY_IX	INDEX	01-JUN-10	VALID
COUNTRY_C_ID_PK	INDEX	01-JUN-10	VALID
DEPT_ID_PK	INDEX	01-JUN-10	VALID
DEPT_LOCATION_IX	INDEX	01-JUN-10	VALID

JHIST_DEPARTMENT_IX	INDEX	01-JUN-10	VALID
JOB_ID_PK	INDEX	01-JUN-10	VALID
LOC_ID_PK	INDEX	01-JUN-10	VALID
EMP_EMAIL_UK	INDEX	01-JUN-10	VALID
EMP_EMP_ID_PK	INDEX	01-JUN-10	VALID
EMP_DEPARTMENT_IX	INDEX	01-JUN-10	VALID
EMP_JOB_IX	INDEX	01-JUN-10	VALID
EMP_MANAGER_IX	INDEX	01-JUN-10	VALID
EMP_NAME_IX	INDEX	01-JUN-10	VALID
SECURE_DML	PROCEDURE	01-JUN-10	VALID
ADD_JOB_HISTORY	PROCEDURE	01-JUN-10	VALID
DEPARTMENTS_SEQ	SEQUENCE	01-JUN-10	VALID

نلاحظ أن قيمة العمود STATUS هي VALID لجميع مكونات قاعدة البيانات وهذا لأنه تم إنشاء هذه المكونات بطريقة صحيحة ولم يطرأ عليها ما يغيرها. وفي حالة الجداول فإن العمود STATUS دائماً يحمل القيمة VALID ولكن في باقي المكونات (العروض والفهارس والسلسل والإجراءات Procedures والدوال Functions والقيود Constraints والخزم Packages والزنانات Triggers فإن هذه الحلة قد تتغير إلى INVALID).

يوجد عرض اسمه **CAT** وهذا يحتوي على أسماء وأنواع جميع مكونات قاعدة البيانات لديك؛ ويستخدم لاستعراض السريع لما لديك من مكونات.

أوراكل مثلا (٤٣)

التعامل مع قاموس أوراكل Managing Oracle Data Dictionary

معلومات الجداول

يوجد عرض خاص اسمه USER_TABLES يحتوي على معلومات الجداول فقط وهذه هي هيكلية بياناته:

Name	Null?	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
TABLESPACE_NAME		VARCHAR2(30)
CLUSTER_NAME		VARCHAR2(30)
IOT_NAME		VARCHAR2(30)
STATUS		VARCHAR2(8)
PCT_FREE		NUMBER
PCT_USED		NUMBER
INI_TRANS		NUMBER
MAX_TRANS		NUMBER
INITIAL_EXTENT		NUMBER
NEXT_EXTENT		NUMBER
MIN_EXTENTS		NUMBER
MAX_EXTENTS		NUMBER
PCT_INCREASE		NUMBER
FREELISTS		NUMBER
FREELIST_GROUPS		NUMBER

LOGGING		VARCHAR2(3)
BACKED_UP		VARCHAR2(1)
NUM_ROWS		NUMBER
BLOCKS		NUMBER
EMPTY_BLOCKS		NUMBER
AVG_SPACE		NUMBER
CHAIN_CNT		NUMBER
AVG_ROW_LEN		NUMBER
AVG_SPACE_FREELIST_BLOCKS		NUMBER
NUM_FREELIST_BLOCKS		NUMBER
DEGREE		VARCHAR2(10)
INSTANCES		VARCHAR2(10)
CACHE		VARCHAR2(5)
TABLE_LOCK		VARCHAR2(8)
SAMPLE_SIZE		NUMBER
LAST_ANALYZED		DATE
PARTITIONED		VARCHAR2(3)
IOT_TYPE		VARCHAR2(12)
TEMPORARY		VARCHAR2(1)
SECONDARY		VARCHAR2(1)
NESTED		VARCHAR2(3)

BUFFER_POOL		VARCHAR2(7)
ROW_MOVEMENT		VARCHAR2(8)
GLOBAL_STATS		VARCHAR2(3)
USER_STATS		VARCHAR2(3)
DURATION		VARCHAR2(15)
SKIP_CORRUPT		VARCHAR2(8)
MONITORING		VARCHAR2(3)
CLUSTER_OWNER		VARCHAR2(30)
DEPENDENCIES		VARCHAR2(8)
COMPRESSION		VARCHAR2(8)
DROPPED		VARCHAR2(3)

ولسنا مطالبين بحفظها فلها المرجع السابق ذكره وهو المرجع لقاعدة البيانات أوراكل. وهذا مثال على استعلام عرض جداول المستعمل : USER_TABLES

SELECT TABLE_NAME FROM USER_TABLES;

TABLE_NAME
TIME_EXAMPLE2
TIME_EXAMPLE
TIME_EXAMPLE3
DEPTS
EMPLOYEES1
COUNTRIES

DEPARTMENTS
EMPLOYEES
JOB_HISTORY
LOCATIONS
REGIONS

وهناك عرض آخر يحتوي أسماء الجداول والعروض ورقم العنقود Cluster ID الذي يتسمى إليه الجدول وهو TAB وهو للاستعراض السريع لأسماء الجداول والعروض.(بحث العناقيد Clusters ليس من ضمن بحثنا هنا).

Column Information

معلومات أعمدة الجداول موجودة في العرض USER_TAB_COLUMNSوها هو جزء من هيكلية بياناته:

DESCRIBE user_tab_columns

Name	Null?	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME	NOT NULL	VARCHAR2(30)
DATA_TYPE		VARCHAR2(106)
DATA_TYPE_MOD		VARCHAR2(3)
DATA_TYPE_OWNER		VARCHAR2(30)
DATA_LENGTH	NOT NULL	NUMBER
DATA_PRECISION		NUMBER
DATA_SCALE		NUMBER
NULLABLE		VARCHAR2(1)

COLUMN_ID		NUMBER
DEFAULT_LENGTH		NUMBER
DATA_DEFAULT		LONG
NUM_DISTINCT		NUMBER
LOW_VALUE		RAW(32)
HIGH_VALUE		RAW(32)
DENSITY		NUMBER
NUM_NULLS		NUMBER
NUM_BUCKETS		NUMBER
LAST_ANALYZED		DATE
SAMPLE_SIZE		NUMBER
CHARACTER_SET_NAME		VARCHAR2(44)
CHAR_COL_DECL_LENGTH		NUMBER
GLOBAL_STATS		VARCHAR2(3)
USER_STATS		VARCHAR2(3)
AVG_COL_LEN		NUMBER
CHAR_LENGTH		NUMBER
CHAR_USED		VARCHAR2(1)
V80_FMT_IMAGE		VARCHAR2(3)
DATA_UPGRADED		VARCHAR2(3)
HISTOGRAM		VARCHAR2(15)

من هذه الأعمدة التي تهمنا:

اسم العمود

نوع العمود

حجم العمود

دقة العمود Precision وعدد الخانات العشرية Scale للأعمدة العددية

إجبارية/عدم إجبارية العمود NULL/NOT NULL

القيمة الافتراضية Default Value

يمكننا قراءة العرض بالاستعلام التالي:

```
SELECT column_name, data_type, data_length,  
       data_precision, data_scale, nullable  
  FROM user_tab_columns  
 WHERE table_name = 'EMPLOYEES';
```

COLUMN_NAME	DATA_TYPE	DATA_LENGTH	DATA_PRECISION	DATA_SCALE	NUL
EMPLOYEE_ID	NUMBER	22	6	0	N
FIRST_NAME	VARCHAR2	20			Y
LAST_NAME	VARCHAR2	25			N
EMAIL	VARCHAR2	25			N
PHONE_NUMBER	VARCHAR2	20			Y
HIRE_DATE	DATE	7			N
JOB_ID	VARCHAR2	10			N
SALARY	NUMBER	22	8	2	Y
COMMISSION_PCT	NUMBER	22	2	2	Y
MANAGER_ID	NUMBER	22	6	0	Y
DEPARTMENT_ID	NUMBER	22	4	0	Y

11 rows selected

أوراكل مثلا (٤٤)

التعامل مع قاموس أوراكل Managing Oracle Data Dictionary

 معلومات قيود التكامل *Constraints Information*

معلومات قيود التكامل موجودة في العرض USER_CONSTRAINTS وفيه معلومات القيود التي تتحكم في الجداول التي تملكها؛ و العرض USER_CONS_COLUMN الذي يحتوي معلومات عن الأعمدة التي تكون القيود. وهذا وصف عرض القيود نفسها.

DESCRIBE user_constraints

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2(30)
R_CONSTRAINT_NAME		VARCHAR2(30)
DELETE_RULE		VARCHAR2(9)
STATUS		VARCHAR2(8)
DEFERRABLE		VARCHAR2(14)
DEFERRED		VARCHAR2(9)
VALIDATED		VARCHAR2(13)
GENERATED		VARCHAR2(14)
BAD		VARCHAR2(3)
RELY		VARCHAR2(4)

LAST_CHANGE		DATE
INDEX_OWNER		VARCHAR2(30)
INDEX_NAME		VARCHAR2(30)
INVALID		VARCHAR2(7)
VIEW RELATED		VARCHAR2(14)

كما تلاحظ، نجد أسماء القيود وأنواعها وأسماء الجداول التي تطبق عليها القيود وشروط التي تخضع لها البيانات Check والجدول المرجع والعمود المرجع وقواعد المسح Deletion Rules وحالة القيد وغير ذلك كثير.

وكمثال استخراج بعض معلومات القيود نكتب الاستعلام التالي:

```
SELECT constraint_name, constraint_type,
```

```
search_condition, r_constraint_name,
```

```
delete_rule, status
```

```
FROM user_constraints
```

```
WHERE table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	CON	SEARCH_CONDITION	R_CONSTRAINT_NAME	DELETE_RULE
EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL		
EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL		
EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL		
EMP_JOB_NN	C	"JOB_ID" IS NOT NULL		
EMP_SALARY_MIN	C	salary > 0		
EMP_EMAIL_UK	U			
EMP_EMP_ID_PK	P			
EMP_MANAGER_FK	R		EMP_EMP_ID_PK	NO ACTION
EMP_JOB_FK	R		JOB_ID_PK	NO ACTION
EMP_DEPT_FK	R		DEPT_ID_PK	NO ACTION

10 rows selected.

قيمة العمود نوع القيد CONSTRAINT_TYPE يمكن أن تكون:



١. C يعني التدقيق على البيانات Check Constraint
٢. P المفتاح الأولي Primary Key
٣. U المفتاح الوحد Unique Key

٤. R مرجعية التكامل Referential Integrity

٥. V خيار التدقيق للعرض فقط Check Option

٦. خيار القراءة فقط للعرض فقط Read Only Option

قيمة العمود نوع القيد DELETE_RULE يمكن أن تكون:

١. CASCADE-التمدد؛ بحيث إذا قمنا بمسح الصف الأب فيتم مسح الصنوف الأبناء تلقائيا

٢. NO_ACTION لا فعل أو لا حدث؛ أي لا يمكن مسح الصف الأب ما دام له صنوف أبناء؛ فيجب مسح الصنوف الأبناء أولا ثم مسح الصف الأب.

قيمة العمود الحالة STATUS يمكن أن تكون:

١. ENABLED-فعال؛ أي أنه يؤثر على البيانات في جمل معالجة البيانات DML.

٢. DISABLED-غير فعال؛ أي لا يؤثر على البيانات في جمل معالجة البيانات DML.

العرض لأعمدة القيود user_cons_columns

وأعمدة هذه العرض قليلة وهي:

DESC user_cons_columns

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME		VARCHAR2(4000)
POSITION		NUMBER

ويمكننا الاستعلام عن الأعمدة التي تكون القييد من هذا العرض وكمثال:

```
SELECT constraint_name, column_name  
FROM user_cons_columns  
WHERE table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_JOB_NN	JOB_ID
EMP_HIRE_DATE_NN	HIRE_DATE
EMP_EMAIL_NN	EMAIL

EMP_LAST_NAME_NN	LAST_NAME
EMP_SALARY_MIN	SALARY
EMP_EMAIL_UK	EMAIL
EMP_EMP_ID_PK	EMPLOYEE_ID
EMP_MANAGER_FK	MANAGER_ID
EMP_JOB_FK	JOB_ID
EMP_DEPT_FK	DEPARTMENT_ID

طبعا لا ننسى أنه يمكن أن يشترك أكثر من عمود في القيد الواحد وبالتالي يتكرر اسم القيد بعدد الأعمدة التي تدخل في تكوينه.

أوراكل مثلاً (٤٥)

التعامل مع قاموس أوراكل Managing Oracle Data Dictionary

معلومات العروض Views Information



توجد معلومات العروض التي تخص مستعمل ما في العرض user_views ومواصفاته هي:

DESC user_views

VIEW_NAME	NOT NULL	VARCHAR2(30)
TEXT_LENGTH		NUMBER
TEXT		LONG
TYPE_TEXT_LENGTH		NUMBER
TYPE_TEXT		VARCHAR2(4000)
OID_TEXT_LENGTH		NUMBER
OID_TEXT		VARCHAR2(4000)
VIEW_TYPE_OWNER		VARCHAR2(30)
VIEW_TYPE		VARCHAR2(30)
SUPERVIEW_NAME		VARCHAR2(30)

أهم الأعمدة هي اسم العرض view_name والنص text والنص هو جملة الاستعلام SELECT التي نشأ منها العرض. فعلى سبيل المثال الاستعلام التالي يجلب لنا نص جملة الاستعلام التي نشأ منها العرض EMP_DETAILS_VIEW

SET LONG 1000

SELECT text FROM user_views

WHERE view_name = 'EMP_DETAILS_VIEW';

TEXT

```
SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.country_id, e.first_name, e.last_name, e.salary, e.commission_pct, d.department_name, j.job_title, l.city, l.state_province, c.country_name, r.region_name
FROM employees e, departments d, jobs j, locations l, countries c, regions r
WHERE e.department_id = d.department_id AND d.location_id = l.location_id AND l.country_id = c.country_id AND c.region_id = r.region_id AND j.job_id = e.job_id WITH READ ONLY
```

لاحظ الأمر SET LONG 1000 و ذلك لأن نص جملة الاستعلام أكبر من حجم الذاكرة المؤقتة BUFFER لذلك تم استعمال هذا الأمر.

معلومات المتسلسلات Sequences Information

و هذه المعلومات موجودة في العرض user_sequences وهذا هو وصفه:

DESCRIBE user_sequences

Name	Null?	Type
SEQUENCE_NAME	NOT NULL	VARCHAR2(30)
MIN_VALUE		NUMBER
MAX_VALUE		NUMBER
INCREMENT_BY	NOT NULL	NUMBER
CYCLE_FLAG		VARCHAR2(1)
ORDER_FLAG		VARCHAR2(1)
CACHE_SIZE	NOT NULL	NUMBER
LAST_NUMBER	NOT NULL	NUMBER

طبعاً لديك معرفة تامة بمعلومات السلسلة من خلال الأعمدة الظاهرة أمامك ما عدا العمود last_number وهو أكبر من آخر قيمة للعمود الزائف CURRVAL راجع بحث السلسلة.

مثال:

```
SELECT sequence_name, min_value, max_value,
increment_by, last_number
FROM user_sequences;
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
LOCATIONS_SEQ	1	9900	100	3300
DEPARTMENTS_SEQ	1	9990	10	280
EMPLOYEES_SEQ	1	1.0000E+27	1	207

معلومات المترادفات Synonyms Information

من خلال استعلام العرض user_synonyms نستطيع الحصول على معلومات المترادفات وهذه المعلومات موجودة في أعمدة العرض السابق.

DESCRIBE user_synonyms

Name	Null?	Type
SYNONYM_NAME	NOT NULL	VARCHAR2(30)

TABLE_OWNER		VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
DB_LINK		VARCHAR2(128)

-١: العمود SYNONYM_NAME : اسم المرادف.

-٢: TABLE_OWNER : اسم المستعمل صاحب الجدول المعمول له مرادف.

-٣: TABLE_NAME : اسم الجدول المعمول مرادف له.

-٤: DB_LINK : اسم رابط قاعدة البيانات بين قاعدة البيانات المحلية Local Database وقاعدة البيانات البعيدة Remote Database وذلك إذا وجدت. (بحث رابط قاعدة البيانات موجودة في مواضع قاعدة البيانات الموزعة).

إضافة ملاحظات على الجداول والأعمدة بعد إنشائهما *Adding Comments*

من المفيد جداً إضافة ملاحظات على الجداول والأعمدة أثناء إنشائهما كنوع من التوثيق؛ ويتم تخزين هذه الملاحظات في قاموس أوركل شأنها شأن أي مكون آخر لقاعدة البيانات. والشكل العام لإضافة ملاحظات هو:

COMMENT ON TABLE table | COLUMN table.column

IS 'comment text';

إضافة نص الملاحظة comment text على جدول TABLE table أو على عمود في جدول COLUMN .table.column

مثال إضافة ملاحظات على جدول الموظفين:

COMMENT ON TABLE employees

IS 'Employee Information';

Comment created.

والإلغاء الملاحظة Drop comment نكتب نفس الأمر السابق ولكن يجعل الملاحظة قيمة لاشيء NULL كما يلي:
COMMENT ON TABLE employees IS '';

أما معلومات الملاحظات فهي موجودة من خلال العروض التالية:

- ALL_COL_COMMENTS
- USER_COL_COMMENTS
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلا (٤٦)

التعامل مع ملفات SQL

١ - والمقصود بالتعامل مع ملفات سكول هو كيفية تخزين جمل سكول SQL على هيئة ملف نصي قابل للتنفيذ من سكول+ أو غيرها. ونستعمل لهذا الغرض الأمر save متبعاً بالمسار واسم الملف. مثال:

```
SQL>SELECT * FROM EMPLOYEES;
```

```
SQL>SAVE D:\E1
```

```
Created file d:\e1.sql
```

تقوم سكول+ بحفظ الجملة السابقة في ملف نصي بالامتداد .sql. على المسار المحدد. نستطيع استعمال أي محرر نصوص مثل المفكرة لكتابة جميع جمل سكول التي نريد تنفيذها دفعه واحد بدلاً من كتابتها كل مرة؛ نحفظ الملف بالامتداد .sql ثم ننفذه من بيئه سكول+ كالتالي على فرض أن اسم الملف هو e1.sql :

```
SQL>@d:\e1
```

أما إذا نسينا وقمنا بحفظ الملف بصيغة txt فلا مشكلة أيضاً؛ فقط نضع الصيغة المناسبة للملف المراد تنفيذه كالتالي على فرض أن اسم الملف هو e2.txt فتكتب الأمر التالي:

```
SQL>@d:\e2.txt
```

فيتم تنفيذ محتويات الملف. يجب أن يكون الملف نصياً بغض النظر عن الأداة التي نستخدمها.

٢ - تخزين نتائج تنفيذ جمل سكول في ملف نصي. نرغب غالباً في تخزين نتائج الاستعلام أو تنفيذ أي جملة من جمل سكول في ملف للتصرف بها لاحقاً. تمكننا بيئه سكول+ من ذلك باستعمال الأمر spool متبعاً بالمسار واسم الملف ثم sool off فينتج لدينا ملف نصي بالامتداد lst كالتالي:

```
SQL>spool empsdata
```

```
SQL>select * from employees;
```

```
SQL>spool off
```

فينتاج لدينا ملف نصي اسمه empsdata.lst على المسار الافتراضي. يمكن فتح الملف من وورد مثلاً وتنسيقه بالطريقة المناسبة. يمكننا تغيير الامتداد من lst إلى أي امتداد آخر وهذا لا يعني تغيير صيغة الملف فيبقى ملفاً نصياً.

٣ - توليد جمل سكول من جملة الاستعلام وتخزينها في ملف. لنفرض أننا نريد مسح بيانات جميع الجداول في المخطط hr. فنقوم عادة بكتابه جملة المسح لكل جدول على حدة ثم نؤكده عملية المسح باستخدام commit. في حالة عندنا

بضعة حداول فالأمر سهل؛ ولكن في حالة وجود عشرات الحداول أو عدة عمليات على الحداول مثل إعطاء الصلاحيات أو إنشاء أسماء متزدفة.... الخ. فالمسألة تصبح صعبة وملة في هذه الحالة. لذلك قدمت لنا أوراكل طريقة لتوليد ما نشاء من جمل سكول بعدد الحداول أو المكونات الأخرى لقاعدة البيانات ومن ثم تخزين هذه الجمل الجديدة في ملف نصي تقوم بتنفيذها من بيئه سكول+ أو أية بيئه متاحة. لتطبيق الفرضية السابقة أي مسح بيانات جميع الحداول كما يلي:

```
SQL>spool delemp.sql
SQL>select 'delete '||table_name||';' from user_tables;
SQL>spool off
SQL>@delemp
SQL>commit;
```

شرح المثال:

١- استخدمنا الأمر spool لإنشاء ملف نصي على المسار الافتراضي ذي الامتداد sql وهو الامتداد الافتراضي الذي تنفذه سكول+ بدون ذكره. تذكر إذا لم نذكر الامتداد فإن سكول تستخدم الامتداد الافتراضي lst وفي هذه الحالة يجب كتابة اسم الملف مع الامتداد مع أمر التنفيذ هكذا: @delemp.lst مثلا.

٢- استخدمنا في جملة الاستعلام الإضافية letiral وهي أمر delete كما سبق شرحه ووصلناه باسم الحدول المأمور من العرض user_tables السابق شرحه في عبارة from . لا حظ أيضاً توصيل اسم الحدول بالفواصل المنقوطة كإضافة لإتمام بناء الجملة

٣- أغلقنا الملف باستخدام spool off

٤- قمنا بتنفيذ الملف باستخدام الأمر @delemp

٥- قمنا بتشبيت عملية المسح باستخدام commit;

((((لا تنفذ هذا المثال كي لا تفقد بياناتك))))

فيما يلي محتويات الملف delemp.sql

```
'DELETE'||TABLE_NAME||';'
```

```
delete LOCATIONS ;
delete PS_TXN1 ;
delete COUNTRIES ;
delete JOB_HISTORY ;
delete REGIONS ;
delete DEPARTMENTS ;
delete EMPLOYEES ;
delete JOBS ;
```

8 rows selected.

```
SQL> spool off
```

المثال التالي سيقوم بتوليد جمل سكول لأنشاء متراادات عامة public synonym للجداول في المخطط hr وسوف تأخذ نفس أسماء الجداول:

```
SQL>select 'create public synonym '||table_name||' for hr.'||table_name||';' from user_tables;
```

جرب هذا المثال فلا خطورة كبيرة منه!

**** أنا شخصياً أستخدم هذا الأسلوب كثيراً

بسم الله الرحمن الرحيم

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٤٧)

السرية(الأمنية) في أوراكل Security in Oracle

في الفقرات التالية سألقي الضوء على السرية في أوراكل؛ وذلك من خلال التحكم في الوصول إلى البيانات؛ حيث سنرى أنواع الصالحيات وكيفية منحها ونزعها من المستخدمين وكيفية إنشاء المستخدمين وإنشاء مجموعة من الصالحيات تسمى أدواراً Roles وغيرها ذلك من أمور شيقة.

+ الصالحيات؛ هي إعطاء الحق للمستخدم بتنفيذ جملة سكول محددة وذلك باستعمال لغة التحكم بالبيانات DCL. ومدير قاعدة البيانات DBA هو أعلى مستوى من المستخدمين؛ فلديه القدرة على إنشاء المستخدمين ومنحهم صالحيات على قاعدة البيانات. فالمستخدم يحتاج إلى صالحيات النظام System Privileges للوصول إلى قاعدة البيانات ويحتاج إلى صالحيات المكونات Object Privileges ليتمكن من الوصول إلى البيانات في هذه المكونات ومن ثم التعامل معها إضافة إلى ذلك يمكن للمستخدم إعطاء صالحيات على المكونات التي يمتلكها إلى مستخدمين آخرين. وعليه فإن أمن قاعدة البيانات Database ينقسم إلى نوعين: Security

١ - صالحيات النظام System Proivilges مثل إنشاء الجداول والفهارس... وإسقاطها وتغييرها.... باستعمال لغة تعريف البيانات DDL

٢ - صالحيات المكونات Object Privilegs أي الوصول إلى البيانات والتعامل معها من إضافة وحذف.... باستعمال لغة معالجة البيانات DML

 المخطط Schema هو تجمع من مكونات قاعدة البيانات مثل الجداول والفالئرس والعروض وغيرها. والمخطط مملوك لمستخدم ما ويأخذ نفس الاسم. مثلاً المخطط hr مملوك للمستخدم hr وهذا يعني أن المستخدم قد لا يملك مخططاً أي ليس لديه أي مكون من مكونات قاعدة البيانات.

صلاحيات النظام System Privileges

يوجد ما يزيد عن ١٠٠ من هذه الصلاحيات؛ مثل إنشاء المستخدمين Users والأدوار Roles وفي الجدول التالي بعض هذه الصلاحيات:

الصلاحية Privilege	وصف الصلاحية Description
CREATE USER	من يملك هذه الصلاحية يمكنه إنشاء مستخدمين
DROP USER	من يملك هذه الصلاحية يمكنه إسقاط مستخدمين
DROP ANY TABLE	من يملك هذه الصلاحية يمكنه إسقاط أي جدول حتى لو لم يملكه
BACKUP ANY TABLE	من يملك هذه الصلاحية يمكنه عمل نسخ احتياطي لأي جدول
SELECT ANY TABLE	صاحب هذه الصلاحية يمكنه الاستعلام من أي جدول
CREATE ANY TABLE	صاحب هذه الصلاحية يمكنه من إنشاء أي جدول على أي مخطط

 إنشاء المستخدمين Creating users في قاعدة البيانات؛ وإنشاء مستخدم ما نستعمل جملة CREATE من لغة تعريف البيانات DDL كالتالي:

CREATE USER *username*

IDENTIFIED BY *password*;

حيث *username* اسم المستخدم ويبدأ بحرف بما لا يزيد عن ٣٠ رمزا

كلمة السر بما لا يزيد عن ٣٠ رمزا  Password

مثال

CREATE USER HR1

IDENTIFIED BY HR1;

User created.

هنا أنشأنا المستخدم hr1 وأعطيته كلمة السر hr1 أيضا.

إلى هنا أنشأنا المستخدم hr1 وهذا لا يعني أنه يستطيع الوصول إلى قاعدة البيانات أو التصرف فيها أو في بعض مكوناتها؛ بل يجب أن يحصل على صلاحيات يحددها مدير قاعدة البيانات وحسب متطلبات العمل.

 منح الصلاحيات على النظام.

في الجملة التالية الصيغة العامة لمنح الصلاحيات:

GRANT *privilege* [, *privilege...*] TO *user* [, *user/ role*, PUBLIC...];

شرح الجملة السابقة:

- ١ GRANT لمنح الصلاحيه وهي من لغه التحكم بالبيانات DCL
- ٢ privilege وهي سلسله الصلاحيات المراد منحها للمستخدم تفصيل تفصل بينها فاصلة. ما بين القوسين المربعين اختياري
- ٣ TO user[,user|role,PUBLIC,...] هنا يتم ذكر أسماء المستخدمين الذين نريد منحهم صلاحيات معينة. أو roles. أو للعامة أي لكل المستخدمين باستعمال الكلمة المحفوظة PUBLIC.

في الجدول التالي بعض أنواع الصلاحيات

الصلاحية <i>privilege</i>	الوصف <i>description</i>
CREATE SESSION	إنشاء جلسة العمل؛ أي الاتصال بقاعدة البيانات
CREATE TABLE	صلاحية إنشاء جداول على المخطط الذي يملكه المستخدم
CREATE SEQUENCE	صلاحية إنشاء متسلسلات على المخطط الذي يملكه المستخدم
CREATE VIEW	صلاحية إنشاء عروض على المخطط الذي يملكه المستخدم
CREATE PROCEDURE	صلاحية إنشاء إجراءات-procedures على المخطط الذي يملكه المستخدم

مثال:

GRANT create session, create table, create sequence, create view

TO hr1;

Grant succeeded.

في هذا المثال تم منح صلاحيات الاتصال بقاعدة البيانات وإنشاء جداول ومتسلسلات وعروض. راجع الجدول أعلاه. هذا وهناك صلاحية منح حصة TABLESPACE مساحة الجداول QUOTA كما يلي:

Alter user hr1 quota 5m on users;

هنا تم منح حصة مقدارها 5 م.ب. على مساحة الجداول الذي اسمه users وعليه فإن المستخدم hr1 لن يستطيع استعمال أكثر من 5 م.ب. لبياناته على مساحة الجدول هذه. في نهاية الدروس سوف أتطرق إلى هذه المفاهيم.

 إنشاء الأدوار Creating roles في قاعدة البيانات؛ والدور هو اسم لمجموعة صلاحيات والتي يمكن منحها لمستخدم أو دور آخر.

ولإنشاء دور role ما نستعمل جملة CREATE من لغة تعريف البيانات DDL كالتالي:

CREATE ROLE rolename;

حيث rolename اسم الدور ويبدأ بحرف بما لا يزيد عن ٣٠ رمزا. وبعد إنشاء الدور يقوم مدير قاعدة البيانات أو المستخدمين منح الصلاحيات للدور.

مثال:

CREATE ROLE managers;

Role created.

والآن سنعطي صلاحيات للدور manager.

GRANT create table, create view

TO managers;

Grant succeeded.

والآن سنعطي الدور بعض المستخدمين.

GRANT managers TO hr1, scott;

Grant succeeded.

إذا أردنا منح الصلاحيات في المثال إلى المستخدمين فسوف نضطر أن تكتب جملة إعطاء الصلاحيات لكل مستخدم على حدة وعلى افتراض أن لدينا مئات المستخدمين بلآلافا منهم فتصير العملية عقيمة؛ فنلجاً إلى إنشاء عدد مناسب من الأدوار وكل دور نعطيه عدداً من الصلاحيات -قد تشتراك الأدوار في أكثر من صلاحية- ثم نعطي كل مجموعة مستخدمين الدور المناسب لهم أو الأدوار المناسبة. وهذا أسهل من كتابة جمل صلاحيات لكل مستخدم.

عند منح صلاحية أو صلاحيات للدور فإن المستخدمين يرثون هذه الصلاحيات من الدور؛ وفي المقابل عند نزع صلاحية معينة من الدور فإن تأثير هذا النزع يمتد إلى المستخدمين الذين ورثوا الصلاحية من الدور.

في مثالنا منحنا صلاحيات إنشاء الجداول والعرض والمستسلسلات إلى الدور managers والمستخدمان hr1 و scott هما نفس الصلاحيات. فلو نزعنا صلاحية إنشاء المستسلسلات من الدور managers فلنلقائياً تنزع من المستخدمين hr1 و scott.

ولنزع الصلاحية نستخدم جملة REVOKE كالتالي:

REVOKE create table from managers;

تلقاءياً يتم نزع صلاحية إنشاء جداول من hr1 و scott.

 تغيير كلمة السر للمستخدم. يمكن للمستخدم تغيير كلمة السر التي تخصه بالجملة التالية:

ALTER USER username IDENTIFIED BY newpassword;

مثال: المستخدم الذي أنشأناه سابقاً يريد تغيير كلمة السر الخاصة به؛ فيكتب بعد الدخول على قاعدة البيانات ما يلي:

ALTER USER hr1 IDENTIFIED BY hrmngr;

لا يستطيع المستخدم hr1 تغيير كلمات السر للمستخدمين الآخرين؛ بينما يستطيع مدير قاعدة البيانات DBA ذلك. ولكن يمكن أن يقوم الـDBA بمنع هذع الصلاحية لمستخدم آخر فيماكنته في هذه الحالة تغيير كلمة السر للمستخدمين الآخرين.

 منح الصلاحيات على مكونات قاعدة البيانات؛ تشمل الصلاحيات على مكونات قاعدة البيانات عاليات الـDML والـDCL وعينة منها في الجدول التالي حيث (نعم) تعني إمكانية إعطاء الصلاحية

View العرض	Table الجدوال	Sequence المتسلسلة	Procedure الإجراءات	الصلاحية/المكون Object/privilgr
	نعم	نعم		ALTER
نعم	نعم			DELETE
			نعم	EXECUTE

نعم	نعم			INDEX
نعم	نعم			INSERT
	نعم			REFERENCES
نعم	نعم	نعم		SELECT
نعم	نعم			UPDATE

والصلاحيات على الجدول ممكن أن تقتصر على أعمدة معينة دون الأخرى. الصيغة التالية هي لإعطاء الصلاحيات على مكونات قاعدة البيانات:

```
GRANT object_priv[(columns)]

ON      object

TO      {user|role|PUBLIC}

[WITH GRANT OPTION];
```

حيث:

١ - نوع الصلاحية واحتيارياً على عمود أو أعمدة معينة في الجدول أو العرض *Object_priv*[(*columns*)]

٢ - اسم مكون قاعد البيانات مثل *Object* employees

٣ - هنا نختار نوع من سمعتي له الصلاحية؛ هل هو مستخدم *user* أو دور *role* أو لل العامة *PUBLIC*

٤ - عبارة اختيارية تعني أنه يمكن لمن تم إعطاؤه صلاحية معينة أن يمنحها لآخرين [WITH GRANT OPTION]

أمثلة:

```
GRANT select ON employees TO sue, rich;
```

Grant succeeded.

في هذا المثال أعطى المستخدم hr صلاحية القراءة على جدول الموظفين employees للمستخدمين sue و rich.

يستطيع المستخدمان sue و rich كتابة جمل الاستعلام على جدول الموظفي في المخطط hr كما يلي:

```
SELECT * FROM HR.employees;
```

في المثال التالي فإن المستخدم hr يعطي صلاحيه تحديث العمودين department_name و location_id في جدول departments . managers للمسخدم scott والدور .

```
GRANT update (department_name, location_id)
```

```
ON departments TO scott, managers;
```

Grant succeeded.

أما المستخدم scott فيمكنه تحدث اسم وموقع القسم إذا أراد كما يلي :

```
UPDATE hr.departments
```

```
SET department_name = 'Torism'
```

```
WHERE department_id = 70;
```

وكذلك أي مستخدم ضمن الدور managers

قواعد ارشادية

١ - منح الصلاحيات على أحد مكونات قاعدة البيانات، يجب أن يكون المكون في خططك الخاص your schema ، أو يجب أن تكون قد حصلت على الصلاحيات على المكون من خلال الخيار .WITH GRANT OPTION

٢ - يمكن مالك المكون منح أي صلاحيه على المكون لأي مستخدم أو دور آخر لقاعدة البيانات.

٣ - يكتسب مالك المكون تلقائياً جميع الصلاحيات على هذا المكون.

يجب ذكر اسم المخطط hr قبل اسم الجدول متبعاً بنقطة لتبين آلية أوراكل Oracle Engin أن هذا المكون مملوك للمستخدم hr.

وبدليل يمكن للمستخدم hr إنشاء مزدوج يعطيه اسمه للمستخدمين الآخرين بدلاً من الاسم الأصلي للجدول كنوع من السرية: كما يلي :

```
CREATE SYNONYM emp FOR hr.employees;
```

وهنا نكتب اسم المزدوج بدلاً من اسم الجدول مسبوقاً باسم المخطط أيضاً

```
SELECT * FROM hr.emp;
```

أما إذا أردنا عدم ذكر اسم المخطط فيتوجب علينا إنشاء مزدوج عام PUBLIC SYNONYM كما مر شرحه سابقاً.

فنكتب مباشرةً اسم المزدوج من غير اسم المخطط

```
SELECT * FROM emp;
```

ويجب ملاحظة أن المستخدم الذي يريد إنشاء المرادف يحتاج إلى صلاحية إنشاء المرادف سواءً كان عاماً أو خاصاً private. والذي يمنح هذه الصلاحية هو مدير قاعدة البيانات DBA كما يلي:

```
GRANT CREATE SYNONYM TO HR;
```

```
GRANT CREATE PUBLIC SYNONYM TO HR;
```

تمرير الصلاحيات إلى مستخدمين آخرين

ويقصد بذلك أن المستخدم الذي تم منحه صلاحيات معينة على بعض مكونات قاعدة البيانات يمكنه منح (تمرير) هذه الصلاحيات إلى مستخدمين آخرين؛ وذلك باستعمال الخيار WITH GRANT OPTION. مثال على ذلك:

```
GRANT select, insert
```

```
ON departments TO scott
```

```
WITH GRANT OPTION;
```

Grant succeeded.

هنا يستطيع المستخدم scott منح (تمرير) الصلاحيات التي أخذها على الجدول departments إلى مستخدمين آخرين.

أما إذا أراد صاحب المخطط منح الصلاحيات لعموم المستخدمين في قاعدة البيانات على أحد المكونات التي يملكتها فإنه يستخدم الكلمة المحوزة PUBLIC لهذا الغرض: كما يلي:

```
GRANT select
```

```
ON alice.departments
```

```
TO PUBLIC;
```

Grant succeeded.

لاحظ أن اسم الجدول departments مسبوق باسم المخطط alice وهذا يعني أن مستخدماً آخر له صلاحية الاستعلام من جدول الأقسام الذي يخص المخطط alice مع صلاحية منح ما اكتسبه من صلاحية إلى مستخدمين آخرين أو إلى عموم المستخدمين.

تأكيد الصلاحيات الممنوحة

لمعرفة الصلاحيات الممنوحة ومن هم الذين تم منح الصلاحيات لهم؛ يمكننا استطلاع العروض المذكورة في الجدول التالي:

الوصف	اسم العرض
-------	-----------

System privileges	صلاحيات النظام المعطاة للأدوار granted to roles	ROLE_SYS_PRIVS
Table privileges	الصلاحيات المعطاة للأدوار على الجداول granted to roles	ROLE_TAB_PRIVS
Roles accessible by the user	الأدوار الممنوحة للمستخدمين user	USER_ROLE_PRIVS
Object privileges granted on the user's objects	الصلاحيات على الجدول التي أعطاها صاحب الجدول privileges granted on the user's objects	USER_TAB_PRIVS_MADE
Object privileges granted to the user	صلاحيات مكونات قاعدة البيانات التي أعطاها المالك لمستخدم Object privileges granted to the user	USER_TAB_PRIVS_REC'D
Object privileges granted on the columns of the user's objects	الصلاحيات التي أعطاها مستخدم ما على ما يملك إلى الأدوار Object privileges granted on the columns of the user's objects	USER_COL_PRIVS_MADE
Object privileges granted on the columns of the user's objects	الصلاحيات الممنوحة للأدوار على أعمدة معينة من الجداول	USER_COL_PRIVS_REC'D
System privileges granted to the user	صلاحيات النظام الممنوحة للمستخدمين granted to the user	USER_SYS_PRIVS

ما يحدث هو أن أوراكل يستطيع قاموس البيانات باستخدام العروض السابقة ليقرر تنفيذ عملية ما على شيء لمخطط آخر؛ فإن وجد أن لك صلاحية على هذا الشيء قام بتنفيذ هذه العملية وإلا أعطاك رسالة الخطأ بأنك لا تملك هذه الصلاحية أو أن الشيء غير موجود!

إلغاء الصلاحيات

يمكن للمالك إلغاء الصلاحيات التي أعطاها الآخرين (مخدمين أو أدوار) باستخدام جملة REVOKE والصيغة العامة هي:

REVOKE {privilege [, privilege...] | ALL}

ON object

FROM {user[, user...] | role | PUBLIC}

[CASCADE CONSTRAINTS];

حيث يمكن نزع صلاحيات معينة ببعاد هذه الصلاحيات أو نزع جميع الصلاحيات على كائن معين باستخدام ALL بدلًا من ذلك. ثم ببعاد المستخدمين أو الأدوار أو العامة PUBLIC. والخيار الأخير CASCADE CONSTRAINTS يعني اصطدام القيود التي تم عملها بواسطة المستخدم الذي له صلاحية إنشاء القيود على جدول ما في المخطط الذي منحه هذه الصلاحية.

ملاحظات:

- ١- إذا تم نزع الصلاحية من العامة فهذا يعني أن جميع المستخدمين لن يتمكنوا من التعامل مع الشيء الذي نزع الصلاحية عليه.
- ٢- إذا نزع الصلاحية من دور معين ROLE فإنه يتم نزع الصلاحية من جميع المستخدمين المندرجين ضمن هذا الدور.
- ٣- إذا تم منح الصلاحية لمستخدم آخر باستخدام الخيار WITH GRANT OPTION وقام هذا المستخدم بمنع الصلاحية لمستخدمين آخرين؛ فإنه إذا نزع منه الصلاحية فإنها تنزع من جميع المستخدمين أيضا الذي أكتسبوا الصلاحية من أول مستخدم. مثلاً؛ قام المالك HR بمنع صلاحية القراء على جدول الموظفين EMPLOYEES إلى الخيار ALICE WITH GRANT OPTION وقام ALICE بمنح الصلاحية ل SCOTT بنفس الخيار وقام SCOTT بمنح الصلاحية ل AHMAD. ثم بسبب ما قام HR بنزع الصلاحية من ALICE فإن الصلاحية تنزع أيضاً من SCOTT و AHMAD.

لفرض أن المستخدم ALICE منح صلاحية الاستعلام وإضافة سطور على جدول الموظفين EMPLOYEES الذي يمتلكه في مخططه للمستخدم SCOTT فيمكنه نزع تلك الصلاحيات من خلال الجملة التالية:

```
REVOKE select, insert
```

```
ON departments
```

```
FROM scott;
```

```
Revoke succeeded.
```

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٤٨)

إدارة مكونات قاعدة البيانات

Managing Database objects

في الدروس التالية سوف نتعلم التصرف بمكونات قاعدة البيانات مثل إضافة قيود وإنشاء الفهارس وغير ذلك من أمور مهمة جداً.

بداية سوف ندرس جملة ALTER تغيير هيكلية مكونات قاعدة البيانات. تستعمل هذه الجملة للأغراض التالية:

- ١- إضافة عمود جديد على الجدول
- ٢- تحديث بنية عمود موجود في الجدول
- ٣- تعريف قيمة افتراضية للعمود
- ٤- إسقاط عمود من الجدول
- ٥- إضافة قيود على الجدول
- ٦- أغراض أخرى تذكر في حينها

إضافة عمود جديد على الجدول؛ لهذا الغرض نستعمل جملة التغيير كما يلي:

ALTER TABLE *table*

ADD *(column datatype [DEFAULT *expr*]
[, column datatype]...);*

حيث *table* اسم الجدول وعبارة ADD لإضافة عمود أو أعمدة وتضمين خيار القيمة الافتراضية DEFAULT. أي يمكننا إضافة أكثر من عمود دفعة واحدة. مثال:

ALTER TABLE dept80

ADD job_id VARCHAR2(9));

Table altered.

على افتراض أن لدينا الجدول dept80 الذي لا يحتوي على رمز الوظيفة job_id. ولعرض هيكلية الجدول نستعمل الأمر DESCRIBE السابق شرحه

DESC dept8

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE
JOB_ID		VARCHAR2(9)

ملاحظات:

- ١- الأعمدة الجديدة تظهر آخر شيء باستعمال الأمر `describe`
- ٢- وجود الأعمدة الجديدة آخراً لا يعني منع استعمالها كأول عمود في جملة `SELECT` كما مر معنا فإننا لا نحتم بترتيب العمود في الجدول ولكن نحتم بما نريد ونظام إدارة قواعد البيانات DBMS يتولى الاستجابة لما نريد.
- ٣- إذا كان الجدول يحتوي بيانات فإن العمود الجديد يحتوي قيمة اللاشيء `NULL` ابتداءً ثم يمكن تحدث قيمته أو إضافة سطور جديدة إلى الجدول تتضمن قيمة للعمود الجديد.
- ٤- في حالة وجود بيانات سابقة في الجدول لا يمكنك وضع قيد منع اللاشيء `NOT NULL` على العمود الجديد. ولكن فقط في حالة أن الجدول فارغ يمكنك ذلك.

تغيير مواصفات عمود؛ بتغيير نوع العمود أو زيادة طول العمود أو إنفاس طول العمود أو تغيير القيمة الافتراضية؛ مثال:

```
ALTER TABLE dept80
MODIFY last_name VARCHAR2(30);
```

Table altered.

هنا غيرنا طول العمود `last_name` إلى ٣٠ رمزاً.

وهذه بعض الإرشادات لتغيير مواصفات عمود:

- ١- يمكننا زيادة طول العمود العددي أو زيادة الدقة
- ٢- يمكننا زيادة طول العمود الرمزي أي نوع `CHAR/VARCHAR2`
- ٣- يمكننا إنفاس طول العمود في الحالات التالية:
 - أ. إذا كان العمود يحتوي قيم اللاشيء فقط `.NULL VALUES`.
 - ب. إذا كان الجدول فارغاً.

ت. إذا كان طول العمود الجديد يستوعب أكبر قيمة موجودة في هذا العمود؛ أي أن لا يقل طول العمود الجديد عن طول القيمة الموجودة أصلاً في العمود.

٤ - يمكننا تغيير نوع العمود بشرط أن يكون فارغاً-يحتوي قيم اللا شيء فقط NULL VALUES إلا في حالة نوع العمود الرمزي من رمز CHAR إلى رمز متغير VARCHAR2 أو بالعكس بشرط عدم تغيير طول العمود.

٥ - تغيير القيمة الافتراضية DEFAULT VALUE يؤثر على السطور (السجلات) الجديدة فقط

 إسقاط (إلغاء) عمود من الجدول؛ يمكننا إسقاط عمود لا نحتاجه في جدول ما - سواءً كان في العمود بيانات لغرض توفير مساحة على قاعدة البيانات - أم لم يكن فيه بيانات. مثال:

```
ALTER TABLE dept80
```

```
DROP COLUMN job_id;
```

Table altered.

في هذا المثال أسلقنا العمود JOB_ID الذي أضفناه سابقاً من الجدول DEPT80 بغض النظر عن وجود أو عدم وجود قيم في العمود.

 إرشادات:

١ - ممكن أن يحتوي العمود المراد إسقاطه على قيم أو لا يحتوي؛ فلا يتشرط لإسقاط عمود أن يحتوي على قيم.

٢ - يتم إسقاط عمود واحد فقط مع كل جملة ALTER DROP .

٣ - لا يمكن إسقاط العمود الوحيد في الجدول؛ فيجب أن يحتوي الجدول على عمود واحد على الأقل.

٤ - لا يمكن استرجاع العمود الذي تم إسقاطه ما لم يكن هناك نسخة احتياطية من الجدول الذي أسلقنا منه العمود ومع ذلك لا يمكن عودة البيانات التي ليس لها نسخة احتياطية

٥ - إذا كان العمود المراد إسقاطه جزءاً من قيد أو فهرس فيجب استخدام الخيار CASCADE في جملة التغيير ALTER .

٦ - من الممكن أن تأخذ عملية إسقاط عمود وقت طويلاً إذا كان الجدول كبيراً؛ وفي هذه الحالة نغير حالة العمود إلى غير مستعمل UNUSERD (انظر الدرس التالي) ثم إسقاطه عندما يكون عدد المستخدمين قليلاً أو أن الضغط على قاعدة البيانات أقل مما يمكن

٧ - هناك أعمدة لا يمكن إسقاطها إذا كانت جزءاً من مفتاح التقسيم partitioned key من جدول مقسم table . أو كان جزءاً من المفتاح الأولي primary key لجدول إذا كان خاضعاً بنظام الفهرسة - لسنا بصدد النقطتين السابقتين فهما ضمن شرح إدارة قاعدة البيانات.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٤٩)

إدارة مكونات قاعدة البيانات

Managing Database objects

التأشير على عمود أنه غير مستعمل Marking column as unused

بدلاً من أن نلجأ لإسقاط عمود من الجدول مما قد يسبب بطءاً في الأنظمة الكبيرة؛ فإننا نضع علامة أو إشارة على العمود أنه غير مستعمل وهذا لا يؤثر على أداء النظام. كل ما في الأمر أن أوراكل تسجل في قاموس البيانات أن هذا العمود غير مستعمل؛ واسم العرض هو USER_UNUSED_COL_TABS. ثم لاحقاً في وقت لا يكون هناك ضغط على النظام تقوم بـإسقاط الأعمدة غير المستعملة. وضع علامة أن العمود غير مستعمل لا يحرر أية مساحة من على الديسک أي أن البيانات تبقى كما هي؛ ولكن لا يمكن الوصول إليها لأن العمود غير موجود ولا يمكن أن نراه عندما نستعرض هيكلية العمود ولا يتم اختياره في جملة SELECT. لذلك من الممكن أن نضيف عموداً جديداً للجدول بنفس اسم العمود الذي أشرنا عليه أنه غير مستعمل؛ والإرشادات التي ذكرناها في عملية إسقاط عمود سابقاً تطبق على عملية التأشير على عمود أنه غير مستعمل.

ولتنفيذ ذلك نكتب الجملة التالية بشكل عام:

```
ALTER TABLE <table_name>
SET UNUSED (<column_name>);
```

```
ALTER TABLE <table_name>
SET UNUSED COLUMN <column_name>;
```

ثم نقوم بـإسقاط هذه الأعمدة كما يلي:

```
ALTER TABLE <table_name>
DROP UNUSED COLUMNS;
```

مثال للتأشير على عمود كغير مستعمل ثم اسقاطه:

```
ALTER TABLE dept80
SET UNUSED (last_name);
```

Table altered.

```
ALTER TABLE dept80
DROP UNUSED COLUMNS;
```

Table altered.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوكل مثلاً (٥٠)

إدارة مكونات قاعدة البيانات

Managing Database objects

التعامل مع القيود؛ وذلك بإضافة قيد أو إسقاط قيد أو تعطيل قيد وإضافة قيد إجبارية الإدخال NOT NULL. لا يوجد تحديث هيكلية قيد بل إسقاطه وإنشاؤه من جديد.

١ - لإضافة قيد نستعمل الجملة التالية:

```
ALTER TABLE <table_name>
```

```
ADD [CONSTRAINT <constraint_name>]
```

```
type(<column_name>);
```

حيث:

tablename : اسم الجدول الذي نريد إضافة قيد عليه

constraint_name : اسم القيد

type : نوع القيد(مفتاح أولي، مفتاح وحيد،....) راجع أنواع القيود

column_name : اسم أو أسماء الأعمدة المشتركة في القيد

ملاحظة: ما بين الأقواس المربعة اختياري

مثال ١ :

```
ALTER TABLE emp2
```

```
Modify employee_id Primary Key;
```

```
Table altered.
```

هنا قمت بإضافة القيد-مفتاح أولي-على الجدول بتغيير هيكلية العمود employee_id. طبعاً نتذكر كيفية تغيير هيكلية عمود باستعمال modify مع alter table . لاحظ أن أوكل ستقوم بتسمية القيد لأننا لم نذكر له اسمًا في جملة التغيير.

-٢

```
ALTER TABLE emp2
```

```
ADD CONSTRAINT emp_mgr_fk
```

```
FOREIGN KEY(manager_id)
```

```
REFERENCES emp2(employee_id);
```

Table altered.

في هذا المثال تمت إضافة قيد اسمه emp_mgr_fk من نوع مفتاح أجنبي foreign key باستخدام العمود

emp2 employee_id العمود references manager_id الذي مرجعه نفس الجدول

٢ - خيار التمدد مع الحذف ON DELETE CASCADE

نتذكر أننا عندما نضيف قيد المفتاح الأجنبي على جدول ما؛ فإن أوراكل تمنع حذف الصف الذي يحتوي على المفتاح الأولي من الجدول الأب ما دام هناك صفات في الجدول الابن تابعة لهذا الصف في الجدول الأب. ولتجاوز هذه الخاصية فإننا نضيف عبارة ON DELETE CASCADE في تعريف المفتاح لأجنبي بحيث يتم مسح الصفات الابن مع مسح الصف الأب. مثال:

```
ALTER TABLE Emp2 ADD CONSTRAINT emp_dt_fk
```

```
FOREIGN KEY (Department_id)
```

```
REFERENCES departments(department_id) ON DELETE CASCADE;
```

Table altered.

هنا تقوم أوراكل بحذف الصفات من جدول الموظفين الذين لهم نفس رقم القسم الذي نحذفه من جدول الأقسام (مثلاً قسم ٣٠ في جدول الأقسام له أبناء ٤ صفات في جدول الموظفين) فعند حذف القسم ٣٠ من جدول الأقسام يتم مسح الـ ٤ صفات من جدول الموظفين.

 إسقاط القيود Dropping constraints واسقاط القيد يعني أن البيانات المدخلة لم تعد تخضع لذلك القيد. فلو

أسقطنا القيد مفتاح أولي مثلاً؛ فإن التكرار وارد في قيمة المفتاح الذي لم يعد أولياً. وكذلك إسقاط المفتاح الأجنبي يمكننا من إضافة صفات في الجدول الذي كان(ابنا) بدون التقيد بقيم العمود الذي كان(أباً) في الجدول الذي كان(أباً).

مثال:

```
ALTER TABLE emp2
```

```
DROP CONSTRAINT emp_mgr_fk;
```

Table altered.

هنا أسقطنا المفتاح الأجنبي من الجدول emp2 الذي هو أب لنفسه. وهذا يعني أننا يمكننا مسح بيانات مدير حتى لو كان تحته موظفون؛ ويمكننا إضافة موظفي برقم مدير غير موجود أصلاً في الجدول. تذكر أن رقم المدير هو رقم موظف أصلاً....

تعطيل وتفعيل القيود DISABLE & ENABLE

أولاً: تعطيل القيود؛ من الممكن تعطيل القيود لفترة ثم تفعيلها مرة ثانية بدلاً من اسقاطها وإنشائها مرة ثانية لأغراض معينة وذلك من خلال جملة التغيير ALTER. ومثال على ذلك:

```
ALTER TABLE emp2  
DISABLE CONSTRAINT emp_dept_fk;  
  
Table altered.
```

هنا قمنا بتعطيل القيد emp_dept_fk والذي يمكننا تنشيطه فيما بعد واستعمال نفس الجملة مع استبدال التنشيط DISABLE بالتعطيل ENABLE هكذا

```
ALTER TABLE emp2  
ENABLE CONSTRAINT emp_dt_fk;  
  
Table altered.
```

إرشادات:

- ١ - استعمال خيار CASCADE يؤدي إلى تعطيل القيود المعتمدة على القيد الذي تم تعطيله
- ٢ - تعطيل القيدين وحيد UNIQUE والمفتاح الأولي PRIMARY KEY يؤدي إلى اسقاط الفهارس الوحيدة التي نتجت عنهما لدى إنشائهما

والصيغة العامة لتعطيل القيد هي:

```
ALTER TABLE table  
DISABLE CONSTRAINT constraint [CASCADE];
```

ثانياً: تنشيط القيود؛ ويمكن تنشيط القيود من خلال جملة التغيير ALTER ومثال ذلك:

```
ALTER TABLE emp2  
ENABLE CONSTRAINT emp_dept_fk;  
  
Table altered.
```

في المثال تم تنشيط القيد emp_dept_fk الذي عطناه سابقاً.

إرشادات:

- ١ - عند تنشيط القيد؛ فإنه يطبق على جميع بيانات الجدول فيجب أن تكون هذه البيانات خاضعة للقيد وإلا لن يتم تنشيط القيد
- ٢ - عند تنشيط أحد القيدين وحيد unique أو مفتاح أولي primary key فيتم إنشاء الفهارس الوحيدة آلياً بشرط أن يكون للمستخدم صلاحية إنشاء الفهارس. إذا كانت الفهارس موجودة أصلاً فيتم استعمالها آلياً بواسطة هذين القيدين.
- ٣ - من الممكن استعمال الخيارين enable و disable من خلال جملة إنشاء الجدول create و جملة تغيير الجدول alter
- ٤ - تنشيط المفتاح الأولي الذي سبق تعطيله لا ينشط أو ينشأ أية قيود كانت تعتمد عليه

 العلاقة بين اسقاط الأعمدة و تعطيل القيود:

- ١ - يمكن استخدام عبارة DROP COLUMN مع عبارة CASCADE CONSTRAINTS
- ٢ - استخدام عبارة CASCADE CONSTRAINTS يؤدي إلى اسقاط جميع القيود المرتبطة بالأعمدة التي تم تعريف المفتاح الأولي PRIMARY KEY عليها أو المفتاح الوحد UNIQUE KEY.
- ٣ - أيضاً يؤدي استعمال الخيار CASCADE CONSTRAINTS إلى اسقاط جميع القيود متعددة الأعمدة المرتبطة بالأعمدة التي تم اسقاطها

ولتوضيح النقاط السابقة؛ نستعرض المثال الطويل التالي:

لنفرض أننا أنشأنا الجدول test1 كما يلي

```
CREATE TABLE test1 (
    pk NUMBER PRIMARY KEY,
    fk NUMBER,
    col1 NUMBER,
    col2 NUMBER,
    CONSTRAINT fk_constraint FOREIGN KEY (fk) REFERENCES test1,
    CONSTRAINT ck1 CHECK (pk > 0 and col1 > 0),
    CONSTRAINT ck2 CHECK (col2 > 0);
```

والآن سنتكتب جملتي التغيير التاليتين من غير الخيار cascade constraints

```
ALTER TABLE test1 DROP (pk);
```

سوف تلقى الخطأ (pk is a parent key) وذلك لأن القيد pk عبارة عن مرجع لقيد المفتاح الأجنبي .fk_constraint

```
ALTER TABLE test1 DROP (col1);
```

سوف تلقى الخطأ التالي .col1 is referenced by multicolumn constraint ck1. وهو أن العمود col1 يشترك مع أعمدة أخرى في القيد ck1.

مثال على استخدام :cascade constraints 

```
ALTER TABLE emp2
```

```
DROP COLUMN employee_id CASCADE CONSTRAINTS;
```

Table altered.

في هذا المثال ستم اسقاط جميع القيود المرتبطة بالعمود employee_id .

لاسقاط أعمدة في الجدول test1 السابق ذكره نكتب:

```
ALTER TABLE test1
```

```
DROP (pk, fk, col1) CASCADE CONSTRAINTS;
```

Table altered.

في هذه الحالة أيضا يتم اسقاط جميع القيود المرتبطة بالأعمدة التي تم اسقاطها. ولكن في هذه الحالة لا داعي لاستخدام عبارة cascade constraints لأن جميع هذه الأعمدة مشتركة في قيد أو أكثر؛ فيتم اسقاطها جميعاً واسقاط أية قيد مرتبطة بها؛ وهذا يسهل علينا اسقاط الأعمدة –أو تحميدها unused كما سبق بيانه– غير المرغوب بها مع القيود المرتبطة بها دفعة واحدة.

تأجيل القيود 

بمجرد إنشاء القيود فإنها تكون عاملة وتؤثر على البيانات فوراً ما لم نضعها في حالة تعطيل disabled عند إنشائها. أما عملية تأجيل القيود فإنها تعني تأجيل اختبار البيانات Data Validation حتى نقرر تثبيتها على قاعدة البيانات باستعمال COMMIT. في الحالة الطبيعية يقوم القيد باختبار البيانات في الحركة transaction مع كل جملة معالجة البيانات DML يقوم بإلغاء التغييرات ROLLBACK للحمل المخالف للقيد أولاً بأول. أما عند تأجيل القيد فيتم إلغاء التعديلات عند إصدار أمر التثبيت باستعمال COMMIT.

وللقيود الخصائص التالية:

١ - قابلية التأجيل NOT DEFERRABLE أو غير قابلة للتأجيل DEFERRABLE

٢ - مؤجلة ابتداءً INITIALLY DEFERRED أو فورية ابتداءً INITIALLY IMMEDIATE

الخاصية الأولى تعني أننا يمكن أنه يوجد في القيد قابلية تأجيل التأثير حتى نهاية الحركة بـ COMMIT؛ أو أننا نجعل القيد غير قابل للتأجيل؛ وبهذا فإنه يختبر البيانات مع كل جملة DML.

أما الخاصية الثانية فهي تحدد حالة القيد من أنه ابتداءً مؤجل أو ابتداءً فوري؛ والحالة العادبة أنه ابتداءً فوري.

ولتطبيق الخصائص السابقة نكتب الأمثلة التالية:

```
ALTER TABLE dept2
```

```
ADD CONSTRAINT dept2_id_pk
```

```
PRIMARY KEY (department_id)
```

```
DEFERRABLE INITIALLY DEFERRED
```

الجملة السابقة أنشأت المفتاح الأولي على الجدول dept2 وخاصيته أن ابتداءً مؤجل؛ وهذا يعني أنه لدى إدخال أو تعديل بيانات العمود department_id فالقيد لا يختبر البيانات مع كل جملة إضافة INSERT أو تعديل UPDATE بل يؤجل ذلك حتى إصدار أمر التثبيت في جملة COMMIT بحيث يلغى ROLLBACK الجملة المخالفة ويثبت الجملة المنسجمة.

```
SET CONSTRAINTS dept2_id_pk IMMEDIATE
```

في هذا المثال صار القيد فوريا IMMEDIATE. لاحظ أننا لم نستخدم جملة ALTER بل مباشرة كتبنا الجملة التي تجعل القيد فوريا.

```
ALTER SESSION
```

```
SET CONSTRAINTS= IMMEDIATE
```

أما هذا المثال فإنه يجعل جميع القيود على مستوى الجلسة SESSION فورية.

ولنأخذ السينario التالي: لنفرض أن الشركة قررت تغيير رقم القسم من ٤٠ إلى ٤٥. طبعاً القسم ٤٥ غير موجود؛ فعملية تعديل رقم القسم إلى ٤٥ سيتم رفضه فوراً. لذلك نقوم بإعطاء المفتاح الأولي في جدول الأقسام والمفتاح الأجنبي في جدول الموظفين الخاصية قابل للتأجيل DEFERRABLE وابتداءً مؤجل INITIALLY DEFERRED. ثم نقوم بعمليات تعديل رقم القسم من ٤٠ إلى ٤٥ في جدول الموظفين ثم إضافة القسم رقم ٤٥ ومسح القسم رقم ٤٠ في جدول الأقسام من غير اعتراض من القيود السابقة. ثم ثبت التعديلات.

ويجدر بنا عدم الخلط بين تأجيل القيد DEFERRING وتعطيل القيد DISABLE. فإن التأجيل ينتهي مع إعطاء الأمر COMMIT أي عند انتهاء الحركة. بينما التعطيل يبقى مستمرا حتى نعيد تنشيط القيد ENABLE مرة أخرى.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥١)

إدارة مكونات قاعدة البيانات

Managing Database objects

التعامل مع الفهارس indexes

في هذه الشرح مراجعة وإضافة لما تعلمناه سابقاً فأرجو مراجعة الدروس السابقة لمعرفة الظرف المناسب لإنشاء أو عدم إنشاء الفهارس.

هناك نوعان من الفهارس وهما الفهرس الوحد unique index والفهرس غير الوحد non-unique index يمكن إنشاؤهما من خلال جملة CREATE INDEX أو جملة CREATE TABLE.

١- إنشاء الفهارس الوحيدة يكون بـ:

أ. بإنشاء قيد المفتاح الأولي PRIMARY KEY أثناء إنشاء الجدول؛ وهنا تقوم أوراكل بإنشاء الفهرس بنفس اسم المفتاح الأولي إذا أعطيناه اسماء وإنما وإن أوراكل تعطيه اسماء.

ب. بإنشاء المفتاح الوحد UNIQUE KEY وينطبق عليه ما ينطبق على المفتاح الأولي -نقطة أ-

إنشاء الفارس باستخدام أثناء إنشاء الجدول؛ لنأخذ المثال التالي:

```
CREATE TABLE NEW_EMP
```

```
(employee_id NUMBER(6)
```

```
    PRIMARY KEY USING INDEX  
(CREATE INDEX emp_id_idx ON  
    NEW_EMP(employee_id)),
```

```
first_name VARCHAR2(20),
```

```
last_name VARCHAR2(25));
```

Table created.

لاحظ عبارة PRIMARY KEY USING INDEX بعد عبارة CREATE. في هذا المثال تم إنشاء الفهرس الوحد emp_id_idx (لماذا هو وحيد؟) أثناء إنشاء الجدول ومتافق مع إنشاء القيد من نوع مفتاح أولي. وللتتأكد من وجود الفهرس نستعمل العرض user_indexes كما يلي:

```

SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'NEW_EMP';

```

طبعا لا بد من ذكر اسم الجدول الذي نريد معرفة الفهارس التي تم إنشاؤها عليه وإلا سنحصل على جميع الفهارس في المخطط .schema

INDEX_NAME	TABLE_NAME
EMP_ID_IDX	NEW_EMP

المثال التالي ينشئ فهرسا من غير أن نعطيه اسماء في جملة إنشاء الجدول وتأكدنا من وجوده:

```

CREATE TABLE EMP_UNNAMED_INDEX
(employee_id NUMBER(6) PRIMARY KEY,
first_name VARCHAR2(20),
last_name VARCHAR2(25));

```

Table created.

```

SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'EMP_UNNAMED_INDEX';

```

INDEX_NAME	TABLE_NAME
SYS_C002835	EMP_UNNAMED_INDEX

وكأسلوب آخر لإنشاء الفهارس نكتب الجمل التالية:

المخطوة الأولى: إنشاء الجدول

```

CREATE TABLE NEW_EMP2
(employee_id NUMBER(6)
first_name VARCHAR2(20),
last_name VARCHAR2(25));

```

المخطوة الثانية: إنشاء فهرس عادي-غير وحيد

```

CREATE INDEX emp_id_idx2 ON
new_emp2 (employee_id);

```

المخطوة الثالثة: إنشاء مفتاح أولي

```
ALTER TABLE new_emp2 ADD PRIMARY KEY (employee_id) USING INDEX  
emp_id_idx2;
```

ينفع هذا الأسلوب في حالة إنشاء جداول من غير فهارس أو مفاتيح أولية ثم تكون هناك حاجة لذلك!

ولإنشاء الفهارس باستعمال جملة إنشاء الفهرس CREATE INDEX يرجى مراجعة الدرس المتعلق بذلك ففيه الكفاية. ولكن أوراكل تنصح بقوة أن يكون إنشاء الفهارس الوحيدة من خلال إنشاء المفتاح الأولي والمفتاح الوحيد.

الفهارس المبنية على دوال Function-based indexes

يمكنا إنشاء فهرس مبني على دالة معينة والدالة عبارة عن تعبير تستخدم فيه الأعمدة من الجدول أو ثوابت أو دوال سكول أو دوال يقوم بإنشائها المستعمل(المبرمج)؛ وفي حالة أن المستعمل هو من ينشئ الدالة التي يريد استعمالها في الفهرس؛ فهذه الدالة يجب أن تكون مخزنة في قاعدة البيانات على نفس المخطط أو للمستخدمين الآخرين صلاحية تنفيذ هذه الدالة (أسارح فكرة الدوال المخزنة في قاعدة البيانات لدى الحديث عن لغة أوراكل الإجرائية Oracle procedural language المعروفة بلغة سكول الإجرائية P/L SQL إن كان لنا في العمر بقية إن شاء الله).

في المثال التالي سيتم إنشاء فهرس على حالة الحرف أنه كبير UPPER على العمود department_name في جدول departments.

```
CREATE INDEX upper_dept_name_idx  
ON dept2 (UPPER (department_name));
```

Index created.

ولاستعمال الفهرس المبني على يجب تضمين الدالة والعمود المستخدم فيها ضمن عبارة WHERE وإلا فلن يكون له مفعول مثلا:

```
SELECT *  
FROM dept2  
WHERE UPPER (department_name) = 'SALES';
```

أما لو كتبنا الاستعلام السابق كما يلي:

```
SELECT *  
FROM dept2  
WHERE department_name = 'SALES';
```

فلن يكون هناك تأثير للفهرس كسرعة البحث؛ فوجب الانتباه. وهذا يقودنا إلى أن نلغى مفعول أي فهرس بالتحايل عليه بإضافة شيء عليه أو عدم ذكر الدالة.... أي تضمين الفهرس في عبارات expressions. راجع بحث الفهارس سابقا.

ملاحظة مهمة: في حالة استعمال الفهارس المبنية على دوال فال وسيط parameter الذي اسمه QUERY_rewrite_enabled يجب أن يكون في حالة صواب TRUE وذلك في إعدادات قواعد البيانات. أسأل مدير قاعدة البيانات فهو المسؤول.

اسقاط الفهارس Dropping indexes

يمكننا اسقاط الفهرس الموجود في مخطتك your schema باستعمال أمر الإسقاط DROP كما يلي:

```
DROP INDEX index_name;
```

لإسقاط الفهرس في المثال السابق نكتب:

```
DROP INDEX upper_dept_name_idx;
```

Index dropped.

أما إذا أردت اسقاط فهرس ليس في مخطتك فيجب أن تكون لديك صلاحية اسقاط أي فهرس .INDEX

في حالة اسقاط جدول ما فإن جميع الفهارس والقيود على الجدول تسقط تلقائيا؛ بينما العروض views والمتسلسلات sequences تبقى ولكن العروض تكون غير مقبولة invalid

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥٢)

إدارة مكونات قاعدة البيانات

Managing Database objects

التنظيف purge. في الإصدارات السابقة لأوراكل 10g جي ١٠ كانت أوراكل تسقط الجدول نهائياً من قاعدة البيانات وتسترجع المساحة التي كان يحتلها الجدول. ولكن ابتداءً من نسخة أوراكل ١٠ جي؛ صارت أوراكل تعطي فرصة للتراجع عن الاسقاط بطريقة سيتم شرحها بعد هذه النقطة؛ وبالتالي لا تتحرر المساحة التي يحتلها الجدول؛ فيتم وضع الجدول في سلة المهملات وتعطيه أوراكل اسمًا غريباً!

وعند رغبتنا بتحرير المساحة التي يحتلها الجدول أي أننا لا نرغب بوجود الجدول كلية؛ فأضافت أوراكل عبارة PURGE أي تنظيف المساحة التي كان يحتلها الجدول ضمن جملة اسقاط الجدول DROP TABLE. مثال:

```
DROP TABLE dept80 PURGE;
```

الجملة السابقة تسقط الجدول وتتحرر المساحة التي كان يحتلها الجدول. ويجب الانتباه أن استعمال الخيار PURGE يقضي على أي أمل لاسترجاع الجدول بنسخته الأخيرة. أما إذا لم يكن هناك نسخة احتياطية من الجدول فعليه العرض.

الاسترجاع FLASHBACK

ناقشتنا في النقطة السابقة كيفية اسقاط الجدول وتحرير المساحة التي كان يحتلها باستعمال drop table ... purge. وقد وفرت أوراكل وسيلة رائعة وسهلة لاستعادة الجدول إلى حاليه السابقة قبل التغيير بل وإلى وقت معين — تاريخ ووقت). وتغيير الجدول يشمل الإسقاط وتحديث الهيكلية وإضافة وحذف وتعديل البيانات... ولاسترجاع جدول ما إلى حالته في وقت معين فهذه هي الصيغة لذلك:

```
FLASHBACK TABLE [schema.]table [, [ schema.]table]...
```

```
TO { TIMESTAMP | SCN } expr
```

```
[ { ENABLE | DISABLE } TRIGGERS ];
```

حيث:

جملة FLASHBACK TABLE هي من لغة تعريف البيانات DDL لاسترجاع الجدول.

الجدول المطلوب استرجاع حالته في المخطط الذي يقع فيه الجدول. إذا كان الجدول في مخططك فلا داعي لذكر اسم المخطط؛ أما إن كان في مخطط آخر فتذكرة اسم المخطط التابع له الجدول. يمكنك استرجاع أكثر من جدول في أكثر من مخطط. لا تحتاج إلى صلاحيات لعمل ذلك.

TO { TIMESTAMP | SCN } expr تغير النظام (أو راكل تعطي كل حركة على قاعدة البيانات رقماً وحيداً اسمه System Change Number(SCN) تحفظ به في قاموس بياناتها وهو عمود وهي Pseudo Column) ويمكننا طبعاً معرفة قيمة هذا العمود كما سيرد في الأمثلة. يجب الاختيار فقط بين رقم تغير النظام أو الوقت وليس كليهما معاً. expr هي القيمة التي نرغب تخصيصها لرقم التغيير أو الوقت.

[] في حالة وجود زناد trigger أو أكثر(System شرح موضوع الزناد في دروس لغة أوراكل الإجرائية PL/SQL إن شاء الله)؛ فتصبح حالتها معطلة disabled لدى اسقاط الجدول وكذلك أية عروض يشترك فيها الجدول. وفي حالة استرجاع الجدول بعد اسقاطه فيمكننا الاختيار بين حالي التعطيل أو التنشيط للزناد.

إن عملية الاسترجاع تتميز بالسهولة أولاً؛ وهي متاحة في أي وقت ثانياً؛ وسرعة التنفيذ ثالثاً. وسهولة التنفيذ هي مجرد كتابة جملة الاسترجاع ويستطيع أي مستخدم القيام بها في مخططه أو مخطط غيره؛ وهذا طبعاً يخفف العبء عن كاهل مدير قاعدة البيانات. لو أردنا استرجاع جدول باستخدام الطرق التقليدية فبحب أن يقوم بها مدير قاعدة البيانات المختص وقد يكون هناك تعاون مع مدير النظام System Administrator.

ويجب ملاحظة أنه في حالة حدوث ضرر في وسائل التخزين فعملية الاسترجاع هذه لا تستطيع معالجة هذا الضرر أو استرجاع حالة الجدول المتضرر؛ بينما في الطرق التقليدية يمكن ذلك إلى حد ما.

لا تقوم أوراكل بتنشيط أي تعديلات على الجداول بصورة نهائية حتى باستعمال أمر التثبيت commit بل تحفظ بها في مساحة تخزين تسمى مساحة الاسترجاع أو التراجع undo segment وذلك لفترة محددة؛ وهذه الفترة يقررها مدير قاعدة البيانات؛ وإذا لم يتم تحديدها فقيمتها الافتراضية هي ١٥ دقيقة أو ٩٠٠ ثانية؛ وبعد هذه المدة لا مجال لاسترجاع الحالة الأصلية باستعمال وسيلة الاسترجاع هذه ولكن بالطرق التقليدية. نعم إن عملية التثبيت commit تجعل التغييرات دائمية على الجدول ولكن على مستوى المستخدمين وليس على مستوى النظام. وهذه ميزة ممتازة تسهل عمل المبرمجين ومديري قواعد البيانات. ولتحديد مدة الاسترجاع هناك وسيط parameter اسمه UNDO_RETENTION.

عملية الاسترجاع السريعة هذه تسترجع الفهارس والقيود على الجدول التي سقطت باسقاطه.

مثال: سنقوم باسقاط الجدول emp2 ثم نستعلم عنه من سلة المهملات

```
DROP TABLE emp2;
```

Table dropped

```
SELECT original_name, operation, droptime, FROM recyclebin;
```

العرض recyclebin فيه معلومات الجدول الذي تم اسقاطه مثل الاسم الأصلي ونوع العملية التي قمت عليه - في حالتنا اسقاط - وقت العملية. كما في الجدول أدناه.

ORIGINAL_NAME	OPERATION	DROPTIME
EMP2	DROP	2004-03-03:07:57:11

تفيد عملية الاستعلام في معرفة أسماء الجداول التي أُسقطت ووأوقات اسقاطها لتقرير استرجاعها من عدمه. لسترجع الجدول:

FLASHBACK TABLE emp2 TO BEFORE DROP;

Flashback complete

لاحظ أننا لم نستعمل وقت الاسترجاع بل اكتفيينا بعبارة before drop أي قبل وقت الاسقاط! وهذا استعمال ضمني للوقت!

للاستعلام عن محتويات سلة المهملات نكتب

SELECT * FROM RECYCLEBIN;

وبديهي أننا لا نستطيع معرفة ما في سلة مهملات الآخرين؛ بل محتويات سلة المهملات الخاصة بنا أي بالمحظط الذي نعمل عليه بالرغم من إمكانية استرجاع الجداول في محططات أخرى.

أما في حالة رغبتنا في حذف البيانات نحائيا فنقوم بإفراغ سلة المهملات بالجملة التالية:

PURGE RECYCLEBIN;

كما هو ملاحظ فهذه العملية بمراحلها تشبه حذف ملف في نظام ويندوز ووضعه في سلة المهملات واسترجاعه أو إفراغ سلة المهملات بحيث لا يمكن استرجاع الملفات المحذوفة بسهولة.

إن ما سبق شرحه عن عملية الاسترجاع السريعة flashback هو لحة سريعة عن إمكانيات هذه التقنية الرائعة؛ فهذه التقنية تمكننا من استرجاع كامل قاعدة البيانات إلى حالتها الأخيرة حسب الوقت المحدد؛ كما تمكننا أيضاً من متابعة التغييرات على الصف أو الصنف في جدول ما ومعرفة القيم التي تغيرت وأوقات التغيير....الخ؛ وهذا شرحه في موضوع إدارة قواعد البيانات.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥٣)

إدارة مكونات قاعدة البيانات

Managing Database objects

الجدوال الخارجية External Tables

الجدوال الخارجية هي جداول عادية كالتى مرت معنا سابقاً؛ ولكن بياناتها ليست مخزنة في قاعدة البيانات بل في ملفات خارجية؛ فهي تمثل بيانات ليست في قاعدة البيانات أوراكل؛ ويتم تعبئتها -تبعية منطقية وليس فизيائياً لأن البيانات تظل في الملف الأصلي ولا تنتقل إلى الجدول الخارجي ولكن من خلال الجدول الخارجي تتم قرائتها- هذه الجداول بأساليب خاصة أثناء إنشاء هذه الجداول. وهذه الجداول للقراءة فقط Read Only ولا يمكن إجراء أي تعديلات على بياناتها باستخدام لغة معالجة البيانات DML. الجدول الخارجي فيه فقط هيكل الجدول data structure.

طبعاً هناك اختلاف في كيفية إنشاء هذه الجداول عن الجداول العادية؛ لأن البيانات في الملفات التي تمثلها تختلف في شكلها عن البيانات التي يتم إضافتها في الجداول العادية. وقد وفرت أوراكل طريقتين؛ الأولى بواسطة محمل أوراكل ORACLE_LOADER والذي يستعمل ضمنياً أداة أوراكل في تحميل البيانات من ملفات ليست من أوراكل إلى قاعدة بيانات أوراكل. والبيانات في هذه الملفات تأخذ شكلاً محدداً وليس متغرياً؛ وهذه الأداة هي SQL*Loader علماً أنها قد لا تدعم بعض أشكال البيانات في الملفات الخارجية.

أما الأداة الثانية أو الطريقة الثانية لربط الجداول الخارجية بالملفات الخارجية فهي مضخة أوراكل البيانات ORACLE_DATAPUMP؛ وهذه غير مرتبطة بشكل البيانات على أي منصة؛ فهي أداة مستقلة تتعامل مع كل البيانات؛ ويمكن استخدامها أيضاً لتصدير واستيراد البيانات. عند إنشاء الجدول الخارجي يتم ذكرها ضمن بناءً Syntax جملة لإنشاء مع جملة SELECT في نفس جملة إنشاء.

خطوات إنشاء الجدول الخارجي:

١- يجب إنشاء دليل DIRECTORY على قاعدة البيانات -وهو أحد مكونات قاعدة البيانات database object مثل الجداول والفهارس-أولاً باستعمال الجملة التالية وبصلاحيات مدير قاعدة البيانات أو من يعطيه هذه الصلاحية:

```
CREATE [OR REPLACE] DIRECTORY directoryname AS 'path_name';
```

حيث:

اسم الدليل على قاعدة البيانات directoryname

.directoryname 'اسم المجلد أو المسار كاملاً الذي يمثله الكائن' path_name'

قد يbedo الأمر غامضاً قليلاً؛ ولكن يزول الغموض إذا نظرنا إليه من الناحية الأمنية. فالذي ينشأ هذا الدليل يخبر المبرمجين عنه اسمه ولكنه لا يخبرهم عن مكان تواجده أي لا يخبرهم عن الجهاز المتواجد عليه هذا الدليل ولا عن اسمه ولا أية معلومات أخرى. بل إن المرجع يمكنه الوصول إلى الدليل من خلال أوراكل فقط بواسطة اسم الدليل directoryname حتى لو لم تكن له صلاحية الوصول إلى المجلد الفعلي الذي يمثله الدليل وحتى لو لم يكن المجلد مشتركاً shared folder. هذا المجلد يحتوي على الملفات التي يريد إنشاء جداول خارجية عليها من غير السماح للوصول إليها وهذا يؤدي إلى حمايتها من أية تدخلات لمن ليس لهم عمل. مثال لإنشاء دليل:

```
CREATE [OR REPLACE] DIRECTORY emp_dir AS 'd:\emp_dir';
```

هنا أنشأنا الدليل emp_dir الذي يمثل المجلد d:\emp_dir. ولا يوجد تعارض بين الاسمين لأنهما يقعان في بيئتين مختلفتين.

ملاحظة: عند إنشاء الدليل حسب الجملة السابقة قد لا يكون المجلد الذي يمثله موجود على القرص فيزيائياً ولا تصدر أية رسالة خطأ؛ ولكن تصدر رسالة خطأ عن تنفيذ جملة إنشاء المجلد. وبمعنى آخر يمكن إنشاء الكائن دليل ثم إنشاء المجلد فيزيائياً لاحقاً.

- ٢- أما كيفية وصول المستعملين (المبرمجين) لهذا الدليل المخفي كلياً عنهم فهم بإعطاء صلاحية القراءة أو الكتابة عليه أو كلامها حسبما يرى صاحب الصلاحية وذلك باستعمال جملة الصلاحية GRANT كما يلي:

```
GRANT READ ON DIRECTORY directoryname TO username;
```

إن الكائن من نوع دليل DIRECTORY مملوك فقط للمستعمل SYS - وهو أعلى من المستعمل SYSTEM - بغض النظر عن من أنشأه بالرغم من أن الذي أنشأه هو من له صلاحية إعطاء صلاحية الكتابة والقراءة لمستعملين آخرين. وصلاحية الكتابة لا تعني الكتابة على الملف الممثل بالجدول الخارجي لأنه أصلاً لا يمكن تعديل الملف من خلال الجدول الخارجي؛ بل تعني أنه يمكن للمستعمل (المبرمج) إنشاء ملفات أخرى على المجلد

 إنشاء الجدول الخارجي؛ يتم من خلال الجملة التالية:

```
CREATE TABLE table_name
```

```
( <col_name> <datatype>, ... )
```

```
ORGANIZATION EXTERNAL
```

```
(TYPE <access_driver_type>
```

```
DEFAULT DIRECTORY <directory_name>
```

```
ACCESS PARAMETERS ( ... )
```

```
LOCATION ('<locationSpecifier>')
REJECT LIMIT [0 | <number> | UNLIMITED];
```

حيث:

- ١ - عبارة CREATE TABLE كما سبق شرحه
- ٢ - عبارة ORGANIZATION EXTERNAL، لإعلام أوراكل بأن هذا الجدول خارجي وبالتالي تمنع الكتابة عليه
- ٣ - عبارة TYPE access_driver_type لتحديد كيفية الوصول إلى الملف الخارجي هل بواسطة ORACLE_DATAPUMP أو ORACLE_LOADER
- ٤ - عبارة DEFAULT DIRECTORY directory_name التي تشير إلى اسم الدليل الذي سبق إنشاؤه
- ٥ - عبارة ACCESS PARAMETERS تشير إلى معلومات الملف الذي ستتعامل معه مثل الفوائل بين السجلات والحقول وأطوال وأنواع الحقول.....
- ٦ - عبارة 'locationSpecifier' تخبر أوراكل عن اسم الملف الذي سيمثله الجدول الخارجي
- ٧ - عبارة REJECT LIMIT [0 | <number> | UNLIMITED] هذه العبارة تخبر أوراكل بواسطتها عن عدد السطور المخالفة للقيود التي قمنا بإنشائها على الجدول أثناء إنشاء الجدول كالمعتاد. وعدد السطور هو صفر وهو العدد في الحالة الافتراضية عند عدم تحديده - أي تتوقف أوراكل عن استقبال البيانات من الملف الخارجي عند أول مخالفة. أما ذا حددنا العدد ب number فإن أوراكل تتوقف بعد تجاوز هذا العدد بسطر واحد. أما عبارة UNLIMITED فتعني استمرار استقبال البيانات بغض النظر عن عدد السطور المخالفة.

مثال:

```
CREATE TABLE oldemp (
    fname char(25), lname CHAR(25))
ORGANIZATION EXTERNAL
(TYPE ORACLE_LOADER
DEFAULT DIRECTORY emp_dir
ACCESS PARAMETERS
(RECORDS DELIMITED BY NEWLINE
NOBADFILE
NOLOGFILE
FIELDS TERMINATED BY ',')
```

```
fname POSITION (1:25) CHAR,  
lname POSITION (27:51) CHAR))  
LOCATION ('emp.dat'))  
PARALLEL 5  
REJECT LIMIT 200;
```

Table created.

شرح المثال:

١- الجملة السابقة تنشئ المجدول oldemp وفيه عمودان طول كل واحد ٢٥ حرفا وهو يمثل ملفا خارجيا عن قاعدة البيانات.

٢- طريقة الوصول من خلال محمل أوراكل ORACLE_LOADER

٣- الدليل الافتراضي الذي يقع فيه الملف هو emp_dir الذي سبق إنشاؤه في المثال السابق

٤- وسائل الوصول ACCESS PARAMETERS كالتالي:

أ. كل سجل مفصول عن الآخر بسطر-أي كل سجل في سطر مستقل

ب. عدم إنشاء الملف السيء (ملف أخطاء) NOBADFILE وهو يتم فيه تخزين السجالات المخالفة

ج. عدم إنشاء ملف السجل NOLOGFILE وهو الملف الذي يتم تسجيل مجريات العملية

د. يفصل بين الحقول الفاصلة العادية (,) , ''

هـ. الحقل fname يبدأ من الموقع ١ وينتهي بالموقع ٢٥ أي بطول ٢٥ ونوعه رموز CHAR

وـ. الحقل الثاني lname يبدأ من الموقع ٢٧ لأن الفاصلة تقع في الموقع ٢٦ وينتهي في الموقع ٥١ ونوعه رموز CHAR

زـ. اسم الملف الخارجي emp.dat هو LOCATION

حـ. عدد قنوات نقل البيانات ٥ PARALLEL 5. أي تقوم أوراكل بنقل البيانات من الملف الخارجي من خلال قنوات حددناها ب ٥ قنوات وقد تكون أكثر وهي على حساب الذاكرة ولكن في صالح السرعة. العدد الافتراضي ١

طـ. حد الرفض REJECT LIMIT تم تحديده ب ٢٠٠ سجل تتجاوزها أوراكل قبل توقف عملية النقل من الملف الخارجي

يمكنكم إنشاء ملف نصي بسيط من عدة سطور يشمل الاسم الأول والثاني حسب الشكل المذكور في المثال ثم إنشاء المجدول الخارجي من خلال المثال.

الاستعلام من الجداول الخارجية Querying the external tables

لا يختلف الاستعلام من الجدول الخارجي عنه في الجدول العادي كما تم شرحه في دروس الاستعلام بكل تفاصيله التي مرت معنا. ويمكن ربطه بغيره من الجداول أيضاً ويمكن إسقاطه.... الخ

ولكن لا مجال لتحديث البيانات عليه!

أظن أن هذا أصعب درس ولكنه مهم من الناحية الأمنية ومن ناحية تصدير البيانات إلى قاعدة البيانات مباشرة؛ مع ملاحظة أن طرق الوصول إلى الملفات الخارجية السابقة تمكّننا من جلب البيانات من ملفات خارجية إلى قاعدة البيانات مباشرة. وقد مرت بي تجربة نقل بيانات من ملفات لغة البرمجة الشهيرة COBOL إلى قاعدة البيانات أوراكل ومن قاعدة البيانات DBASEIII إلى قاعدة البيانات أوراكل ولكن من غير استعمال تقنية الجداول الخارجية لأن هذه التقنية جديدة في نسخة أوراكل 10 جي وتجربتي كانت على نسخة قاعدة البيانات أوراكل نسخة 6؛ على نظام يونيكس سنة ١٩٩٦ على ما ذكر.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أولاً مثلاً (٤)

معالجة مجموعات من البيانات الضخمة

Manipulating Large Data Sets

في السلسلة التالية من الدروس سوف يتم التعامل مع كميات كبيرة من البيانات من حيث المعالجة أي بالإضافة والتعديل والحذف والدمج merging. والموضوع شيق.

+ معالجة البيانات من خلال الاستعلامات الجزئية؛ وقد مرت معنا من خلال إنشاء الجداول وإضافة البيانات. وفي السطور التالية بعض التوسيع في ذلك.

أولاً: نسخ بيانات من جدول إلى جدول آخر؛ والصيغة العامة هي كالتالي:

```
INSERT INTO table [ column, column ] subquery;
```

حيث table الجدول المدفوع الذي سنضيف إليه بيانات

أسماء الأعمدة المستهدفة—وذكرها اختياري وعدم ذكرها يعني جميع الأعمدة مستهدفة column, column...

الاستعلام الجزئي الذي يجلب البيانات من الجداول الأخرى Subquery

ملاحظات:

أ. لا تذكر عبارة values في جملة insert

ب. يجب أن يكون عدد الأعمدة في عبارة insert مطابقاً لعدد الأعمدة في الاستعلام الجزئي

ج. مراعاة مطابقة أنواع البيانات في الأعمدة المناظرة للتعابير في الاستعلام الجزئي. بعض البيانات قد تظهر أخطاء إذا كانت البيانات غير متطابقة. مثلاً العمود في الجدول يكون من نوع عدددي وما يناظره في الاستعلام يكون من نوع تاريخ. عموماً هناك تحويل ضمني بين أنواع البيانات مكان شرحه في اللغة الإجرائية PL/SQL.

مثال: لنفرض أننا نريد تعبئة الجدول sales_rep ببيانات مماثلة للمبيعات من جدول الموظفين على فرض أن هذا الجدول موجود وبالهيكلية المذكورة في عبارة insert

```
INSERT INTO sales_reps (id, name, salary, commission_pct)
```

```
SELECT employee_id, last_name, salary, commission_pct
```

```
FROM employees
```

```
WHERE job_id LIKE '%REP%';
```

33 rows created.

لاحظ مطابقة عدد الأعمدة في عباري insert و select. علما بأن أساليب insert و select جملة الاستعلام سابقة الشرح تستعمل هنا بكل بساطة وسترد أمثلة أيضا عليها.

أما إذا أردنا نسخة كاملة من بيانات جدول ما فنستخدم حينئذ * select في جملة الاستعلام الجزئي. مثلا:

```
INSERT INTO EMP_COPY
```

```
SELECT * FROM employees;
```

ثانياً: استعمال الاستعلام الجزئي كهدف لإدخال البيانات أي بدلاً من الجدول نفسه؛ مثال ذلك:

```
INSERT INTO
```

```
(SELECT employee_id, last_name, email, hire_date, job_id, salary, department_id  
FROM emp_copy  
WHERE department_id = 50)
```

```
VALUES (99999, 'Taylor', 'DTAYLOR', TO_DATE('07-JUN-99', 'DD-MON-RR'),  
'ST_CLERK', 5000, 50);
```

1 row created.

كما تلاحظ لا داعٍ لذكر اسم الجدول مباشرة في عبارة insert بل استعملنا الاستعلام الجزئي كهدف؛ وهذا يعني عن إنشاء عرض view من أجل إضافة بيانات على أعمدة محددة. وغني عن البيان أنه يجب ذكر جميع الحقول الإجبارية في كلا الجزئين من جملة إضافة البيانات مع مراعاة قيود التكامل أيضا.

وللتتأكد من نجاح الجملة السابقة نستعمل كما يلي:

```
SELECT employee_id, last_name, email, hire_date,  
job_id, salary, department_id  
FROM emp_copy  
WHERE department_id = 50;
```

ثالثاً: استخراج البيانات من خلال الاستعلام الجزئي وليس الجدول كمصدر. مثال:

```
SELECT a.last_name, a.salary, a.department_id, b.salavg  
FROM employees a, (SELECT department_id, AVG(salary) salavg  
FROM employees  
GROUP BY department_id) b  
WHERE a.department_id = b.department_id  
AND a.salary > b.salavg;
```

لاحظ الاستعلام الجزئي الذي عاملناه كجدول؛ ولكن في الحقيقة يمكن النظر إليه كعرض ويسمى بالعرض الضمني inline view وأعطيتني هذا الاستعلام اسمًا بديلا Alias هو b تماماً كما أعطينا الجدول employees الاسم البديل a. ثم جرت عملية الربط بين (العرض b) والجدول (a) وهذا الأسلوب مفيد في حالات الإحصائيات المعقّدة وعمليات الربط الصعبة بدلاً من إنشاء جداول وعروض خاصة تزيد العبء على قاموس البيانات.

هل يُكنّكم معرفة وظيفة الجملة السابقة؟ أخبرونا!

بسم الله الرحمن الرحيم

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٤٥)

معالجةمجموعات من البيانات الضخمة

Manipulating Large Data Sets

تحديث بيانات عمودين فأكثر من استعلام جزئي؛ والمقصود هو تعديل بيانات عمودين فأكثر بقيم مأخوذة من استعلام جزئي لكل عمود. والمثال التالي يوضح العملية؛ نريد تعديل مهنة job_id الموظف رقم ١١٤ لتكون مثل مهنة الموظف رقم ٢٠٥؛ ونريد تعديل الراتب salary ليكون مثل راتب الموظف ١٦٨ لنفس الموظف رقم ١١٤. لا يهمنا أن نعرف تلك القيم:

```
UPDATE emp13  
SET job_id = (SELECT job_id  
FROM emp13  
WHERE employee_id = 205),
```

```
Salary = (SELECT salary  
FROM emp13  
WHERE employee_id = 168)  
WHERE employee_id = 114;  
1 row updated.
```

أما الصيغة العامة فهي:

```
UPDATE table  
SET column =  
(SELECT column  
FROM table  
WHERE condition)  
[ ,  
column =  
(SELECT column  
FROM table  
WHERE condition)]  
[WHERE condition] ;
```

واضح من الصيغة إمكانية تعديل قيم أي عدد من الأعمدة. إذا لم يتحقق الشرط الموجود في الاستعلام الجزئي فالنتيجة رسالة تقول بأنه تم تعديل صفر سطر/سطور

“0 rows updated.”

تعديل صفوف جدول باستعمال قيم مأخوذة من جدول آخر. في النقطة السابقة كانت القيم مأخوذة من نفس الجدول؛ ولكن هنا ستكون القيم مأخوذة من جدول آخر وكذلك شروط الاستعلام.

نريد تعديل رقم القسم في الجدول emp13 باستخدام رقم القسم الذي ينتمي إليه الموظف رقم ١٠٠ في جدول employees بشرط أن المهنة job_id في جدول emp13 هي نفسها مهنة الموظف رقم ٢٠٠ في جدول employees . والجملة هي التالية:

```
UPDATE empl3  
SET department_id = (SELECT department_id  
                      FROM employees  
                     WHERE employee_id = 100)  
WHERE job_id      = (SELECT job_id  
                      FROM employees  
                     WHERE employee_id = 200);
```

1 row updated.

 حذف بيانات من جدول اعتمادا على استعلام جزئي من جدول آخر. لنفرض أننا نريد مسح بيانات الموظفين في القسم Public Relations ولكننا لا نعرف رقم القسم department_id فاضطررنا إلى استعلام جدول الأقسام لهذا الغرض من غير أن نعرف رقم القسم كذلك. راقب هذا المثال:

```
DELETE FROM emp13  
WHERE department_id =  
      (SELECT department_id  
       FROM departments  
      WHERE department_name  
            LIKE '%Public%');
```

1 row deleted.

 استعمال خيار التحقق WITH CHECK OPTION

تعرفنا على كيفية الاستعلام الجزئي INSERT كهدف لجملة الإضافة subquery وسنضيف خيار التتحقق إلى جملة الاستعلام الجزئي بحيث لا يتم إضافة الصفوف التي لا تتحقق هذا الخيار. لندرس المثال التالي:

```
INSERT INTO (SELECT employee_id, last_name, email,
```

```

hire_date, job_id, salary

FROM emp13

WHERE department_id = 50
      WITH CHECK OPTION,
VALUES (99998, 'Smith', 'JSMITH',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000);

```

INSERT INTO

*

ERROR at line 1:

ORA-01402: view WITH CHECK OPTION where-clause violation

جملة الإضافة السابقة تستعمل الاستعلام الجزئي كهدف لإضافة صف على الحدود emp13 بشرط أن يكون رقم القسم ٥٠ وباستخدام خيار التتحقق. ولكن رسالة خطأ ظهرت بأن هناك مخالفة لشرط التتحقق لأن رقم القسم لم يذكر في الاستعلام الجزئي ولا في القيم values ولهذا فالقيمة الافتراضية ستكون اللا شيء null وهذا يخالف شرط التتحقق بأن رقم القسم يجب أن يكون ٥٠.

 التصريح بخاصية القيمة الافتراضية explicit DEFAULT feature. وهذه الخاصية أضيفت للتطابق مع سكول SQL:1999 standard ١٩٩٩. واستعمال عبارة DEFAULT يفترض أن العمود المراد التعامل معه تم تخصيص قيمة افتراضية له أثناء إنشاء الجدول.

لو أضفنا صفاً جديداً إلى جدول وفيه أحد الأعمدة معرف بقيمة افتراضية؛ فلا داعي لذكر ذلك العمود في جملة الإضافة INSERT ولكن يمكن استعمال هذه الخاصية في جملة الإضافة وجملة التحديث UPDATE. في المثالين التاليين سنفترض أن رقم المدير MANAGER_ID قد تم تعريفه بقيمة افتراضية معينة.

INSERT INTO deptm3

(department_id, department_name, manager_id)

VALUES (300, 'Engineering', DEFAULT

UPDATE deptm3

```
SET manager_id = DEFAULT  
WHERE department_id = 10;
```

لاحظ أننا لا نعرف ما هي القيمة الافتراضية في كلتا الحالتين وهذا لا يهم بل ويوفر وقتا علينا لمراجعة الجدول. إذا لم يكن هناك قيمة افتراضية للعمود المعنى فإن قيمته ستكون الالاشيء NULL في كلتا الحالتين.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥٥)

معالجةمجموعات من البيانات الضخمة

Manipulating Large Data Sets

جملة الإضافة المتعددة المداول Multitable INSERT statement

عندما ناقشنا جملة الإضافة INSERT رأينا أننا نكتب اسم جدول واحد فقط لإضافة صف أو صفوف إليه. في المناقشة التالية سنرى إمكانية استعمال أكثر من جدول لإضافة البيانات إليها بصورة متوازية. وهذه الفكرة مفيدة في تعبئة مستودعات البيانات ETL-Extraction Transformatin Loading التي تستعمل عملية data warehouse.

نستخدم في جملة الإضافة جملة الاستعلام من جدول أو جداول كما سبق بيانه لإضافة البيانات إلى أكثر من جدول. وهذا يسهل علينا كثيراً ويسرع عملية إضافة البيانات من حيث:

- لسنا مضطرين لكتابية جملة إضافة INSERT كل مرة لكل جدول لكمية كبيرة من البيانات IF...THEN.. ESLE... procedural statements تعتمد على عبارات الشرط ...

أنواع جملة الإضافة متعددة المداول:

١- إضافة الكل من غير شروط Unconditional ALL INSERT

٢- إضافة الكل بشروط Conditional ALL INSERT

٣- الإضافة عند أول شرط Conditional FIRST INSERT

٤- الإضافة المحورية Pivoting INSERT

الكلمات المكتوبة بالحرف الكبير هي جزء من تركيب الجملة. وسنأخذ أمثلة لكل نوع ثم نكتب الصيغة العامة فهذا أسهل في الفهم. استعمال أي نوع يعتمد على حاجة العمل.

- إضافة الكل من غير شروط Unconditional INSERT. نريد تعبئة كل من الجداولين sal_history وmgr_history للموظفين الذي أرقامهم أكبر من ٢٠٠ من غير شروط باستخدام فكرة الإضافة متعددة المداول كما يلي:

INSERT ALL

```
INTO sal_history VALUES (EMPID, HIREDATE, SAL)  
INTO mgr_history VALUES (EMPID, MGR, SAL)
```

```
SELECT employee_id EMPID, hire_date HIREDATE, salary SAL, manager_id MGR  
FROM employees  
WHERE employee_id > 200;
```

12 rows created.

عدد الصفوف قد يختلف. لاحظ عبارة INSERT ALL وعبارة INTO المطبقتين على جدولين؛ وأخيرا لاحظ بعانياً جملة الاستعلام SELECT. هنا يتم إضافة جميع الصفوف التي تتحقق الشرط إلى كلا الجدولين من غير شروط.

- إضافة الكل ولكن بشروط. وفي المثال التالي نزيد إضافة جميع الصفوف من جدول الموظفين بشرط أن يكون رقم الموظف أكبر من ٢٠٠ حسب شرط جملة الاستعلام. سنضيف هذه البيانات بشرط أنه إذا كان الراتب sal الذي يمثل العمود salary في حملة الاستعلام أكبر من ١٠٠٠٠ يتم إضافة هذه البيانات إلى الجدول sal_history؛ والشرط الثاني أنه إذا كان رقم المدير mgr الذي يمثل العمود manager_id في حملة الاستعلام أكبر من ٢٠٠ يتم إضافة البيانات إلى الجدول mgr_history كما يلي:

```
INSERT ALL  
WHEN SAL > 10000 THEN  
    INTO sal_history VALUES(EMPID,HIREDATE,SAL)  
WHEN MGR > 200 THEN  
    INTO mgr_history VALUES(EMPID,MGR,SAL)  
  
SELECT employee_id EMPID,hire_date HIREDATE,salary SAL, manager_id MGR  
FROM employees  
WHERE employee_id > 200;
```

4 rows created.

هنا الشرطان في عبارة WHEN غير مرتبطين فستستمر عملية الإضافة على الجدولين طالما أن أي شرط يتحقق على الجدول الذي يخصه. فإذا تحقق الشرط أن الراتب أكبر من ١٠٠٠٠ تتم الإضافة على الجدول sal_history أما إذا لم يتحقق يتنتقل العمل إلى الشرط الثاني وهكذا لكل صف من الصفوف الناتجة من الاستعلام.

لاحظ أننا استعملنا الأسماء البديلة للأعمدة في جملة الاستعلام في هذا المثال والسابق والأمثلة اللاحقة لسهولة التتبع.

- إضافة عند تحقق أول شرط ثم التوقف عن اختبار باقي الشروط ويتم الانتقال إلى الصيغ التالي واختباره وهكذا. في المثال التالي يوجد ثلاث عبارات INTO تعمل على ثلاثة جداول؛ فعند أول شرط يتحقق يتم تنفيذ عبارة INTO على الجدول المرتبط بها ثم الانتقال إلى الصيغ الثاني دون المرور على باقي الشروط... وإذا لم يتحقق أي شرط ينتقل التنفيذ إلى عبارة ELSE وهي اختيارية. لندرس المثال التالي:

INSERT FIRST

```
WHEN SAL > 25000      THEN
    INTO special_sal VALUES(DEPTID, SAL)
WHEN HIREDATE like ('%00%') THEN
    INTO hiredate_history_00 VALUES(DEPTID, HIREDATE)
WHEN HIREDATE like ('%99%') THEN
    INTO hiredate_history_99 VALUES(DEPTID, HIREDATE)
ELSE
    INTO hiredate_history VALUES(DEPTID, HIREDATE)

SELECT department_id DEPTID, SUM(salary) SAL, MAX(hire_date) HIREDATE
FROM employees
GROUP BY department_id;
```

12 rows created.

في المثال يوجد عبارة FIRST مرافق لعبارة INSERT وهذا يعني يتم اختبار أول صفات ناتج من الاستعلام. فإن تحقق أي شرط يتم تنفيذ عبارة INTO المرتبطة بالشرط فإن لم يتحقق أي شرط ينتقل التنفيذ إلى ELSE إن وجدت ثم يتم الانتقال إلى الصيغ التالي. فمثلاً إذا كان الراتب sal أكبر من ٢٥٠٠٠ يتم إضافة صفات إلى الجدول special_sal ثم الانتقال إلى الصيغ التالي الناتج من جملة الاستعلام؛ وإلا ينتقل الاختبار إلى الشرط التالي وهو إذا كان تاريخ التعيين يحتوي على '00' تتم إضافة صفات إلى الجدول hiredate_history_00 ثم الانتقال إلى الصيغ التالي الناتج من جملة الاستعلام؛ وإذا لم يتحقق يتم اختبار الشرط التالي وهو أن يحتوي تاريخ التعيين على '99' فيتم إضافة صفات إلى الجدول hiredate_history_99 ثم الانتقال إلى الصيغ التالي الناتج من جملة الاستعلام؛ أما إذا لم يتحقق هذا الشرط الأخير ينتقل التنفيذ إلى عبارة ELSE لإضافة صفات إلى الجدول hiredate_history ثم يبدأ العمل على الصيغ التالي وهكذا.

٤ - الإضافة متعددة الجداول المحوسبة Pivoting MultiTables Insert

وهذا الأسلوب يتم التعامل من خلاله مع الجداول غير العلائقية Non-Relational لتحويلها إلى جداول علائقية يكون فيها السجل الواحد في الجدول غير العلائقي محور Pivot العمل. وكمثال على ذلك نفرض أن هناك جدول اسمه sales_source_info وفيه الحقول التالية:

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
WEEK_ID		NUMBER(2)
SALES_MON		NUMBER(8,2)
SALES_TUE		NUMBER(8,2)
SALES_WED		NUMBER(8,2)
SALES_THUR		NUMBER(8,2)
SALES_FRI		NUMBER(8,2)

وهذه عينة من البيانات

EMPLOYEE_ID	WEEK_ID	SALES_MON	SALES_TUE	SALES_WED	SALES_THUR	SALES_FRI
176	6	2000	3000	4000	5000	6000

لاحظ أنه توجد سبعة أعمدة منها خمسة تمثل المبيعات اليومية من الاثنين إلى الجمعة وكلها في صف أو سجل واحد لكل موظف؛ ونريد تحويله إلى جدول علائقى يتكون من 3 ثلاثة أعمدة فقط بحيث تحول كل سجل في الجدول غير العلائقى إلى خمسة صفوف في الجدول العلائقى. هيكل البيانات للجدول العلائقى sales_info كالتالى:

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
WEEK		NUMBER(2)
SALES		NUMBER(8,2)

وبعد التحويل وتبعية الجدول العلائقى أعلاه فهذا هو شكل الصدفوف بعد التحويل:

EMPLOYEE_ID	WEEK	SALES
176	6	2000
176	6	3000
176	6	4000
176	6	5000
176	6	6000

لاحظ عدد الصدفوف الناتجة هو خمسة صدفوف. والجملة التالية تحقق عملية التحويل:

INSERT ALL

```
INTO sales_info VALUES (employee_id,week_id,sales_MON)
INTO sales_info VALUES (employee_id,week_id,sales_TUE)
INTO sales_info VALUES (employee_id,week_id,sales_WED)
INTO sales_info VALUES (employee_id,week_id,sales_THUR)
INTO sales_info VALUES (employee_id,week_id, sales_FRI)
SELECT EMPLOYEE_ID, week_id, sales_MON, sales_TUE,
       sales_WED, sales_THUR,sales_FRI
FROM sales_source_data;
```

5 rows created.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أولاً كل مثلاً (٥٦)

معالجةمجموعات من البيانات الضخمة

Manipulating Large Data Sets

MERGER جملة الدمج 

في أنظمة مستودعات البيانات تحتاج باستمرار إلى دمج بيانات الجداول العاملة التي عليها العمل الفعلي وجدائل بيانات المستودعات والتي ليس عليها حركة عادة. ولكن بيانات الجداول العاملة قد يطرأ عليها تعديل لما هو موجود فضلاً عن إضافة صفوف إليها؛ والطرق العادية صعبة لتنفيذ عملية الإضافة والتعديل معاً. جملة الدمج MERGE توفر علينا وقتاً وجهداً كبيرين من حيث:

- ١ - أنها تقوم بإضافة الصفوف غير الموجودة في الجدول المهدف Target Table من الجدول المصدر Source Table
- ٢ - تقوم بتحديث بيانات الصفوف الموجودة في الجدول المهدف بآخر نسخة من بيانات الجدول المصدر
- ٣ - ترفع مستوى أداء النظام فضلاً عن سهولة الاستعمال

والشكل العام لجملة الدمج هو:

MERGE INTO *table_name table_alias*

USING (*table/view/sub_query*) *alias*

ON (*join condition*)

WHEN MATCHED THEN

UPDATE SET

col1 = col_val1,

col2 = col2_val

WHEN NOT MATCHED THEN

INSERT (*column_list*)

VALUES (*column_values*);

حيث:

- ٢ اسم الجدول أو اسمه البديل الذي ستتم عليه عملية الدمج (الجدول المدف) *table_name table_alias*

- ٣ اسم الجدول المصدر أو العرض أو حتى الاستعلام الجرئي مع إمكانية استعمال الاسم البديل *USING table/view/sub_query alias*

- ٤ الشرط الذي يربط بين الجدولين المدف والمصدر وبناءً عليه تتم عملية الدمج (إضافة أو تعديل) *ON (join condition)*

- ٥ إذا وجدت صفات مشتركة يتم تحديث الصفات في الجدول المدف بقيم الصفات المطابقة في الجدول الأصل *WHEN MATCHED THEN UPDATE SET...*

- ٦ *WHEN NOT MATCHED THEN INSERT VALUES....*
إذا لم يكن تطابق أي أن الصفات في الجدول المصدر ليست موجودة في الجدول المدف يتم إضافتها إلى الجدول المدف

مثال: نريد دمج بيانات الجدول employees -الجدول الأصل مع بيانات الجدول emp13-الجدول المدف والذي نفرض أنه فارغ. جملة الدمج التالية تتحقق المدف المطلوب:

```
MERGE INTO emp13 c
```

```
USING employees e
```

```
ON (c.employee_id = e.employee_id)
```

```
WHEN MATCHED THEN
```

```
UPDATE SET
```

```
c.first_name      = e.first_name,
```

```
c.last_name      = e.last_name,
```

```
c.email          = e.email,
```

```
c.phone_number   = e.phone_number,
```

```
c.hire_date      = e.hire_date,
```

```
c.job_id         = e.job_id,
```

```
c.salary         = e.salary,
```

```
c.commission_pct = e.commission_pct,
```

```
c.manager_id     = e.manager_id,
```

c.department_id = e.department_id

WHEN NOT MATCHED THEN

INSERT VALUES(e.employee_id, e.first_name, e.last_name,
e.email, e.phone_number, e.hire_date, e.job_id,
e.salary, e.commission_pct, e.manager_id,
e.department_id);

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥٧)

معالجةمجموعات من البيانات الضخمة

Manipulating Large Data Sets

تتبع التحديقات على البيانات Tracking Changes in Data

كثيراً ما يحدث أن تجري تحديقات على صنوف في جدول وتكون غير مرغوبة سواء بإضافة صنف أو حذف صنف أو تعديل بعض الأعمدة عدة مرات بقيم غير مرغوبة؛ ونريد مراجعة هذه التغييرات وإرجاعها إلى أصلها مثلاً.

ومن الجميل أن أوراكل توفر وسيلة للاستعلام عن مختلف التغييرات التي تحد على صنف أو صنوف خلال فترة زمنية يحددها المستخدم؛ حيث تظهر كل التغييرات خلال هذه الفترة؛ وذلك من خلال استعلام يسمى استعلام استرجاع النسخ Falshback version query.

```
SELECT salary FROM employees3
```

```
WHERE employee_id = 107; (1)
```

SALARY
4200

```
UPDATE employees3 SET salary = salary * 1.30
```

```
WHERE employee_id = 107;
```

```
COMMIT; (2)
```

```
SELECT salary FROM employees3 (3)
```

VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE

```
WHERE employee_id = 107;
```

SALARY
5460
4200

في الجملة رقم (١) استعلممنا عن راتب الموظف رقم ١٠٧ فكان ٤٢٠٠ . في الجملة رقم (٢) تمت زيادة الراتب بمقدار ٣٠٪ . لاحظ جملة التثبيت commit لأن لي عليها تعقيباً . أما في الجملة رقم (٣) فتبيننا التغييرات (النسخ versions) التي حدثت على الراتب؛ فأول مرة كان ٤٢٠٠ وبعد تنفيذ الجملة (٢) صار الراتب ٥٤٦٠ وكان ترتيب التحديقات الأحدث

فالأقدم. لنلق نظرة أعمق على الجملة رقم (٣) خاصة عبارة **VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE**

١- تقع هذه العبارة بعد اسم الجدول/الجدوال مباشرة في عبارة **FROM** وقبل عبارة **WHERE**

٢- عبارة **BETWEEN SCN** تعطي تعليمات لخادم أوراكل Oracle Server (نظام إدارة قواعد البيانات) بالبحث عن التغييرات باستخدام رقم تغير النظام (تعطي أوراكل لأية عملية تجاري على قاعدة البيانات رقمًا وحيدًا يسمى رقم تغيير النظام System Change Number SCN)

٣- العمودان الوهيان **MINVALUE AND MAXVALUE** يعنيان أن يقوم خادم أوراكل بالبحث ابتداءً من أصغر رقم تغير النظام وانتهاءً بآخر رقم تغير النظام وهذا يعني جلب كل التغييرات ويعني أيضًا أننا لسنا بحاجة لمعرفة هذين الرقمين

ذكرنا سابقاً أن المستخدم الحالي يرى التغييرات التي أحدثتها على قيم الأعمدة من غير COMMIT ولكن كي يراها الآخرون يجب استعمال جملة التثبيت COMMIT. إذن لماذا أضفنا جملة التثبيت في جملة رقم ٢ - التحديث UPDATE؟

السبب في ذلك أن عبارة **VERSIONS** لا تعرف عن التغييرات إلا بعد تثبيتها شأنها شأن المستخدمين الآخرين؛ والحقيقة أن البحث عن القيم الجديدة يتم بعد اكتشافها من قبل المستخدمين الآخرين فتحتاج عبارة التثبيت.

ولكن هل يمكننا معرفة تواريخ وأوقات هذه التحديثات؟ والجواب نعم كما في المثال التالي:

```
SELECT VERSIONS_STARTTIME "Start Date",
```

```
VERSIONS_ENDTIME "End Date", salary
```

```
FROM EMPLOYEES
```

```
VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
```

```
WHERE last_name = 'Lorentz';
```

Start Date	End Date	Salary
07/09/20 12:25:40.000000000 pm		5187
07/09/20 12:23:19.000000000 pm	07/09/20 12:25:40.0000000 pm	5460
	07/09/20 12:23:19.0000000 pm	4200

لقد عرفنا تواريخ وأوقات وقيم عمود ما خلال فترة معينة مما يمكننا من استعادة القيم التي نريدها. حيث أن العمود **version_starttime** يمثل بداية التغيير مع الأخذ بعين الاعتبار أنه إذا كانت قيمته اللاحية NULL فتعني أن هذه النسخة وجدت قبل أن يبدأ وقت الاستعادة (المحافظة على البيانات) undo retention time. أما قيمة العمود

فتعني أن النسخة بقىت حتى هذا الوقت؛ أما إن كانت قيمته الالشيه NULL فيعني أن النسخة موجودة وقت الاستعلام.

إن جملة الاستعلام عن التغييرات (النسخ) تعكس التغييرات التي حدثت خلال فترة الاستعادة (المحافظة على البيانات) undo retention والتي من خلالها نعرف القيم المرغوبة للحفاظ عليها كقيم دائمة.

ويمكننا معرفة نوع العملية التي حدثت على الصفي (إضافة أو تحرير) المستخدم الذي قام بها وهذا موضوع طويل يبحث في إدارة قواعد البيانات.

وفترة الاستعادة يتم تحديدها تلقائياً بمدة ٩٠٠ ثانية يقوم مدير قاعدة البيانات بتغييرها حسب متطلبات العمل من خلال تعديل ملف الوسائط database parameters file

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥٨)

مراجعة وإضافات على استخراج التقارير الإحصائية

التعامل مع دوال المجموعات Group functions

في هذا الدرس سنراجع ونتوسع قليلاً في دوال المجموعات لاستخراج احصائيات أكثر معنىً مثل المحاميع الجزئية.

مراجعة جملة الاستعلام المحتوية على دوال المجموعات وصيغتها العامة كما نعلم هي:

```
SELECT [column,] group_function(column) . . .
FROM      table
[WHERE   condition];
```

وهذا هو الشكل البسيط وكمثال على ذلك:

```
SELECT AVG(salary), STDDEV(salary), COUNT(commission_pct), MAX(hire_date)
FROM employees
WHERE job_id LIKE 'SA%';
```

كما هو واضح استخدمنا أربع دوال هي المتوسط AVG والانحراف المعياري COUNT والتعداد STDDEV والأكبر(MAX). وقد مر علينا أن بعض هذه الدوال يتعامل مع الأعداد فقط بينما الآخر يتعامل مع كل أنواع البيانات. فالدوال COUNT, AVG, SUM, STDDEV, SUM, VARIANCE لا تقبل إلا الأعداد كمدخلات؛ بينما الدوال التعداد MAX وأصغر MIN تقبل أي نوع من أنواع البيانات مع ملاحظة أن قيم اللاشيء لا تؤخذ بعين الاعتبار ما لم تُضف عبارة ALL ما عدا التعداد COUNT. المثال يستخرج احصائيات عن الموظفين الذين تبدأ وظائفهم بـ SA.

مراجعة عبارة التصنيف GROUP BY وعبارة الترتيب ORDER BY

نضيف عبارة التصنيف GROUP BY ليكون التجميع أكثر تفصيلاً بينما نضيف عبارة الترتيب لترتيب النتائج حسب رغبتنا بأن خادم أوراكل Oracle server يربّ التنتائج تصاعدياً حسب العمود المذكور في عبارة SELECT. والصيغة العامة لإضافة العبارتين السابقتين هي:

```
SELECT [column,] group_function(column) . . .
FROM      table
[WHERE   condition]
```

[GROUP BY *group_by_expression*]

[ORDER BY *column*];

في عبارة SELECT نذكر الأعمدة التي نريد التصنيف على أساسها. مثال:

```
SELECT department_id, job_id, SUM(salary), COUNT(employee_id)
```

```
FROM employees
```

```
GROUP BY department_id, job_id ;
```

في المثال أعلاه نريد استخراج مجموع الرواتب وعدد الموظفين في كل قسم وحسب الوظيفة. لذلك ذكرنا العمودين GROUP BY department_id و job_id في عبارتي SELECT و WHERE. وهذا يعني إعطاء التعليمات لخادم أوراكل أن يستخرج المجموع والعدد حسب القسم ثم عرض المجموع والعدد حسب الوظيفة في نفس القسم.

HAVING عبارة مراجعة

مررت معنا عبارة WHERE وتستعمل لتقييد عدد الصفوف الناتجة من الاستعلام؛ وعبارة HAVING مشابهة ولكنها تؤثر على مجموعات الصفوف وليس صفا واحدا فقط كما في WHERE وتحديدا تستعمل مع دوال المجموعات GROUP FUNCTIONS لتقييد عدد المجاميع الناتجة. والشكل العام لها هو:

```
SELECT [column,] group_function(column)...
```

```
FROM table
```

```
[WHERE condition]
```

```
[GROUP BY group_by_expression]
```

```
[HAVING having_expression]
```

```
[ORDER BY column];
```

وللمراجعة لنأخذ مثلا كما يلي:

```
SELECT SUM(salary),job_id
```

```
FROM employees
```

```
GROUP BY job_id
```

```
HAVING SUM(salary) > 30000
```

ORDER BY SUM(salary) DESC

هنا قيدنا نتائج مجموع الرواتب التي تظهر بحيث يكون المجموع أكبر من ٣٠٠٠٠ . لاحظ أن المثال يرتب النتائج حسب مجموع الراتب من الأعلى فال أقل عكس الترتيب الافتراضي الذي يرتب من الأصغر فال أكبر . ولا حظ أيضاً أن يجب ذكر العمود المذكور في عبارة SELECT في عبارة GROUP BY وإلا سيظهر لك خطأ يدعو إلى الحيرة .

العامل ROLLUP

يستخدم هذا العامل مع group by على أكثر من عمود لاستخراج الإحصائيات الجزئية على مستوى عمود ثم مستوى العمود الذي يندرج تحته هذا العمود ثم الإحصائية الكلية في الاستعلام . فمثلاً لو أردنا مجموع الرواتب حسب القسم وحسب نوع الوظيفة في كل قسم فإننا نستخدم group by rollup مع group by ثم يتم استخراج المجموع الكلي الناتج من الاستعلام . مثال :

```
SELECT department_id, job_id, SUM(salary)
```

```
FROM employees
```

```
WHERE department_id < 60
```

```
GROUP BY ROLLUP(department_id, job_id);
```

وهذه هي النتائج .

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10		4400
20	MK_MAN	(1) 13000
20	MK_REP	6000
20		(2) 19000
30	PU_MAN	11000
30	PU_CLERK	13900
30		24900
40	HR_REP	6500
40		6500

50	ST_MAN	36400
50	SH_CLERK	64300
50	ST_CLERK	55700
50		156400
	(٣)	211200

شرح النتائج:

مجموع رواتب الوظيفة MK_MAN ضمن القسم ٢٠ هو ١٣٠٠٠ (١) ومجموع رواتب الوظيفة MK_REP في نفس القسم هو ٦٠٠٠ (٢) أيضاً ومجموع رواتب دوال القسم كلها هو ١٩٠٠٠ رقم (٣). أما رقم (٣) فيمثل مجموع رواتب دوال جميع الأقسام الواردة في جملة الاستعلام.

CUBE العامل

أيضاً هذا العامل هو توسيع لعمل group by بحيث يستخرج لنا مجموعات إحصائيات أكثر من cube. فلو أردنا معلومات أكثر في المثال السابق؛ نستعمل cube بحيث نحصل على مجموع الرواتب الكلية ومجموع رواتب كل وظيفة لوحدها في القسم ومجموع الرواتب لكل قسم ومجموع الرواتب للقسم والرواتب معاً. لنكتب المثال السابق باستعمال :cube

```

SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY CUBE (department_id, job_id);
  
```

وهذه هي النتائج:

DEPARTMENT_ID	SALARY	SUM(SALARY)
		(١) 211200
	HR_REP	6500
	MK_MAN	(٢) 13000
	MK_REP	6000
	PU_MAN	1100

	ST_MAN	36400
	AD_ASST	4400
	PU_CLERK	13900
	SH_CLERK	64300
	ST_CLERK	55700
10		4400
10	AD_ASST	4400
20		19000
20	MK_MAN	(٣) 13000
20	MK_REP	6000
30		(٤) 24900
30	PU_MAN	11000
30	PU_CLERK	13900
40		6500
40	HR_REP	6500
50		156400
50	ST_MAN	36400
50	SH_CLERK	64300
50	ST_CLERK	55700

(١) مجموع رواتب جميع موظفي الشركة بغض النظر عن القسم أو الوظيفة. لاحظ أن الحقلين DEPARTMENT_ID و JOB_ID فارغان أي أن قيمتيهما هي الالاشيء NULL

(٢) مجموع رواتب الوظيفة MK_MAN بغض النظر عن القسم. لاحظ أن حقل رقم القسم DEPARTMENT_ID قيمته الالاشيء NULL وهذا يعني يهمنا مجموع الرواتب الوظيفة فقط في الشركة بغض النظر عن القسم

(٣) مجموع رواتب الوظيفة MK_MAN ضمن القسم رقم ٢٠ . لاحظ أن كلا الحقلين JOB_ID و DEPARTMENT_ID يحملان قيمة وهذا يعني أخذهما معا بعين الاعتبار.

(٤) مجموع الرواتب في القسم رقم ٣٠ مثلا وحده بغض النظر عن الوظيفة. لاحظ أيضا أن حقل الوظيفة لا يحمل قيمة NULL.

إن العاملين ROLLUP و CUBE يوفران علينا كتابة أكثر من جملة استعلام لاستخراج المجاميع الجزئية والكلية حسب الأعمدة المذكورة في جملة الاستعلام. كما يوفران وقتنا لكتابه برامج خاصة أو إضافة أعمدة ملخصات summary columns كما هو الحال في تقارير أوراكل Oracle Reports بالرغم من سهولتها.

وهنالك أساليب أخرى لاستخراج المللخصات والإحصائيات تركتها لتشعبها وتعقيدها مما يخرج هذه الدروس عن بساطتها.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٥٩)

إدارة المناطق الزمنية المختلفة

Managing Data in Different Time Zones

في عالمنا الواسع صار للمؤسسات حكومية أو خاصة فروع في مختلف المناطق في العالم. فشركات الطيران مثلاً لها فروع في بلدان مختلفة؛ وشركات التصدير وشركات الشحن كذلك. وتختلف الأوقات من بلد لآخر ومن منطقة جغرافية لأخرى. وتم تقسيم الكره الأرضية إلى ٢٤ منطقة زمنية أساسها بعد عن خط الطول المار في منطقة غرينيتش البريطانية. فمثلاً في السعودية الوقت يفرق ٣ ساعات عن غرينيتش؛ وبين جدة والرياض يوجد نصف ساعة فرق ولكن التوقيت الواحد؛ بينما في الولايات المتحدة التوقيت ليس واحداً بل يعتمد على المنطقة الجغرافية فيوجد أربع ساعات بين الشاطئ الشرقي (نيويورك مثلاً) والشاطئ الغربي (سان فرانسيسكو مثلاً) وهكذا.

والمؤسسات ذات الفروع المتعددة تراعي اختلاف التوقيت في مختلف المناطق؛ وقدمت أوراكل دعماً قوياً للتوقیتات المختلفة حسب المناطق الجغرافية بحيث يتم متابعة الحركة على البيانات بما حسب غرينيتش أو ما يعرف التوقيت العالمي مثلاً أو حسب التوقيت المحلي أو حسب أي توقيت تختاره الشركة سواءً موحداً في جميع الفروع أو تعطي الحرية لكل فرع أن يتعامل مع التوقيت المحلي. مع ملاحظة أنه عند التنقل بين التوقيتين الصيفي والشتوي توجد فترة مفقودة. فيتم التحول عادة الساعة ١٢ منتصف الليل (بداية يوم جديد) فتصبح الساعة بدلًا من ١٢ نصف الليل تصبح ١ ليلاً (صباحاً) فالوقت بين الساعة ١٢ منتصف الليل والساعة ١ صباحاً غير معروف أو مفقود؛ وكذلك الحال عند الرجوع إلى التوقيت الشتوي من ١٢ منتصف الليل إلى ١١ ليلاً توجد ساعة مفقودة أو غير معروفة وبالتالي فالتوقيت غير معروف.

وعلى كل حال قدمت أوراكل العديد من الدوال للتعامل مع الوقت سنتعرف عليها واحداً واحداً: وقد لا يستخدمها الكثير منها ولكن معرفتها مهمة لمن يتعاملون في بيانات من مختلف المناطق الجغرافية.

+ أول شيء نبحثه هو تغيير توقيت الجلسة session أي بداية الدخول على قاعدة البيانات أوراكل؛ ويتم ذلك من خلال الوسيط parameter الذي اسمه TIME_ZONE. ويتم ذلك باستخدام:

١ - عدد الساعات عن جرينيتش أو التوقيت العالمي absolute offset مثال:

```
ALTER SESSION SET TIME_ZONE = '-05:00';
```

عدد الساعات يكون بالمحجب يعني قبل التوقيت العالمي (Coordinated Universal Time) UTC أي أنها نسبتاً غرينيتش بينما العدد السالب يعني أنها بعد التوقيت العالمي

٢ - منطقة توقيت قاعدة البيانات Database time zone مثال:

```
ALTER SESSION SET TIME_ZONE = dbtimezone;
```

٣- منطقة توقيت نظام التشغيل OS local time zone مثال:

```
ALTER SESSION SET TIME_ZONE = local;
```

٤- اسم المنطقة الزمنية A named region . مثال:

```
ALTER SESSION SET TIME_ZONE = 'America/New_York';
```

الدوال: LOCALTIMESTAMP و CURRENT_TIMESTAMP و CURRENT_DATE

١- الدالة CURRENT_DATE: تستعمل لعرض تاريخ وقت جلسة العمل؛ حيث أن الوقت حسب توقيت المنطقة الزمنية لجلسة العمل. إذا لم يجبر تغيير المنطقة الزمنية لجلسة العمل ف تكون هذه المنطقة هي الافتراضية. سلسلة الجمل التالية لتوضيح نتائج هذه الدوال:

```
ALTER SESSION
```

```
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

هذه الجملة لتغيير شكل مخرجات التاريخ والوقت بدلاً من الشكل الافتراضي. نذكر أننا استعملنا سابقاً الدالة TO_CHAR لتشكيل مخرجات التاريخ (يرجى مراجعتها). ولكن باستعمال الوسيط NLS_DATE_FORMAT لا نضطر إلى تلك الدالة طوال جلسة العمل وهذا يعني أننا إذا أنهينا الجلسة يتغير علينا إما تغيير الشكل باستعمال الوسيط سالف الذكر أو التحويل التي درسناها سابقاً.

نريد هنا استخراج التاريخ حسب الشكل يوم-اسم اليوم المختصر-السنة من أربعة أرقام والوقت حسب نظام الـ ٢٤ ساعة والدقائق والثواني.

```
SELECT CURRENT_DATE FROM DUAL;
```

ستكون المخرجات هي:

١٦-سبتمبر-٢٠٢٠ ١٣:٣٩:٥٣

في جهازي نظام تاريخ الشهور باللغة العربية لذلك لم يتم اختصار اسم الشهر إلى ٣ حروف كما هو في اللغة الإنجليزية. وقد تختلف هذه النتيجة لديكم.

والآن سأغير المنطقة الزمنية لجلسة العمل لتكون بعد التوقيت العالمي (UTC)(Coordinated Universal Time) بخمس ساعات.

```
ALTER SESSION SET TIME_ZONE = '-5:0';
```

```
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_DATE
05:00-	١٦-سبتمبر-٢٠٢٠ ٠٥:٥١:٣٠

وهذا مثال آخر:

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_DATE
-----------------	--------------

-08:00	18-SEP-2020 10:00:27
--------	----------------------

٣- الدالة CURRENT_TIMESTAMP: تستعمل لعرض تاريخ ووقت جلسة العمل؛ إضافة إلى بعد المنطقة الزمنية عن التوقيت العالمي بالساعات والدقائق وعرض أجزاء الثانية في حقل الوقت بحد أقصى ٩ خانات عشرية إذا حددها المستخدم أو ٦ خانات عشرية وهي الدقة الافتراضية. علما بأن الوقت يكون حسب توقيت المنطقة الزمنية لجلسة العمل. إذا لم يجر تغيير المنطقة الزمنية لجلسة العمل فتكون هذه المنطقة هي الافتراضية. سلسلة الجمل التالية لتوضيح نتائج هذه الدالة

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP
FROM DUAL;
```

في الجملة الأولى غيرنا المنطقة الزمنية لجلسة الحالية بحيث تكون بعد التوقيت العالمي بخمس ساعات؛ وفي الجملة الثانية طلبنا عرض المنطقة الزمنية لجلسة العمل الحالية(للتأكد) وطلبنا أيضا عرض مخرجات ال current_timestamp.

SESSIONTIMEZONE	CURRENT_TIMESTAMP
-----------------	-------------------

-05:00	20-SEP-20 04.11.52.719000 AM -05:00
--------	-------------------------------------

نلاحظ دقة الثواني ل ٦ خانات عشرية ونلاحظ عرض بعد المنطقة الزمني عن التوقيت العالمي ب (٥-) ساعات

```
SELECT CURRENT_TIMESTAMP(8) FROM DUAL;
CURRENT_TIMESTAMP(8)
```

20-SEP-20 04.18.32.03600000 AM -05:00

في المثال السابق أردنا عرض الوقت بحيث تكون دقة الثواني إلى ٨ خانات عشرية أي بدقة جزء من عشرة ملايين جزء من الثانية! مثل هذه الدقة مطلوبة في كثير من الأبحاث العلمية في المختبرات وأبحاث الميكروبات... كما أنها تفيد في دقة نتائج المباريات الرياضية مثل رياضة الجري!!!!

٣- الدالة LOCALTIMESTAMP

تعرض لنا هذه الدالة التوقيت المحلي باليوم والشهر والسنة وال دقائق والثواني وأجزاء الثانية حتى جزء من ١٠٠ مليون جزء أي ٩ خانات عشرية وإذا لم تحدد يكون عدد الخانات الافتراضي ٦ خانات فقط. والفرق بينها وبين الدالة CURRENT_TIMESTAMP هو أن هذه الدالة تعطينا بيانات التوقيت المحلي دون عرض المنطقة الزمنية بينما الدالة CURRENT_TIMESTAMP تعرض لنا المنطقة الزمنية. مثال:

```
SELECT CURRENT_TIMESTAMP "Current", LOCALTIMESTAMP "Local" FROM DUAL;
```

Current	Local
-----	-----

21-SEP-20 10.21.39.913000 AM +03:00	21-SEP-20 10.21.39.913000 AM
-------------------------------------	------------------------------

لاحظ نفس التوقيت في الحالتين ولكن في الأولى تم عرض المنطقة الزمنية-البعد عن التوقيت العالمي

٤- الدالة DBTIMEZONE وهذه الدالة تعرض المنطقة الزمنية الخاصة بقاعدة البيانات أي الخاصة بخادم أوراكل عند تنصيب أوراكل على السيرفر. وهذه يمكن تغييرها بالجملة

```
ALTER DATABASE SET TIME_ZONE =requiredtimezone
```

ونحتاج إلى إغلاق shutdown قاعدة البيانات ثم بدأها startup مرة ثانية. ولا يمكن تغييرها باستعمال الجملة ALTER SESSION. مثال:

```
SELECT DBTIMEZONE FROM DUAL;
```

```
DBTIME
```

```
-----
```

```
+00:00
```

في هذا المثال فإن وقت المنطقة الزمنية لقاعدة البيانات هي منتصف الليل للتوقيت العالمي لأن DBA لم يحدد منطقة زمنية معين أثناء تنصيب أوراكل.

٥- الدالة SESSIONTIMEZONE وهي تعرض توقيت (المنطقة الزمنية) جلسة العمل الحالية التي تختلف عن توقيت

قاعدة البيانات؛ مثال:

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

```
SESSIONTIMEZONE
```

+03:00

لاحظ اختلاف المنطقة الزمنية لكل من جلسة العمل وقاعدة البيانات علما بأن المثالين تم تطبيقهما على نفس الجهاز. إن المنطقة الزمنية تعرض أو تستعمل حسب الصيغة TIME_ZONE = '[+ | -] hh:mm' مع ملاحظة أن علامة الزائد والناقص اختياريتان وعدم ذكرهما يعني أن التوقيت قبل التوقيت العالمي وهو يعني + وهو الأمر الافتراضي.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٠)

إدارة المناطق الزمنية المختلفة

Managing Data in Different Time Zones

مراجعة البيانات من نوع دمجة الوقت

وهذا النوع هو توسيع لنوع البسيط `date` والذي يحتوي على السنة والول يوم والشهر والساعات والدقائق والثواني؛ فإضافة إلى ذلك يحتوي على أجزاء الثانية من صفر خانة عشرية إلى ٩ خانات عشرية كحد أقصى بعدد خانات افتراضي ٦ خانات عشرية. وله الصور التالية:

١- أن يتضمن على أجزاء الثانية إضافة إلى باقي الأجزاء علما بأن ما بداخل الأقواس المربعة اختياري؛ وشكله هو :

`TIMESTAMP[fractional_seconds_precision]`

٢- أن يتضمن على المنطقة الزمنية وشكله هو :

`TIMESTAMP [(fractional_seconds_precision)]_ WITH TIME ZONE`

٣- أن يتضمن على المنطقة الزمنية المحلية وشكله هو :

`TIMESTAMP [(fractional_seconds_precision)]_ WITH LOCAL TIME ZONE`

مع العلم أن البيانات تخزن في قاعدة البيانات حسب إعدادات المنطقة الزمنية لقاعدة البيانات ولكنها تعرض حسب المنطقة الزمنية جلسة العمل. والجدول التالي يصف حقول النوع `TIMESTAMPE`

الحقل	وصف الحقل
YEAR	السنة من -٤٧١٢ (قبل الميلاد) إلى ٩٩٩٩ علماً بأنه لا يوجد شيء اسمه السنة صفر
MONTH	ويتراوح بين ١ إلى ١٢ (٠١-١٢)
DAY	رقم اليوم في الشهر من ١ إلى ٣١ (٠١-٣١)
HOUR	الساعة بالتوقيت العالمي من ٠ إلى ٢٣ (٠٠-٢٣)
MINUTE	الدقيقة من ٠ إلى ٥٩ (٠٠-٥٩)

الثواني من ٠ إلى (٥٩٠٩٠٠) حيث نعبر عن عدد الخانات العشرية لأجزاء الثانية وهي من صفر خانة إلى ٩ خانات عشرية	SECOND
ساعات المنطقة الزمنية من ١٢ إلى ١٤ (من -١٢ إلى ١٤). السالب يعني بعد التوقيت العالمي والوجب قبل التوقيت العالمي	TIMEZONE_HOUR
دقائق المنطقة الزمنية من ٠ إلى ٥٩ (٥٩-٥٠). السالب يعني بعد التوقيت العالمي والوجب قبل التوقيت العالمي	TIMEZONE_MINUTE

لأخذ مثلا لنرى الفرق بين النوع تاريخ البسيط DATE والنوع دمجة الوقت TIMESTAMP. سنضيف عمودا على جدول الموظفين EMPLOYEES من نوع دمجة الوقت TIMESTAMP ثم ننسخ إليه قيم العمود HIRE_DATE الذي نوعه تاريخ بسيط ثم نعرض التاريخين لنرى الفرق بينهما.

```
ALTER TABLE employees ADD hire_date1 TIMESTAMP;
```

Table altered.

```
UPDATE employees set hire_date1 = hire_date;
```

107 rows updated.

```
SELECT HIRE_DATE1 , HIRE_DATE FROM EMPLOYEES;
```

وهذه عينة من النتائج:

HIRE_DATE1	HIRE_DATE
17-JUN-03 12.00.00.000000000 AM	17-JUN-03
21-SEP-05 12.00.00.000000000 AM	21-SEP-05
13-JAN-01 12.00.00.000000000 AM	13-JAN-01
03-JAN-06 12.00.00.000000000 AM	03-JAN-06
21-MAY-07 12.00.00.000000000 AM	21-MAY-07
25-JUN-05 12.00.00.000000000 AM	25-JUN-05
05-FEB-06 12.00.00.000000000 AM	05-FEB-06
07-FEB-07 12.00.00.000000000 AM	07-FEB-07

17-AUG-02 12.00.00.000000000 AM	17-AUG-02
16-AUG-02 12.00.00.000000000 AM	16-AUG-02

واضح كيف تم عرض العمود hire_date1 بدمغة الوقت بحيث يشمل كل الحقول ولاحظ أن عدد الخانات العشرية لأجزاء الثانية هو ٦ خانات وهو الوضع الافتراضي ؛ بينما تم عرض العمود hire_date فقط بثلاثة حقول هي اليوم والشهر والسنة

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثala (٦١)

إدارة المناطق الزمنية المختلفة

Managing Data in Different Time Zones

مراجعة البيانات من نوع دمجة الوقت مع المنطقة الزمنية  TIMESTAMP WITH TIME ZONE data type

ولا يختلف كثيراً عن سابقه TIMESTAMP إلا بإضافة المنطقة الزمنية عند تعريف البيانات بحيث يمكن تخزين البيانات في قاعدة البيانات حسب المنطقة الزمنية. وصيغته العامة هي :

TIMESTAMP[(fractional_seconds_precision)] WITH TIME ZONE

وهو يعبر عن بعد المنطقة الزمنية من التوقيت العالمي Coordinated Universal Time(UTC) المعروف سائقاً بالتوقيت المتوسط لجرينيش Greenwich Mean Time(GMT). ولنأخذ مثلاً بإنشاء جدول يحتوي على تاريخ

TIMESTAMPW WITH TIME ZONE الحركة من نوع

```
CREATE TABLE web_orders
(ord_id number primary key,
order_date TIMESTAMP WITH TIME ZONE);
```

ونضيف صفاً إليه كما يلي:

```
INSERT INTO web_orders values (ord_seq.nextval, current_date);
```

أضفنا رقم الأمر ord_id باستخدام السلسلة ord_seq ووقت النظام الحالي كتاريخ للأمر. ثم عرضنا الصيغة التي أضفناها

```
SELECT * FROM web_orders;
```

```
ORD_ID ORDER_DATE
```

```
----- -----  
101 25-SEP-20 12:02:57.000000 PM +03:00
```

بكل بساطة تم عرض التاريخ متضمناً بعد المنطقة الزمنية عن التوقيت العالمي بـ ٣+ ساعات أي أنها تسبق التوقيت العالمي بثلاث ساعات. وبهذه الطريقة يمكن للشركة أن تتوقع الوقت الذي يتم فيه تسليم شحنتها من خلال الوقت الذي تم إدخال البيانات فيه من أي منطقة زمنية تواجد فيها الشركة

النوع دمغة الوقت بالتوقيت المحلي

وهنا لا يتم تخزين البعد displacement عن التوقيت العالمي في الجدول ولكن يتم تخزين التوقيت حسب توقيت قاعدة البيانات ويتم عرض البيانات حسب توقيت جلسة العمل. والشكل العام هو:

TIMESTAMP WITH LOCAL TIMEZONE

سنأخذ مثلاً :

```
CREATE TABLE shipping (delivery_time TIMESTAMP WITH LOCAL TIME  
ZONE);
```

أنشأنا الجدول بعمود نوعه التوقيت المحلي وسوف نضيف سطراً إليه بعد التاريخ الحالي بيومين :

```
INSERT INTO shipping VALUES(current_timestamp + 2);
```

سنعرض الآن محتويات السطر

```
SELECT * FROM shipping;
```

```
DELIVERY_TIME
```

```
-----  
27-SEP-20 12:26:20.000000, PM
```

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٢)

إدارة المناطق الزمنية المختلفة

Managing Data in Different Time Zones

النوع الفترة INTERVAL وهو يحدد فترة زمنية معينة كنوع بيانات في لعمود في جدول وينقسم إلى قسمين؛ قسم يحدد السنوات والشهور وقسم يحدد الأيام وال ساعات والثانية وأجزاء الثانية والمدخل التالي يلخص هذا النوع

المقول	نوع البيانات
السنة، الشهر (Year, Month)	INTERVAL YEAR TO MONTH
يوم، ساعة، دقيقة، ثانية، أجزاء الثانية (Day, Hour, Minute, Second with fractional seconds)	INTERVAL DAY TO SECOND

النوع فترة سنة-شهر

هذا النوع يستخدم لتخزين الفترات الزمنية المكونة من سنوات وشهور وهو يتكون من حقلين؛ أحدهما للسنوات والآخر للشهور.
وشكله القياسي العام كما يلي:

YEAR [(year_precision)] TO MONTH

حيث (year_precision) هو عدد الخانات التي تعبّر عن عدد السنين في الحقل YEAR بحد أقصى ٩ خانات والافتراضي خانتان فقط. والحقل YEAR الذي يمثل عدد السنين قد يكون بالسالب والوجب. أما حقل الشهر MONTH فعدد صحيح موجب تتراوح قيمته من صفر إلى ١١ فقط.

تقييد

يجب أن يكون الحقل الأول أكبر من الحقل الثاني . فعلى سبيل المثال التركيب التالي غير مقبول

MONTH TO YEAR

لأن الحقل الأول MONTH هو صفر بينما الحقل الثاني YEAR هو واحد والتركيب يقول أن الدقة هي صفر شهر إلى سنة(أقرب إلى سنة) وهو غير صحيح. فالدقة سنة ثم شهر. الأمثلة التالية صحيحة:

YEAR(3) TO MONTH

يعني فترة ١٢٣ سنة وثلاثة شهور

INTERVAL '123' YEAR(3)

يعني فترة ١٢٣ سنة وصفر شهر

INTERVAL '3' MONTH

يعني فترة ثلاثة شهور. في هذا المثال الأخير لم يتم ذكر الحقل YEAR فعرفنا أن الفترة المذكورة بالشهور أما في المثال الثاني لم يتم ذكر الحقل MONTH فذلك أخبرنا أن العدد المذكور يمثل فترة سنوات. أما في المثال الأول فتم ذكر الحقلين فهذا يعني أن الفترة هي بالسنوات والشهر وهنا مثال لجدول مبني على نوع البيانات السابق:

CREATE TABLE warranty

(prod_id number, warranty_time INTERVAL YEAR(3) TO MONTH);

INSERT INTO warranty VALUES (123, INTERVAL '8' MONTH);

INSERT INTO warranty VALUES (155, INTERVAL '200' YEAR(3));

INSERT INTO warranty VALUES (678, '200-11');

بعد إنشاء الجدول أضفنا ثلاثة صفوف لفترة الضمان في الأول كانت ٨ شهور وفي الثاني ٢٠٠ سنة أما في الثالث فكانت الفترة ٢٠٠ سنة و ١١ شهر وبعد تنفيذ جملة الاستعلام تم عرض الصنوف كالتالي (لاحظ كيفية عرض حقل الضمان :WARRANTY_TIME)

SELECT * FROM warranty;

PROD_ID	WARRANTY_TIME
123	+000-08
155	+200-00
678	+200-11

لاحظ إشارة الموجب (+) قبل حقل السنة.

النوع الفترة يوم إلى ثانية 

ويستعمل هذا النوع لتخزين بيانات الوقت متضمنا عدد الأيام وال ساعات وال دقائق والثواني وأجزاء الثانية بحد ٩ خانات على يمين الفاصلة وعدد الخانات الافتراضي هو ٦ ما لم يذكر خلاف ذلك. وشكله العام

INTERVAL DAY[(day_precision)] TO Second

بحيث أن day_precision يمثل عدد الخانات التي تعبّر عن عدد الأيام وبحد أقصى 9 أرقام والافتراضي رقمان. فمثلاً ٣٤٥ تعني ثلاثة وخمسة وأربعين يوماً. أما الساعات فهي من ٠ إلى ٢٣. وعدد الدقائق من ٠ إلى ٥٩ والثواني من ٠ إلى ٥٩ (ن) حيث ن تمثل عدد الخانات على يمين الفاصلة. المثال التالي يوضح:

INTERVAL '6 03:30:16.481' DAY TO SECOND

يمثل ٦ أيام و ٣ ساعات و ٣٠ دقيقة و ١٦ ثانية و ٤٨١ جزءاً من ألف جزء من الثانية. والمثال التالي يمثل جدول المختبر كما يلي:

CREATE TABLE lab

```
( exp_id number,
test_time INTERVAL DAY(2) TO SECOND);
```

INSERT INTO lab VALUES (100012, '90 00:00:00');

INSERT INTO lab VALUES (56098, INTERVAL '6 03:30:16' DAY TO SECOND);

ولدى الاستعلام ينتج لدينا

SELECT * FROM lab;

EXP_ID	TEST_TIME
100012	+90 00:00:00.000000
56098	+06 03:30:16.000000

لاحظ إشارة الموجب التي تعني أنه يمكننا إدخال عدد الأيام بالسالب أيضاً. لاحظ أيضاً عدد الخانات العشرية في حقل الثواني ظهر تلقائياً ولو لم ندخله لأنه ضمن خصائص هذا النوع.

استخراج الـ EXTRACT Function

تستخدم هذه الـ EXTRACT لاستخراج مكونات فترة زمنية. فستخرج السنة YEAR والشهر MONTH على شكل رقم؛ وستخرج رقم اليوم DAY وال الساعة HOUR والدقيقة MINUTE والثانية SECOND كما يتم استخراج ساعة المنطقة الزمنية TIMEZONE_HOUR ودقيقة المنطقة الزمنية TIMEZONE_MINUTE واسم المنطقة TIMEZONE_REGION والشكل العام TIMEZONE_ABBR و اختصار اسم المنطقة TIMEZONE_ABBR هو:

```
SELECT EXTRACT ([YEAR] [MONTH][DAY] [HOUR] [MINUTE][SECOND]
[TIMEZONE_HOUR] [TIMEZONE_MINUTE]
[TIMEZONE_REGION] [TIMEZONE_ABBR]
```

```
FROM [datetime_value_expression] [interval_value_expression]);
```

حيث يتم استخراج حقل واحد فقط في المرة الواحدة. في حال استخرجنا الحقلين TIMEZONE_REGION و TIMEZONE_ABBR فإن نوع النتيجة هو رموز؛ أما حقول السنة والشهر واليوم والسنة والدقيقة والثانية فهي من نوع تاريخ-وقت DATETIME حسب التقويم الجريجوري GREGORIAN CALENDAR أما في حالة حقول المنطقة الزمنية TIMEZONE فالنتيجة تكون حسب التوقيت العالمي UTC . مثال:

```
SELECT EXTRACT(YEAR FROM SYSDATE) "Year" ,EXTRACT(MONTH  
FROM SYSDATE) "Month, "
```

```
EXTRACT(DAY FROM SYSDATE)"Day" FROM DUAL
```

Year	Month	Day
2020	10	11

مثال آخر: نريد معرفة الشهر الذي تم فيه توظيف الموظفين الذين مديرهم رقم ١٠٠ :

```
SELECT last_name, hire_date,
```

```
EXTRACT (MONTH FROM HIRE_DATE)
```

```
FROM employees
```

```
WHERE manager_id = 100;
```

LAST_NAME	HIRE_DATE	EXTRACT(MONTHFROMHIRE_DATE)
Kochhar	21-SEP-89	9
De Haan	13-JAN-93	1
Mourgos	16-NOV-99	11
Zlotkey	29-JAN-00	1
Hartstein	17-FEB-96	2

الدالة TZ_OFFSET . والمهدف منها عرض بعد المنطقة الزمنية المحددة كمدخلات للدالة بالساعات. فلو أردنا معرفة عدد الساعات عن التوقيت العالمي لمنطقة الزمنية 'US/Eastern' نكتب الاستعلام التالي:

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;
```

فتشاهد النتيجة

TZ_OFFSET
-04:00

يعني أربع ساعات بعد التوقيت العالمي. أما بعد عن التوقيت العالمي للمنطقة الزمنية 'Asia/Riyadh' فنكتب الاستعلام التالي:

```
SELECT TZ_OFFSET('Asia/Riyadh') FROM DUAL;
```

TZ_OFFSET

+03:00

لاحظ إشارة الناتج التي تكون موجبة في حالة أن المنطقة الزمنية قبل التوقيت العالمي وسالبة في حالة أن المنطقة الزمنية بعد التوقيت العالمي. يمكننا معرفة أسماء المناطق الزمنية واحتصارها باستعلام العرض V\$TIMEZONE_NAMES كما يلي:

```
SELECT * FROM V$TIMEZONE_NAMES;
```

وفي الشكل التالي عينة من النتائج:

TZNAME	TZABBREV
Africa/Algiers	LMT
Africa/Algiers	PMT
Africa/Algiers	WET
Africa/Algiers	WEST
Africa/Algiers	CET
Africa/Algiers	CEST
Africa/Cairo	LMT
Africa/Cairo	EET
Africa/Cairo	EEST
Africa/Casablanca	LMT
Africa/Casablanca	WET
Africa/Casablanca	WEST
Africa/Casablanca	CET
Africa/Ceuta	LMT
Africa/Ceuta	WET
Africa/Ceuta	WEST

كما يمكننا معرفة بعد المنطقة الزمنية بجلسة العمل SESSIONTIMEZONE ولقاعدة البيانات DBTIMEZONE

```
SELECT TZ_OFFSET(DBTIMEZONE) "DB Offset",
       TZ_OFFSET(SESSIONTIMEZONE) "Session Offset" FROM DUAL;
```

DB Offset Session Offset

----- -----

+00:00

+03:00

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٣)

أساليب إضافية لاستعمال الاستعلامات الجزئية

More using subqueries to retrieve data

سبق وأن ناقشنا الاستعلام الجزئي وقلنا بأنه استعلام عادي نتائجه أساس للاستعلام الأساسي؛ فإننا لا نرى نتائج الاستعلام الجزئي بل هي مختبئة في الذاكرة يستعملها الاستعلام الرئيسي لعرض ما نريد. وكل المبرمجين لا يستغنون عن الاستعلام الجزئي. في هذه السلسلة سوف نلقي الضوء على استعمالات إضافية للاستعلامات الجزئية.

الاستعلامات الجزئية متعددة الأعمدة multiple-column subqueries

درسنا فيما سبق كتابة الاستعلامات الجزئية التي ينتج عنها صفات واحد وتلك التي ينتج عنها أكثر من صفات واستعملنا العوامل المناسبة لكل نوع؛ ولكنها جميعاً تشتراك بمقارنة عمود واحد في الاستعلام الرئيسي بعمود واحد في الاستعلام الجزئي. ماذا لو أردنا أن نقييد نتائج الاستعلام الرئيسي بأكثر من عمود؟ سنكتب الاستعلام الرئيسي متضمناً أكثر من شرط باستعمال الاستعلام الجزئي بعدد الأعمدة المرغوب التقييد بناءً عليها. ولكن الأسهل كتابة الاستعلام الرئيسي متضمناً شرطاً واحداً يحتوي الأعمدة التي نريدها ونقارنها بالأعمدة المذكورة في الاستعلام الجزئي. والصيغة العامة لذلك هي:

```
SELECT    column, column, ...
FROM      table
WHERE     (column, column, ...) IN
          (SELECT column, column, ...
           FROM   table WHERE condition);
```

هنا يتم مقارنة الأعمدة الواردة في عبارة WHERE في الاستعلام الرئيسي مع نظائرها في الاستعلام الجزئي في عبارة SELECT. وسنقتصر على نوعين من الاستعلام الجزئي متعدد الأعمدة. وهما

١ - مزدوج الأعمدة Pairwise columns

٢ - غير مزدوج الأعمدة Non-Pairwise columns

كمثال على النوع الأول؛ نريد بيانات الموظفين الذين مدیرهم والذين يعملون في نفس القسم الذي يعمل فيه كل من الموظف رقم ١٩٩ أو ١٧٤ فنكتب:

```
SELECT    employee_id, manager_id, department_id
```

```

FROM      employees
WHERE (manager_id, department_id) IN
       (SELECT manager_id, department_id
        FROM   employees
        WHERE employee_id IN (199,174))
AND employee_id NOT IN (199,174);

```

هنا يتم مقارنة رقم المدير ورقم القسم الواردين في كل صف من جدول الموظفين في الاستعلام الرئيسي بقيمهما المقابلة الواردة في الاستعلام الخزئي للموظف رقم ١٩٩ أو ١٧٤ وهذه عينة من النتائج

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
141	124	50
142	124	50
...		
11 rows selected.		

وأما في النوع الثاني؛ غير مزدوج الأعمدة؛ فإننا نوزع الأعمدة على أكثر من استعلام جزئي؛ نفس المثال السابق سنكتبه بالصيغة الثانية كما يلي:

```

SELECT employee_id, manager_id, department_id
FROM   employees
WHERE manager_id IN
       (SELECT manager_id FROM employees
        WHERE employee_id IN (174,199))
AND   department_id IN
       (SELECT department_id
        FROM   employees WHERE employee_id IN (174,199))
AND employee_id NOT IN(174,199);

```

لاحظ الرابط بواسطة AND وليس OR؛ لأن المطلوب كلا العمودين MANAGER_ID و DEPARTMENT_ID نفسها لو لاحظتم

والمقصود بالعدي بأن للاستعلام الجزئي قيمة معينة هي التعبير الناتج عن الاستعلام الجزئي؛ بمعنى آخر فهو يعطي قيمة عمود واحد وهو المذكور في الاستعلام. فإذا لم ينتج صفوف من الاستعلام الجزئي فإن قيمة الاستعلام الجزئي هي الالاشيء NULL وإذا أعطت أكثر من الصف ففيتسبب ذلك بخطأ. وبالتالي الاستعلام متعدد الأعمدة يخرج من كونه استعلاماً عددياً. وإذا كان الاستعلام عددياً فيمكن استعماله في الحالات التالية:

١- كجزء من شرط أو تعبير في كل من CASE و DECODE سابقتي الذكر

٢- في كل عبارات جملة الاستعلام SELECT ما عدا عبارة GROUP BY

٣- في عبارتي SET و WHERE في جملة التحديث UPDATE

ولكن لا يمكن أن يستعمل في الحالات التالية:

١- قيمة افتراضية DEFAULT في تعريف الأعمدة عند إنشاء جدول

٢- التعبارات الجزئية HASH Expressions لدى إنشاء العناقيد clusters

٣- في عبارة RETURNING في جمل معالجة البيانات DML

٤- كجزء من دوال الفهارس المبنية على دوال (سق مناقشتها)

٥- في عبارة GROUP BY وقيد التدقيق CHECK وكشرط في عبارة WHEN

٦- في عبارة CONNECT BY. ستمر معنا لاحقاً

٧- في جميع الحالات التي لا تستعمل فيها الاستعلامات الجزئية مثل إنشاء الملف الشخصي CREATE PROFILE .

وهنا مثالان لاستعمال الاستعلام العددي في تعبير CASE وفي عبارة ORDER BY

١- تعبير CASE

SELECT employee_id, last_name,

(CASE

WHEN department_id =

(SELECT department_id

FROM departments WHERE location_id = 1800)

THEN 'Canada' ELSE 'USA' END) location

FROM employees;

في المثال السابق يقوم الاستعلام الجزئي باستخراج رقم القسم في الموقع رقم ١٨٠٠ وهو رقم القسم ٢٠ . وفي الاستعلام الرئيسي يستعمل هذه القيمة لعرض بيانات الموظفين واسم الموقع CANADA في حالة أن القسم رقم ٢٠ و USA في حالة غير ذلك.

وهذه عينة من النتائج

EMPLOYEE_ID	LAST_NAME	LOCATION
196	Walsh	USA
197	Feeney	USA
198	OConnell	USA
199	Grant	USA
200	Whalen	USA
201	Hartstein	Canada
202	Fay	Canada
203	Mavris	USA
204	Baer	USA
205	Higgins	USA
206	Gietz	USA

107 rows selected.

٢- عبارة ORDER BY

```
SELECT employee_id, last_name
FROM employees e
ORDER BY (SELECT department_name
FROM departments d
WHERE e.department_id = d.department_id);
```

هنا فإن الاستعلام الجزئي يستخرج اسم القسم ليقوم الاستعلام الرئيسي بترتيب النتائج بناءً عليه. لاحظ أن الاستعلام الرئيسي يستخرج الصف الأول ثم يقوم الاستعلام الجزئي باستخراج اسم القسم بناءً على رقم القسم من الاستعلام الرئيسي؛ أي أن العملية عكس ما سبق ودرسته من أن الاستعلام الجزئي يتم تنفيذه أولاً ثم يتم تنفيذ الاستعلام الرئيسي بناءً على نتيجة الاستعلام الجزئي. إن الأسلوب الوارد في المثال يسمى الاستعلام الجزئي المرتبط Correlated Subquery وسيتم بحث هذا النوع بالتفصيل في دروس قادمة. عينة من النتائج:

EMPLOYEE_ID	LAST_NAME
205	Higgins
206	Gietz
200	Whalen
100	King
101	Kochhar
102	De Haan
108	Greenberg
109	Faviet

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٤)

أساليب إضافية لاستعمال الاستعلامات الجزئية

More using subqueries to retrieve data

الاستعلامات الجزئية المرتبطة

سبق وأن ناقشنا الاستعلام الجيئي؛ وقلنا إن نتائج الاستعلام الرئيسي تعتمد على نتائج الاستعلام الجيئي. وهذا يعني أن خادم أوراكل Oracle Server يقوم بتنفيذ أول صفت من الاستعلام الجيئي ثم يستخرج من الاستعلام الرئيسي الصنوف المطابقة لقيمة أو قيم الأعمدة المستخرجة من الاستعلام الجيئي.

ولكن في الاستعلام الجيئي المرتبط يحدث العكس إلى حد ما؛ حيث يقوم خادم أوراكل بتنفيذ الاستعلام الجيئي مقابل كل صفت من الاستعلام الرئيسي والخطوات التي يقوم بها خادم أوراكل لتنفيذ مثل هذا الاستعلام هي كما يلي:

١- يستخرج خادم أوراكل الصفت من الاستعلام الرئيسي؛ ويسمى هذا الصفت بالصف المرشح Candidate Row أي أنه مرشح للتعامل معه

٢- تفريذ الاستعلام الجيئي اعتماداً على القيمة من الصفت المرشح (من الاستعلام الرئيسي)

٣- يتم استعمال القيمة من الاستعلام الجيئي لتأهيل Qualify أو عدم تأهيل Disqualify الصفت المرشح. ومعنى تأهيل الصفت المرشح أنه يتم استعمال هذا الصفت بعرضه أو معاجلات أخرى. وعدم تأهيله يعني إهماله

٤- تكرار الخطوات السابقة حتى لا يبقى صنوف مرشحة في الاستعلام الرئيسي.

ونظراً لارتباط نتيجة الاستعلام الجيئي بقيمة عمود في الاستعلام الرئيسي قلنا إنه استعلام مرتبط Correlated

سندرس مثالين ثم نصل إلى الصيغة العامة لكتابة الاستعلام المرتبط.

١- نريد عرض بيانات الموظفين الذين رواتبهم أعلى من متوسط رواتب الأقسام التي يعملون فيها؛ فنكتب:

```
SELECT last_name, salary, department_id
```

```
FROM employees outer
```

```
WHERE salary >
```

```
(SELECT AVG (salary)
```

```
FROM employees
```

```
WHERE department_id = outer.department_id);
```

وخطوات تنفيذ الاستعلام السابق هي:

- ١- ترشيح أول صف من الاستعلام الرئيسي بيانات الموظف (الاسم الأخير؛ والراتب ورقم القسم-ركرز على رقم القسم)
- ٢- تم استعمال رقم القسم من الصف المرشح من الاستعلام الرئيسي لحساب متوسط رواتب هذا القسم. لاحظ استعمال الاسم البديل outer لجدول الموظفين لتمييز رقم القسم في الاستعلام الجزئي عن رقم القسم الذي نريد حساب متوسط رواتبه
- ٣- بعد حساب متوسط رواتب القسم يتم استعمال هذا المتوسط لتقرير عرض الصف أم لا وذلك بمقارنة الراتب في الاستعلام الرئيسي salary بالمتوسط؛ فإن حقق الشرط بأنه أكبر من المتوسط يتم استخراج الصف وإلا انتقل البحث إلى الصنوف التالية
- ٤- يتم ترشيح الصف التالي من الاستعلام الرئيسي بتكرار الخطوات السابقة

٢- المثال الثاني نريد عرض بيانات الموظفين الذين غيروا وظيفتهم على الأقل مرتين؛ فنكتب:

```
SELECT e.employee_id, last_name,e.job_id  
FROM employees e  
WHERE 2 <= (SELECT COUNT(*)  
FROM job_history  
WHERE employee_id = e.employee_id);
```

يتم تكرار نفس الخطوات. ادرسها وجرب سواءً في المثال الأول أو الثاني. حاول السير في خطوات الاستعلام السابق.

العاملان يوجد EXISTS ولا يوجد NOT EXISTS 

سبق وأن ناقشتنا العديد من العوامل operators مع الاستعلامات الجزئية. وهنا نضيف العاملين NOT EXISTS و EXISTS والذين يستعملان غالباً مع الاستعلامات المرتبطة corrolated subqueries

١- العامل EXISTS

يستعمل هذا العامل للبحث عن وجود القيمة المرشحة من الاستعلام الرئيسي في مجموعة نتائج الاستعلام الجزئي. فعند أول عشر على هذه القيمة في الاستعلام الجزئي ينتهي البحث ويصير الشرط صحيحاً TRUE وينتقل إلى القيمة المرشحة التالية. أي أننا لا نبحث عن قيمة بل عن وجود القيمة. مثال:

نريد عرض بيانات الموظفين الذين يوجد موظف واحد على الأقل تُفعِّل إليه التقارير؛ وبمعنى آخر نريد معرفة من هم المدراء.

```
SELECT employee_id, last_name, job_id, department_id
```

```

FROM employees outer
WHERE EXISTS (SELECT 'anything'
               FROM employees
              WHERE manager_id = outer.employee_id);

```

لاحظ اننا لا نبحث عن قيمة معينة في الاستعلام الجزئي؛ بل عن وجود رقم مدير مطابق لرقم الموظف المرشح من الاستعلام الرئيسي(الخارجي) والذي أعطيناه الاسم البديل outer؛ لذلك لم نكتب في عبارة SELECT اسم عمود في الاستعلام الجزئي بل كتبنا ثابت حرفيا Letiral هو anything ويمكنك كتابة ما تشاء حسب ما درسنا سابقا في بحث الاصفات الحرفية لأننا لا نبحث عن قيمة بل عن وجود صفات!

وهذه عينة من النتائج

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
108	Greenberg	FI_MGR	100
114	Raphaely	PU_MAN	30
120	Weiss	ST_MAN	50
121	Fripp	ST_MAN	50
122	Kaufling	ST_MAN	50
123	Vollman	ST_MAN	50
124	Mourgos	ST_MAN	50
145	Russell	SA_MAN	80
146	Partners	SA_MAN	80
147	Errazuriz	SA_MAN	80
148	Cambrault	SA_MAN	80
149	Zlotkey	SA_MAN	80
201	Hartstein	MK_MAN	20
205	Higgins	AC_MGR	110

18 rows selected.

٢- العامل لا يوجد NOT EXISTS

والمدف منه عكس المدف من العامل EXISTS أي أننا نبحث عن عدم وجود القيمة المرشحة من الاستعلام الرئيسي. فلو أردنا معرفة الأقسام التي لا يوجد فيها موظفون؛ فيعني ذلك أننا سنرشح رقم القسم من جدول الأقسام ونرى إن كان غير موجود في جدول الموظفين فنظهر بياناته وإلا ينتقل البحث إلى المربع التالي. لاحظ الاستعلام التالي:

```

SELECT department_id, department_name
      FROM departments d
     WHERE NOT EXISTS (SELECT 150
                           FROM employees
                          WHERE department_id = d.department_id);

```

لاحظ استعمال الإضافة الحرفية 150 أو أي شيء آخر لأننا لا نبحث عن قيمة بل نبحث عن عدم وجود القيمة المرشحة من الاستعلام الرئيسي في الاستعلام الجزئي. وهذه عينة من النتائج

DEPARTMENT_ID	DEPARTMENT_NAME
120	Treasury
130	Corporate Tax
140	Control And Credit
150	Shareholder Services
160	Benefits
170	Manufacturing

لنجعل العامل NOT IN بدلًا من NOT EXISTS كما يلي:

```
SELECT department_id, department_name
FROM departments
WHERE department_id NOT IN (SELECT department_id
                             FROM employees);
```

No rows selected.

لا نتائج! رغم وجود صفات تحقق الشرط والسبب في ذلك أن NOT IN تصبح قيمتها خطأ FALSE إذا وجد أي صفات لا يتحقق الشرط في الاستعلام الجزئي وبهذا لا ينتهي أي صفات علماً بأن هناك صفات تتحقق الشرط.
توجد طريقة أخرى لتحقيق الغرض حاول أن تجدها.

التحديث المرتبط

وذلك باستعمال الاستعلام المرتبط لتحديث قيمة عمود في جدول باستخدام قيمة من جدول آخر من الاستعلام الجزئي. والصيغة العامة هي:

```
UPDATE table1 alias1
SET column = (SELECT expression
               FROM table2 alias2
               WHERE alias1.column = alias2.column);
```

لنفرض أن لدينا الجدول emp3 المطابق للجدول employees. نريد إضافة عمود يمثل اسم القسم إلى الجدول emp3. طبعاً هذا الإجراء لا يتبع قواعد القياسية Normalization في تصميم قواعد البيانات ولكن قد نضطر أحياناً إلى مخالفة هذه القواعد لتبسيط عمليات الاستعلام. وهذه المخالفة تسمى عدم القياسية Denormalization. والآن لنطبق الخطوات:

```

ALTER TABLE emp3
ADD (department_name VARCHAR2(25));
UPDATE emp3 e
SET department_name =
(SELECT department_name
FROM departments d
WHERE e.department_id = d.department_id);

```

الحذف المرتبط

وفيه يتم حذف صفوف من جدول بناءً على صفات من جدول آخر بصورة مشابهة للتحديث المرتبط.
والصيغة العامة:

```

DELETE FROM table1 alias1
WHERE column operator
      (SELECT expression
       FROM table2 alias2
       WHERE alias1.column = alias2.column);

```

نريد حذف صفوف من الجدول emp3 السابقة والتي هي متواجدة في الجدول emp_history كما يلي:

```

DELETE FROM empl6 E
WHERE employee_id =
      (SELECT employee_id
       FROM emp_history
       WHERE employee_id = E.employee_id);

```

لتسهيل عمليات الاستعلام المعقدة تم تطوير فكرة جعل كل استعلام في كتلة بيانات Data Block باستخدام عبارة WITH التي تحول من كتل استعلام إلى حدول مؤقت أو عرض VIEW في الذاكرة بحيث يسهل الربط بين تلك الجداول أو العروض. ويمكننا إنشاء أي عدد من كتل البيانات باستخدام WITH؛ فنقوم عبارة WITH ب تخزين الصدفوف الناتجة من الاستعلام في مساحة الجدول المؤقتة Temporary Tablespace وهذا يسرع من أداء عمليات الاستعلام بشكل كبير لأنها تتم في الذاكرة.

لدرس المثال التالي؛ لنفرض أننا نريد عرض أسماء الأقسام ومجموع الرواتب فيها والتي تحقق أن مجموع الرواتب للقسم أعلى من متوسط رواتب الأقسام في الشركة. وهذا يحتاج إلى الحسابات الوسيطة التالية باستخدام عبارة WITH

- ١ - حساب مجموع رواتب كل قسم وتخزينها في المخزن المؤقت
- ٢ - حساب متوسط رواتب جميع الأقسام وتخزينها في المخزن المؤقت
- ٣ - مقارنة مجموع القسم بمتوسط الأقسام وعرض بيانات القسم الذي يتحقق شرط أن مجموع رواتبه أعلى من متوسط جميع الأقسام.

لذلك سنكتب الاستعلام التالي:

```

WITH
dept_costs AS (
    SELECT d.department_name, SUM(e.salary) AS dept_total
    FROM employees e JOIN departments d
    ON e.department_id = d.department_id
    GROUP BY d.department_name),
avg_cost AS (
    SELECT SUM(dept_total)/COUNT(*) AS dept_avg
    FROM dept_costs)
SELECT *
FROM dept_costs
WHERE dept_total >
(SELECT dept_avg FROM avg_cost)

```

ORDER BY department_name;

شرح الاستعلام السابق:

- ١ - بدأنا بعبارة WITH وهذا مهم جدا لأنها هي التي تخبر خادم أوراكل عن نيتنا إنشاء جداول مؤقتة أو كتل بيانات مؤقتة
 - ٢ - أعطينا اسم لأول استعلام(dept_costs) الذي فيه أنشأنا جدولًا مؤقتًا يحمل هذا الاسم ويحتوي على اسم القسم ومجموع الرواتب فيه. لاحظ الربط بين جدول الأقسام departments وجدول الموظفين employees.
 - ٣ - أعطينا اسمًا للستعلام الثاني(avg_cost) والذي أنشأنا فيه حدولًا يحمل نفس الاسم ولكنه يحتوي على عمود واحد فقط هو متوسط الرواتب لجميع الأقسام. لاحظ مصدر الاستعلام وهو الجدول المؤقت من الاستعلام الأول dept_costs
 - ٤ - كتبنا جملة استعلام عاديّة جداً لعرض الأقسام التي تحقق الشرط باستعمال الجداول المؤقتة السابقة. ضمنياً نكون أنشأنا جداول واستخدمناها في جملة الاستعلام العاديّة كأي جدول.
- أسلوب مريح. أليس كذلك؟ وهذه هي نتيجة الاستعلام السابقة

DEPARTMENT_NAME	DEPT_TOTAL
Sales	304500
Shipping	156400

ملاحظات على استعمال عبارة WITH

- ١ - تستعمل فقط مع جملة الاستعلام SELECT فلا تستعمل مع جملة التحديث UPDATE أو جملة الحذف DELETE....
- ٢ - أسماء الجداول المؤقتة (أسماء الاستعلامات) متاحة لجميع عناصر عبارة WITH التي تلي تلك الأسماء؛ بما فيها الاستعلامات الجزئية التي تتبع كل استعلام جزئي.
- ٣ - إذا تطابق اسم الاستعلام في عبارة WITH مع اسم جدول أو عرض في قاعدة البيانات فإن المخلل Parser يعطي الأولوية لاسم الاستعلام على اسم الجدول في قاعدة البيانات فوجب الانتباه إلى تطابق الأسماء
- ٤ - يمكن لعبارة WITH أن تحمل أي عدد من الاستعلامات المفصولة بفواصله فيما بينها.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٥)

الاستعلام المرمي

Hierarchical Retrieval

والمقصود بالاستعلام المرمي استرجاع البيانات على شكل هرم أو شجرة مقلوبة؛ جذرها root في القمة وأغصانها branches متفرعة عن هذا الجذر وآخر عنصر في الشجرة هذه هي الأوراق leaves. فمثلاً في جدول الموظفين يوجد رئيس الشركة؛ وتحت إدارته عدد من المدراء؛ وهؤلاء بدورهم ممكن أن يكونون تحت إدارتهم مدراء آخرين حتى يصل إلى الموظفين. فالرئيس هو جذر الشجرة والمدراء الذين تحت يده والذين تحت أيديهم من مدراء هم أغصان الشجرة؛ وأخيراً الموظفون الذين لا يوجد تحت إدارتهم موظفون آخرون هم أوراق الشجرة. وقواعد البيانات العائمة لا تخزن البيانات على شكل شجرة بالرغم من وجود علاقة هرمية بينها كما في جداول الموظفين سالف الذكر. لذلك نلجأ إلى أسلوب ما لاسترجاع البيانات على شكل شجرة ويسمى هذا الأسلوب بالمشي خلال الشجرة Tree Walking . والأسلوب المرمي يطبق في كثير من مجالات الحياة؛ كإدارة الشركات؛ وتربية الماشي والدراسات المتعلقة بالسلالات البشرية وفي صناعة تجميع المنتجات مثل تجميع الكمبيوترات وغير ذلك كثير.

الشكل العام للاستعلام المرمي(الشجري) 

SELECT [LEVEL], *column*, *expr*...

FROM *table*

[WHERE *condition(s)*]

[START WITH *condition(s)*]

[CONNECT BY PRIOR *condition(s)*] ;

حيث:

١ - عبارة الاستعلام الأساسية المعروفة SELECT

٢ - LEVEL عمود وهي pseudocolumn يمثل مستوى الصف الناتج من الاستعلام؛ فالجذر يأخذ المستوى الأول أي ١؛ ثم الذي يليه يأخذ المستوى الثاني أي ٢.. وهكذا. وهو اختياري وليس إجباريا

٣ - column,expr القيم التي نريد استخراجها من الاستعلام

٤ - FROM table اسم الجدول الذي نريد استخراج الشجرة منه. جدول واحد فقط غير مرتبط بأي جدول؛ أي لا يذكر اسم جدول آخر أو عرض في عبارة FROM. يمكن استخدام عرض VIEW بدلاً من الجدول بشرط أن يكون بسيطاً؛ (راجع إنشاء العروض)

٥ - WHERE وهي العبارة التي نستعملها لتقيد عدد الصفوف الناتجة من الاستعلام

٦ - WHERE شرط أو شروط الاستعلام في عبارة Condition(s)

٧ - START WITH condition(s) وهذه العبارة تحدد لنا نقطة البداية للشجرة أي تحدد لنا الجذر root الذي نبدأ منه إنشاء الشجرة. وهذه العبارة ضرورية للاستعلام المترافق الشجري (الشجري)

٨ - CONNECT BY PRIOR TO condition(s) لتحديد العمود الذي تكون العلاقة بين الأب والإبن قبله. وهذه ضرورية في الاستعلام المترافق الشجري

ملاحظة: يجب أن لا تحتوي جملة الاستعلام المترافق الشجري على أكثر من جدول سواء بالربط بينها JOIN أو من خلال استعمال عرض مركب COMPLEX.

أيضاً عبارتا START WITH و CONNECT BY PRIOR TO ليستا ضمن سلسلة القياسية ANSDI SQL

المشي في الشجرة WALKING the tree

للمشي في الشجرة تحتاج نقطة البداية starting point التي هي جذر الشجرة root. ويتم تحديدها بواسطة شروط معينة. وهذه الشروط أية شروط سبق وأن تحدثنا عنها بما فيها الاستعلامات الجزئية. ولهذا الغرض نستعمل عبارة START WITH ؛ والصيغة العامة لهذه العبارة هي:

START WITH *column* = *value*

حيث أن القيمة *value* هي نتيجة الشرط الذي نضعه ليتطابق معه العمود *column*

ولنأخذ جدول الموظفين على سبيل المثال ونختار نقطة البداية للسير فيه؛ مثلاً:

START WITH *last_name* = 'Kochhar'

هنا اختبرنا نقطو البداية باستعمال العمود last_name بحيث تكون قيمته Kochhar فيكون جذر الشجرة هو .Kochhar القيمة

ومثال آخر؛ نريد أن تكون نقطة البداية هي قيمة العمود manager_id اللاشيء؛ فنكتب:

START WITH manager_id IS NULL

أي أنها سنبدأ بأعلى شخص في الشركة. ويمكن أن يحتوي الشرط على استعلامات جزئية؛ فمثلاً نريد أن تكون نقطة البداية هي قيمة العمود employee_id للموظف Kochhar فنكتب:

```
START WITH employee_id = (SELECT employee_id  
                           FROM employees  
                          WHERE last_name = 'Kochhar')
```

وهكذا نختار نقطة البداية للمشي في الشجرة.

العثور على الأبناء التابعين لأب. المشي في الشجرة قد يكون من الأب إلى الابن أو بالعكس. وفي كل الأحوال نحتاج معرفة العلاقة التي تربط الأب بالأبناء. لهذا الغرض نستعمل عبارة CONNECT BY PRIOR حيث CONNECT BY PRIOR column1 = column2 condition(s) على الشكل يمثل الأب و column2 يمثل الإبن. فمثلاً لإيجاد الموظفين الذين تحت إدارة موظف ما (مديرهم) نكتب CONNECT BY PRIOR employee_id = manager_id وبشكل عام:

- الاتجاه من أعلى (من الجذر) إلى أسفل (الأغصان والأوراق) فإن:

Column1 = Parent Key

Column2 = Child Key

- الاتجاه من أسفل إلى أعلى فإن:

Column1 = Child Key

Column2 = Parent Key

مثال: للسير في الشجرة من أعلى (من الجذر) إلى الأسفل نكتب:
CONNECT BY PRIOR employee_id = manager_id

فتكون العلاقة أن نبدأ برقم المدير الذي هو رقم موظف ثم نكمل بارقام الموظفين الذين تحت إدارته

مثال للسير من أسفل (الأغصان والأبناء) إلى أعلى (الجذر) نكتب:

CONNECT BY employee_id = PRIOR manager_id

هنا يعني نبدأ بالأبناء (الموظفين) ثم المدراء.

ملاحظة: لا يمكن استعمال الاستعلام الجزئي مع عبارة CONNECT BY PRIOR

مثال للمشي من أسفل إلى أعلى؛ أي نريد عرض الموظفين (الأبناء) أولاً ثم مدراهم:

SELECT employee_id, last_name, job_id, manager_id

```

FROM employees
START WITH employee_id = 101
CONNECT BY PRIOR manager_id = employee_id ;

```

وهذه هي النتيجة:

EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
101	Kochhar	AD_VP	100
100	King	AD_PRES	

لاحظ أنه تم عرض الموظف أولا ثم المدير تاليا. لاحظ الشرط في عبارة WHERE . وأيضا يمكننا إضافة شروط أخرى في عبارة CONNECT BY PRIOR مثل:

```
CONNECT BY PRIOR employee_id = manager_id
```

```
AND salary > 15000;
```

والآن لنمش في الشجرة من الأعلى(الجذر) إلى أسفل لمعرفة المدير الذي يرفع له الموظف تقاريره:

```

SELECT last_name || ' reports to ' ||
PRIOR last_name "Walk Top Down"
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR employee_id = manager_id ;

```

وهذه عينة من النتائج:

Walk Top Down
King reports to
King reports to
Kochhar reports to King
Greenberg reports to Kochhar
Faviet reports to Greenberg
Chen reports to Greenberg

ادرس المثال جيدا وطبقه بشروط مختلفة.

تصنيف الصفوف حسب المستوى باستعمال العمود الوهمي LEVEL

Psuedo column Ranking rows باستعمال العمود الوهمي LEVEL

لجعل تقريرنا الشجري أوضح يمكننا استعمال العمود الوهمي LEVEL لتشكيل الشجرة بصورة أوضح. حيث يأخذ الجذر رقم 1 والابن رقم 2 وابن الابن رقم 3 ... وهكذا. والعقد forks التي تتفرع منها الأغصان تسمى تقاطعات أو nodes والترجمة العربية واحدة أي العقد. وأخر شيء في العقدة يسمى الورقة leaf. وقد يكون الابن child يتفرع منه ابن كمستوى أخير الذي يسمى الورقة.

لنعرض شجرة الإدارة في كل الشركة باستعمال العمود الوهمي LEVEL

COLUMN org_chart FORMAT A20

```
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2,'*')
```

```
AS org_chart
```

```
FROM employees
```

```
START WITH last_name='King'
```

```
CONNECT BY PRIOR employee_id=manager_id
```

شرح المثال:

أول أمر A12 COLUMN org_chart FORMAT A12 هو عنوان التقرير كما سبقت الإشارة إليه

```
LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2,'*')
```

نريد أن نضيف الرمز * إلى شمال الاسم الأخير باستعمال ال LPAD التي سبق بحثها؛ بحيث يكون الطول الكلي للنتائج هو طول الاسم الأخير + عدد من الرمز *. عدد مرات تكرار الرمز يحدده المستوى حسب الصيغة وهي المستوى مضروبا ب 2 ومطروحا منه 2 أي $2 - 2^{*2}$. فلو حسبنا عدد تكرار الرمز * الذي يجب إلحاقه بأول سجل أي المدير King الذي هو رئيس الشركة؛ فرقم المستوى الذي يحمله هو 1؛ وعليه فإن $2 - 2^1 = 0$ يساوي صفرًا؛ أي لا تحتاج إضافة أي عدد من الرموز إلى الاسم الأخير King. بينما المستوى الثاني رقمه 2؛ لذلك نحسب عدد * الذي يجب إضافته له وهو $2 - 2^2 = -2$. أي نضيف **. وهكذا باقي المستويات. النتائج كما يلي:

ORG_CHART

King

King

**Kochhar

****Greenberg

*****Faviet

*****Chen

*****Sciarra

*****Urman

*****Popp

****Whalen

****Mavris

****Baer

****Higgins

*****Gietz

**De Haan

****Hunold

*****Ernst

*****Austin

*****Pataballa

*****Lorentz

**Raphaely

****Khoo

****Baida

****Tobias

****Himuro

****Colmenares

**Weiss

****Nayer

****Mikkilineni

****Landry

****Markle

****Taylor

****Fleur

****Sullivan

****Geoni

**Fripp

****Bissot

****Atkinson

****Marlow

****Olson

****Sarchand

****Bull

****Dellinger

****Cabrio

**Kaufling

****Mallin

****Rogers

****Gee

****Philtanker

****Chung

****Dilly

****Gates

****Perkins

**Vollman

****Ladwig

****Stiles

****Seo

****Patel

****Bell

****Everett

****McCain

****Jones

**Mourgos

****Rajs

****Davies

****Matos

****Vargas

****Walsh

****Feeney

****OConnell

****Grant

**Russell

****Tucker

****Bernstein

****Hall

****Olsen

****Cambrault

****Tuvault

**Partners

****King

****Sully

****McEwen

****Smith

****Doran

****Sewall

**Errazuriz

****Vishney

****Greene

****Marvins

****Lee

****Ande

****Banda

**Cambrault

****Ozer

****Bloom

****Fox

****Smith

****Bates

****Kumar

**Zlotkey

****Abel

****Hutton

****Taylor

****Livingston

****Grant

****Johnson

**Hartstein

****Fay

108 rows selected

Pruning Branches ترشيد (تقليل) لأغصان

كما في الشجرة العادمة يمكننا ترشيدتها بإزالة ورقة مثلاً وإزالة أغصان أيضاً. وفي الاستعلام الممرمي يمكننا استثناء عقدة من الشجرة ولكن مع المحافظة على على الصفوف التابعة لها؛ وبالمقابل يمكننا إلغاء غصن كامل من الشجرة وبالتالي جميع ما تحته من صفوف. لاستثناء عقدة نستعمل عبارة CONNECT BY ولاستثناء غصن كامل نستعمل عبارة WHERE. مثال نريد استثناء العقدة Higgins (الموظف-المدير) فقط مع المحافظة على الصفوف التابعة له نكتب:

WHERE last_name != 'Higgins'

CONNECT BY PRIOR

employee_id = manager_id

AND last_name != 'Higgins'

مثال على التخلص من العقدة 'Higgins' فقط:

SELECT department_id, employee_id, last_name, job_id, salary

FROM employees

WHERE last_name != 'Higgins'

START WITH manager_id IS NULL

CONNECT BY PRIOR employee_id = manager_id;

للتخلص من الغصن كاملاً:

SELECT department_id, employee_id, last_name, job_id, salary

FROM employees

```
START WITH manager_id IS NULL  
CONNECT BY PRIOR employee_id = manager_id  
AND last_name != 'Higgins';
```

وهكذا نكون قد ألقينا نظرة عميقه على الاستعلام الشجري وأهميته وهو جدير بالدراسة.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٦)

دعم التعبيرات القياسية

Regular Expressions Support

مقدمة

تقدم قاعدة البيانات أوراكل ١٠ جي فصاعداً دعماً للتعبيرات القياسية Regular Expressions وهذا يتوافق مع معايير نظام التشغيل المحمول يونيكس (POSIX) Portable Operating System؛ والذي يتحكم فيه معهد مهندسي الكهرباء والإلكترونيات (IEEE)، لطابقة بيانات الآسكنـي ASCII من ناحية التركيب syntax والدلـلات semantics وتعمل إمكانـيات أوراـكل متعدـدة اللغـات على توسيـع قدرـات المطـابـقة للعـوامل بما هو أبعـد من معيـار POSIX. والتـعبـيرـات الـقـيـاسـيـة هي طـرـيقـة لـوـصـف كـل مـن الـأـنـماـط patterns البـسيـطة والمـعـقـدة لـلـبـحـث والمـعـالـجـة.

تساهم معالجة السلسلـات strings والـبـحـث في نـسـبة كـبـيرـة من الـعـمـل دـاخـل تـطـيـقـات الـوـبـ. حيث يتـراـوح الاستـخدـام من البـسيـط: اـبـحـث عن كـلـمة "سان فـرانـسيـسـكـو" في نـص مـحدـد؛ إـلـى المـعـقـد لـجـمـيع عـنـاوـين URL من النـص؛ إـلـى الأـكـثـر تـعـقـيـداً: مـثـل اـبـحـث عن كـلـ الكلـمـات التي يـكـون كـلـ حـرـف ثـانـٍ فـيـها حـرـكة تـشكـيل vowel.

في سـكـول SQL الأـصـلـية، فإن استـخدـام التـعبـيرـات العـادـية يـسـمح بـإـجـراء عـمـلـات بـحـث وـمـعـالـجـة قـوـيـة جـداً عـلـى أـيـة بـيـانـات مـخـزـنـة في قـاعـدة بـيـانـات أـورـاـكل. يمكنـك استـخدـام هـذـه المـيـزة لـحـلـ المشـكـلـات التي قد تكون مـعـقـدة لـلـغاـية.

رموز الوصفية

الرمـوز الوـصـفـية هي رـمـوز خـاصـة لها معـنـى خـاصـ، مـثـل رـمـز التـعمـيم wildcards أو رـمـز التـكرـار أو رـمـز عدم التـطـابـق أو رـمـز ليـدـلـ على نـطـاقـ من الأـحـرـفـ؛ وهـكـذا.... يمكنـك استـخدـام العـدـيد من رـمـوز الأـحـرـفـ الوـصـفـية المـخـدـدة مـسـبـقاً في مـطـابـقة النـمـطـ. وفي الجـدولـ التـالـي شـرـحـ لـهـذـه الرـمـوزـ.

الرمز	الشرح	Description
*	للبحث عن تطابق أكثر من التكرار	
	عامل الاختيار بين البديلـات لـتـحـديـدـ التـطـابـقـاتـ الـبـدـيلـةـ	
^/\$	ـلـتـحـديـدـ بـدـاـيـةـ سـطـرـ وـ\$ـ لـتـحـديـدـ نـهاـيـةـ سـطـرـ	

لـطـابـقـة أي تـعـبـيرـات دـاـخـلـ القـوسـين المـرـبعـين	[]
لـطـابـقـة m مـرـة بـالـضـبـط	{m}
لـطـابـقـة لـعـدـد m مـرـة عـلـى الـأـقـلـ وـلـكـنـ لاـ يـزـيدـ عـن n مـرـة	{m,n}
تـحـدـيدـ أيـ جـمـوـعـةـ رـمـوزـ بـحـيـثـ يـتـمـ التـطـابـقـ مـعـ أيـ رـمـزـ مـنـ هـذـهـ الـجـمـوـعـةـ	[: :]
لـهـ أـرـبـعـةـ اـسـتـعـمـالـاتـ مـخـتـلـفـةـ: ١ـ يـعـبـرـ عـنـ نـفـسـهـ. ٢ـ كـحـاـصـرـةـ لـلـرـمـزـ التـالـيـ لـهـ. ٣ـ تـقـدـسـ عـامـلـ آـخـرـ. ٤ـ لـاـ تـعـمـلـ شـيـئـاـ	\
لـطـابـقـةـ لـمـرـةـ وـاحـدـةـ أـوـ أـكـثـرـ	+
لـطـابـقـةـ صـفـرـ مـرـةـ أـوـ أـكـثـرـ	?
لـطـابـقـةـ أيـ رـمـزـ فـيـ جـمـوـعـةـ رـمـوزـ مـاـ عـدـاـ الـلـاشـيـءـ NULL	.
معـاـلـجـةـ عـدـدـ تـعـبـيرـاتـ كـتـعـبـيرـ وـاحـدـ	()
لـطـابـقـةـ التـصـنـيـفـاتـ الـمـتـكـافـئـةـ لـلـرـمـوزـ	[==]
لـطـابـقـةـ لـلـمـرـةـ nـ قـبـلـ التـعـبـيرـ الجـزـئـيـ حـيـثـ nـ عـدـدـ صـحـيـحـ بـيـنـ ١ـ وـ ٩ـ	\n
تـحـدـيدـ عـنـصـرـ جـمـعـ مـثـلـ العـنـصـرـ الـمـكـونـ مـنـ عـدـدـ رـمـوزـ	[..]

أمثلة:

- نـيـدـ الـبـحـثـ عـنـ السـلـسـلـةـ 'abc'ـ فـيـ سـلـسـلـةـ أـخـرـىـ مـنـ الرـمـوزـ

فـيـ السـلـسـلـةـ abcـ يـوـجـدـ تـطـابـقـ

فـيـ السـلـسـلـةـ 'def'ـ لـاـ يـوـجـدـ تـطـابـقـ

- نـيـدـ الـبـحـثـ عـنـ الرـمـزـ 'a'ـ مـتـبـوعـاـ بـأـيـ رـمـزـ ثـمـ مـتـبـوعـاـ بـالـرـمـزـ 'c'

نستعمل لذلك الرمز الوصفي النقطة '.'

الحل: نريد إذن البحث عن '**a.c'**

في السلسلة **abc** يوجد تطابق

في السلسلة **adc** يوجد تطابق

في السلسلة **alc** يوجد تطابق

في السلسلة **a&c** يوجد تطابق

في السلسلة **abb** لا يوجد تطابق لاختفاء الرمز **c**

- نريد البحث عن تواجد الرمز '**a**' مرة أو أكثر

نستعمل لهذا الغرض الرمز الوصفي علامة الجمع '+' لتحقيق المطلوب

الحل: '**a+**'

في السلسلة **a** يوجد تطابق لمرة واحدة

في السلسلة **aa** يوجد تطابق لمرتين

في السلسلة **bbb** لا يوجد تطابق

- نريد أن نبحث عن أي رموز غير '**abc**' فنستعمل الرمز الوصفي '^' كما يلي:

التعبير المطلوب: **[^abc]**

في السلسلة **abcdef** يوجد تطابق لوجود رموز غير **abc**

في السلسلة **ghi** يوجد تطابق لوجود رموز غير **abc**

في السلسلة **abc** لا يوجد تطابق لعدم وجود رموز غير **abc**

- للبحث عن أية رموز ليست بين **a** و **i** نستعمل التعبير **[^a-i]** حيث أن الرمز '^' يستعمل لنفي التطابق ورمز عملية الطرح - تشير إلى المدى بين الحرفين **a** و **i** ونضع الكل بين قوسين مربعين

الحل: **[^a-i]**

في السلسلة **hijk** يوجد تطابق لوجود الحرفين **jk** وهمما لبسا ضمن المدى

في السلسلة **lmn** يوجد تطابق لأنها ليست ضمن المدى

في السلسلة **abcdefghijklm** لا يوجد تطابق لأن عناصر هذه السلسلة ضمن المدى

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٧)

دعم التعبيرات القياسية

Regular Expressions Support

دوال التعبيرات القياسية في سكول Regular Expressions Functions in SQL

قامت أوراكل بتوفير عدة دوال للبحث في قاعدة البيانات لأي نوع من أنواع البيانات مثل char varchar2, nchar, nvarchar2, clob, nclob اعتباراً من نسخة أوراكل 10 جي. وفي الجدول التالي ملخص هذه الدوال ومن ثم بعض الأمثلة.

اسم الدوال	عملها
REGEXP_LIKE	مشابه للعامل LIKE الذي سبق وناقشناه في دروس سابقة. يرجى الرجوع إليه. ولكن تمت إضافة التعبيرات القياسية بدلاً من النمط البسيط
REGEXP_REPLACE	للبحث والاستبدال باستعمال التعبيرات القياسية مشابه لـ REPLACE المار ذكرها
REGEXP_INSTR	للبحث عن موقع نص معين داخل نص باستعمال التعبيرات القياسية مشابه لـ INSTR المار ذكرها
REGEXP_SUBSTR	للبحث عن نص في نص آخر وإرجاع النص المبحوث عنه المطابق. وهو مشابه لـ SUBSTR المار ذكرها ولكن باستعمال التعبيرات القياسية

شرح الدوال:

١ - الدوال العامة REGEXP_LIKE وصيغتها هي

REGEXP_LIKE (srcstr, pattern [,match_option])

حيث:

- srcstr : النص المصدر؛ المبحث فيه
- pattern النمط (كيفية البحث باستعمال التعبيرات القياسية)
- match_option خيار المطابقة وهو اختياري وله المعاني التالية:
 - ١ - "C" مراعاة حالة الحرف (صغرى أو كبيرة) وهو الخيار الافتراضي
 - ٢ - "I" عدم مراعاة حالة الحرف (البحث عن كل الحروف ضئيل أو كبرى)
 - ٣ - "n" السماح باستعمال رموز أخرى كعوامل
 - ٤ - "m" معاملة النص المصدر كنص من عدة سطور -إذا كان كذلك فإضافة هذا الخيار تصبح ضرورية

مثال ١: البحث البسيط Basic Search

```
SELECT first_name, last_name  
FROM employees  
WHERE REGEXP_LIKE (first_name, '^Ste(v|ph)en$')
```

شرح المثال:

نريد عرض الاسم الأول والأخير لجميع الموظفين الذين أول اسم لهم يحتوي على Steven أو Stephen. لهذا الغرض استعملنا التعبيرات القياسية كما يلي:

- '^Ste(v|ph)en\$' التعبير القياسي

- ^ يشير إلى أن البحث يبدأ من بداية السطر؛ هنا first_name

- \$ يشير إلى نهاية السطر أي first_name

- | يشير إلى أن البحث عما قبله أو عما بعده. أي مطابقة تتحقق المطلوب

- البحث عن Ste متبوعاً ب v أو ph ومتنتها ب en

النتيجة هي:

FIRST_NAME	LAST_NAME
Steven	King
Steven	Markle
Stephen	Stiles

يوجد في أوراكل دالة اسمها SOUNDDEX وهي تعرض النصوص ذات اللفظ المتشابه مثل المثال الوارد وهي ليست مجال بحثنا .

٢ - ال REGEXP_INSTR وصيغتها العامة:

REGEXP_INSTR (srcstr, pattern [, position [, occurrence

[, return_option [, match_option]]])

حيث:

- srcstr النص الأصلي(المصدر) الذي نبحث فيه

- pattern نمط البحث (كيفية البحث باستعمال التعبيرات القياسية)

- position عدد يمثل موقع بداية البحث في النص الصلي. وهو اختياري. والافتراضي من أول النص أي ١

- occurrence عدد مرات البحث. وهو اختياري والافتراضي مرة واحدة

- return_option خيار إرجاع بداية التواجد أو نهاية التواجد وهو اختياري.

- match_option خيار المطابقة وهو اختياري وله المعاني التالية:

١ - "C" مراعاة حالة الحرف (صغير أو كبير) وهو الخيار الافتراضي

٢ - "I" عدم مراعاة حالة الحرف (البحث عن كل الحروف ضعيف أو كبير)

٣ - "n" السماح باستعمال رموز أخرى كعامل

٤ - "m" معاملة النص المصدر كنص من عدة سطور-إذا كان كذلك فإضافة هذا الخيار تصبح ضرورية

مثال: البحث عن تواجد نص؛ نريد في هذا المثال البحث عن موقع أول تواجد لأي رمز غير الحروف سواء كانت كبيرة أو صغيرة في عمود street_address في جدول المواقع locations

SELECT street_address,

REGEXP_INSTR(street_address,'[^[:alpha:]]')

FROM locations

WHERE

REGEXP_INSTR(street_address,'[^[:alpha:]]')> 1;

شرح المثال:

- الأقواس المربعة الخارجية [] تشير إلى بداية ونهاية التعبير القياسي
- ^ رمز النفي NOT
- الأقواس المربعة الداخلية [] تحصر التصنيف(المجموعة) class
- alpha: مجموعة الحروف: مجموعات المجموعات classes
- إذن فالتعبير '^[:alpha:]]' يستثني من البحث أي حرف بغض النظر عن حالته. وهذه عينة من النتائج

STREET_ADDRESS	REGEXP_INSTR(STREET_ADDRESS,'[^[:ALPHA:]])'	
Magdalen Centre, The Oxford Science Park		9
Schwanthalerstr. 7031		16
Rua Frei Caneca 1360		4
Murtenstrasse 921		14
Pieter Breughelstraat 837		7
Mariano Escobedo 9991		8

المحفوظات classes يمكن البحث عنها ومعرفتها بالبحث عن الجروف الكبيرة فقط نكتب .[[upper:]]+.

٣- ال REGEXP_SUBSTR وصيغتها العامة

REGEXP_SUBSTR (srcstr, pattern [, position

[, occurrence [, match_option]]])

شرح الصيغة:

- srcstr النص الأصلي(المصدر) الذي نبحث فيه
- pattern نمط البحث (كيفية البحث باستعمال التعبيرات القياسية)
- position عدد يمثل موقع بداية البحث في النص الصلي. وهو اختياري. والافتراضي من أول النص أي 1
- occurrence عدد مرات البحث. وهو اختياري والافتراضي مرة واحدة
- return_option خيار إرجاع بداية التواجد أو نهاية التواجد وهو اختياري.
- match_option خيار المطابقة وهو اختياري وله المعاني التالية:

١ - "C" مراعاة حالة الحرف (صغير أو كبير) وهو الخيار الافتراضي

٢ - "I" عدم مراعاة حالة الحرف (البحث عن كل الحروف ضغير أو كبير)

٢- "n" السماح باستعمال رموز أخرى كعوامل

٤- "m" معاملة النص المصدر كنص من عدة سطور-إذا كان كذلك فإضافة هذا الخيار تصبح ضرورية

مثال: نريد عرض اسم الشارع فقط من عمود street_address وهو قبل أي فراغ space في اسم الشارع الكامل. وهذا يعني أن يتوقف البحث عند العثور على أول فراغ space. لذلك نستعمل الاستعلام التالي:

```
SELECT REGEXP_SUBSTR(street_address , ' [^ ]+' ) "Road" FROM locations;
```

شرح المثال:

نعرض اسم الشارع الذي يسبق أي فراغ مباشرة؛ فاستعملنا التعبير القياسي ' [^]+' والذي فيه:

- الأقواس المربعة [] بداية ونهاية التعبير

- NOT ^ النفي

- الفراغ space على يمين النفي ^ سنقف عنده

- علامة الجمع + تعني أكثر من تواجد واحد للفراغ

عينة من النتائج في الجدول التالي:

Road
Via
Calle
Jabberwocky
Interiors
Zagora
Charade

٤- التبديل REGEXP_REPLACE وصيغتها العامة:

```
REGEXP_REPLACE(srcstr, pattern [,replacestr [, position
```

```
 [, occurrence [, match_option]]])
```

شرح الصيغة:

- srcstr النص الأصلي(المصدر) الذي نبحث فيه

- pattern نمط البحث (كيفية البحث باستعمال التعبيرات القياسية)

- position عدد يمثل موقع بداية البحث في النص الصلي. وهو اختياري. والافتراضي من أول النص أي ١

- occurrence عدد مرات البحث. وهو اختياري والافتراضي مرة واحدة

- return_option خيار إرجاع بداية التواجد أو نهاية التواجد وهو اختياري.

- replacestr النص الذي نريد وضعه بدلاً من النمط pattern المبحوث عنه

- match_option خيار المطابقة وهو اختياري وله المعاني التالية:

١ - "C" مراعاة حالة الحرف (صغرى أو كبرى) وهو الخيار الافتراضي

٢ - "I" عدم مراعاة حالة الحرف (البحث عن كل الحروف ضعفأ أو كبرى)

٣ - "n" السماح باستعمال رموز أخرى كعوامل

٤ - "m" معاملة النص المصدر كنص من عدة سطور-إذا كان كذلك فإضافة هذا الخيار تصبح ضرورية

مثال: نريد إعادة تشكيل اسم الدولة بفصل حروف الاسم عن بعضها بأي رمز ول يكن الفراغ space بحيث يكون اسم الدولة n J o r d a n مثلا بدلاً من Jordan. لذلك نبحث عن أي رمز غير اللاشيء NULL ونضيف إليه الفراغ Space. نكتب الاستعلام التالي:

```
SELECT REGEXP_REPLACE(country_name, '(.|\')' ) "REGEXP_REPLACE"
```

```
FROM countries;
```

شرح المثال: بحثنا عن أي رمز في اسم الدولة بواسطة التعبير القياسي (.|.'). ثم بدلناه بنفسه زائدا الفراغ بواسطة التعبير القياسي (\1) حيث أن \1 تعني الرجوع للرمز والفراغ بعد ال 1 يعني أن أوراكل بدللت الرمز الوارد في اسم الدولة بنفس الرمز مضافاً إليه الفراغ. والشكل الناتج كما هو في العينة التالية:

REGEXP_REPLACE(COUNTRY_NAME,'(. \')')
Argentina
Australia
Belgium
Brazil
Canada
Switzerland
China

أسلوب جميل!

يمكننا استعمال التعبيرات القياسية في قيود التدقيق لحل مشاكل هذه القيود عندما تكون معقدة نوعاً ما. لنفرض أن لدينا الجدول المطابق للجدول employees الشهير. نريد وضع قيد على إدخال الإيميل بحيث يتضمن إجبارياً الرمز @. فيمكننا إضافة القيد كما يلي:

```
ALTER TABLE emp8
```

```
ADD CONSTRAINT email_addr
CHECK (REGEXP_LIKE (email,'@')) NOVALIDATE;
```

لاحظ أننا استعملنا التعبير القياسي REGEXP_LIKE لاختبار وجود الرمز @ في الإيميل بحيث يتم رفض الإيميل إذا لم يكن هذا الرمز موجوداً. وعبارة NOVALIDATE تمت إضافتها حتى لا تؤثر على البيانات الموجودة سابقاً. الآن سنضيف صفاً إلى الجدول ولكن الإيميل لا يحتوي على الرمز @ وسوف نرى أن العملية تم رفضها مع عرض رقم خطأ.

```
INSERT INTO emp8 VALUES
```

```
(500,'Christian','Patel', 'ChrisP2creme.com', 1234567890, '12-Jan-2004', 'HR_REP', 2000,
null, 102, 40) ;
```

```
INSERT INTO emp8 VALUES
*
```

ERROR at line 1:
ORA-02290: check constraint (ORA20.EMAIL_ADDR) violated

ORA-02290: check constraint (ORA20.EMAIL_ADDR) violated

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٨)

خادم أوراكل Oracle Server

نظرة عامة Overview

في هذه السلسلة سنلقي نظرة على

١- مكونات قاعدة البيانات أوراكل من حيث وصف عام لعمارية(هيكلية) Architecture خادم أوراكل والمكونات الرئيسية له

٢- ذكر الهيكليات المستخدمة لربط المستخدم بنموذج أو حدث أوراكل Oracle Instance

٣- شرح مراحل معالجة كل من:

- الاستعلامات Queries

- جمل معالجة البيانات DML Statements

- عمليات التثبيت Commits

بشكل عام تكون عمارية قواعد البيانات أوراكل من جزئين أساسين:

١- قواعد البيانات أوراكل Database أو ما يسمى بالتركيب الفيزيائي Physical Structure

٢- نموذج (حدث) أوراكل Oracle Instance أو ما يسمى بهيكلية الذاكرة Memory Structure - التركيب المنطقي

* وتكون قواعد البيانات من ملفات فيزيائية على وسائط التخزين مثل الأقراص الصلبة ويمكننا معرفة أسمائها وأحجامها ولكن

لا يمكن التعامل معها مباشرة؛ وأي محاولة الفتحتها أو تحريرها تؤدي إلى تحطيمها Corruption ومنها على سبيل المثال:

- ملف التحكم Control File الذي يحتوي على آخر تعريفات لقواعد البيانات مثل معلومات الجداول والمستخدمين... الخ

- ملف سجلات الإعادة Redo log File وهو يحتوي على المعلومات اللازمة لعملية الاستعادة Recovery في حالة فشل النظام

- ملفات البيانات Data Files حيث يتم تخزين البيانات التي يدخلها المستخدم

- ملف الوسائط Parameters File الذي يحتوي على وسائط تتحكم في حجم وخصائص نموذج أوراكل Oracle Instance

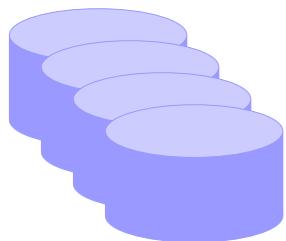
- ملف كلمة المرور Password File الخاصة بالمستخدم السوبر Super User الذي له أعلى الصلاحيات SYSDBA

مدير النظام

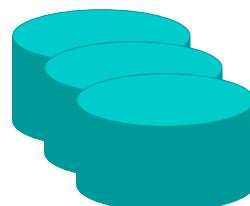
* والنماذج أو الحدث Instance ويكون من مساحة النظام الشاملة System Global Area(SGA) ومجموعة عمليات System Processes والتي تقوم بمهام عديدة في قاعدة البيانات. وهي عبارة عن برمجيات مقيدة تقيم في الذاكرة لدى تشغيل أوراكل Startup وتنتهي بإغلاق أوراكل Shutdown فيتم تحرير الذاكرة.



ملفات التحكم
Control files

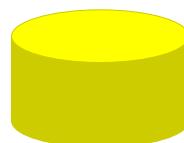
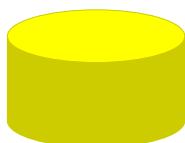


ملفات البيانات
Data files



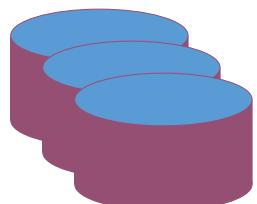
ملفات سجلات الاستعادة
Online redo log files

سجلات الاستعادة



ملف الوسائط
Parameter file

ملف كلمات السر
Password file



ملفات الأرشفة
Archive log files

الأرشفة

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٦٩)

خادم أوراكل Oracle Server

الميكيلية الفيزيائية لقاعدة البيانات

Database Physical Architecture

فيزيائياً تكون قاعدة البيانات من أنواع الملفات التالية:

١ - ملفات التحكم Control Files: تحتوي هذه الملفات على بيانات حول قاعدة البيانات نفسها، وتسمى بالبيانات الوصفية Metad Data.

هذه الملفات مهمة لقاعدة البيانات فبدونها لا يمكن فتح ملفات البيانات

للوصول إلى البيانات الموجودة في قاعدة البيانات؛ وهي البيانات التي يتعامل بها المستخدمون

٢ - ملفات البيانات Data Files: تحتوي هذه الملفات على بيانات الأنظمة والتطبيقات؛ أي البيانات التي يتعامل معها المستخدمون من إضافة وحذف وتعديل وبحث مثل بيانات الموظفين.

٣ - ملفات سجل الإعادة Redo Log Files: تتيح هذه الملفات استعادة آخر تحديات لقاعدة البيانات لم يتم تخزينها في ملفات البيانات أو ملفات التحكم. فعلى سبيل المثال إذا تعطلت قاعدة البيانات ولم تفقد أي ملف من ملفات البيانات Data Files، فسيكون النموذج-الحدث Instance قادرًا على استعادة قاعدة البيانات بالمعلومات الموجودة في هذه الملفات.

وهناك ملفات أخرى ليست جزءاً رسمياً من قاعدة البيانات ولكنها مهمة للتشغيل الناجح لقاعدة البيانات. وهذه هي:

٤ - ملف الوسائل Parameters File: يتم استخدام ملف الوسائل لتحديد كيفية تكوين النموذج-الحدث عند بدء تشغيله.

٥ - ملف كلمة المرور Password File: يتيح هذا الملف للمستخدمين بالاتصال عن بعد Remotely بقاعدة البيانات وتنفيذ المهام الإدارية وبمعنى آخر إدارة قاعدة البيانات البعيدة Remote Database.

٦ - ملفات أرشيف السجل Archive Log Files: تحتوي هذه الملفات على نسخة احتياطية من ملفات سجلات الإعادة Redo Log Files التي تم إنشاؤها بواسطة النموذج-الحدث Instance. تسمح هذه الملفات باستعادة البيانات المفقودة في حال فشل النظام ومن الممكن استعادة ملفات البيانات المفقودة أي التي لم تعد صالحة. وهذه ليست إجبارية ويتم إنشاؤها إذا اشتغلت أوراكل في طور الأرشفة Archive Log Mode.

وفيما يلي وصف عام للمكونات الفيزيائية لخادم أوراكل.

 ملفات التحكم Control Files: كما أسلفنا سابقاً فإن ملفات التحكم هذه تحتوي على البيانات الوصفية لقاعدة البيانات مثل أسماء ملفات البيانات و مواقعها؛ وبدونها لا يمكن لأوراكل أن تشغّل بصورة صحيحة أي لا يمكن المستخدمون من الاتصال بقاعدة البيانات أوراكل وبالتالي لا يمكن العمل فيها.

فبعد بدء تشغيل النموذج-الحدث Database Mount Instance وتحتاج قاعدة البيانات تتم قراءة ملف التحكم؛ و محتوياته تحدّد الجرّيات التي تشكّل قاعدة البيانات.

فمثلاً عند إضافة ملفات إضافية إلى قاعدة البيانات الخاصة، يتم تحديد ملف التحكم تلقائياً.

يتم تحديد موقع ملفات التحكم في ملف الوسائط File Parameters .

وللحماية من فشل تشغيل قاعدة البيانات بسبب فقدان ملف التحكم، فيجب تكرار ملف Multiplexing التحكم على ثلاثة أجهزة مختلفة في الحالة المثالبة أو على ثلاثة أقراص فيزيائية Physical Drives مختلفة في الحالة العاديّة أو على ثلاثة أقراص منطقية Logical Drives مختلفة على الأقل وهو تصرف مقبول إلى حد ما؛ -ولكنه غير آمن خاصّة إذا تلف القرص-؛ حيث يتم تسجيل بيانات موقع الملفات في ملف الوسائط File Parameters .

 ملفات سجل الإعادة Redo Log Files؛ وتكمّن أهميّة هذه الملفات في اصلاح واسترجاع أيّة بيانات في الحركات المؤكدة Committed Transactions وأية تغييرات على قاعدة البيانات مثل الجداول... .

عادةً يتسبّب انحصار النظام في فقد البيانات والتغييرات الأخرى على قاعدة البيانات. وانحصار النظام أو فشله ينبع عن انقطاع الكهرباء مثلاً عن الخادم server أو تحطم الأقراص أو غير ذلك.

ولضمان عدم فقدان التغييرات على قاعدة البيانات أو فقدان الحد الأدنى منها يجب تكرار Multiplexing هذه الملفات أي عمل أكثر من نسخة للملف على أكثر من وسيط للتخزين فيزيائياً أو منطقياً كما في حالة ملفات التحكم؛ كما أنّ عدد الملفات يجب يكون ثلاثة ملفات فأكثر.

هناك مساحة في الذاكرة اسمها المخزن المؤقت لسجل الإعادة Log Buffer يقوم خادم أوراكل بكتابته أي تغيير على قاعدة البيانات تمهدًا لتخزينه على ملفات البيانات

هناك عملية في الخلفية background process تسمى كاتب السجل LGWR تقوم بالكتابة تلقائياً على هذه الملفات خلال أحداث معينة مثل كل ثلث ثوانٍ أو عندما يمتلأ ثلث المخزن المؤقت لسجل الإعادة.... ليس مجال بحثنا هنا.

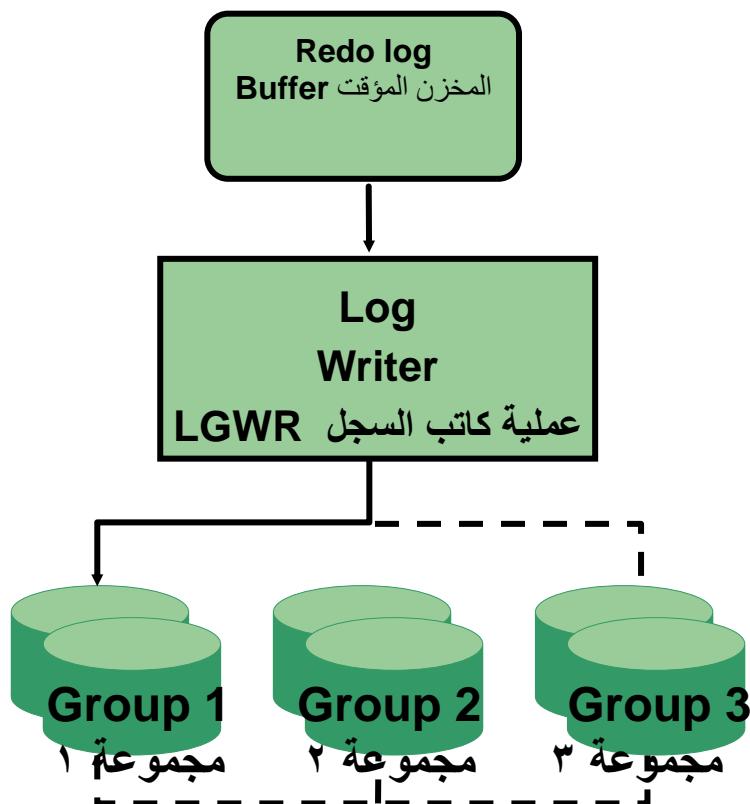
ولزيادة الحرص على التغييرات التي تحصل على قاعدة البيانات قامت أوراكل بعمل طريقة سجل الاستعادة؛ وذلك بإنشاء عدةمجموعات من هذه الملفات تحمل أرقاماً مثل ١ و ٢ ... وكل مجموعة تحتوي على ملفات - ثلاثة ملفات مثلاً - من هذا النوع تسمى أعضاء members. وكل عضو في المجموعة هو نسخة من العضو الآخر من المجموعة الأخرى. وهذا العضو الآخر يكون ضمن المجموعة النشطة(current) (الحالية)

group. يعني لو أن المجموعة رقم 1 هي الحالية وفيها الملفات الأعضاء $\log1$ و $\log2$ و $\log3$ فإن التغييرات على قاعدة البيانات تتم كتابتها على العضو $\log1$ الذي له نسخة في المجموعة 2 و 3 مثلاً. وإذا امتلاء العضو $\log1$ ينتقل خادم أوراكل ليكتب على العضو $\log2$ وهكذا حتى تمتلأ كل الأعضاء؛ فحينئذ يبدأ خادم أوراكل بالكتابة مرة أخرى على العضو $\log1$ بعد تصفيره وهذه العملية هي عملية دائرة circular.

أثناء العمل على المجموعة الحالية قد يتم الانتقال العمل إلى إلى المجموعة التالية أو التي بعدها قبل امتلاء أي عصو؛ وهذه العملية تسمى تحول switching وقد تتم آلياً أو بواسطة مدير قاعدة البيانات.

يجب توزيع ملفات المجموعات على أكثر من جهاز أو أكثر من ثلاثة أقراص صلبة فيزيائياً أو منطقياً بالحد الأدنى.

عملية الاستعادة تتم آلياً بواسطة خادم أوراكل في غالب الأحيان لدى إعادة تشغيل restarting. والشكل التالي يوضح العلاقة بين هذه الملفات والملف المؤقت وعملية كتابة السجل LGWR. تقرأ عملية كتابة السجل مهنن السجل المؤقت Redo Log Buffer وتكتبه في أول ملف من المجموعة الأولى إذا كانت هي المجموعة النشطة ولا زال هناك مساحة كافية في أول عضو من هذه المجموعة.



مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٠)

خادم أوراكل Oracle Server

الميكيلية الفيزيائية لقاعدة البيانات

Database Physical Architecture

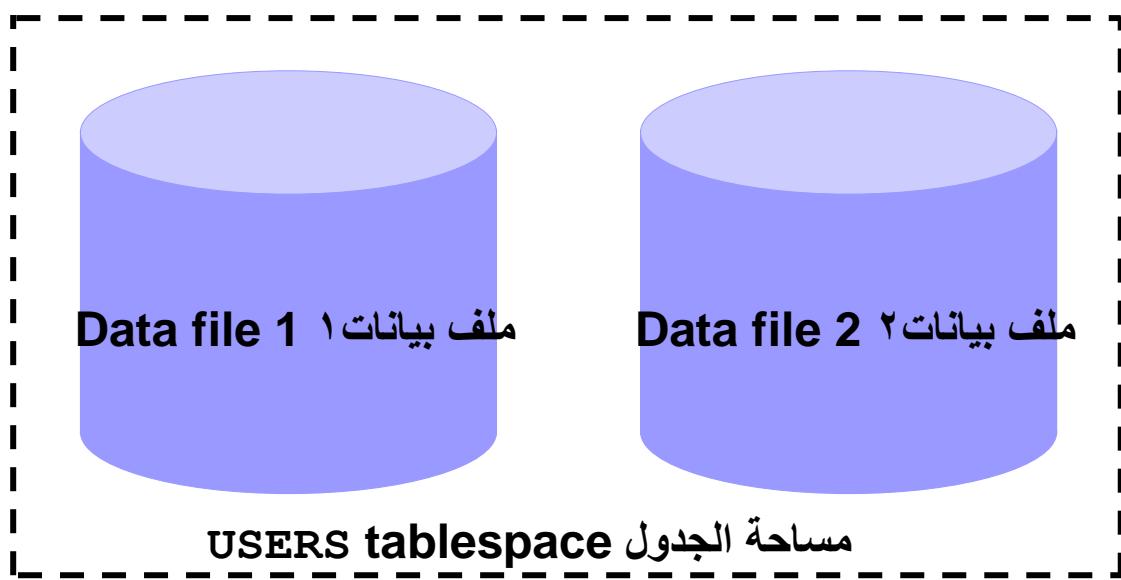
مساحة الجداول وملفات البيانات Tablespace & Data Files

ت تكون قاعدة البيانات أوراكل من وحدات منطقية تسمى مساحة الجدول **tablespace** يرتبط بها فيزيائياً ملفات البيانات والتي يتم تخزين مختلف البيانات فيها. ويتم اتباع النقطتين التاليتين بكل بساطة:

١ - جدول المساحة (يتكون) من ملف بيانات أو أكثر؛ أي يرتبط به ملف فأكثر من ملفات البيانات

٢ - ملف البيانات يتبع إلى جدول مساحة واحد فقط - لا يشترك أكثر من مساحة جدول بملف بيانات واحد

الشكل التالي يبين هذه العلاقة :



على سبيل المثال مساحة الجداول ذات الاسم **USERS** يرتبط بها ملفاً بيانات هما **file1** و **file2** واضح أن هذين الملفين لا يرتبطان بغير مساحة الجدول هذه. فعلاقة مساحة الجدول بالملفات هي واحد متعدد بينما علاقة الملفات بمساحة الجداول هي واحد لواحد.

القطع Segments والامتدادات Extents والكتل Blocks



إن مكونات قاعدة البيانات مثل الجداول والفهارس والعناقيد Clusters يتم تخزينها على شكل قطع Segments في جدول المساحة. والقطعة هذه تتكون من امتدادات Extents وكل امتداد يتكون من كتل أوراكل Oracle وهذه الكتل بدورها تفتقرن بكتل نظام التشغيل OS Blocks.

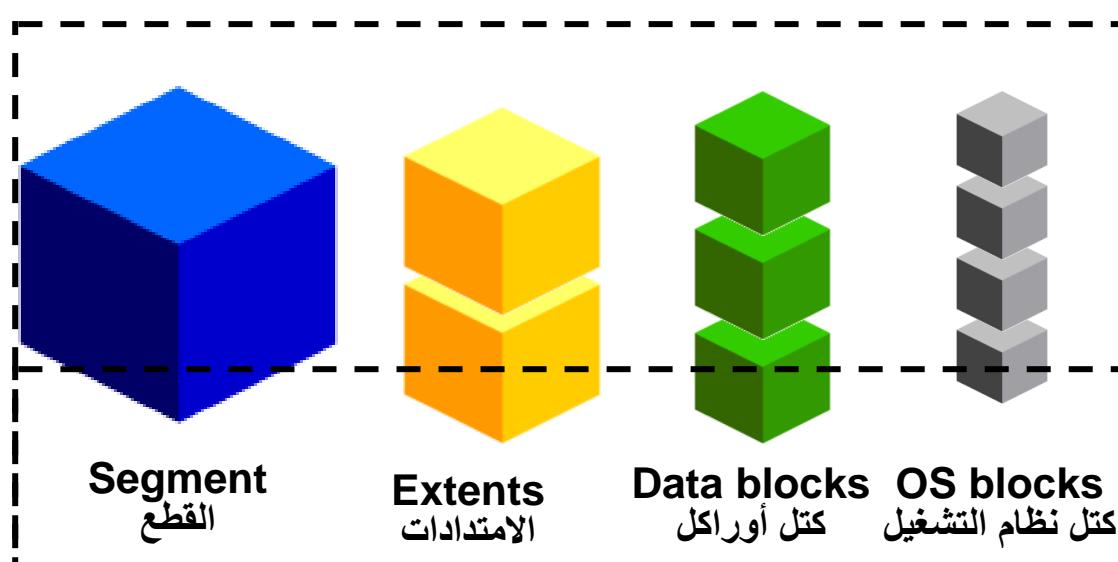
ما يحدث هو أنه عندما يطلب خادم أوراكل كتل بيانات من نظام التشغيل فإن نظام التشغيل OS يستجيب بتخصيص الكمية المطلوب وعمل اقتراض لها مع كتل البيانات التي طلبها خادم أوراكل. مثلاً عن إنشاء جدول مساحة فيجب تحديد اسم وحجم الملف المطلوب؛ فلما تبدأ عمليات الإنشاء يطلب أوراكل من نظام التشغيل عدداً من كتل البيانات المناسبة لكتل البيانات التي تكون الملف. وهذا يوفر علينا عناء التعامل مباشرةً مع نظام التشغيل لأن خادم أوراكل هو الذي يتولى ذلك عنا. هذا فقط تبسيط شديد لما يحدث والهدف منه تقرير الصورة فقط.

حجم كتل البيانات نوعان؛ نوع يحدده نظام التشغيل والأخر يحدده نظام إدارة قواعد البيانات. إن الأحجام الحقيقة للملفات ووسائل التخزين التي نستعملها مبنية على نظام الـ ١٠٢٤؛ فنعرف أن البايت Byte عبارة عن ٨ بت 8 Bits وأل ١٠٢٤ بايت عبارة عن ١ كيلوبايت KB ونظام التشغيل يقسم وسائل التخزين إلى كتل Blocks أدنى حجم لها هو ٢ كيلوبايت. يعتمد حجم الكتلة على نوع وإصدار نظام التشغيل؛ وهذا هو التركيب الفيزيائي للبيانات التي على وسائل التخزين.

أما التركيب المنطقي فيأتي من نظام إدارة قواعد البيانات؛ ففي أوراكل أقل حجم لكتلة البيانات هو نفس حجم كتلة البيانات لنظام التشغيل؛ ويزداد الحجم حسب إصدار قاعدة البيانات. في أوراكل يبدأ حجم الكتلة من حجم كتلة نظام التشغيل ومضارعاتها؛ وفي الإصدارات الحديثة أقل حجم هو ٨ كيلوبايت.

عند إنشاء جدول المساحة في حالة الجداول الضخمة في مستودعات البيانات فإن كبير حجم كتلة البيانات يكون مفيداً فيكون مثلاً ١٦ أو ٣٢ كيلوبايت. بينما في حالة الأنظمة التي يوجد فيها الكثير من الحركات من إضافة وتعديل وحذف فإن صغر حجم كتلة البيانات يكون أفضل بحد أدنى مساواً لكتلة نظام التشغيل.

يمكن إنشاء مساحة الجدول باستعمال أكثر من حجم لكتلة البيانات طبعاً كل حجم يتم تحديده ملف معين ويعني هذا أن هذه المساحة مكونة من عدة ملفات بيانات. والشكل التالي يبين مكونات مساحة الجدول



أوراكل مثلاً (٧١)

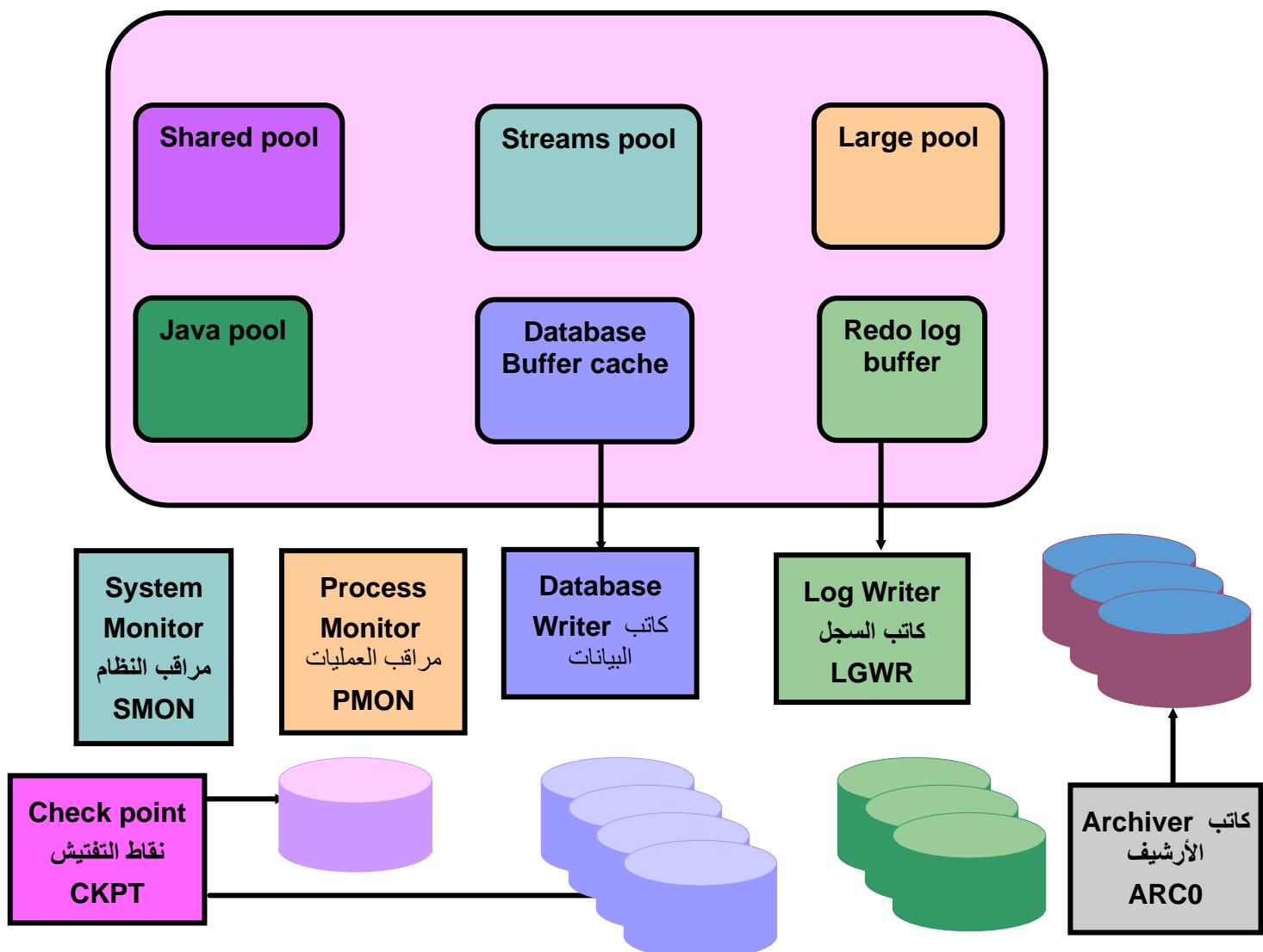
خادم أوراكل Oracle Server

إدارة نموذج – حدث أوراكل

Oracle Instance Management

الشكل التالي يمثل أجزاء نموذج-حدث أوراكل Oracle Instance يليه تعقب على كل جزء منه.

مكونات حدث-نموذج أوراكل Oracle Instance Components



سبق القول إن خادم أوراكل أو نظام إدارة قواعد البيانات أوراكل Oracle Database Server يتكون من قواعد البيانات Database ونموذج – حدث أوراكل Oracle Instance. ويكون نموذج-حدث أوراكل من قسمين هما هيكلية الذاكرة

وهو عبارة عن مجموعة من مخازن الذاكرة المؤقتة Memory Buffers ويطلق عليها اسم مساحة النظام الشاملة System Global Area(SGA) والتي سنشير إليها بالاسم المختصر SGA.

والقسم الثاني هو ما يعرف بعمليات الخلفية Background Processes مثل LGWR سالفة الذكر.

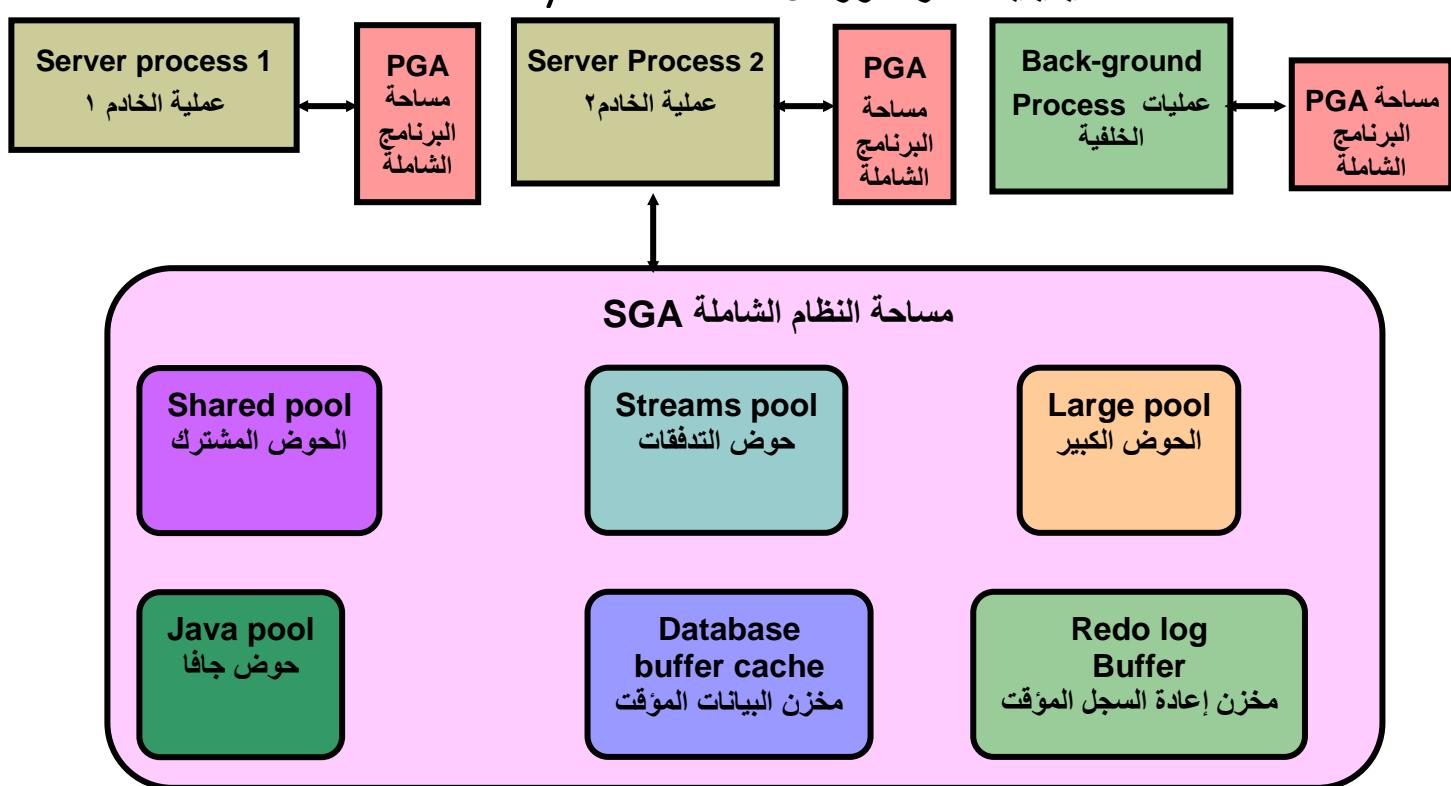
نموذج — حدث أوراكل يكون ساكناً أو غير موجود حتى يتم تشغيله وعندما يتم قراءة ملف الوسائط Parameters File الذي بناءً على مدخلاته يتم تحية نموذج — حدث أوراكل Oracle Instance.

بعد أن يشتعل نموذج — حدث أوراكل وتفتح ملفات البيانات — في الحالة الطبيعية — يستطيع المستخدمون الاتصال بقاعدة البيانات وممارسة أعمالهم عليها.

هيكلية ذاكرة أوراكل Oracle Memory Structure

الشكل التالي يمثل هيكلية ذاكرة أوراكل:

هيكلية ذاكرة أوراكل Oracle Memory Structures



يتكون هيكل ذاكرة أوراكل المرافق لنموذج — حدث أوراكل Oracle Instance من جزأين أساسين:

- مساحة النظام الشاملة System Global Area(SGA) وهي مساحة مشتركة بين جميع عمليات الخادم Server Background Processes وعمليات الخلفية Processes

- مساحة البرنامج الشاملة Program Global Area وبكل Server Process وهي خاصة بكل عملية للخادم One PGA. لكل عملية يوجد مساحة واحدة فقط لخلفية Background Process.

 تحتوي مساحة النظام المشتركة SGA على معلومات السيطرة وبيانات النموذج؛ وتتكون من الأجزاء التالية:

- ١- المخبار المؤقت لقاعدة البيانات Database Buffer Cache ويتم تخزينه أو تخزين البيانات المستخرجة من قاعدة البيانات بواسطة الاستعلام مثلاً.
- ٢- مخزن مؤقت لسجل الإعادة Redo Log Buffer ويستعمل لتخزين معلومات الاستعادة حتى نقلها إلى ملفات الاستعادة على القرص. تستعمل هذه المعلومات لاستعادة النظام.
- ٣- المخزن المشترك Shared Pool تخزن فيه التركيبات Constructs التي يمكن أن يتشارك فيها المستخدمون
- ٤- المخزن الكبير Large Pool: مساحة اختيارية مستخدمة لتخزين طلبات الإدخال / الإخراج الكبيرة مؤقتاً
- ٥- مخزن جافا Java Pool: يستخدم لجمع برمج وبيانات جافا الخاصة بالجلسة آلية جافة الافتراضية Java Virtual Machine (JVM)
- ٦- مخزن التدفقات Stream Pool ويستخدم في تقنية التدفقات من أوراكل وهي تقنية شاملة لإدارة قواعد البيانات مثل تكرار البيانات Replication ومستودعات البيانات Messageing وراسلة Data Wharehouses... الخ
- + عند بدء تشغيل نموذج حدث أوراكل Oracle Instance باستخدام المدير الشامل Enterprise Manager يتم عرض بيانات ذاكرة النظام الشاملة SQL * Plus.

بنية الـ SGA تسمح بتغيير حجم ذاكرة التخزين المؤقت لقاعدة البيانات Database Buffer، والمخزن المشترك Shared Pool، والمخزن الكبير Large Pool، ومخزن التدفقات Streams Pool دون الاضطرار لإعادة تشغيل النموذج Instance.

تم تجهيز قاعدة البيانات مسبقاً بإعدادات مناسبة لوسائل الذاكرة Memory Parameters. ومع ذلك، مع التوسع في استخدام قاعدة البيانات، قد تجد أنه من الضروري تغيير إعدادات وسائل الذاكرة.

وتوفر أوراكل تنبؤات Alerts ونصائح Advises لتحديد مشكلات حجم الذاكرة وللمساعدة في تحديد القيم المناسبة لوسائل الذاكرة

 مساحة البرنامج الشاملة Program Global Area(PGA) وهي منطقة في الذاكرة تحتوي على بيانات ومعلومات التحكم لكل عملية خادم Server Process. وعملية الخادم تخدم طلب العميل Client Request مثل طلب الاستعلام Query. وكل عملية خادم لها منطقة PGA الخاصة بها والتي يتم إنشاؤها عند بدء عملية الخادم. والوصول إليها خاص بعملية الخادم هذه أي أنها غير مشتركة. ولا تتم قراءتها وكتابتها إلا بواسطة برمجيات أوراكل فقط Oracle Code الذي يعمل نيابة عنها. ويعتمد مقدار ذاكرة PGA المستخدمة ومحفوتها على ما إذا كان قد تم تكوين النموذج instance في وضع الخادم المشترك Sahred Server Mode أو الوضع المخصص Dedicated. وبشكل عام تكون الـ PGA من قسمين:

Bind Information Private SQL Area: تحتوي على بيانات؛ مثل معلومات الربط
Memory Structure وهيكلية الذاكرة.

ذكرة الجلسة session memory ولكل جلسة تنفذ جملة سكول تنشأ منطقة سكول SQL Area خاصة بها.
وهذه هو الشكل العامل لنموذج أوراكل Oracle Instance

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٢)

نخدم أوراكل

إدارة نموذج — حدث أوراكل

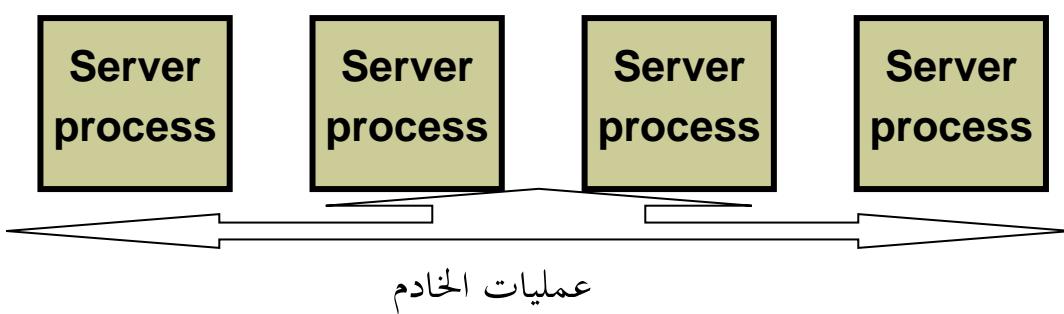
Oracle Instance Management

عمليات أوراكل Oracle Processes

عندما تشتعل أوراكل Started تقوم بإنشاء مجموعة من العمليات الخلفية Background Processes التي تتفاعل مع بعضها البعض ومع نظام التشغيل لإدارة الذاكرة، وإجراء عمليات الإدخال/الإخراج بشكل غير متزامن لكتابه البيانات على القرص، وإجراءات أخرى لإدارة النظام وصيانته.

وعندما يقوم المستخدم بتشغيل أي تطبيق أو أداة من أدوات التطوير مثل سكول+ SQL+ أو نماذج أوراكل Forms فإن خادم أوراكل ينشأ عملية خادم server process لتقوم بتنفيذ الأوامر القادمة من التطبيق أو أداة التطوير. الشكل التالي يوضح عمليات أوراكل الأساسية.

عمليات أوراكل Oracle Processes



System Global Area
SGA
مساحة النظام الشاملة



التي تدعى عمليات أساسية

هي:

Background processes

-١ مراقب النظام (SMON): و تقوم هذ العملية باستعادة النظام (نظام أوراكل) بعد الفشل Failure عندما تتم إعادة تشغيل النموذج Instance بعد الفشل

-٢ مراقب العمليات (PMON): وتقوم بتنظيف آثار العملية التي ينشأها (المستخدم) عندما تفشل؛ مثل تنظيف الذاكرة.

-٣ كاتب قاعدة البيانات (DBWN): وهذه العملية تكتب الكتل المحدثة Modified Blocks من ذاكرة التخزين المؤقت Databas Buffer Cache لقاعدة البيانات؛ إلى الملفات الموجودة على القرص.

-٤ مراقب نقاط التفتيش (CKPT): وهذا يعطي إشار لكاتب قاعدة البيانات ليكتب DBWN عند نقاط معينة تسمى نقاط التفتيش وهي علامات على الحركة transaction تفيد بأن البيانات مؤهلة لكتابتها على الملفات الفизيائية. كما يقوم بتحديث جميع ملفات البيانات وملفات التحكم في قاعدة البيانات لتشير إلى أحدث نقاط تفتيش.

-٥ كاتب السجل (LGWR): تكتب هذه العملية إدخالات إعادة السجل إلى القرص أي إلى ملفات إعادة العمل Redo Log Files

-٦ المؤرشف (ARCN): هذه العملية تقوم بنسخ ملفات سجل الإعادة إلى ملفات الأرشيفية عندما تكون ملفات السجل Redo Log Files متباينة أو يحدث تبديل Switch بين المجموعات التي تتبع إليها. وهذه العملية توجد عند تحيئة أوراكل لتعمل في طور الأرشفة Archive Log Mode. وهذا الطور اختياري؛ والطور الافتراضي ليس الأرشفة Non-Archive Log Mode

يشير حرف ال n في أسماء العمليات إلى أنه ممكن إنشاء أكثر من عملية من نفس النوع حسب حجم العمل لضمان عدم ضياع البيانات إلى الحد الأقصى.

هيكلية فизيائية أخرى؛ توجد ملفات أخرى ليست جزءاً من قاعدة البيانات أي أن نموذج أوراكل ممكن أن يعمل بذوئنا ولكنها قد تستعمل لأغراض أخرى ومثل هذه الملفات:

-١ ملف الوسطاء Parameters File؛ وفيه تعريف خصائص نموذج أوراكل. ويستخدم مع بداية تشغيل نموذج أوراكل Oracle Instance Startup ويحتوي على معلومات مثل أحجام مكونات مساحة النظام الشاملة .SGA

-٢ ملف كلمات السر Password File وفيه يتم تخزين بيانات مدراء قاعدة البيانات DBA's مثل الاسم وكلمات السر والمخولين مثلاً بيء Startup أو إنهاء(إيقاف) Shutdown نموذج أوراكل Oracle Instance

-٣ ملفات أرشيف إعادة العمل Archive Redo Log Files؛ سبقت الإشارة إليها بأنها تحتوي على نسخ احتياطية لملفات إعادة العمل Redo Log Files والتي تفيد جداً في استعادة أقصى قدر من البيانات في حالة فشل النظام مثل تلف بعض أو أحد وسائل تخزين البيانات أو تلف ملف بيانات...الخ.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٣)

معالجة جمل سكول SQL Statements

سيتم بحث النقاط التالية:

١- الاتصال بالنماذج Connect to Instance من خلال:

- عملية المستخدم User Process

- عملية الخادم Server Process

٢- مكونات خادم أوراكل المنخرطة في كل من:

- عملية الاستعلام Queries

- لغة معالجة البيانات DML

- عملية التثبيت COMMIT

٣- بعض مكونات الخادم Server Components التي لا تنخرط مع معالجة جمل سكول SQL.

لا يتم استخدام جميع مكونات نموذج(حدث) أوراكل Oracle Instance لمعالجة جمل سكول SQL. مثلاً يتم استخدام عمليات المستخدم user processes وعمليات الخادم server processes لغرض اتصال المستخدم بنموذج(حدث) أوراكل. وهذه العمليات ليست جزءاً من نموذج(حدث) أوراكل، ولكنها مطلوبة لمعالجة جمل سكول Oracle Instance.

تُستخدم بعض عمليات الخلفية وهي أكل مساحة النظام الشاملة SGA وملفات قاعدة البيانات لمعالجة عبارات SQL. اعتماداً على نوع جمل سكول SQL، يتم استخدام مكونات مختلفة كما يلي:

- تتطلب الاستعلامات معالجة إضافية لإرجاع الصنوف إلى المستخدم.

- تتطلب عبارات لغة معالجة البيانات (DML) معالجة إضافية لتسجيل التغييرات التي تتم على البيانات.

- تضمن معالجة التثبيت commiti إمكانية استرداد البيانات المحدثة من خلال الحركة transaction. يرجى مراجعة مفهوم الحركة في دروس سابقة.

- بعض عمليات الخلفية background processes والتي هي جزء أساسي من نموذج(حدث) أوراكل لا تشارك بشكل مباشر في معالجة جمل سكول SQL ولكنها تستخدم لتحسين الأداء واستعادة قاعدة البيانات.

فمثلاً تُستخدم عملية الأرشفة الخلفية الاختيارية، ARC0، لضمان إمكانية استرداد قاعدة البيانات التي تحدث عليها التغييرات في الحياة .production database

الاتصال بنموذج(حدث) أوراكل Connecting to an Oracle Instance

قبل أن يبدأ المستخدمون من إرسال جمل SQL إلى خادم أوراكل Oracle Server، يجب عليهم الاتصال بالنماذج .Instance

عندما يقوم المستخدم بتشغيل أداة مثل سكول+ SQL*Plus أو يشغل تطبيقاً تم تطويره باستخدام أداة من أدوات التطوير مثل نماذج أوراكل Oracle Forms. يتم تنفيذ هذا التطبيق أو الأداة في عملية المستخدم.

في التكوين الأساسي Basic configuration، عندما يقوم المستخدم بتسجيل الدخول إلى خادم أوراكل، يتم إنشاء عملية process على الكمبيوتر الذي يقوم بتشغيل خادم أوراكل. هذه العملية تسمى عملية الخادم server process. هذه العملية تتواصل مع نموذج أوراكل Oracle Instance نيابة عن عملية المستخدم user process التي يتم تشغيلها على كومبيوتر المستخدم client. فتقوم عملية الخادم بتنفيذ جمل سكول SQL نيابة عن المستخدم.

الاتصال Connection

الاتصال هو طريق للتواصل بين عملية المستخدم وخادم أوراكل. ويمكن لمستخدم قاعدة البيانات الاتصال بخادم أوراكل بإحدى الطرق الثلاث:

١ - يقوم المستخدم بتسجيل الدخول إلى نظام التشغيل الذي يقوم بتشغيل نموذج أوراكل Oracle Instance ويشغل تطبيقاً أو أداة تطوير مثلاً؛ يتصل إلى قاعدة البيانات الموجودة على هذا النظام. يتم إنشاء طريق الاتصال باستخدام آليات الاتصال بين العمليات المتاحة على نظام التشغيل المضيف Host Operating System. وهذه الطريقة تعني أن المستخدم يشغل تطبيقاته أو أدوات التطوير من نفس الكومبيوتر الذي تم تركيب قاعدة البيانات أوراكل فيه.

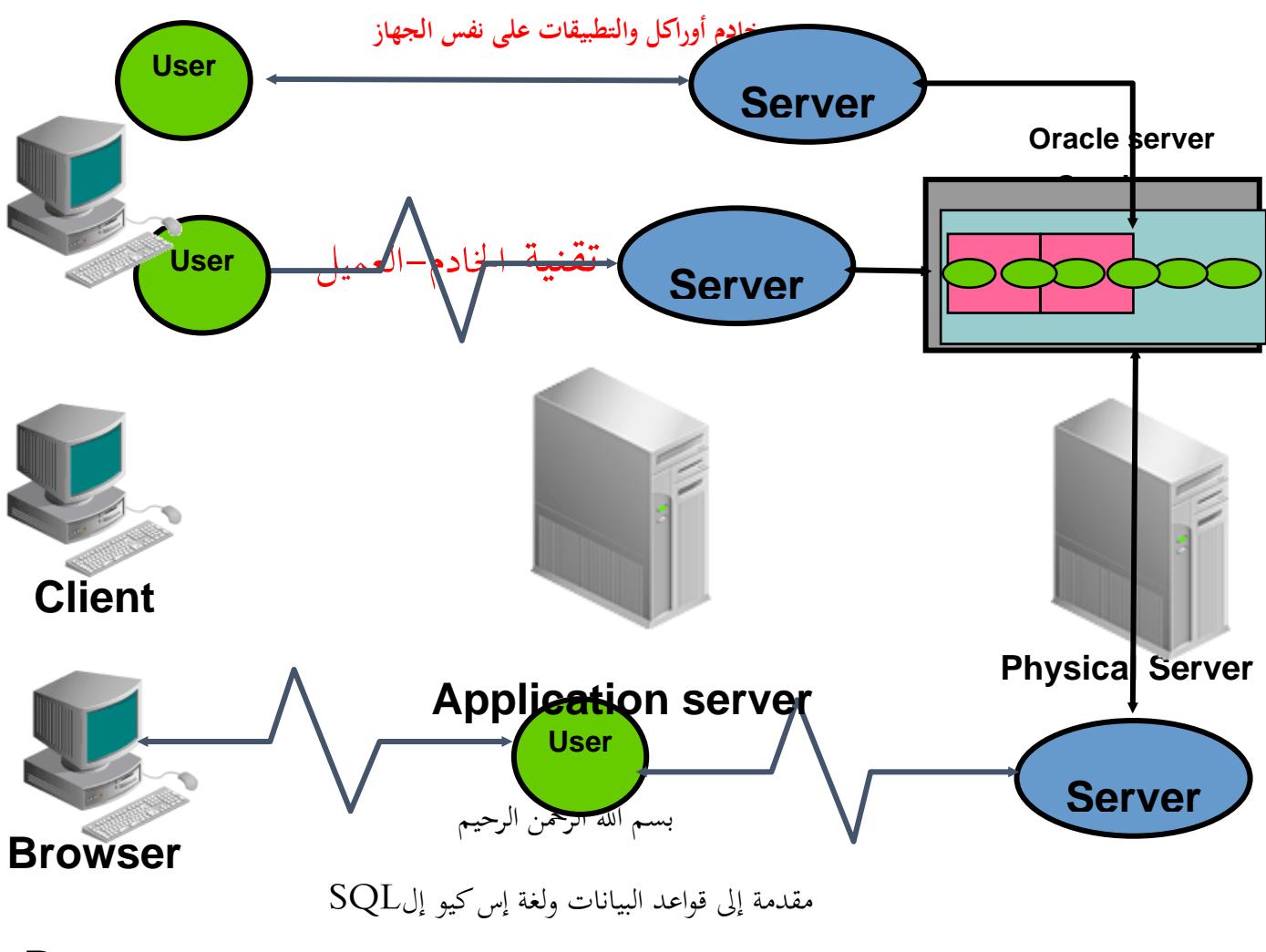
٢ - يشغل المستخدم تطبيقاته أو أدوات التطوير من كومبيوتر مستقل عن الكومبيوتر الذي عليه أوراكل. ولكن الاتصال يتم من خلال الشبكة ويوجد برمجيات تنظم عملية الاتصال بين الجهازين بحيث يتم الاتصال بين عملية المستخدم user process وعملية الخادم server process. وتسمى خدمة التقنية باسم الخادم-العميل Client-Server.

٣ - في هذه الطريقة يتصل جهاز المستخدم بجهاز آخر يسمى خادم التطبيقات Application Server والذي بدوره يتصل بخادم قاعدة البيانات Database Server. فمثلاً يقوم المستخدم بتشغيل المستعرض Browser من جهازه ليتصل بخادم التطبيقات - الذي قد يكون يعمل بنظام التشغيل ويندوز-؛ فيتصل خادم التطبيقات بخادم قاعدة البيانات - الذي يعمل بنظام التشغيل يونيكس أو لينوكس - لتحقيق طلبات المستخدم.

 مفهوم جلسة العمل session. وجلسة العمل تبدأ عندما يتم الاتصال بين المستخدم ونموذج أوراكل وهذا يتم بعد أن تتحقق أوراكل من صحة بيانات المستخدم (مستخدم قاعدة البيانات) وتنتهي عندما يقطع المستخدم جلسة العمل هذه. ويمكن لمستخدم قاعدة بيانات معين، إحداث العديد من جلسات العمل في نفس الوقت إذا قام بتسجيل الدخول من عدة أدوات أو تطبيقات أو محطات الطرفية في نفس الوقت. باستثناء بعض أدوات إدارة قواعد البيانات الخاصة التي تحدد عدد هذه الجلسات أو تجعلها جلسة واحدة فقط.

 ملاحظة: يُسمى نوع الاتصال الموضح هنا الذي يتم الاتصال بصورة فردية بين المستخدم user process وعملية الخادم server process، بالخادم المخصص Dedicated Server. في هذا السيناريو يكون لكل عملية مستخدم واحدة عملية خادم واحدة أي علاقة واحد إلة واحد one to one. وهناك سيناريو آخر تقوم عملية الخادم بخدمة أكثر من عملية مستخدم one to many. وتسمى بالخادم المشترك Shared Server. والشكل التالي يوضح عمليات الاتصال:

Connecting to an Instance



أوراكل مثلاً (٧٤)

معالجة جمل سكول SQL Statements

 معالجة الاستعلام Processing Query تختلف الاستعلامات عن الأنواع الأخرى من جمل SQL لأنها، في حالة نجاحها، تُرجع البيانات كنتائج. في حين أن الجمل الأخرى تنتهي ببساطة بالنجاح أو الفشل، يمكن للاستعلام إرجاع صف واحد أو آلاف الصفوف أو لا شيء حسب شروط الاستعلام. وفيها يتم ما يلي:

- عملية التحليل Parsing وتم الاجراءات التالية فيها:

١- البحث عن جملة مماثلة Identical

٢- التتحقق من الصيغة Syntax وأسماء الكيانات Objects المذكورة وصلاحيات المستخدم عليها

٣- غلق Lock للكيانات المستخدمة في عملية التحليل

٤- إنشاء وتخزين الخطة Create and Store plan

- عملية التنفيذ وفيها يتم تعريف Identify الصفوف المختارة

- الجلب Fetch إرسال الصفوف إلى عملية المستخدم

تبدأ عملية الاستعلام عندما يتم تمرير جملة الاستعلام من عملية المستخدم user process إلى عملية الخادم server process. فتبدأ عملية التحليل ثم يتم تحميل الجملة التي تم تحليلها في مساحة سكول المشتركة Shared SQL Area. تقوم عملية اسيفر بالبحث عن جملة مماثلة (نسخة) في مساحة الحوض المشتركة Shared Pool Area. وإذا لم توجد نسخة مماثلة يتم التتحقق من صحة أو سلامية تركيب الجملة. ثم تبدأ عملية البحث في قاموس أوراكل عن أسماء الجداول والأعمدة والصلاحيات.

فإذا مرت الأمور بسلام تقوم مرحلة الجلب بتنفيذ الجملة باستخدام أفضل خطة يقررها مُحسن أوراكل Oracle Optimizer فيتم إرجاع الصفوف إلى المستخدم.

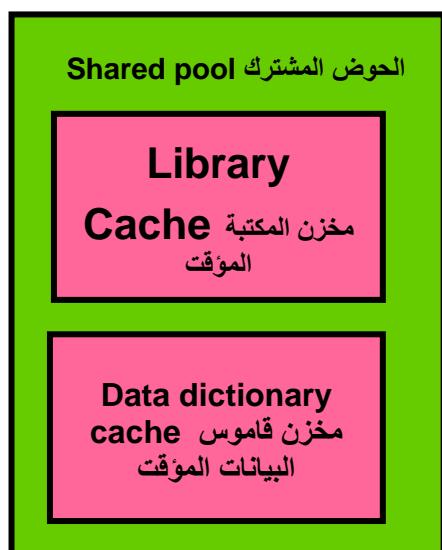
مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٥)

معالجة جمل سكول Processing SQL Statements

مكونات مساحة الحوض المشتركة Shared Pool Components

أثناء مرحلة التحليل parsing، تستخدم عملية الخادم server process هذه المنطقة الموجودة في مساحة النظام الشاملة SGA لترجمة compiling جملة سكول. تحتوي مساحة الحوض المشتركة على مكونين أساسين كما في الشكل التالي:



١ - مكتبة ذاكرة التخزين المؤقت Library Cache. وفي هذه المكتبة يتم تخزين معلومات عن أحدث جملة سكول تم تنفيذها في منطقة تسمى مساحة سكول المشتركة Shared SQL Area والتي تحتوي الأشياء التالية:

أ. نص جملة سكول SQL Text

ب. شجرة التحليل parse tree وهي النسخة المترجمة إلى لغة الآلة Compiled copy لجملة سكول

ت. خطة التنفيذ Execution Plan وهي خطة تنفيذ جملة سكول. وهذه الخطة يحددها محسن أوراكل Oracle Optimizer وهو المسؤول عن اختيار الطريقة المثلثي لتنفيذ الجملة. وإذا تم تنفيذ الجملة مرة أخرى فلا تحتاج عملية الخادم إلى تحليل الجملة؛ وبالتالي فإن مساحة المكتبة المؤقتة Library Cache تحسن أداء الجملة بشكل كبير باختصار وقت التحليل وتقليل كمية الذاكرة المطلوبة؛ وهذا ما يلاحظه المبرمجون عندما ينفذون جملة الاستعلام مثلاً لأول مرة، فتأخذ بعض الوقت وعند تنفيذها مرة أخرى فيلاحظون سرعة التنفيذ. أما إذا لم يتم إعادة استخدام جملة سكول أي تنفيذها مرة أخرى فيتم إخراجها من المكتبة.

ذاكرة التخزين المؤقت لقاموس البيانات، المعروفة أيضًا باسم ذاكرة التخزين المؤقت للقاموس أو ذاكرة التخزين المؤقت للصف، هي مجموعة من أحدث التعريفات المستخدمة في قاعدة البيانات. والتعريفات definitions هذه تعني هيكل البيانات data structure للكائنات؛ فتتضمن معلومات حول ملفات قاعدة البيانات والجداول والفهارس والأعمدة المستخدمين والامتيازات وكائنات قاعدة البيانات الأخرى.

أثناء مرحلة التحليل، تبحث عملية الخادم عن المعلومات الموجودة في ذاكرة التخزين المؤقت للقاموس لتحليل أسماء الكائنات المحددة في جملة SQL وللحصول على امتيازات الوصول. وإذا لزم الأمر، تبدأ عملية الخادم في تحميل هذه المعلومات من ملفات البيانات إن لم يتم التحقق منها.

* تحديد حجم الحوض المشتركة

يتم تحديد حجم الحوض المشتركة بواسطة وسيط التهيئة SHARED_POOL_SIZE الموجود في ملف التهيئة.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٦)

معالجة جمل سكول Processing SQL Statements

ذاكرة التخزين المؤقت لقاعدة البيانات Database Buffer Cache

وظيفة ذاكرة التخزين المؤقت لقاعدة البيانات.

عند معالجة الاستعلام، تبحث عملية الخادم server process في ذاكرة التخزين المؤقت لقاعدة البيانات cache عن أي كتل بيانات datablocks تحتاجها. فإذا لم يتم العثور على الكتلة في ذاكرة التخزين المؤقت لقاعدة البيانات، فإن عملية الخادم تقرأ الكتلة من ملف البيانات وتضع نسخة منها في ذاكرة التخزين المؤقت. ونظراً لأن الطلبات اللاحقة لنفس الكتلة قد تجد الكتلة في الذاكرة، فقد لا تتطلب الطلبات القراءات من القرص الصلب أو أي وسيلة تخزين أخرى. يستخدم خادم أوراكل Oracle Server خوارزمية algorithm الأقل استخداماً مؤخراً Least Used physical reads لتحرير المساحات في المخازن المؤقتة التي لم يتم الوصول إليها مؤخراً لافساح المجال للكتل الجديدة في ذاكرة التخزين المؤقت.

تحديد ذاكرة التخزين المؤقت لقاعدة البيانات Sizing of Database Buffer Cache

يتم تحديد حجم المخزن المؤقت بناءً على حجم كتلة أوراكل Oracle Block والتي سبق مناقشتها، فمبدئياً يكون حجم المخزن المؤقت مساوياً لحجم كتلة أوراكل أو أكبر-أي من مضاعفاته- وقد تكون ضعفها ويتم تحديده بواسطة الوسيط DB_BLOCK_BUFFERS. أما عدد المخازن المؤقتة فهو يساوي قيمة الوسيط DB_BLOCK_SIZE

بسم الله الرحمن الرحيم

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٧)

معالجة جمل سكول Processing SQL Statements

مساحة البرنامج الشاملة Program Global Area(PGA)

خصائص مساحة البرنامج الشاملة PGA

- ١ - ليست مشتركة Not Shared

- ٢ - لا يمكن الكتابة عليها إلا بواسطة عملية الخادم server process

-٣ تحتوي على ما يلي:

أ. مساحة الترتيب sort area

ب. معلومات الجلسة session information

ت. حالة المؤشر cursor area

ث. مساحة التكديس stack area

وفيما يلي نبذة عن هذه المساحة.

فهي مساحة ذاكرة تحتوي على بيانات ومعلومات التحكم لعملية الخادم server process. وهي ذاكرة غير مشتركة يتم إنشاؤها بواسطة خادم أوركل Oracle Server عند بدء عملية الخادم server process. والوصول إليها حصري لعملية الخادم هذه، ولا تتم قراءتها وكتابتها إلا بواسطة برمجيات خادم أوركل Oracle Code الذي يعمل نيابة عنها.

ويشار إلى ذاكرة البرنامج الشاملة PGA المخصصة بواسطة كل عملية خادم متصلة بنموذج(حدث) أوركل Instance بذاكرة PGA المجمعة المخصصة بواسطة النموذج(الحدث). وهذا يعني أن هناك العديد من عمليات الخادم التي يقوم نموذج أوركل بإنشائها وبالتالي العديد من مساحة البرنامج الشاملة.

تتضمن مساحة البرنامج الشاملة PGA للخادم المكونات التالية:

١- منطقة الترتيب sort area: تُستخدم لأية عمليات ترتيب قد تكون مطلوبة لمعالجة جملة سكول SQL

٢- معلومات الجلسة session information: تتضمن امتيازات المستخدم user privilages وإحصائيات الأداء للجلسة session performance

٣- حالة المؤشر cursor state: تشير إلى المرحلة التي وصلت إليها معالجة جمل SQL المستخدمة حالياً بواسطة الجلسة

٤- مساحة التكديس stack area: تحتوي على متغيرات أخرى للجلسة

يتم تخصيص مساحة البرنامج الشاملة PGA عند إنشاء العملية وإلغاء تخصيصها(تحريرها) عند إنتهاء العملية.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٨)

معالجة جمل سكول Processing SQL Statements

معالجة عمليات الإضافة والتعديل والحذف DML

يتطلب تنفيذ لغة معالجة البيانات (DML) مرحلتين فقط من المعالجة:

- ١- مرحلة التحليل وهي نفس مرحلة التحليل المستخدمة لمعالجة الاستعلام؛ فيرجى مراجعتها هناك
- ٢- مرحلة التنفيذ تتطلب بعض المعالجة الإضافية لإجراء التغييرات على البيانات.

 مرحلة التنفيذ Execute phase - جملة التحديث UPDATE :

- إذا لم تكن البيانات وكتل التراجع rollback blocks موجودة بالفعل في ذاكرة التخزين المؤقت buffer، فإن عملية الخادم server process تقرأها من ملفات البيانات data files إلى ذاكرة التخزين المؤقت. والمقصود بالبيانات الجديدة هو البيانات التي يتم جلبها من ملف البيانات؛ وكتل التراجع هي النسخ الحالية من البيانات.
- تقوم عملية الخادم بإغلاق الصفوف التي سيتم تعديلها؛ وهذا لا يمكن تغييرها من قبل مستخدمين آخرين (يرجى مراجعة بحث الحركة transaction)
- في المخزن المؤقت لإعادة السجل redo log buffer، تسجل عملية الخادم التغييرات التي يجب إجراؤها على التراجع والبيانات - أي تسجيل البيانات الحالية والجديدة.
- تقوم كتلة التراجع rollback block بتسجيل قيم البيانات قبل تعديلها؛ ويتم استخدام كتلة التراجع لتخزين الصورة السابقة للبيانات، بحيث يمكن التراجع عن عبارات التغييرات واسترجاع الصورة السابقة إذا لزم الأمر.
- تسجل كتلة البيانات المتغيرة data block changes القيم الجديدة للبيانات.

تسجل عملية الخادم الصورة السابقة في كتلة التراجع وتقوم بتحديث كتلة البيانات. يتم إجراء كل من هذه التغييرات في ذاكرة التخزين المؤقت data buffer cache لقاعدة البيانات. يتم وضع علامة على أي كتل تم تغييرها في ذاكرة التخزين المؤقت على أنها مخازن غير نظيفة dirty buffers؛ أي أن المخازن المؤقتة ليست مماثلة للكتل المقابلة على القرص.

 عمليتا الإضافة INSERT والحذف DELETE تستخدمان خطوات مماثلة. فتحتوي الصورة السابقة لـ DELETE على قيم الأعمدة في الصف المحذف، وتحتوي الصورة السابقة لعملية الإضافة على معلومات موقع الصف.

ونظراً لأن التغييرات التي تم إجراؤها على الكتل يتم تسجيلها فقط في هيكل الذاكرة ولا تتم كتابتها على الفور على القرص، فإن فشل الكمبيوتر الذي يتسبب في فقدان الذاكرة الشاملة SGA والتي هي جزء من ذاكرة الكمبيوتر المؤقتة RAM يمكن أن يفقد هذه التغييرات أيضاً. لكن لا تقلق؛ راجع بحث ملفات إعادة السجل REDO LOG FILES

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٧٩)

معالجة جمل سكول Processing SQL Statements

مخزن إعادة السجل المؤقت Redo Log Buffer

لهذا المخزن الخصائص التالية:

- ١ - تم تحديد حجمه بواسطة الوسيط LOG_BUFFER
- ٢ - يسجل التغييرات التي تم إجراؤها على جميع مكونات قاعدة البيانات الفизيائية والمنطقية.
- ٣ - التسجيل عليه يتم بصورة متتابعة Sequential
- ٤ - يتم استعماله بصورة دائرة Circular.

تقوم عملية الخادم server process بتسجيل معظم التغييرات التي تتم على كتل ملفات البيانات في المخزن المؤقت لإعادة السجل Redo Log Buffer، والذي يعد جزءاً من ذاكرة النظام الشاملة SGA. وهذه الخصائص بصورة مبسطة كما يلي:

- ١ - يتم تحديد حجمها بالبايت بواسطة الوسيط LOG_BUFFER parameter
- ٢ - يسجل الكتلة التي تم تغييرها وموقع التغيير والقيمة الجديدة في مدخلات الإعادة redo entry. لا تميز مدخلات الإعادة بين أنواع الكتل التي تم تغييرها؛ بل تسجل فقط البايتات المتغيرة في الكتلة.
- ٣ - يتم استخدام المخزن المؤقت لإعادة السجل بالترتيب sequentialy، وقد يتم تضمين التغييرات التي يتم إجراؤها بواسطة حركة معينة transaction مع التغييرات التي تم إجراؤها بواسطة حركات أخرى.
- ٤ - وهو عبارة عن مخزن مؤقت دائري يتم إعادة استخدامه بعد ملئه، ولكن فقط بعد تسجيل كافة إدخالات الإعادة القديمة في ملفات سجل الإعادة.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٨٠)

معالجة جمل سكول Processing SQL Statements

قطعة التراجع RollBack Segment

قطع التراجع Rollback Segments

قبل إجراء تغيير على البيانات المخزنة على ملفات البيانات، تقوم عملية الخادم server process بحفظ قيم البيانات القديمة في قطعة التراجع إلى الحالة السابقة. في الشكل التالي آلية عمل قطع التراجع ويليه شرح لها.

قطعة التراجع Rollback Segment



- ١- التراجع عن التغييرات إذا تم الغاء الحركة transaction rolled back
- ٢- تحقيق التنسق في القراءة read consistency عن طريق التأكد من أن الحركات الأخرى-المستخدمين الآخرين- لا ترى التغييرات غير المثبتة uncommitted changes التي تم إجراؤها بواسطة جمل DML
- ٣- استعادة قاعدة البيانات إلى حالة متناسقة consistent status في حالة فشل أو انهيار النظام

وقطع التراجع هذه هي أجزاء من الجداول والفهارس وغيرها من مكونات قاعدة البيانات، وهي تتوارد في ملفات البيانات، أي هي البيانات الأصلية. ويتم إحضار كتل التراجع rollback blocks - أي صورة من البيانات الأصلية - إلى ذاكرة التخزين المؤقت لقاعدة البيانات database buffer cache حسب الطلب. ويتم إنشاء قط التراجع بواسطة مدير قاعدة البيانات DBA. ويتم تسجيل التغييرات على قطع التراجع في المخزن المؤقت لسجل الإعادة redo log buffer.

مقدمة إلى قواعد البيانات ولغة إس كيو إل SQL

أوراكل مثلاً (٨١)

معالجة جمل سكول Processing SQL Statements

معالجة التثبيت COMMIT Processing

التثبيت السريع Fast COMMIT 

يستخدم خادم أوراكل آلية التثبيت السريع التي تضمن المحافظة على التغييرات المثبتة في حالة فشل النموذج Instance وبالتالي ضمان عدم ضياع التغييرات التي يقوم بها المستخدمون إلى الحد الأعلى.

رقم تغير النظام System Change Number(SCN) 

عندما يتم تثبيت حركة transaction committed Oracle server، يقوم خادم أوراكل بتخصيص رقم معين للحركة يسمى رقم تغير النظام SCN. يتم زيادة هذا الرقم بشكل رتيب متسلسل وهو رقم وحيد unique في قاعدة البيانات فلا يتم تخصيصه لحركات أخرى. يتم استخدامه بواسطة خادم أوراكل كطابع زمني داخلي internal time stamp لزمانة البيانات data عند استرداد البيانات من ملفات البيانات read consistency ولتوفير تناسق القراءة synchronization.

يمكن خادم أوراكل oracle server باستخدام رقم تغير النظام SCN من إجراء فحوصات التناسق دون الاعتماد على تاريخ وقت نظام التشغيل. وهذا بدوره يحد من التلاعب بالبيانات عن طريق تغيير تاريخ وقت النظام إلى تاريخ أقدم أو أحدث من تاريخ الحركة.

خطوات معالجة التثبيت steps to process COMMITS 

عند إصدار أمر التثبيت COMMIT، يتم تفريذ الخطوات التالية:

١- تضع عملية الخادم commit record سجل التثبيت server process، مع رقم تغير النظام SCN، في المخزن

المؤقت لسجل الإعادة redo log buffer

٢- يقوم كاتب السجل LOG WRiter LGWR بإجراء كتابة متحاورة contiguous لجميع إدخالات المخزن المؤقت لإعادة تسجيل الإدخالات redo log buffer entries بما في ذلك سجل التثبيت إلى ملفات سجل الإعادة redo log files. بعد هذه النقطة، فإن خادم أوراكل يضمن عدم فقدان التغييرات حتى في حالة فشل النموذج instance.

٣- يتم إرسالة رسالة للمستخدم بأن عملية تثبيت التغييرات قد اكتملت comit completed

٤- تقوم عملية الخادم بتسجيل المعلومات للإشارة إلى أن الحركة قد اكتملت وأنه يمكن تحرير الموارد (الجداؤل...) من عملية الإقفال unlock.

يتم إجراء دفع المخازن المؤقتة المتسخة إلى ملفات البيانات بشكل مستقل بواسطة عملية كاتب قاعدة البيانات DataBase Writer DBW0 والصفر في اسم العملية يعبر عن أول عملية لأنه قد يكون هناك أكثر من عملية كاتب؛ ويمكن أن يحدث هذا إما قبل أو بعد التثبيت.

ميزات التثبيت السريع

تضمن آلية الالتزام السريع استعادة البيانات عن طريق كتابة التغييرات على المخزن المؤقت لإعادة التسجيل بدلاً من ملفات البيانات. ولهذه الآلية الميزات التالية:

١- تكون عمليات الكتابة المتسلسلة في ملفات السجل أسرع من الكتابة إلى كتل مختلفة في ملف البيانات. حيث أن الحركات transaction قد تضمن جمل سكول تعامل مع أكثر من جدول على أكثر من ملف بيانات على القرص فتحتاج وقتاً أطول للتسجيل على مختلف الملفات. بينما الكتابة على ملف إعادة سجل Redo Log File أسرع؛ لأنها تكتب على ملف واحد فقط ولأنها الكتابة متسلسلة وليس مشتتة على أكثر من ملف؛ وتقوم أوراكل تلقائياً بـ تكرار الكتابة على الملفات الاحتياط.

٢- تتم كتابة الحد الأدنى من المعلومات الضرورية فقط لتسجيل التغييرات في ملفات السجل وهذا يقلل الوقت مما يساهم في الحفاظة على البيانات، بينما تتطلب الكتابة إلى ملفات البيانات كتابة كتل كاملة من البيانات والتي طبعاً تأخذ وقتاً أطول ومتيناً.

٣- إذا طلبت عدة حركات transactions عملية التثبيت commit في نفس الوقت، فإن النموذج instance يقوم بـ التجميع بيانات بإعادة السجل في عملية كتابة واحدة وليس متعددة بـ تعدد الحركات.

٤- ما لم يكن المخزن المؤقت لإعادة تسجيل الدخول ممتنعاً، يلزم عملية كتابة متزامنة write synchronized فقط لكل حركة. في حالة حدوث التجميع، يمكن أن يكون هناك أقل من كتابة متزامنة واحدة لكل حركة.

٥- لأنه قد يتم مسح المخزن المؤقت لإعادة التسجيل قبل عملية التثبيت COMMIT transaction، فلا يؤثر حجم الحركة على مقدار الوقت اللازم لعملية التثبيت الفعلية. يرجى مراجعة بحث ملفات إعادة السجل redo log files في موضوع الهيكل الفيزيائي لأوراكل وكاتب إعادة السجل LWGR في موضوع الهيكل المنطقي لأوراكل بـ بند العمليات background processes الخلفية

ملاحظة: لا يؤدي التراجع عن الحركة إلى تشغيل LGWR لـ الكتابة على القرص. وذلك لأن خادم أوراكل Oracle server يلغى دائماً للتغييرات غير المثبتة عند استعادة النموذج instance recovery من حالات الأنياب؛ وهذا يضمن توافقية البيانات. إذا كان هناك أنياب بعد التراجع rollback، قبل تسجيل إدخالات التراجع على القرص، فإن عدم وجود سجل التثبيت commit record (أي عدم وجود علامة على أنه هذه الحركة مثبتة) يكون كافياً لضمان التراجع عن التغييرات التي تم إجراؤها بواسطة الحركة.

وبحذا تمت هذه السلسة من الدروس بحمد الله؛ وكان فيها فكرة عامة عن مفهوم قواعد البيانات. ودروس واسعة عن لغة الاستعلام الميكيلية SQL من خلال قاعدة البيانات أوراكل كمثال؛ وأخيراً فكرة عن نظام إدارة قواعد البيانات أوراكل وهو أصعب موضوع وهو عادة للمبرمجين المتقدمين والذي يميلون إلى العمل كمدرباء لإدارة قواعد البيانات DBAs.