

قواعد البيانات

(2)

إعداد

أ.د عبد المجيد أمين علي

عميد كلية الحاسبات والمعلومات بالمنيا

فهرس الكتاب

الصفحة	الموضوع
3	الفصل الأول: مقدمة في قواعد البيانات
4	• مقدمة
6	• نظم إدارة قواعد البيانات
6	• أهمية تصميم قواعد البيانات
10	• تطوير نظم قواعد البيانات
13	• خصائص قواعد البيانات
28	• مكونات نظم قواعد البيانات
32	الفصل الثاني: أساسيات لغة الاستفسارات الهيكلية
37	• لغة الاستفسارات الهيكلية SQL
40	• استخدام SQL
41	• جملة SELECT
47	• جملة الشرط Where
60	• ترتيب النتائج باستخدام order by
64	• القيمة null وتأثيراتها
65	• استخدام محرر النصوص في SQL
67	• الأسئلة والتمارين
73	الفصل الثالث: تقنيات لغة SQL
76	• الربط بين جدولين أو أكثر
87	• استخدام Multi-row functions
92	• استخدام single-row functions
102	• استخدام group by clause
106	• استخدام having clause
108	• استخدام و عمل إستعلام فرعي sub query
117	• الأسئلة والتمارين
122	الفصل الرابع: معالجة البيانات باستخدام SQL
126	• إضافة سجلات جديدة للجدول
129	• حذف بيانات موجودة في قاعدة البيانات
131	• تعديل بيانات موجودة في قاعدة البيانات
136	• إنشاء الجداول وإضافة المحددات لها
149	• الأسئلة والتمارين
158	الفصل الخامس: ورشة عمل

الفصل الاول

مقدمة في قواعد البيانات

1-1 مقدمة

يحتاج اتخاذ القرار في الشركات والمصالح الحكومية إلى معلومات Information صحيحة وجيدة يتم استنتاجها من الحقائق الأولية Raw Facts والتي يطلق عليها بيانات Data . ويجب إدارة هذه البيانات ومعالجتها بطريقة منظمة حتى يمكن استنتاج المعلومات المطلوبة لاتخاذ القرار السليم والمناسب مع الالتزام بالدقة والسرعة المناسبة ، وأنسب الطرق لحفظ هذه البيانات هو استخدام قواعد البيانات .

وفى هذه الوحدة سنركز علي معرفة الفرق بين البيانات والمعلومات وأهمية قواعد البيانات.

2-1 البيانات والمعلومات

يعتبر التمييز بين البيانات والمعلومات المدخل الرئيسي لتصميم البيانات . وتعرف البيانات Data بأنها عبارة عن حقائق أولية Raw Facts وكلمة أولية تدل علي أن هذه الحقائق لم يتم معالجتها بعد للحصول علي المعلومات المطلوبة .

مثال

افترض أن شركة لها فرعين وان مديرها يرغب في الحصول علي معلومات عن مبيعات كل فرع من خلال مراجعة صور الفواتير التي تم إصدارها عند البيع .

وهذه الفواتير تحتوي بعض الحقائق الأساسية مثل :

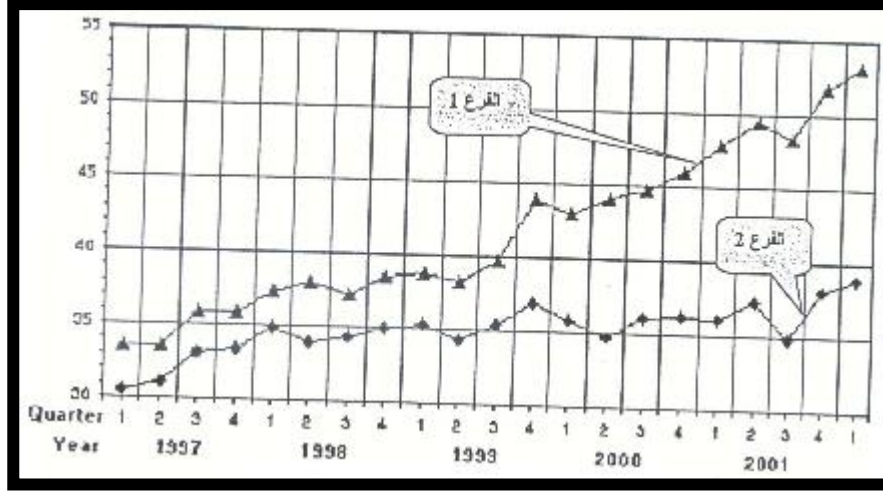
رقم الفاتورة (Invoice Number) = 7234 .

تاريخ الفاتورة (Invoice Date) = 2004/1/12

قيمة المبيعات (Sales amount) = 314.21\$

فإذا فرضنا أن فرع الشركة الأول قد أصدر 1.380.456 فاتورة والفرع الثاني 1.453.907 فاتورة في الفترة ما بين الربع الأول First quarter من العام 1997 والربع الأول من عام 2001 فإننا سنجد أن المدير سيكون أمامه عدد من الفواتير يساوي 1.453.907 + 1.380.456 أي 2.9834.363 وبالتالي فان المدير سيكون أمامه حوالي 3 مليون تاريخ و3 مليون قيمة مبيعات بالإضافة إلي أسماء البائعين الخ ، ويجب عليه دراسة كل منهم قبل اتخاذ أي قرار أو الخروج بخلاصة ما .

ولكن إذا أمكن معالجه هذه البيانات للحصول علي مجموع المبيعات كل ربع عام مثلا (أي كل ثلاثة اشهر) بالنسبة لكل فرع ومن ثم قسمة هذه المبيعات علي عدد البائعين فانه سيصير في الامكان عمل مخطط كما في الشكل (4-1) .



شكل (1-1)

ومن الواضح انه يمكن الخروج بعده حقائق من الشكل السابق منها علي سبيل المثال :

- 1- أن البائعين من الفرع الأول قد أتموا صفقات بيع أكثر من البائعين في الفرع الثاني .
- 2- أن الفرق يأخذ في الزيادة بين الفرعين مع تقدم الزمن .
- 3- أن المدير لديه الآن معلومات نافعة لاتخاذ قرار .

ولهذا من الواضح أن

- 1- البيانات هي أساس تكوين المعلومات .
- 2- المعلومات تستخلص من البيانات .
- 3- المعلومات تستخدم لإظهار معني البيانات .
- 4- المعلومات الجيدة وذات العلاقة هي مفتاح اتخاذ القرارات السليمة .

ومن السهل أن نستنتج أن المعلومات المفيدة تحتاج إلي بيانات دقيقة وهذه البيانات يجب أن يكون أساسها سليم ويجب أن تخزن في صورة يسهل الوصول إليها ومعالجتها ثم توفيرها للمستخدم للاستفادة منها لاحقاً .

وفي الواقع فإننا نجد أن الإدارة الجيدة للبيانات تحتاج إلي قواعد بيانات وقواعد البيانات تشكل في مضمونها نظام إلكتروني لحفظ وإدارة البيانات وهذا النظام يتكون من مجموعة برمجيات

تشكل نظام إدارة قواعد البيانات Data Base Managemant Systems أو (DBMS) .

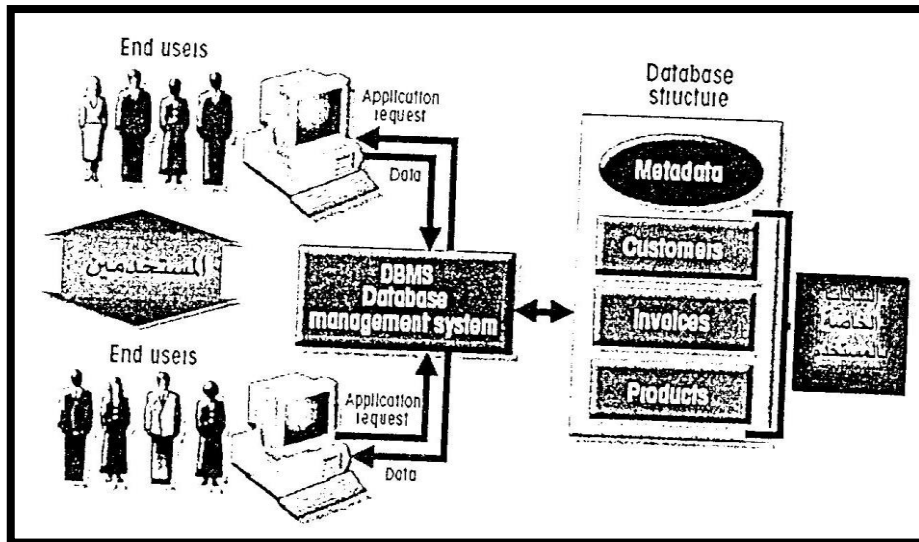
3-1 نظم إدارة قواعد البيانات

يوضح الشكل الآتي أن نظم إدارة قواعد البيانات تقع ما بين مستخدمي تطبيقات نظم الحاسب والتي تتكون من عدد من الجداول تحتوي على البيانات مثل العملاء Customers - الفواتير Invoices والمنتجات Products . وفي الواقع فإن نظم إدارة قواعد البيانات يمكن تصورها علي أنها وسيط بين المستخدم والبيانات حيث تقوم بترجمة متطلبات المستخدم إلي كود مركب وإرساله إلي قواعد البيانات ولهذا يمكن القول أن نظم قواعد البيانات تقوم بتسهيل مهمة التعامل مع ملفات أو جداول قواعد البيانات .

4-1 أهمية تصميم قواعد البيانات

يحتاج تصميم قواعد البيانات إلي عناية خاصة و يجب اخذ الكثير من الاعتبارات في عملية تصميمها قبل البدء في عملية البناء . فقاعدة البيانات التي يتم تصميمها بصورة جيدة تؤدي إلي الإدارة الجيدة للبيانات حتى تصبح مصدرا جيدا للمعلومات المطلوبة والتي تساعد الإدارات في اتخاذ القرارات بصورة جيدة .

أما قواعد البيانات التي تصمم بطريقة سيئة فإنها تولد بيانات متكررة تؤدي إلي معلومات خاطئة وأحيانا مضللة .



شكل 2-1

(مثال)

إذا تم تخزين رقم تليفون عميل مثلاً في ملف العملاء وكذلك ملف المبيعات وملف الفواتير فإنه عند تغيير رقم تليفون العميل فإنه يلزم تعديل الرقم في لملفات الثلاث فإن لم يتم ذلك فإن التقارير قد تحتوي علي أرقام تليفونات خاطئة (والتي سيطلق عليها فيما بعد " البيانات الشاذة ") .

5-1 تعريفات أساسية

■ البيانات Data

هي حقائق أساسية مثل رقم التليفون وتاريخ الميلاد ، اسم العميل والبيانات في صورتها المجردة لا تحتوي علي معني إلا إذا تم ترتيبها في صورة منطقية محددة ، وتكون البيانات في اصغر صورة لها علي هيئة حرف مثل A , F , M أو علي هيئة رقم مثل 5 أو حتى بعض العلامات مثل / .

■ الحقل Field

عبارة عن رمز (ربما حرف) أو مجموعة من الرموز أو الأرقام والتي لها قيمة محددة ويستخدم الحقل لتخزين بيان ما .

■ السجل Record

عبارة عن حقل أو أكثر علي صلة منطقية ويصف شيء محدد مثل شخص أو مكان مثل سجل العميل الذي قد يحتوي علي اسم العميل ، عنوانه ورقم تليفونه وتاريخ ميلاده .

■ الملف File

عبارة عن مجموعة من السجلات ذات علاقة مشتركة مثل ملف الطالب .

6-1 مفهوم قواعد البيانات

تمثل البيانات شيئاً هاماً في نشاط أي كائن ويمتلك كل كائن سواء طبيعياً مثل الأفراد أو اعتبارياً مثل الشركات والمؤسسات - مجموعة من البيانات التي يحتاج إلي تبويبها وحفظها وتحليلها وتحديثها وإدارتها بكفاءة لاستخراج المعلومات والتقارير التي تساعد في اتخاذ القرار المناسب في الوقت المناسب .

قواعد البيانات (2)

تمثل قواعد البيانات الحل الأمثل للمتطلبات السابق ذكرها ومن هنا تتبع أهمية دراسة هذا الموضوع . سنتطرق هنا إلي دراسة وتطبيق نوع محدد من قواعد البيانات ألا وهو قواعد البيانات العلاقة .

قبل الدخول في التفاصيل دعنا نتكلم قليلا عن جداول البيانات المعروفة (Spread Sheets) مثل تلك المستخدمة في البرنامج الشهير مايكروسوفت اكسيل (Microsoft Excel) .

مقارنه سريعة بين قواعد البيانات وجداول البيانات

كما هو معروف يتكون جدول البيانات من مجموعة من الأعمدة ومجموعة أخرى من الصفوف . يتم التعامل مع جدول البيانات من خلال الخلية وهي تمثل تقاطع عمود ما مع صف ما . ويتم حفظ البيانات في خلايا جدول البيانات ومن الممكن إنشاء علاقات بين الخلايا التي تقع في نفس جدول البيانات أو مع خلايا أخرى تقع في جدول بيانات آخر أيضا من الممكن إنشاء صيغ (Expressions) لإجراء عمليات رياضية علي القيم المحفوظة في الخلايا .

أيضا من الممكن فرض تنسيق ما (Format) علي إدخال البيانات في خليه ما إضافة إلي ذلك يمكن استخدام لغة برمجة مثل (Visual Basic for Applications VBA) لكتابة برامج تحتوي علي حسابات معقدة وشروط ومصنفات وغيرها ويسمح جدول البيانات أيضا بإنشاء نماذج لإدخال البيانات مما يسهل عملية إدخال البيانات في جدول البيانات ويضمن صحتها .

وبسبب هذه الخصائص المذكورة أعلاه والتي تتوفر عادة في قواعد البيانات كان الإقبال كبيرا علي استخدام برامج جداول البيانات لسهولة التعامل معها ورخص ثمنها ، رغم وجود هذا التشابه الواضح اعلاء فهناك اختلافات جوهرية بين جداول البيانات وقواعد البيانات .

ويوضح الجدول الآتي بعض نقاط الاختلاف الهامة بين قواعد البيانات وجداول البيانات :

جداول البيانات	قواعد البيانات
ليس هناك ما يسمى مفتاحا أو صفا فريدا .	يتم تمييز الصفوف عن بعضها البعض باستخدام قيمة فريدة تسمى المفتاح الرئيسي (Primary Key) .
يمكن حفظ البيانات في جدول واحد . وفي حالة تعدد الجداول	يتم حفظ البيانات في عدة جداول مترابطة فيما بينها (بواسطة

المفاتيح (بطريقة محددة مسبقا قبل إدخال .	فليس هناك ترابط هيكلي لغياب ما يسمى مفتاحا .
يحتوي أي عامود علي نوع واحد من البيانات .	يمكن لأي عامود أن يحتوي علي أكثر من نوع واحد من البيانات .
يوجد فصل بين قاعدة البيانات وبين البرامج التي تتعامل مع قاعدة البيانات .	لا يوجد فصل بين البيانات والبرامج ويحفظ كلاهما في نفس جدول البيانات .
تدعم التحكم في العمليات .	لا تدعم التحكم في العمليات .

1-6-1 نظام تشغيل الملفات

تم تطوير نظام قاعدة البيانات كبديل لنظام تشغيل الملفات التقليدي المدعوم من قبل نظم التشغيل ويسمح نظام تشغيل الملفات بعمل بحث متعاقب (Sequential) وعشوائي (Random) للبحث عن سجلات تخضع لشروط محددة .

من خصائص نظام تشغيل الملفات انه يتم حفظ السجلات في عدة ملفات ويجب كتابة برامج مختلفة للتعامل مع سجلات كل ملف وهذا يؤدي إلي عدة عيوب من أهمها :

- تكرارية البيانات وعدم توافها Data Redundancy and inconsistency : يتم إنشاء الملفات والبرامج التطبيقية حسب الطلب بواسطة عدة مبرمجين خلال فترة زمنية ما مما يؤدي في اغلب الأحوال إلي وجود نفس البيان في عدة ملفات وبتنسيق مختلف وبرامج تطبيقية مكتوبة بلغات مختلفة .
- انعزال البيانات : بما أن البيانات مبعثرة في عدة ملفات وقد تكون بتنسيق مختلف فانه من الصعب كتابه برنامج تطبيقي جديد للبحث عن بيانات مطلوبة.
- مشكلة أمن البيانات : تنشأ مشكلة أمن البيانات عند تعدد المستخدمين لان ليس لكل مستخدم الحق في التعامل مع كل البيانات بواسطة عدة مبرمجين وبدون تخطيط كامل مسبقا يصبح من الصعب تأكيد قيود امن البيانات .

قواعد البيانات (2)

- التحكم في العمليات المتزامنة : وهي العمليات التي تتم بواسطة أكثر من مستخدم علي نفس البيانات في نفس الوقت .ومثال ذلك حساب بنكي مشترك لشخصين وكل من الشخصين يطلب سحب الرصيد بالكامل .
- صعوبة إنشاء قيود تكامل جديدة علي البيانات (Integrity Constraint) لان هذا يستدعي تعديل البرامج التطبيقية .

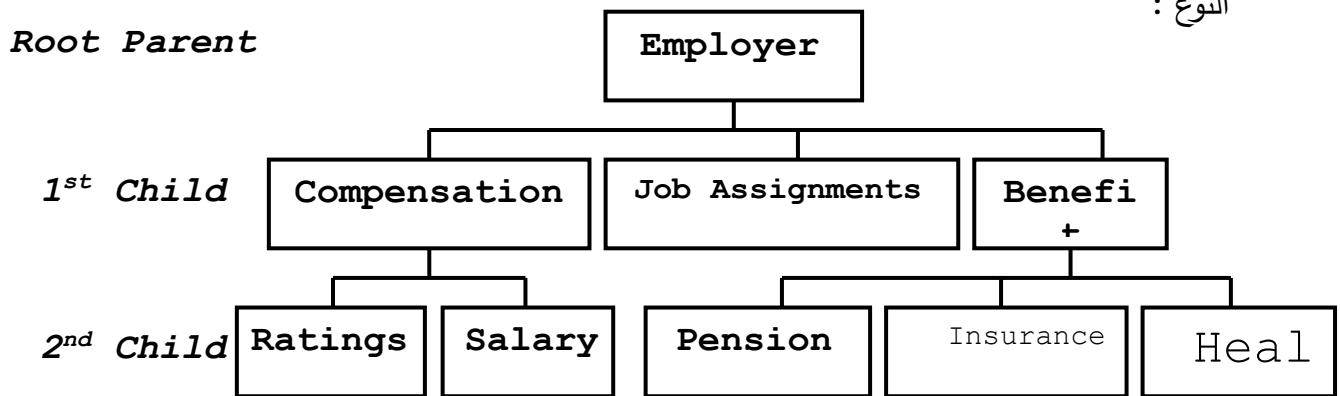
7-1 تطور نظم قواعد البيانات

تطورت نظم قواعد البيانات مع الزمن علي النحو التالي :

1-7-1 نظم قواعد البيانات الهرمية (Hierarchical)

يوضح الشكل 3-1 تركيب البيانات في نظم قواعد البيانات الهرمية ومن خصائص هذا

النوع :



شكل 3-1

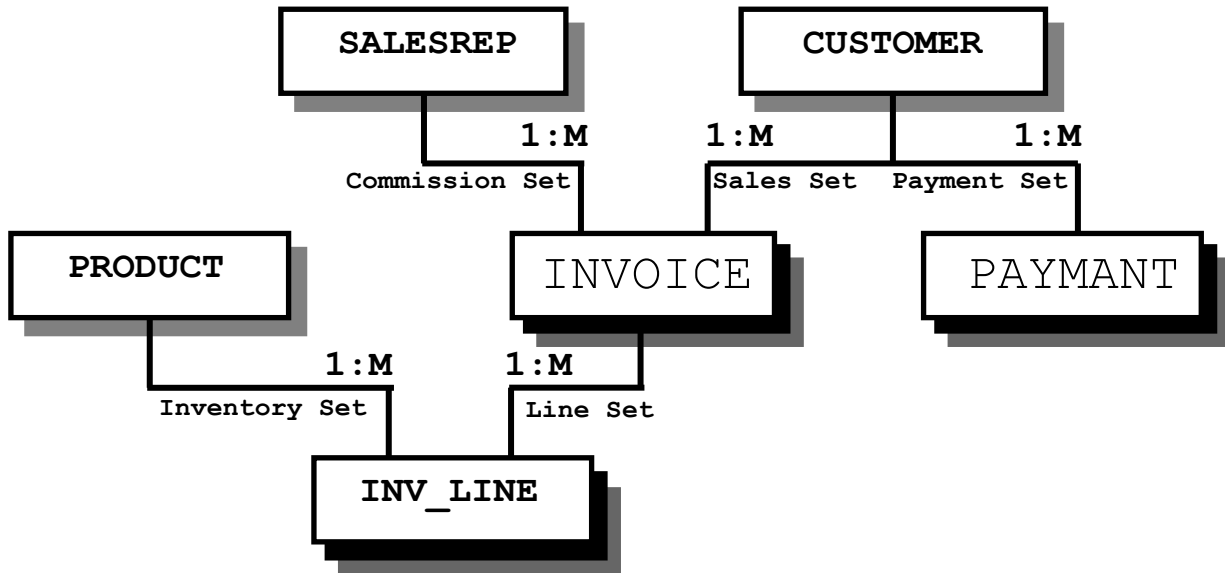
- هذا النوع هو الأقدم وهو مشابه لنظام الملفات علي الحاسب حيث يوجد عقدة جذرية (Root) يقع تحتها عقد فرعية وترتبط البيانات مع بعضها البعض كما في شجرة العائلة بحيث يكون لكل بيان أب واحد بينما يمكن أن يتفرع عن البيان عدة أبناء .
- من مزايا هذا النوع السرعة الفائقة للوصول إلي البيانات ولكن يعيبه عدم المرونة وتكلفة الصيانة العالية وصعوبة إنشاء استعلامات خاصة للمستخدم (يتطلب مهارة عالية في البرمجة ليست متوفرة لدي المستخدم العادي) .
- عدم المرونة وتكلفة الصيانة العالية وصعوبة إنشاء استعلامات خاصة للمستخدم (يتطلب مهارة عالية في البرمجة ليست متوفرة لدي المستخدم العادي) .

2-7-1 نظم قواعد البيانات الشبكية (Network)

وترتبط البيانات مع بعضها البعض بأسلوب شبكي بحيث يمكن أن يكون لكل بيان أكثر من أب واحد ويمكن أن يتفرع عن البيان عدة أبناء .

ويوضح الشكل (4-1) تركيب البيانات في نظم قواعد البيانات الشبكية .

ويتميز هذا النوع عن نظم قواعد البيانات الهرمية باستخدام لغة تعريف للبيانات (Data Definition Language " DDL ") ولغة التعامل مع البيانات (Data Manipulation Language " DML ") ولكن يعيبه عدم المرونة لتعديل هيكل الشبكة وتكلفة الصيانة العالية .



شكل 4-1

3-7-1 نظم قواعد البيانات العلاقية (Relational)

تم بناء هذا النوع علي مفهوم النموذج العلاقي (أو العلائقي كما في بعض المراجع العربية) ويعتبر الأكثر استخداما في الوقت الحالي ومن أمثلة هذا النوع قاعدة البيانات Access .
وتتميز نظم قواعد البيانات العلاقية بالآتي :

- سهولة إنشاء استعلامات خاصة بواسطة المستخدم .
- المرونة في تعديل هيكل قاعدة البيانات العلاقية .

انخفاض تكلفة الصيانة .

استخدام لغة تعريف للبيانات (DDL) .

قواعد البيانات (2)

استخدام لغة التعامل مع البيانات (DML) .

وسوف نتناول نظم قواعد البيانات العلاقية بالتفصيل لاحقا .

4-7-4 نظم قواعد البيانات الشيئية Object oriented

تم تطوير نظم قواعد البيانات الشيئية للتعامل مع :

- الإنترنت التي تتطلب تنفيذا موزعا (Distributed Processing) .
- الوسائط المتعددة التي تشمل الصور والفيديو والصوت والرسوم المتحركة والخرائط .

يبين الجدول التالي تاريخ تطوير الأجيال الأربعة لتنفيذ قواعد البيانات :

العام	الجيل	التقنية	أهم الخصائص
1960	الأول	نظام تشغيل الملفات	ملفات وبرامج تطبيقية خاصة بكل إدارة .
1970	الثاني	شبكات / هرمية	شبكة من السجلات المتعلقة ببعضها
1980	الثالث	علاقية	لغة الاستعلام الهيكلي و Transaction .
1990	الرابع	شيئية	الوسائط المتعددة والانترنت .

تعريفات خاصة بقواعد البيانات

ما هي قواعد البيانات ؟

- هي مجموعة من البيانات المترابطة فيما بينها والمقصود بكلمة بيانات هي الحقائق التي يمكن تسجيلها وحفظها ولها معنى مثل أسماء وأرقام هواتف وعناوين أصدقائك .

- هي مجموعة من السجلات المتكاملة الذاتية الوصف (Self- Descriptio) .

- هي عبارة عن مجموعة منظمة من البيانات الثابتة أو الدائمة التي يتم حفظها بواسطة برنامج تطبيقي . والمقصود بكلمة ثابتة أو دائمة انه في حالة إنهاء البرنامج التطبيقي الذي يتعامل مع قاعدة البيانات المحفوظة لا تتغير إلا إذا قام المستخدم بتغييرها بواسطة برنامج تطبيقي مصمم لهذا الغرض .

8-1 خصائص قواعد البيانات

من أهم خصائص قواعد البيانات الآتي :

- 1- تمثل قواعد البيانات بعض أوجه العالم الحقيقي .
مثال : قاعدة بيانات العاملين في شركة ما .
- 2- تمثل قواعد البيانات حالة للبيانات في وقت معين .
مثال : أعلى مؤهل علمي لموظف حيث يمكن أن يتغير المؤهل مع الوقت يتم تصميم وإنشاء وملئ بيانات قاعدة بيانات ما لخدمة غرض معين .
- 3- لكل قاعدة بيانات مجموعة من المستخدمين بحقوق دخول محددة .
- 4- لكل قاعدة بيانات مجموعة من البرامج التطبيقية لملئ وتحديث البيانات وللاستعلام .
- 5- يتعامل المستخدم مع قاعدة البيانات من خلال مجموعة البرامج التطبيقية المصممة لتلبية متطلباته .

9-1 نظام إدارة قواعد البيانات Database Management System

ما هو نظام إدارة قواعد البيانات ؟

هو مجموعة من البرامج الجاهزة التي تسهل مهمة المستخدم في إنشاء وصيانة قواعد البيانات . ويتم تصميم نظام اداره قواعد البيانات لمساعدة المستخدم في تعريف وإنشاء ومعالجه قواعد البيانات .

ما هو نظام قاعدة البيانات ؟

هو نظام اداره قاعدة البيانات مضافا إليه قاعدة البيانات نفسها . ويتكون نظام قاعدة البيانات من خمسة أجزاء وهي :

- البيانات
- الأجهزة (Hardware)
- البرمجيات (Software)
- المستخدمين
- الإجراءات والعمليات .

1-9-1 واجهات التعامل مع نظم إدارة قواعد البيانات

DBMS InterFaces

يتم تعامل المستخدم مع نظم اداره قواعد البيانات من خلال الواجهات التالية :

- 1- واجهة القوائم (Menu - Based Interface) : يتيح النظام مجموعة من القوائم لتنفيذ أوامر المستخدم .
- 2- واجهة النماذج (Forms-Based Interface) : يتيح النظام مجموعة من النماذج لإدخال البيانات الجديدة أو استرجاع البيانات المطلوبة .
- 3- واجهة الرسومات (Graphical User Interface) : يتيح النظام مجموعة من المخططات ومنها مخطط العلاقات بين الجداول ومخطط الجداول التي يرغب المستخدم في استخدامها لإنشاء الاستعلامات .
- 4- واجهة اللغات الطبيعية (Natural Language Interfac) : يتيح النظام إمكانية التعامل مع اللغات الطبيعية (المشابهة للغة المستخدم) لإنشاء الاستعلامات .
- 5- واجهة مدير قاعدة البيانات (Database Administrator Interface) : يوفر النظام واجهة خاصة ليقوم من خلالها مدير قاعدة البيانات بالمهام الخاصة به مثل أمن النظام وصلاحيات المستخدمين وصيانة قاعدة البيانات .

2-9-1 تصنيف نظم اداره قواعد البيانات

يمكن تصنيف نظم اداره قواعد البيانات تبعا للنقاط التالية :

- 1- نموذج البيانات المستخدم في قاعدة البيانات : هرمي - شبكي - علاقي - شيئي .
- 2- عدد المستخدمين لقاعدة البيانات
- 3- عدد مواقع توزيع قاعدة البيانات
- 4- تكلفة قاعدة البيانات
- 5- الغرض من قاعدة البيانات : عام / خاص

3-9-1 وظائف نظم إدارة قواعد البيانات

يمكن تلخيص الوظائف الرئيسية لنظم اداره قواعد البيانات كالآتي:

1- تخزين البيانات في قاعدة البيانات وذلك يشمل إضافة ملفات جديدة وإضافة بيانات للملفات الموجودة .

2- استرجاع البيانات المحفوظة في قاعدة البيانات .

3- تعديل البيانات الموجودة في قاعدة البيانات وذلك يشمل إلغاء البيانات والملفات.

بالإضافة إلي ذلك تقوم اداره قواعد البيانات بالاتي :

إنشاء كتالوج يشمل :

- اسم ونوع وحجم البيانات المحفوظة في قاعدة البيانات .

- أسماء العلاقات بين الجداول (Relationships) .

- قيود التكامل علي البيانات (Integrity Constraints) . وهذه القيود يتم وضعها

من قبل مصمم البيانات ويتم برمجتها من خلال لغة خاصة تعرف باسم لغة الاستعلام الهيكلية

“SQL” Structured Query Language.

- أسماء المستخدمين وحقوقهم في الدخول الي قاعدة البيانات .

- مخططات قاعدة البيانات (Schema)

- التحكم في العمليات المتزامنة (Concurrency Control) والتي تتم بواسطة أكثر من

مستخدم علي نفس البيانات في نفس الوقت ومثال ذلك حساب بنكي مشترك لشخصين وكل

من الشخصين يطلب سحب الرصيد بالكامل .

- التحكم في عمليات Transactions .

ويمكن تعريف ال Transactions بأنه عبارة عن مجموعة من الأوامر التي يجب تنفيذها

كمجموعة متكاملة أي لا يجوز تنفيذ جزء منها وبالتالي أما يتم تنفيذها كلها أو لا يتم تنفيذ أي

جزء منها . ومثال ذلك نظام بنك حيث يتم تحديث مجموعة مترابطة من البيانات فإذا توقف

النظام بعد البدء في تنفيذ جزء من هذه المجموعة فستكون حالة البيانات غير مقبولة ولذلك يتم

إلغاء ما تم تنفيذه جزئياً للعودة إلي حالة البيانات الأولى .

4-9-1 مميزات استخدام نظم اداره قواعد البيانات

- 1- تسهيل مهمة المستخدم في إنشاء قواعد البيانات وصيانتها (تخزين وتعديل البيانات) .
- 2- تسهيل مهمة المستخدم في إنشاء الاستعلامات
- 3- التحكم في عدم تكرار البيانات مما يؤدي إلي صحة البيانات وسهولة صيانتها .
- 4- دعم قيود التكامل (Integrity Constraints) .
- 5- وذلك عن طريق منع تجاوز الشروط الموضوعة من قبل مصمم النظام.
- 6- دعم مجموعة عمليات Transactions Processing .
- 7- تحسين الأمن (Security) لحماية النظام وقاعدة البيانات.
- 8- سهولة عمل نسخ احتياطية لقاعدة البيانات (Backup) .
- 9- سهولة إعادة تشغيل قاعدة البيانات (Recovery) في حالة توقفها .
- 10- إمكانية تعدد واجهة المستخدم (Views) للتعامل مع قاعدة البيانات (يمكن عرض مجموعة من البيانات لكل مستخدم تبعاً لحقه في رؤية البيانات والتعامل معها) .
- 11- إنقاص زمن تطوير البرامج التطبيقية التي تتعامل مع قاعدة البيانات (وذلك بتوفير بعض الوظائف اللازمة للتعامل مع قاعدة البيانات مع المرونة تعديل البيانات) .

بعض نظم إدارة قواعد البيانات المتوفرة في السوق

هناك العديد من نظم إدارة قواعد البيانات المتوفرة في السوق ومن أكثرها انتشاراً الآتي :

رقم	اسم نظام إدارة قواعد البيانات	اسم الشركة المنتجة
1	IBM DB2	IBM
2	Oracle	Oracle
3	Sql Sever	Microsoft
4	Informix	Informix
5	Sybase SQL Server	Sybase
6	Access	Microsoft

1-9-5 عيوب استخدام نظم اداره قواعد البيانات

- 1- تكلفة شراء نظم إدارة قواعد البيانات (Software) .
- 2- تكلفة شراء أجهزة خاصة لتشغيل نظم اداره قواعد البيانات (HardWare)
- 3- تكلفة تدريب المستخدمين علي نظم اداره قواعد البيانات .
- 4- سرعة الأداء تكون اقل بسبب الزمن المضاف لتنفيذ البرامج الخاصة بنظم اداره قواعد البيانات .
- 5- زيادة تأثير توقف نظم اداره قواعد البيانات وذلك بسبب المستخدمين علي تلك النظم بصفة أساسية .

متى يسمح بعدم استخدام نظام إدارة قواعد بيانات ؟

ينصح بعدم استخدام نظام إدارة قواعد بيانات في الحالات التالية :

- 1- بساطة قاعدة البيانات والبرامج التطبيقية التي تتعامل معها .
- 2- قاعدة بيانات محددة جيدا ومن غير المتوقع أن تتغير في المستقبل .
- 3- عامل السرعة هام للغاية (وهذا متطلب بنظام الوقت الحقيقي).
- 4- تعدد دخول المستخدمين غير مطلوب .
- 5- التكلفة المرتفعة لنظام إدارة قواعد بيانات .

1-9-6 أمثلة تطبيقية لاستخدام قواعد البيانات

- البنوك : نظم حسابات العملاء - التحويلات - الاستثمار .
- الطيران : نظام الحجز - الجدولة
- المستشفيات : نظام الحجز - العيادات الخارجية - العيادات الداخلية - الصيدلة - السجل الطبي - المعامل - صيانة الأجهزة .
- الجامعات والمعاهد : نظام التسجيل - السجل الأكاديمي .
- الشركات التجارية : نظام المخازن - الحسابات - الموردين - العملاء .
- الشركات الصناعية : نظام المخازن - المواد الخام - المنتجات المصنعة - أوامر التصنيع - متابعة الإنتاج - الحسابات - صيانة الآلات - الموردين - العملاء .
- الموارد البشرية : نظام المرتبات - الحضور والانصراف - السجل الوظيفي .

7-9-1 أوجه تمييز نظم قواعد البيانات علي نظم الملفات

من أهم ما تتميز به نظم قواعد البيانات علي نظم الملفات الآتي :

- 1- تواجد نسخة واحدة من كل بيان في قاعدة البيانات بدلا من تكرار نفس البيان في عدة ملفات كل منها خاص بمجموعة من المستخدمين في ادارته ما . وهذا يؤدي إلي سهولة تعديل البيان لأنه محفوظ في مكان واحد وهذه النسخة يستخدمها أكثر من مستخدم .
- 2- عزل البيانات عن البرامج (Program/Data Isolation) في قواعد البيانات يتم فصل بين البرامج عن البيانات حيث لا تحتوي البرامج علي وصف للبيانات المستخدمة ومن مزايا هذا الفصل إتاحة المرونة في تعديل شكل ونوع البيانات دون المساس بالبرامج .
- 3- الوصف الذاتي (Self description) : حيث تحتوي قاعدة البيانات علي كتالوج يشمل وصفا للبيانات التي تتضمنها قاعدة البيانات والتي تسمى بيانات عن البيانات (Meta data).
- 4- تعدد المستخدمين والمشاركة في البيانات Multi-user System & Data Sharing .
- 5- إتاحة عدة مناهير للبيانات (Views) : حيث يكون لكل مستخدم منظور خاص للبيانات التي يتعامل معها .

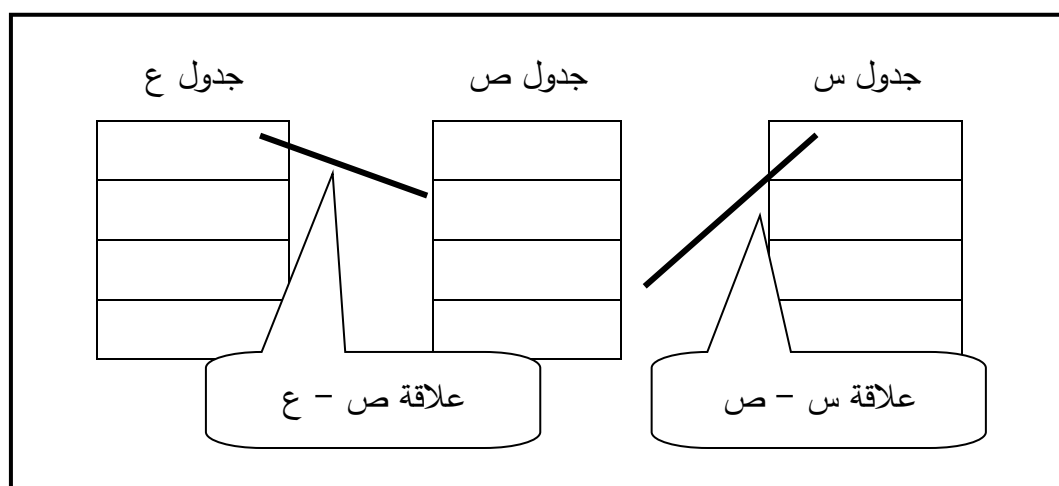
مثال : بيانات الموظفين في شركة ما :

- القسم الإداري يري واجهة عن الموظف ومؤهلاته وخبراته.
- القسم المالي يري واجهه عن المرتب والسلفيات والعلاوات المرتقبة .

10-1 قواعد البيانات العلاقية Relational Databases

تعتبر قواعد البيانات العلاقية النوع الأكثر استخداما في الوقت الحالي وذلك بسبب بساطتها وارتكازها علي خلفية رياضية تنتمي إلي نظرية المجموعات والجبر العلاقي Sets Theory and Relational Algebra .

يمكن تعريف قاعدة البيانات العلاقية بأنها مجموعة من جداول البيانات Tables المرتبطة فيما بينها بعلاقات Relations كما في الشكل (1-5) .



شكل (1-5) العلاقات بين الجداول

وقد تكون قاعدة البيانات العلاقية منشأه في ملف واحد كما هو الحال في Microsoft Access ومثال ذلك قاعدة البيانات شركة نورثونيد المصاحبة لبرنامج Microsoft Access (North wind . mbd) أو في عدة ملفات كما في Microsoft SQL Server .

1-10-1 عناصر قاعدة البيانات العلاقية

بالإضافة إلى البيانات نفسها تشمل قاعدة البيانات العلاقية عناصر أخرى كما هو موضح فيما يلي

- 1- الجداول والأعمدة والصفوف .
- 2- أنواع البيانات (Data Types) .
- 3- الإجراءات المحفوظة / ماكرو (Stored procedures / Macros) .
الإجراءات المحفوظة تعتبر من الموضوعات المتقدمة في قواعد البيانات ويتم كتابتها باستخدام لغة خاصة بها مثل لغة Transact-SQL المستخدمة مع قاعدة البيانات Microsoft SQL Server أو SQL*Plus المستخدمة مع قاعدة البيانات Oracle .
- 4- القداحات (Triggers) وهي إجراءات محفوظة يتم إطلاقها (تفعيلها) إما قبل أو بعد إضافة تعديل أو حذف بيانات من قاعدة البيانات .
- 5- المناظر (Views) ويتم إنشائها من جدول أو عدة جداول بغرض حجب بعض الأعمدة (البيانات) عن بعض المستخدمين .
- 6- الفهارس (Indexes) وهي تنظيم البيانات بغرض جعل تنفيذ عملية الاستعلام سريع وسيتم شرحها لاحقاً .

قواعد البيانات (2)

- 7- المفاتيح الأساسية أو الرئيسية (Primary Keys) تستخدم المفاتيح الأساسية بغرض تمييز الصفوف في الجداول وسيتم شرحها لاحقا .
- 8- المفاتيح الأجنبية (Foreign Keys) : تستخدم المفاتيح الأجنبية بغرض ربط الجداول وسيتم شرحها لاحقا .
- 9- القيود (Constraints) : وتستخدم علي مستوي البرنامج الخادم (Server) بدلا من كتابتها في البرامج التطبيقية وذلك لوضع قيود علي تكامل البيانات وسيتم شرحها لاحقا .

2-10-1 الجداول والأعمدة والصفوف

كما ذكر سابقا تتكون قاعدة البيانات العلاقية من مجموعة من جداول البيانات (Tables) المرتبطة فيما بينها بعلاقات Relations . وتستخدم الجداول لحفظ البيانات ، كما تستخدم لتمثيل كائنات وعلاقات Entities & Relations قاعدة بيانات .

يتكون جدول البيانات (الجدول) من :

- مجموعة من الأعمدة (Columns) وتسمي أيضا حقول (Fields) أو صفات أو عناصر (Attributes) .

- مجموعة من الصفوف (Rows) وتسمي أيضا سجلات (Records) أو (Topless) .

لكل جدول في قاعدة البيانات بنية (هيكل) وهي مجموعة الأعمدة التي يتكون منها الجدول . ويمكن التعبير عن الجدول باستخدام اسم الجدول مقرونا ببنية الجدول بين قوسين كما يلي :

اسم الجدول (اسم العمود 1 - اسم العمود 2 - اسم العمود 3 ، ...)

مثال : الطالب (الاسم ، العنوان ، رقم الهاتف ، التخصص)

3-10-1 المفتاح الأساسي والمفتاح الأجنبي

سننظر هنا باختصار إلي التعريفات وكيفية استخدام المفاتيح .

المفتاح الأساسي (الرئيس) (Primary Key (PK)

يعرف المفتاح الأساسي للجدول بأنه عمود أو أكثر في الجداول يحتوي علي قيم فريدة (أي لا تتكرر) .

قواعد البيانات (2)

ما هي فائدة المفتاح الأساسي ؟

يوفر المفتاح الأساسي وسيلة لتمييز صفوف الجدول أي انه يمكن تحديد صف معين في الجدول بمعرفة قيمة المفتاح الأساسي .

كيف يتم تمثيل المفتاح الأساسي للجدول ؟

يتم تمثيل المفتاح الأساسي للجدول عن طريق وضع خط تحت العامود (أو الأعمدة في حال كونه من النوع المركب) الذي يعرف المفتاح الأساسي .

أمثلة

- الجدول التالي يمثل المدرس بصفاته الثلاثة : رقم المدرس ، اسم المدرس ، الهاتف . المفتاح الأساسي هو رقم المدرس .

المدرس (رقم المدرس ، اسم المدرس ، الهاتف)

- الجدول التالي يمثل الطالب بصفاته الأربعة : رقم الطالب ، اسم الطالب ، التخصص ، مدينة السكن ، المفتاح الأساسي هو رقم الطالب .

الطالب (رقم الطالب ، اسم الطالب ، التخصص ، مدينة السكن)

مثال : الجدول التالي يحتوي على بعض بيانات الطلبة :

بنية الجدول				اسم الجدول
رقم الطالب	اسم الطالب	التخصص	مدينة السكن	الطالب
277	فهد	برمجة	الرياض	صفوف سجلات Tuples
418	سعد	قواعد	جدة	
222	جمال	شبكات	الدمام	
111	أحمد	برمجة	جدة	
421	سعيد	قواعد	الرياض	
أعمدة - حقول - Attributes				

- اسم الجدول : الطالب .
- الأعمدة (الحقول وعددها 4) هي : رقم الطالب ، اسم الطالب ، التخصص ، مدينة السكن
- الصفوف (السجلات وعددها 5) : هي بيانات 5 طلاب .

قواعد البيانات (2)

- بنية (هيكل) الجدول : هي الأعمدة الأربعة التي يتكون منها الجدول (وهي : رقم الطالب ، اسم الطالب ، التخصص ، مدينة السكن) .

4-10-1 تمثيل بنية الجدول

يتم تمثيل بنية الجدول بدون البيانات كما في Microsoft Access و Microsoft SQL Server كالتالي :

مثال :

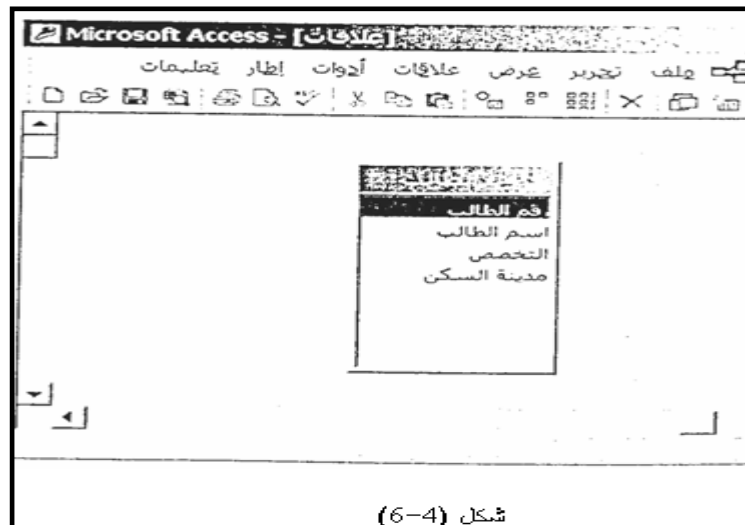
اسم الجدول	الطالب
اسم العمود الأول	رقم الطالب
اسم العمود الثاني	اسم الطالب
اسم العمود الثالث	التخصص
اسم العمود الرابع	مدينة السكن
.....
اسم العمود الأخير

وتظهر بنية الجدول في شاشة العلاقات في Microsoft Access كما في الشكل

(6-1) .

ترتيب الصفوف والأعمدة في الجداول

عند إدخال قيم البيانات في أحد الصفوف يجب مراعاة أن يتوافق ترتيب إدخال قيم البيانات مع بنية الجدول .

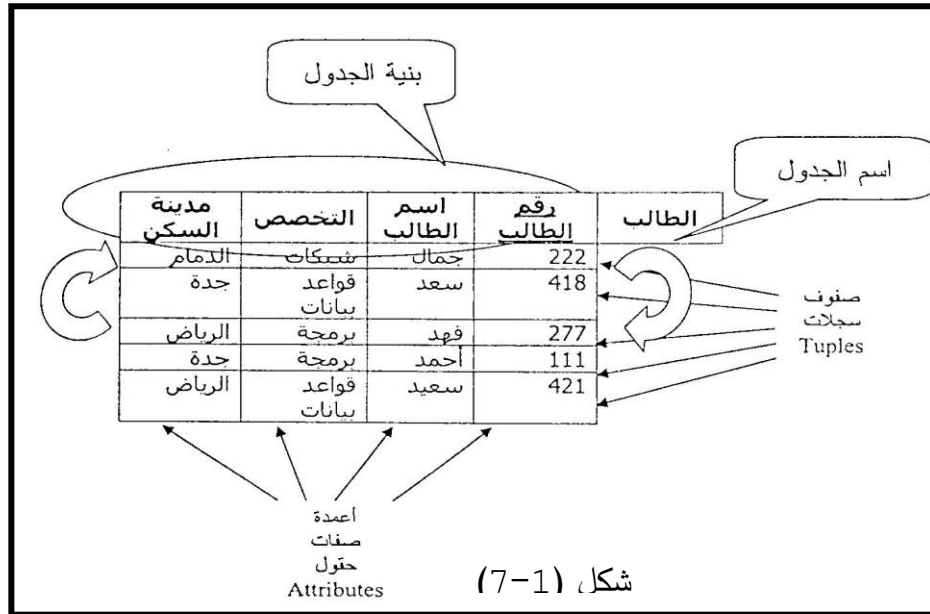


شكل (6-4)

5-10-4 ترتيب الصفوف والأعمدة في الجدول

عند إدخال قيم البيانات في أحد الصفوف يجب مراعاة أن يتوافق ترتيب إدخال القيم مع بنية الجدول . وعلى العكس من ذلك ، فإن ترتيب الصفوف في الجدول ليس له أهمية .

مثال : جدول الطالب المعطى في المثال السابق تم تغيير ترتيب الصفوف كما في الشكل التالي (7-1) بحيث تم تبديل صف الطالب جمال مع الطالب فهد . ونلاحظ أن هذا التبديل لا يؤثر على جواب السؤال التالي : ما هو اسم وتخصص الطالب رقم 222 ؟ فالإجابة في الحالتين هو جمال وشبكات .

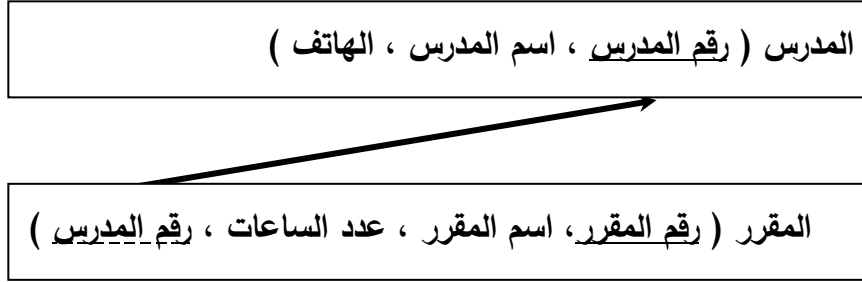
**6-10-1 المفتاح الأجنبي "FK" Foreign Key**

يعرف المفتاح الأجنبي للجدول بأنة عمود أو أكثر في الجدول يشير إلي المفتاح الأساسي في جدول آخر . ما هي فائدة المفتاح الأجنبي ؟ يوفر المفتاح الأجنبي وسيلة لربط جدولين أي صفوف الجدولين ، أي انه يمكن تحديد صف معين في الجدول بمعرفة قيمة المفتاح الأساسي . كيف يتم تمثيل المفتاح الأجنبي لجدول ؟ يمكن تمثيل المفتاح الأجنبي لجدول بطريقتين :

- 1- عن طريق وضع خط متقطع تحت العامود (أو الأعمدة) الذي يعرف المفتاح الأجنبي .
- 2- في حالة استخدام مخطط البيانات يتم رسم سهم يبدأ عند المفتاح الأجنبي وينتهي براس السهم عند المفتاح الأساسي .

قواعد البيانات (2)

مثال : اعتبر الجدولين التاليين : المدرس الواحد يمكنه أن يدرس أكثر من مقرر والمقرر الواحد يتم تدريسه بواسطة مدرس فقط .



7-10-1 أنواع العلاقات بين الجداول

هناك ثلاثة أنواع من العلاقات (الارتباطات) بين الجداول في قاعدة البيانات العلاقية كما هو مبين في الجدول التالي :

رمز العلاقة	اسم العلاقة باللغة الإنجليزية	اسم العلاقة باللغة العربية	
1 : 1	One to one	علاقة واحد إلي واحد	1
1 : M	One to many	علاقة واحد إلي متعدد	2
M : N	Many to many	علاقة متعدد إلي متعدد	3

ويلاحظ أن رمز العلاقة يجب قراءته في الاتجاه الموضح بالسهم (اتجاه قراءة اللغة الإنجليزية)
 أمثلة علي أنواع العلاقات بين الجداول في قواعد البيانات العلاقية
 سنقدم فيما يلي بعض الأمثلة التي توضح أنواع العلاقات بين الجداول في قواعد البيانات العلاقية كما سنعرض شاشة العلاقات في ميكروسوفت اكسيس المناظرة .

مثال 1 : علاقة واحد إلي واحد (1 : 1)

العلاقة بين القسم والمدير (مدير القسم) هي من النوع واحد إلي واحد وهذا يعني:

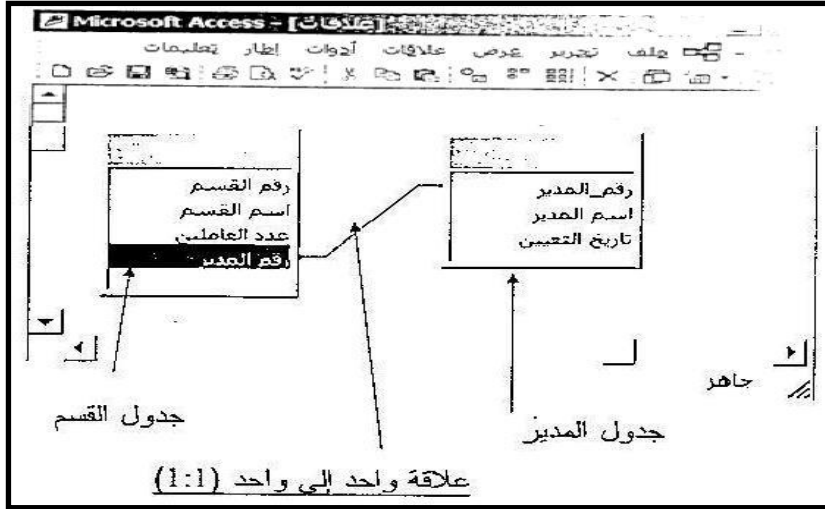
- أن لكل قسم مدير واحد .
- أن كل مدير يرأس قسم واحد .

سنعتبر هنا الجدولين الآتين :

القسم (رقم القسم ، اسم القسم ، عدد العاملين ، رقم المدير)

المدير (رقم المدير ، اسم المدير ، تاريخ التعيين)

وتظهر بنية جدول الطالب في شاشة العلاقات في Microsoft Access كما في الشكل (8-1) .



الشكل (8-1) : علاقة بين جدول المدير وجدول

مثال 2 : علاقة واحد إلي متعدد (1 : many)

سنعتبر في هذا المثال الجدولين الآتيين :

المدرس (رقم المدرس ، اسم المدرس ، الهاتف)

المقرر (رقم المقرر ، اسم المقرر ، عدد الساعات ، رقم المدرس)

لاحظ في هذا المثال :

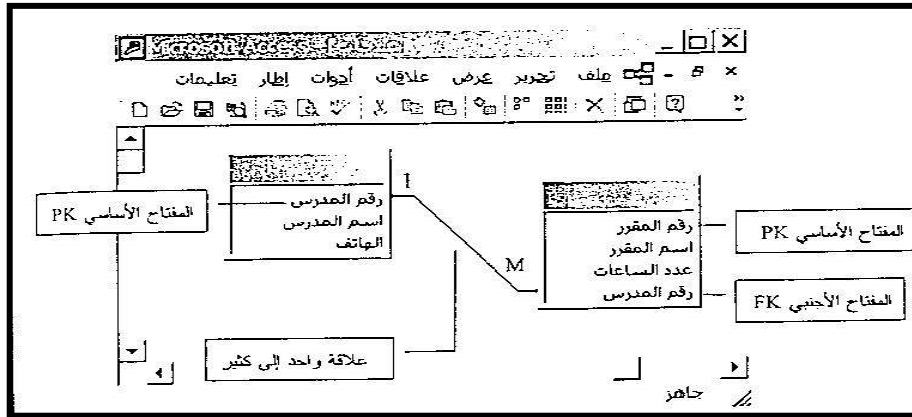
- المقرر الواحد يتم تدريسه بواسطة مدرس واحد فقط .
- المدرس الواحد يمكنه أن يدرس أكثر من مقرر .
- المفتاح الأساسي لجدول المدرس هو رقم المدرس وتحتة خط.
- المفتاح الأساسي لجدول هو رقم المقرر وتحتة خط .
- المفتاح الأجنبي :

قواعد البيانات (2)

- جدول المدرس لا يحتوي علي مفتاح أجنبي .
- جدول المقرر يحتوي علي مفتاح أجنبي اسمه " رقم المدرس " وتحتة خط متقطع ويشير إلي المفتاح الأساسي في جدول المدرس الذي له نفس المسمى " رقم المدرس " .
- السهم يبدأ عند المفتاح الأجنبي في جدول المقرر ويشير إلي المفتاح الأساسي في جدول المدرس .

يوضح الشكل التالي (9-1) كيفية تمثيل علاقة واحد إلى متعدد في شاشة العلاقات في

Microsoft Access .



شكا ، (9-1)

مثال 3 : علاقة متعدد إلي متعدد (Many : Many)

سنعتبر في هذا المثال الجدولين الآتيين :

الطالب (رقم الطالب ، التخصص ، مدينة السكن)

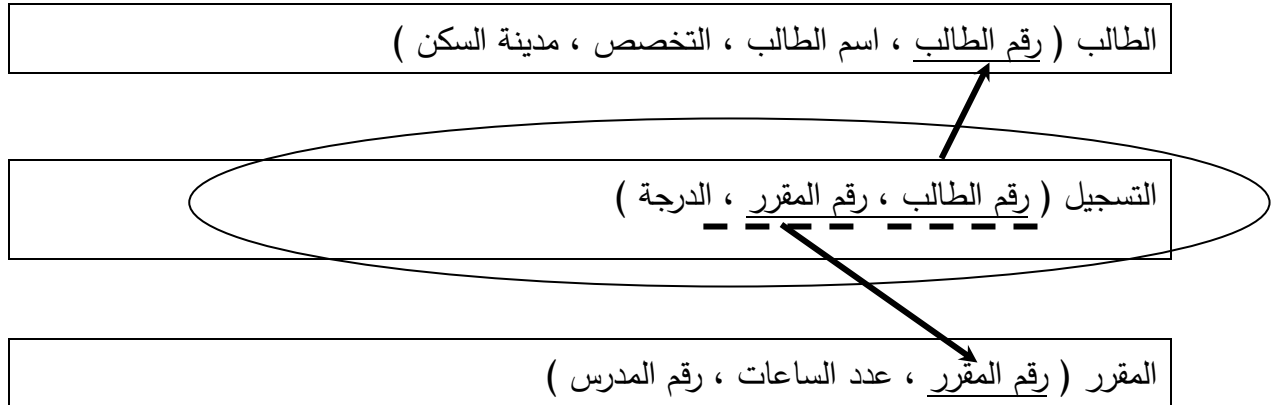
المقرر (رقم المقرر ، اسم المقرر ، عدد الساعات ، رقم المدرس)

لاحظ الآتي في هذا المثال :

- المقرر الواحد يمكن أن يسجل فيه أكثر من طالب .
- الطالب الواحد يمكنه أن يسجل أكثر من مقرر .

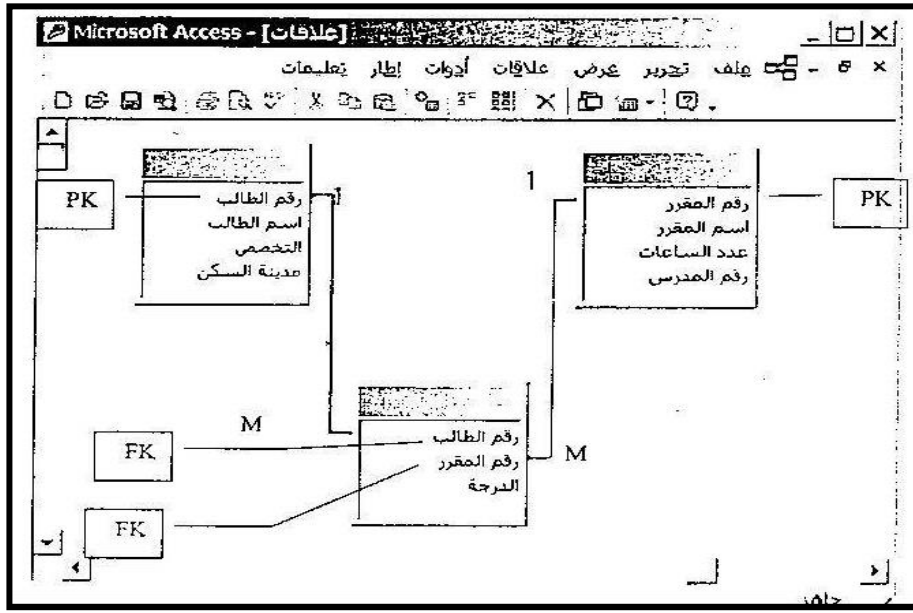
قواعد البيانات (2)

في علاقة متعدد إلي متعدد لا يمكن التعامل مباشرة مع الجدولين الأصليين (جدول الطالب و جدول المقرر) ولذلك يجب في هذا المثال إضافة جدول وسط بين الجدولين باسم جدول التسجيل بحيث يصبح لدينا ثلاثة جداول والعلاقات من نوع واحد إلي متعدد .



الشكل (10-1) يبين كيفية تمثيل علاقة متعدد إلي متعدد في شاشة العلاقات في Microsoft Access وإضافة جدول وسيط بين الجدولين (التسجيل) . لاحظ في هذا الشكل :

- كيف تظهر بنية الجدولين الأصليين و جدول علاقة التسجيل .
- استخدام المفتاح الأساسي والمفتاح الأجنبي للربط
- أن عنصر ربط جدولي المقرر والتسجيل هو رقم المقرر المتواجد في الجدولين - حيث يشير المفتاح الأجنبي باسم رقم المقرر في جدول التسجيل إلي المفتاح الأساسي باسم رقم المقرر (نفس الاسم) في جدول المقرر .
- أن عنصر ربط جدولي الطالب والتسجيل هو رقم الطالب المتواجد في الجدولين - حيث يشير المفتاح الأجنبي باسم رقم الطالب في جدول التسجيل إلي المفتاح الأساسي باسم رقم الطالب (نفس الاسم) في جدول الطالب .



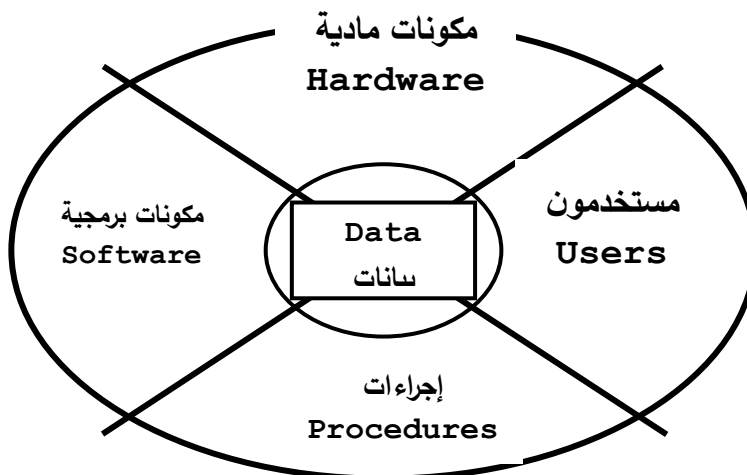
شكل (1) -1

11-1 مكونات نظم قواعد البيانات

Database Systems Components

تتكون نظم قواعد البيانات ذات علاقات منطقية مع بعضها البعض ومخزنه في مخزن واحد يطلق عليه Repository بعكس نظم الملفات التي تعتمد علي ملفات مستقلة ولا تربطها علاقة واضحة وتمثل قواعد البيانات تغيرا كبيرا في الطريقة التي تخزن بها البيانات وطرق الولوج إليها وكذلك في طريقة إدارتها والمحافظة علي سلامتها . وعبارة " نظم قاعدة البيانات " ترمز إلي مجموعة العناصر التي تعرف وتنظم مجموعة من البيانات من ناحية إدارتها وتخزينها واستخدامها في محيط بيئة عملها .

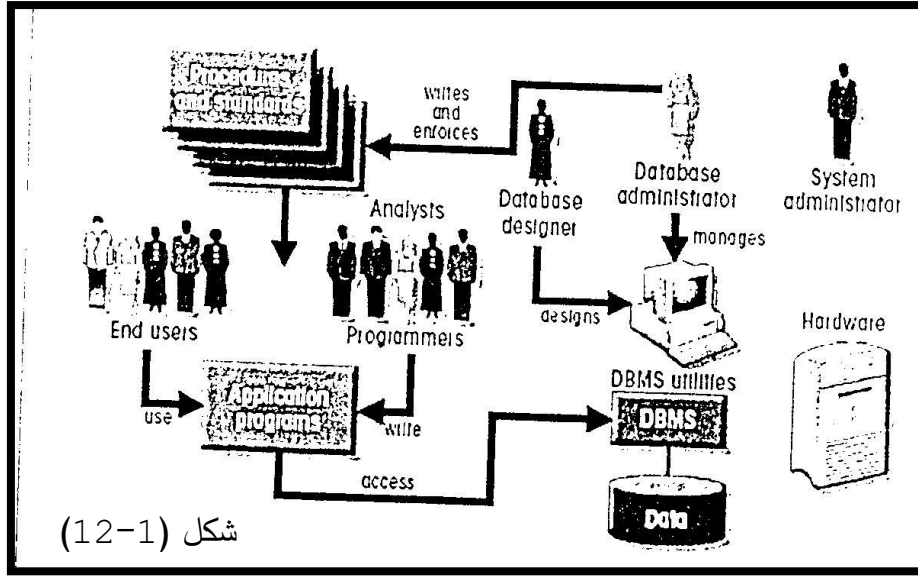
وتتكون قواعد البيانات في صورتها العامة من خمسة مكونات رئيسية كما في الشكل التخطيطي (11-1) والشكل الواقعي لها كما بالشكل (12-1) وهي :



شكل (11-1)

قواعد البيانات (2)

- 1- المكونات المادية Hardware .
- 2- المكونات البرمجية Software .
- 3- البيانات Data .
- 4- الإجراءات والعمليات Procedures .
- 5- المستخدمون Users .



1-11-1 المكونات المادية Hardware

تحتاج نظم قواعد البيانات إلى مكونات مادية Hardware أي حاسب ومكوناته Computer peripherals والذي يطلق عليهم "عتاد" حتى تصبح قلة للاستخدام . وهذه المكونات الحاسوبية تتراوح بين أجهزة حاسبات شخصية وأجهزة حاسبات رئيسية مثل Servers or Mainframes .

مكونات الحاسب تشمل العتاد المعتاد كالمطابعات وشبكات الحاسب وتعد شبكات الحاسب من العتاد الضروري لقواعد البيانات الموزعة والتي يلزم الولوج بها من أماكن بعيدة عن الحاسب مثل نظم حجز مقاعد الطائرات وأجهزة ATM والصرف الآلي .

2-11-1 المكونات البرمجية Software

وتشمل نظم البرمجة الخاصة بقواعد البيانات وغيرها من نظم التشغيل . ويمكن تقسيم المكونات البرمجية إلى ثلاث أنواع رئيسية هي :

قواعد البيانات (2)

1- نظم التشغيل Operating System Software ووظيفتها الرئيسية هي تنظيم وإدارة المكونات المادية Hardware مع إمكانية تشغيل البرمجيات الأخرى علي أجهزة الحاسب ومن أمثلتها نظام windows Xp ونظم Unix ونظم MVS لإدارة الحاسبات الرئيسية .

2- نظم إدارة قواعد البيانات DBMS : ووظيفتها تنظيم قواعد البيانات داخل نظام إدارة قواعد البيانات مثل Microsoft Access .

3- البرامج التطبيقية والمساعدة Application Programs and Utility Software : وتستخدم في عمليات الولوج وتنظيم البيانات في قواعد البيانات وغالبا ما تستخدم في عمل التقارير والجداول وتنظيم المعلومات بصورة ما لعمليات اتخاذ القرارات .

3-11-1 Data البيانات

وتعتبر البيانات من أهم مكونات نظم قواعد البيانات من وجهة نظر المستخدم وهي العنصر الرئيسي الذي يقوم بخدمته باقي عناصر نظم إدارة قواعد البيانات . أنظر الشكل (1-11) . ولذلك فهي العنصر الأساسي الذي نركز عليه في تصميم نظم قواعد البيانات .

4-11-1 Procedures الإجراءات والعمليات

وهي عبارة عن التعليمات والقوانين التي تحكم التصميم الجيد واستخدام قواعد البيانات علي الوجه الأمثل حيث أن مستخدم نظم قواعد البيانات والعاملين علي إدارة تلك الأنظمة وصيانتها بحاجة إلي إجراءات وتعليمات مسجلة حول طبيعة الاستخدام والتشغيل والتنفيذ للنظام وذلك تبعا لنوع ومستوي المستخدم .

كما توفر الإجراءات الوسيلة الصحيحة لمراقبة البيانات التي يتم إدخالها إلي قواعد البيانات والمعلومات التي تم الحصول عليها من خلال استخدامها وطرق استحداثها.

5-11-1 Users المستخدمين

من الممكن تقسيم المستخدمين إلي أربعة أنواع :

1- مدير قواعد البيانات Database Administrator : وهي مهمة يقوم بها شخص متخصص أو فريق عمل وتشمل مسؤولياتهم :

- تحديد متطلبات قواعد البيانات المطلوبة من برمجيات وتجهيزات .
- العمل علي توفر النظام للمستخدم والتنسيق الدائم في عمليات الاستخدام.

قواعد البيانات (2)

- توفير الأمن والحماية لقواعد البيانات وتوزيع ومراقبة صلاحيات الاستخدام.
 - الرقابة المستمرة وضبط أداء النظام .
 - تصميم آليات المحافظة علي قواعد البيانات وإنشاءها لتوافق احتياجات المستخدمين .
- 2- مصمم قواعد البيانات Data base Designer : وهي مهمة يقوم بها شخص أو فريق متخصص ومن مهامه :
- تحديد البيانات الواجب تخزينها .
 - تحديد أفضل التراكيب البنائية للبيانات الواجب استخدامها .
 - الوصول إلي أقل درجة ممكنة من الأخطاء وإهدار المصادر .
 - تحديد طرق تخاطب المستخدمين مع قاعدة البيانات ويشمل ذلك تعريف محتويات وتصميم شاشات التخاطب وتوثيقها .
- 3- مبرمجو قواعد البيانات Database Programmers : وهو فريق عمل من المبرمجين وتتلخص أعمالهم في :
- تحويل تصميم قواعد البيانات إلي لغات قواعد بيانات مناسبة بحيث تكون جاهزة للتجميع وإدخالها إلي الحاسب .
 - تنفيذ الأنظمة والبرمجيات اللازمة لها والتأكد من صحتها وخلوها من الأخطاء وذلك باختبارها مفردة ومجمعة .
 - صياغة شاشات التخاطب مع المستخدم والإدخال والإخراج التي تحتاجها نظم قواعد البيانات وتنفيذها .
 - صياغة أنماط وأشكال التقارير المطلوبة للنظام وتنفيذها .
- 4- مستخدمو قواعد البيانات Database Users : وهم مجموعة من الموظفين لدي الشركة أو المؤسسة أو المصلحة والتي تستخدم أنظمة قواعد البيانات وتطبيقاتها في مجال محدد مثل موظفي البنوك وموظفي تسجيل الطلاب ... الخ .

الفصل الثاني

أساسيات لغة الاستفسارات الهيكلية

SQL

الأهداف:

إعطاء الطالب صورة موضحة عن أساسيات لغة الاستعلامات الهيكلية SQL وكيفية الدخول إلي برنامج SQL*PLUS وإدخال تعليمات SQL وكيفية كتابة جملة SELECT واستخدام كلا من WHERE, ORDER BY مع جملة SELECT

المحتويات:

- 1- مقدمة
- 2- الولوج إلي برنامج SQL*PLUS
- 3- إدخال تعليمات SQL
- 4- استخدام جملة Select، تعريف Alias
- 5- جملة الشرط Where
- 6- ترتيب النتائج باستخدام Order by Clause
- 7- مفهوم القيمة Null وتأثيراتها
- 8- استخدام محرر النصوص في برنامج SQL*PLUS

ماذا سنتعلم في هذا الفصل:

في نهاية هذا الفصل يكون الطالب قد اكتسب المهارات والمعارف التالية:

- (1) التفرقة بين قواعد البيانات ونظم إدارة قواعد البيانات
- (2) التعرف علي لغة الاستعلامات الهيكلية (SQL)
- (3) التفرقة بين SQL Statement ، SQL*Plus
- (4) استخدام جملة Select، تعريف Alias
- (5) استخدام جملة الشرط Where
- (6) ترتيب النتائج باستخدام Order by
- (7) استخدام القيمة NULL في التعبيرات الحسابية
- (8) استخدام محرر النصوص في برنامج SQL*PLUS

مقدمة

برامج قواعد البيانات من أوسع برامج الحاسب انتشارا ويمكن أن تستخدم سواء في المجالات التجارية أو الصناعية ومنها على سبيل المثال مجالات النظم المحاسبية والمالية ومجال شئون الأفراد والمشتريات، وتعتبر لغة الاستعلامات الهيكلية من أهم وأشهر اللغات المستخدمة في برمجة قواعد البيانات من حيث إنشاء ملفات قواعد البيانات واسترجاع البيانات منها بطرق مختلفة ومتعددة تلائم معظم احتياجات المبرمجين وتتميز بسهولة تركيبها وسهولة الفهم للجميع.

وتستخدم لغة الاستعلام مع الكثير من التطبيقات حيث صممت في الأصل لتستخدم لغات البرمجة الأخرى، ولقد استطاعت SQL أن تحقق طموحات معظم مطوري أنظمة قواعد البيانات حيث أنها لغة قياسية وفعالة في نفس الوقت لبناء ومعالجة قواعد البيانات، ولهذا السبب فإن أي شخص يريد الدخول عالم قواعد البيانات أن يتعلم أولا SQL، ويعتبر نظام أوراكل من أفضل نظم قواعد البيانات العلائقية وهو يتميز بإمكانية إدارة قواعد البيانات العلائقية وهو يتميز بإمكانية إدارة قواعد البيانات العلائقية وهو يتميز بإمكانية إدارة قواعد بيانات كبيرة جداً من المعلومات والتي قد تصل حتى تيرابايت من المعلومات وإمكانية التعامل مع عدد كبير جداً من المستخدمين بشكل متزامن بالإضافة إلي ميزات الأمان والوثوقية العالية جداً.

1- قواعد البيانات

1-1 المقصود بقاعدة البيانات

هي مجموعة مترابطة من البيانات يمكن استخدامها لاستخراج المعلومات التي تريدها أو هي عبارة عن تجمع لكمية كبيرة جداً من البيانات والمعلومات وعرضها بطريقة تسهل من الاستفادة منها فعلي سبيل المثال فإن دليل الهاتف هو أبسط قاعدة بيانات يمكن أن تتعامل معها فيمكنك تسجيل أسماء أصدقائك وأرقام التليفونات الخاصة بهم حتى يمكنك فيما بعد أن تبحث عن اسم أحد الأصدقاء بإدخال رقم الإشتراك أو العنوانوهكذا.

2-1 نظم إدارة قواعد البيانات

نظم إدارة قواعد البيانات (DBMS) هي مجموعة من البرامج الجاهزة التي تقوم بتنفيذ جميع الوظائف والمهام المطلوبة من قاعدة البيانات فعلي سبيل المثال بعد إدخال بيانات العاملين إلي قاعدة بيانات شؤون العاملين فإنك قد تحتاج إلي ترتيب أسماء العاملين أبجدياً فإن مثل هذا العمل يطلق عليه إدارة قاعدة البيانات، ويتكون نظام إدارة قواعد البيانات من مجموعة من البرامج والملفات التي تتشابك مع بعضها لحل مشكلة أو تحويل نظام يدوي إلي نظام يعمل بالحاسب مثل تحويل حسابات المخازن من أنظمة ودفاتر يومية إلي نظام وملفات تستخدم بواسطة الحاسب الآلي.

3-1 أنواع قواعد البيانات

- قواعد بيانات هرمية (Hierarchy Database)
- قواعد بيانات شبكية (Network Database)
- قواعد بيانات علائقية (Relational Database)

يقتصر استخدام كلاً من قواعد البيانات الهرمية والشبكية علي الحاسبات الكبيرة، أما قواعد البيانات العلائقية Relational Database فإنها أكثر استخداماً وشهرة مع الحاسب

2- لغة الاستفسارات الهيكلية (SQL)

تمتلك لغة SQL (اختصاراً للكلمات Structured Query Language) من إدارة قواعد البيانات بشكل كامل وإجراء جميع العمليات القياسية كإنشاء الجداول وتعبئتها بالبيانات، أو إجراء الاستعلامات عليها وكذلك الربط بين الجداول المختلفة، ولكن كيف يمكننا عمل ذلك بها؟

كما وضعنا سابقاً فإن قاعدة البيانات عبارة عن مكان أو مستودع كبير لتخزين البيانات المختلفة، تريحك قاعدة البيانات من عناء تخزين بياناتك في ملفات منفصلة وكما أنك تحتاج إلى إجراء عمليات البحث والتصنيف لهذه البيانات عن طريق خوارزميات البحث المعقدة فتسهل لك ذلك بأنها تعطيك واجهة سهلة للتعامل مع البيانات، بحيث لا تحتاج إلى كل هذا وتكون البيانات في قاعدة البيانات مخزنة في عدة جداول Tables وكما نعلم فالجدول يتكون من صفوف Rows وأعمدة Columns هكذا:

الجدول السابق يتكون من ثلاثة صفوف وثلاثة أعمدة، وفي قواعد البيانات فإننا نسمي الصفوف بالسجلات Records ونسمي الأعمدة بالحقول Fields، فما هي الحقول وما هي السجلات

يقوم الحقل الواحد في الجدول بتخزين معلومة معينة، فمثلاً عندما يكون لدينا قاعدة بيانات بأرقام الهواتف فإننا سنحتاج إلى حقلين (أو عامودين) واحد للاسم والثاني لرقم الهاتف، أما السجلات (أو الصفوف) فيحتوي كل منها على مجموعة من الحقول، فترى بأن السجل الواحد في مثالنا يحتوي على معلومتين مختلفتين هما الاسم ورقم الهاتف، لذلك يمكننا القول بأن قاعدة البيانات تتكون من الجداول والجدول تتكون من السجلات والسجلات تتكون من الحقول، وكل حقل يحتوي على معلومة واحدة، أي أن الحقل هو أصغر وحدة قاعدة البيانات.

1-2 عبارات SQL

تنقسم عبارات SQL إلى ثلاث فئات رئيسية وهي كالتالي:

1-1-2 لغة تعريف البيانات DDL

هي مجموعة من أوامر SQL تستطيع من خلالها إنشاء وتعريف الكائنات في قاعدة البيانات، وتقوم هذه الأوامر بإنشاء أو إسقاط أو تغيير كائن قاعدة البيانات، وهي اختصاراً للكلمات (Data Definition Language)، ومن أهم أوامرها Create, Alter, Drop, Rename, Truncate

2-1-2 لغة معالجة البيانات DML

هي مجموعة من العبارات التي تستطيع من خلالها معالجة البيانات الموجودة في قاعدة البيانات من إدراج البيانات وتحديثها أو تحديد أو حذف البيانات، وهي اختصاراً للكلمات (Data Manipulation Language)، ومن أهم أوامرها Insert, Update, Delete

3-1-2 لغة التحكم والصلاحيات DCL

هي مجموعة من أوامر SQL تستطيع من خلالها منح أو سحب إمتيازات استخدام كائن قاعدة البيانات، وهي اختصاراً للكلمات، وهي اختصاراً للكلمات (Data Control Language)، ومن أهم أوامرها Grant, Revoke

وإيجازاً للقول فإن الشكل التالي يوضح الفئات الرئيسية لعبارات SQL

INSERT UPDATE DELETE	لغة معالجة البيانات
CREATE ALTER DROP RENAME TRUNCATE	لغة تعريف البيانات
GRANT REVOKE	لغة التحكم والصلاحيات
COMMIT ROLLBACK SAVEPOINT	لغة معالجة البيانات Transaction control

3 - استخدام SQL

يعتبر برنامج SQL أحد منتجات شركة Microsoft، وهو واجهة التخاطب مع قاعدة البيانات، ويستخدم في إنشاء قواعد البيانات ومنح السماحيات، وإنشاء الجداول وتعديلها واسترجاع البيانات من الجداول.

يستخدم في إنشاء نظام لإدارة قواعد البيانات والتعامل مع المعلومات فيها وتنفيذ الأوامر التي يحتاجها مستخدم قاعدة البيانات تشمل هذه الأوامر:

—تنظيم وتعديل البيانات بالإضافة، والحذف، والتعديل والأرشفة

—البحث في قواعد البيانات والوصول إلى المعلومات

—تستخدم لغة البرمجة SQL في التأكد من دقة المعلومات وحماية البيانات

—تستخدم في التحكم الإذونات، و الصلاحيات للمستخدمين الذين يتعاملون مع قواعد البيانات.

4- جملة Select

الأمر Select يمكنك من استعادة البيانات من داخل قاعدة البيانات، وتأخذ جملة Select الشكل التالي

```
SELECT [DISTINCT] {*, column
[alias], ...}
FROM table;
```

4-1 اختيار كافة الأعمدة

تمثل عبارة * Select طريقة مختصرة لإخبار SQL بأننا نريد إظهار جميع

أعمدة الجدول، فعلي سبيل المثال

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

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7788	SCOTT	ANALYST	7566	19/04/87	3000		20
7839	KING	PRESIDENT		17/11/81	5000		10
7844	TURNER	SALESMAN	7698	08/09/81	1500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3000		20
7934	MILLER	CLERK	7782	23/01/82	1300		10

تم اختيار 14 صف

SQL>

- 1) Select: تحدد ماهي الأعمدة المراد إظهارها .
- 2) From: تحدد المكان الذي نريد استعادة البيانات منه (جدول أو أكثر)
- 3) الفاصلة المنقوطة: تخبر SQL بأن العبارة كاملة وجاهزة للتنفيذ .
- 4) الأقواس [] تشير علي أن ما بداخل هذه الأقواس اختياريًا .

2-4 اختيار أعمدة محددة

يمكننا اختيار أعمدة محددة، وهو أمر في غاية السهولة حيث تقوم بكتابة أسماء الأعمدة

بعد عبارة Select، مع وضع فاصلة بين أسماء الأعمدة، ثم تحدد الجدول بعد From

```
SQL> SELECT empno, ename, job
        2 FROM      emp;
```

مثال 2

EMPNO	ENAME	JOB
7369	SMITH	CLERK
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7566	JONES	MANAGER
7654	MARTIN	SALESMAN
7698	BLAKE	MANAGER
7782	CLARK	MANAGER
7788	SCOTT	ANALYST
7839	KING	PRESIDENT
7844	TURNER	SALESMAN
7876	ADAMS	CLERK
7900	JAMES	CLERK
7902	FORD	ANALYST
7934	MILLER	CLERK

تم اختيار 14 صف

✓ تلميح لا تنس إنهاء جملة Select بالفاصلة المنقوطة لكي تخبر SQL بأن الجملة

كاملة وجاهزة للتنفيذ .

✓ تظهر الأعمدة المطلوبة بدءاً من اليسار إلي اليمين بنفس ترتيب كتابتها في عبارة

Select

```
SQL> SELECT deptno, dname, loc
      2 FROM      dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

3-4 التعبيرات الحسابية (Arithmetic Expressions)

- ضع التعبيرات الحسابية بعد عبارة Select كأي عمود تريد استعادة بياناته، حيث تستخدم التعبيرات الحسابية مع تلك الأعمدة التي نوع معطياتها عددي مثل الأرقام والتاريخ.

- يجب عليك الانتباه لأسبقية المعاملات (Operator Precedence) في التعبيرات الحسابية وفيما يلي بيان بأسبقية المعاملات في التعبيرات الحسابية

(1) ينفذ أولاً الضرب أو القسمة أولاً، ومن اليسار إلى اليمين

(2) بعد ذلك ينفذ الجمع أو الطرح لاحقاً، ومن اليسار إلى اليمين.

تلميح

✓ عند استخدام التعبيرات الحسابية يمكنك تغيير تسلسل أسبقية المعاملات الحسابية

باستخدام الأقواس (Parentheses)، حيث تأخذ الأقواس الأسبقية في التعبيرات

الحسابية.

1-3-4 أمثلة علي التعبيرات الحسابية

```
SQL> SELECT ename, sal, sal+500
2 FROM emp;
```

في هذا المثال يقوم باستخدام معامل الجمع لزيادة مرتبات جميع الموظفين ب 500 ريال، ويقوم بعرض الناتج في عمود جديد .

ENAME	SAL	SAL+500
SMITH	800	1300
ALLEN	1600	2100
WARD	1250	1750
JONES	2975	3475
MARTIN	1250	1750
BLAKE	2850	3350
CLARK	2450	2950
SCOTT	3000	3500
KING	5000	5500
TURNER	1500	2000
ADAMS	1100	1600
JAMES	950	1450
FORD	3000	3500
MILLER	1300	1800

14 row selected

```
SQL> SELECT ename, sal, 12*sal+500
2 FROM emp;
```

تتجاهل SQL*Plus المسافات Blank Spaces قبل وبعد ✓

تلميح

المعاملات الحسابية .

في هذا المثال سيقوم طبقاً لأسبقية المعاملات كما وضعنا سابقاً بحساب عملية الضرب

أولاً وهي $12*sal$ ، ثم إجراء عملية الجمع ثانياً وهي إضافة 500 ريال

ENAME	SAL	12*SAL+500
SMITH	800	10100
ALLEN	1600	19700
WARD	1250	15500
JONES	2975	36200
MARTIN	1250	15500
BLAKE	2850	34700
CLARK	2450	29900
SCOTT	3000	36500
KING	5000	60500
TURNER	1500	18500
ADAMS	1100	13700
JAMES	950	11900
FORD	3000	36500
MILLER	1300	16100

تم اختيار 14 صف

SQL>

✓ في المثال السابق كنا نريد زيادة مرتبات جميع الموظفين ب 500 ريال ثم حساب المرتب السنوي بعد عملية الزيادة فكيف يمكننا ذلك . . . ؟

تلميح

✓ كما تعلمنا سابقاً يمكنك تغيير تسلسل أسبقية المعاملات الحسابية باستخدام الأقواس (Parentheses)، حيث تأخذ الأقواس الأسبقية في التعبيرات الحسابية، ويتم ذلك علي النحو التالي

```
SQL> SELECT ename, sal, 12*(sal+500)
      2 FROM emp;
```

ENAME	SAL	12*(SAL+500)
SMITH	800	15600
ALLEN	1600	25200
WARD	1250	21000
JONES	2975	41700
MARTIN	1250	21000
BLAKE	2850	40200
CLARK	2450	35400
SCOTT	3000	42000
KING	5000	66000
TURNER	1500	24000
ADAMS	1100	19200
JAMES	950	17400
FORD	3000	42000
MILLER	1300	21600

تم اختيار 14 صف

SQL> |

5-تعريف A column Alias

يمكننا إعادة تسمية رؤوس الأعمدة باستخدام a column alias، بمعنى إبدال الاسم الفعلي للحقل باسم اعتباري يتم تحديده بواسطة المستخدم ويكون ذلك مفيداً وخاصة عند التعامل مع التعبيرات الحسابية، وفيما يلي عدة اعتبارات لابد من مراعاتها عند إعادة تسمية رؤوس الأعمدة

(1) يتم كتابة alias في جملة Select يلي اسم العمود أو التعبير الحسابي المراد إعادة تسميته مع ترك مسافة.

قواعد البيانات (2)

(2) يتم استخدام as بين اسم العمود وال alias، ويكون ذلك اختياريًا.

(3) إذا احتوي alias علي مسافات أو أحرف وعلامات خاصة مثل (# or\$)

من وضعة بين علامتي تنصيص (" double quotation)

```
SQL> SELECT ename, sal AS Salary
2 FROM emp;
```

مثال 1

ENAME	SALARY
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	1250
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950
FORD	3000
MILLER	1300

تم اختيار 14 صف

SQL>

تلميح

✓ في المثال السابق تم إعادة تسمية عمود Sal في جدول emp ب salary

✓ تم وضع كلمة AS قبل alias name، وكما تعلمنا أن ذلك اختياريًا.

✓ يتم ظهور alias name بحالة أحرف كبيرة Uppercase

مثال 2

```
SQL> SELECT ename, sal*12 "annual salary"
2 FROM emp;
```

ENAME	Annual Salary
SMITH	9600
ALLEN	19200
WARD	15000
JONES	35700
MARTIN	15000
BLAKE	34200
CLARK	29400
SCOTT	36000
KING	60000
TURNER	18000
ADAMS	13200
JAMES	11400
FORD	36000
MILLER	15600

تم اختيار 14 صف

في هذا المثال تم إعادة تسمية التعبير الحسابي `sal*12` بـ `Annual Salary`، كما نلاحظ تم وضع "Annual Salary" بين علامتي تنصيص لأنه يحتوي مسافات `Spaces`.

6- جملة الشرط Where Clause

كان اختيارنا سابقاً للأعمدة يجري علي كافة الصفوف التي يحتويها الجدول، أردنا إظهار صفوفاً تحتوي علي قيماً معينة، كإظهار الموظفين الذين تتجاوز مرتباتهم 3000 ريال أو الذين يقطنون مدينة ما، لفعل مثل ذلك سنحتاج إلي وضع كلمة `Where` ضمن عبارة `Select`، يأخذ الاختيار الشرطي الشكل التالي

```
SELECT      [DISTINCT] {*, column [alias], ...}
FROM        table
[WHERE      condition(s)];
```

تلميح

✓ تأتي جملة `Where Clause` بعد جملة `From Clause`

✓ يلي `Where` شرط أو عدة شروط

✓ توجه عبارة `Where` الأمر إلي أوراكل بالبحث في قاعدة البيانات واستعادة

الصفوف التي تتوافق مع شرط البحث .

مثال 1

```
SQL> SELECT *
      2 FROM   emp
      3 WHERE  ename='tarek';
```

في هذا المثال نريد استرجاع جميع الأعمدة بجدول `emp`، ولكن بشرط أن يكون

اسم الموظف `tarek`

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	tarek	MANAGER	7839	01/01/99	5000	3000	30

مثال 2

```
SQL> SELECT *
      2 FROM emp
      3 WHERE hiredate='23-JAN-82';
```

في هذا المثال نريد استرجاع جميع الأعمدة بجدول emp، ولكن بشرط أن يكون تاريخ تعيين الموظف هو '23-JAN-82'

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	tarek	MANAGER	7839	01/01/99	5000	3000	30

تلميح

✓ الأحرف النصية Character Strings، والتاريخ في جملة Where

لابد من وضعها بين علامتي تنصيص (' ') Single.

quotation

مثال 3

```
SQL> SELECT ename, job, deptno
      FROM emp
      WHERE ename = 'MILLER';
```

ENAME	JOB	DEPTNO
MILLER	CLERK	10

SQL>

1-6 استخدام عوامل المقارنة (Comparison Operators)

تمحنا SQL ستة عوامل للمقارنة نستطيع استخدامها مع Where، كما هو مبين بالجدول التالي

المعامل Operator	المعني Meaning
=	يساوي
> or <	أكبر من أو أقل من
>=	أكبر من أو يساوي
<=	أقل من أو يساوي
<> Or !=	لا يساوي

مثال 1

```
SQL> SELECT ename, job, sal
2 FROM emp
3 WHERE sal >= 3000;
```

ENAME	JOB	SAL
SCOTT	ANALYST	3000
KING	PRESIDENT	5000
FORD	ANALYST	3000

SQL> |

في هذا المثال قد استخدمنا أحد عوامل المقارنة في جملة Where، حيث يتم استعادة

ename, job, sal عندما يكون sal أكبر من أو يساوي 3000 ريال.

مثال 2

```
SQL> SELECT ename, job, sal
2 FROM emp
3 WHERE sal <> 3000;
```

ENAME	JOB	SAL
SMITH	CLERK	800
ALLEN	SALESMAN	1600
WARD	SALESMAN	1250
JONES	MANAGER	2975
MARTIN	SALESMAN	1250
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
KING	PRESIDENT	5000
TURNER	SALESMAN	1500
ADAMS	CLERK	1100
JAMES	CLERK	950
MILLER	CLERK	1300

12 row selected

6-2 استخدام عوامل المقارنة الأخرى

(Other Comparison Operator)

تتضمن SQL أربعة معاملات أخرى بالإضافة للمعاملات التي الإشارة إليها في النقطة

السابقة يمكن استخدامها مع جملة Where، كما هو مبين بالجدول التالي

المعامل Operator	المعني Meaning
BETWEEN....AND.....	بين قيمتين
IN(list)	تتيح لك اختيار أي قيمة من List
LIKE	يشابه أو يماثل
IS NULL	فارغاً

6-2-1 استخدام معامل Between

يمكنك إجراء عمليات البحث داخل نطاق معين من القيم وذلك باستخدام معامل Between

مع جملة Where، فعلي سبيل المثال قد ترغب في استعادة بيانات جميع الموظفين الذين

يتقاضون راتب ما بين 2000، 5000 يمكنك القيام بهذا وذلك بإجراء المثال التالي

```
SQL> SELECT *
      FROM emp
      WHERE sal BETWEEN 2000 AND 5000;
```

قيمة أدنى

قيمة أعلى

في هذا المثال تم استخدام معاملة Between مع جملة Where لاستعادة السجلات التي تقع نطاق معين من القيم

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02/04/81	2975		20
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7788	SCOTT	ANALYST	7566	19/04/87	3000		20
7839	KING	PRESIDENT		17/11/81	5000		10
7902	FORD	ANALYST	7566	03/12/81	3000		20

تم اختيار 6 صف

SQL> |

2-2-6 استخدام معاملة IN

يمكنك معاملة IN القيام بعمليات البحث عن بنود داخل إحدى القوائم، حيث يمكنك استرجاع السجلات التي قيم عمود فيها (محدد في جملة Where) تساوي إحدى القيم المحددة ضمن مجموعة، فعلي سبيل المثال لإظهار كافة السجلات الخاصة بالموظفين ذوي الأرقام 7369 أو 7499 أو 7521 يتم إجراء المثال التالي

```
SQL> SELECT *
      2 FROM emp
      3 WHERE empno IN (7369, 7499, 7521);
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7369	SMITH	CLERK	7902	17/12/80	800		20

SQL>

قواعد البيانات (2)

في هذا المثال السابق تم استرجاع السجلات التي فيها العمود empno يساوي إما

7369 أو 7499 أو 7369، فنجد أن IN تكافئ سلسلة مقارنات مربوطة بالمعامل Or،

فيمكنك إجراء المثال السابق باستخدام or علي النحو التالي

```
SQL> SELECT *  
      2 FROM emp  
      3 WHERE empno=7369 or empno=7499  
      or empno=7369;
```

مثال 2

```
SQL> SELECT *  
      2 FROM emp  
      3 WHERE ename IN ('tarek', 'ayman', 'said');
```

في هذا المثال سيتم استعادة كافة السجلات الخاصة بالموظفين الموجودين في القائمة التي

تلي معامل IN، حيث يتم استعادة جميع السجلات من جدول emp عندما يكون ename

يساوي إما 'tarek' Or 'ayman' Or 'said'

تلميح

وفيما يلي عدة اعتبارات لابد من مراعاتها عند إعادة استخدام معامل IN مع جملة Where

✓ وضع قيم المجموعة بين قوسين .

✓ فصل قيم المجموعة عن بعضها بالفاصلة .

✓ يمكن وضع القيم بأي ترتيب .

✓ عندما تكون قيم المجموعة حرفية أو تاريخ لابد من وضعها بين علامتي

تنصيص (' ') .

✓ يمكن استخدام NOT IN لعكس الاختيار

3-2-6 استخدام معامـل Like

يمكنك استخدام Like مع جملة Where لاستعادة واسترجاع السجلات التي قيم عمود فيها تشابه سلسلة أحرف نبحت عنها، ليس هذا فحسب بل يمكننا من استرجاع السجلات التي تكون قيم عمود فيها مطابقة جزئياً لمجموعة أحرف نبحت عنها

مثال 1

لاسترجاع جميع الأسماء التي تبدأ بحرف (A) يتم كتابة الكود التالي

```
SQL> SELECT  ename
      2 FROM    emp
      3 WHERE   ename LIKE 'A%';
```

ENAME

ALLEN
ADAMS

للبحث عن الأسماء التي تشتمل علي أحرف are يتم ذلك من خلال كتابة الكود التالي:

```
SQL> SELECT  *
      2 FROM    emp
      3 WHERE   ename LIKE '%are%';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
-----	-----	-----	-----	-----	-----	-----	-----
1	tarek	MANAGER	7839	01/01/99	5000	3000	30

مثال 3

لاسترجاع بيانات الاسم والوظيفة والمرتب للموظفين الذي تم تعيينهم خلال عام 1982

من شهر يناير إلي شهر ديسمبر يتم كتابة الكود التالي

```
SQL> SELECT  ename,job,sal,hiredate
2 FROM    emp
3 WHERE   HIREDATE LIKE '%1982';
```

ENAME	JOB	SAL	HIREDATE
MILLER	CLERK	1300	23/01/82

مثال 4

لاسترجاع جميع الأسماء التي يكون حرف (A) فيها يأخذ الترتيب الثاني يتم كتابة الكود

التالي:

```
SQL> SELECT  ename
2 FROM    emp
3 WHERE   ename LIKE '_A%';
```

ENAME	
WARD	✓
MARTIN	✓
JAMES	✓

تلميح

✓ تستخدم LIKE مع الأعمدة التي يكون نوع بياناتها حرفي .

✓ % اختصار لأي تتابع من سلسلة أحرف غير معينة مؤلفة من أي قيمة

وبأي طول

✓ _ اختصار لأي حرف وحيد

4-2-6 استخدام معامل IS NULL

قبل استخدام معامل IS NULL مع Where لابد من الإجابة علي ماهي القيمة

NULL ؟ تستخدم القيمة NULL للإشارة إلي عمود لا يحتوي علي بيانات، حيث لا تعني القيمة

Blank Space NULL صفر أو مسافة

مثال 1

لاسترجاع جميع بيانات الموظفين الذين لا يتقاضون حوافز Commission يتم

كتابة الكود التالي

```
SQL> SELECT *
      2 FROM      emp
      3 WHERE      comm IS NULL;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7566	JONES	MANAGER	7839	02/04/81	2975		20
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7788	SCOTT	ANALYST	7566	19/04/87	3000		20
7839	KING	PRESIDENT		17/11/81	5000		10
7876	ADAMS	CLERK	7788	23/05/87	1100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3000		20
7934	MILLER	CLERK	7782	23/01/82	1300		10

تم اختيار 10 صف

3-6 استخدام عوامل المقارنة الأخرى (Logical Operators)

تتضمن SQL ثلاثة معاملات منطقية بالإضافة للمعاملات التي الإشارة إليها في سابقاً

يمكن استخدامها مع جملة Where، كما هو مبين بالجدول التالي

المعامل Operator	المعني Meaning
AND	و
OR	أو
NOT	النفى

1-3-6 استخدام معامـل AND

يستخدم AND مع Where لاستعادة السجلات التي تحقق كافة الشروط المحددة في

جملة Where، حيث يعني AND أنه لاستعادة السجلات لابد من تحقق كافة الشروط.

مثال 1

لاسترجاع جميع بيانات الموظفين بالإدارة رقم 10 الذين يتقاضون مرتباً أكبر من 3000

ريال يتم كتابة الكود التالي:

```
SQL> SELECT *
      2 FROM emp
      3 WHERE sal >3000
      4 AND deptno=10;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17/11/81	5000		10

مثال 2

لاسترجاع جميع بيانات محلي النظم الذين يتقاضون مرتباً أكبر من 3000 ريال يتم كتابة

الكود التالي

```
SQL> SELECT ename, job, sal
      2 FROM emp
      3 WHERE sal>1000
      4 AND job='analyst';
```

ENAME	JOB	SAL
SCOTT	ANALYST	3000
FORD	ANALYST	3000

2-3-6 استخدام معام OR

نستخدم معام Or مع Where لاستعادة السجلات التي تحقق شرطاً واحداً من عدة

شروط، حيث يعني المعامل Or أن استعادة البيانات تتم إذا تحقق أي شرط من الشروط

مثال 1

لاسترجاع بيانات الموظفين (الاسم، الوظيفة، المرتب) يتقاضون مرتباً أكبر من

3000 ريال أو الوظيفة analyst، يتم كتابة الكود التالي

```
SQL> SELECT ename, job, sal
2 FROM emp
3 WHERE sal>3000
4 OR job='ANALYST';
```

ENAME	JOB	SAL
SCOTT	ANALYST	3000
KING	PRESIDENT	5000
FORD	ANALYST	3000

تلميح

✓ تم عرض بيانات الموظفين الذين يتقاضون مرتباً أكبر من 3000 أو الوظيفة

Analyst

3-3-6 استخدام معام NOT

استخدام المعامل NOT يمكننا من استثناء سجلات معينة من اختيارنا، حيث يمكننا من

استرجاع السجلات المعاكسة لاختيارنا، فعلي سبيل المثال لاختيار كافة السجلات من جدول

emp باستثناء التي يكون ال Commission بها قيمته NULL يتم كتابة الكود التالي

```
SQL> SELECT ename, job
2 FROM emp
3 WHERE comm IS NOT NULL;
```

مثال 1

ENAME	JOB
-----	-----
ALLEN	SALESMAN
WARD	SALESMAN
MARTIN	SALESMAN
TURNER	SALESMAN

مثال 2

```
SQL> SELECT ename, job
2 FROM emp
3 WHERE job NOT IN
('ANALYST', 'ACCOUNTANT', 'ENGINEER');
```

لاسترجاع بيانات الموظفين (الاسم، الوظيفة، المرتب)، باستثناء الوظيفة
analyst, accountant, engineer يتم كتابة الكود التالي:

ENAME	JOB
-----	-----
SMITH	CLERK
ALLEN	SALESMAN
WARD	SALESMAN
JONES	MANAGER
MARTIN	SALESMAN
BLAKE	MANAGER
CLARK	MANAGER
KING	PRESIDENT
TURNER	SALESMAN
ADAMS	CLERK
JAMES	CLERK
MILLER	CLERK

12 ROW selected

تلميح

يتم استخدام معامل NOT مع المعاملات الأخرى مثل
(BETWEEN, LIKE, and NULL)

✓ وإليك التلميح التالي:

```
.... Where sal NOT BETWEEN 3000 AND 4000
.... Where ename NOT LIKE '%T%'
.... Where job NOT IN ('analyst','accountant')
.... Where comm IS NOT NULL
```

4-6 قواعد أسبقية المعاملات Rules of Precedence

الجدول التالي يوضح قواعد أسبقية المعاملات

الترتيب Order Evaluated	المعامل Operator
1	جميع عوامل المقارنة
2	NOT
3	AND
4	OR

تلميح

كما تعلمنا سابقاً يمكنك تغيير تسلسل أسبقية المعاملات باستخدام

الأقواس (Parentheses)، حيث تأخذ الأقواس الأسبقية في التعبيرات الحسابية

مثال 1

```
SQL>SELECT ename, job, sal
2 FROM emp
3 WHERE job='ANALYST'
4 OR job='MANAGER'
5 AND sal >3000;
```

الثاني

الأول

Job تكون MANAGER و Sal أكبر من 3000 ريال

✓ الشرط الأول هو

تلميح

job تكون ANALYST

✓ الشرط الثاني هو

ENAME	JOB	SAL
SCOTT	ANALYST	3000
FORD	ANALYST	3000

7- ترتيب النتائج باستخدام Order by

ترتب السجلات باستخدام Order by وفقاً لقيم العمود الذي نحدده ترتيباً تصاعدياً أو تنازلياً، ويمكن الترتيب بناءً على قيم عمود أو عدة أعمدة، وعند كتابة جملة Order by يجب إتباع الآتي:

✓ Order by تتبع جملة from {table name} مباشرة في حالة عدم

وجود Where {Condition(s)}

✓ Order by تتبع جملة Where {Condition(s)} مباشرة.

✓ يوجد لدينا خياران عند استخدام Order by

(1) Order by {column name} ASC: يسمح هذا الخيار

بالترتيب التصاعدي

(2) Order by {column name} desc: يسمح هذا الخيار

بالترتيب التنازلي.

✓ يمكن الترتيب على أساس عدة أعمدة، ويجب الفصل بين أسماء الأعمدة بفاصلة

(,)

✓ يوجد لدينا طريقتان لكتابة جملة Order by وهما

1. عن طريق اسم العمود

2. عن طريق رقم العمود النسبي (موقعه في جملة SELECT)

7-1 الترتيب التصاعدي باستخدام Order by

مثال 1

لاستعادة بيانات جميع الموظفين مرتبة علي أساس تاريخ التعيين، يتم كتابة الكود التالي

```
SQL> SELECT *
      2 FROM      emp
      3 ORDER BY hiredate asc;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975		20
7698	BLAKE	MANAGER	7839	01/05/81	2850		30
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7844	TURNER	SALESMAN	7698	08/09/81	1500	0	30
7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
7839	KING	PRESIDENT		17/11/81	5000		10
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3000		20
7934	MILLER	CLERK	7782	23/01/82	1300		10
7788	SCOTT	ANALYST	7566	19/04/87	3000		20
7876	ADAMS	CLERK	7788	23/05/87	1100		20

تم اختيار 14 صف

SQL> |

مثال 2

```
SQL>SELECT *
      2 FROM      emp
      3 ORDER BY hiredate DESC;
```

7-2 الترتيب التنازلي باستخدام Order by

لاستعادة بيانات جميع الموظفين مرتبة تنازلياً علي أساس تاريخ التعيين، يتم كتابة الكود

التالي

```
SQL> SELECT *
      2 FROM      emp
      3 ORDER BY  hiredate desc;
```

الترتيب الافتراضي في جملة Order by هو الترتيب التصاعدي مللم نصرح

تلميح

عكس ذلك بالكلمة DESC

✓ يتم عرض القيم null مؤخراً عند الترتيب التصاعدي، والعكس يتم ظهور القيم

null أولاً عند الترتيب التنازلي.

مثال 3

يمكنك إجراء الترتيب باستخدام Order by علي أساس قيم أعمدة ليست بجملة

SELECT، كما بهذا المثال

```
SQL> SELECT  ename, sal
2 FROM      emp
3 ORDER BY  deptno, sal DESC;
```

ENAME	SAL
KING	5000
CLARK	2450
MILLER	1300
SCOTT	3000
FORD	3000
JONES	2975
ADAMS	1100
SMITH	800
BLAKE	2850
ALLEN	1600
TURNER	1500
WARD	1250
MARTIN	1250
JAMES	950

14 row selected

يمكننا إجراء الترتيب باستخدام Order by علي أساس قيم أكثر من

تلميح

عمود .

3-7 الترتيب باستخدام A Column Alias

يمكننا إجراء الترتيب في جملة Order by باستخدام A column Alias كما

بالمثال التالي:

مثال

```
SQL> SELECT empno, ename, sal*12 total
2 FROM emp
3 ORDER BY total;
```

EMPNO	ENAME	TOTAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7521	WARD	15000
7654	MARTIN	15000
7934	MILLER	15600
7844	TURNER	18000
7499	ALLEN	19200
7782	CLARK	29400
7698	BLAKE	34200
7566	JONES	35700
7788	SCOTT	36000
7902	FORD	36000
7839	KING	60000

14 row selected

8- القيمة Null وتأثيراتها

تستخدم القيمة null للإشارة إلى عمود لا يحتوي على بيانات بمعنى وجود قيم فارغة داخل الحقول ومن أشهر عيوب القيمة null صعوبة إجراء أي عمليات حسابية على سجلات تحتوي بعضها على قيم فارغة

8-1 استخدام القيمة Null

(1) تستخدم القيمة null للإشارة إلى عمود لا يحتوي على بيانات، ولا تعني القيمة

blank space أو المسافة الفارغة null

(2) عند استخدام القيمة null في التعبيرات الحسابية يكون الناتج null

مثال 1

```
SQL> SELECT ename, sal, comm
2 FROM emp;
```

في هذا المثال نجد أن العمود comm يحتوي قيما من النوع null

ENAME	SAL	COMM
SMITH	800	
ALLEN	1600	300
WARD	1250	500
JONES	2975	
MARTIN	1250	1400
BLAKE	2850	
CLARK	2450	
SCOTT	3000	
KING	5000	
TURNER	1500	0
ADAMS	1100	
JAMES	950	
FORD	3000	
MILLER	1300	
14 row selected		


```
SQL> select ename,sal,comm,sal+comm
2 from emp
3 WHERE ename='SMITH';
```

ENAME	SAL	COMM	SAL+COMM
SMITH	800		

في هذا المثال تم تناول العمليات الحسابية علي القيم null، حيث أن الcomm الخاص بالموظف SMITH يأخذ القيمة null، وكما تعلمنا أن ناتج أي عملية حسابية طرفها القيمة null يكون null

9- استخدام محرر النصوص في البرنامج SQL*PLUS

بعد إدخال تعليمة ما في محث SQL فإنها تبقى في الذاكرة المؤقتة (BUFFER)، نقوم بكتابة التعليمة List أو الحرف L لإظهار آخر تعليمة أرسلت إلي ORACLE، ويمكن إعادة تنفيذها بإدخال الكلمة RUN أو الحرف (r)، ويمكنك تعديل التعليمات التي تم إدخالها إلي الذاكرة المؤقتة ، حيث يمتلك sql*plus سلسلة من التعليمات التي تسمح للمستخدم بتعديل محتوى التعليمات المخزنة في الذاكرة المؤقتة كما هو موضح في الجدول التالي :

التعليمة	الوظيفة	مثال
L	إظهار محتوى الذاكرة المؤقتة	l
I	السماح للمستخدم بالبدء بإدخال نص بعد السطر الحالي	I
run أو /	تنفذ التعليمة المخزنة حالياً في الذاكرة المؤقتة	/ RUN
Save filename	حفظ التعليمة الحالية في ملف	Save c:\t.sql

get c:\t.sql	تحميل محتوى ملف ما إلي الذاكرة المؤقتة	Get filename
start c:\t.sql	تحميل الملف إلي الذاكرة المؤقتة وتنفيذ التعليمات الموجودة به	Start filename أو @filename
edit c:\t.sql أو edit أو ed	تشغل محرر النصوص الخارجي مثل المفكرة، فإذا تم تحديد إسم الملف يتم فتح محرر النصوص متضمنا الملف	Edit filename

الأسئلة

المجموعة الأولى:

ضع علامة (/) أمام العبارة الصحيحة وعلامة (x) أمام العبارة الخاطئة مع تصحيح الخطأ.

- (1) لغة DDL تستطيع من خلالها معالجة البيانات الموجودة في قاعدة البيانات من إدراج البيانات وتحديثها أو تحديد أو حذف البيانات.
- (2) تمثل عبارة `Select *` طريقة مختصرة لاسترجاع أعمدة محددة من الجدول.
- (3) تظهر الأعمدة المطلوبة بدءاً من اليسار إلي اليمين بنفس ترتيب كتابتها في عبارة `Select`.
- (4) تتجاهل `SQL*Plus` المسافات `Blank Spaces` قبل وبعد المعاملات الحسابية.
- (4) إذا احتوي `alias` علي مسافات أو أحرف وعلامات خاصة لا بد من وضعة بين علامتي تنصيص (" ").
- (5) تأتي جملة `Where` قبل جملة `From`.
- (6) يلي `Where` شرط أو عدة شروط.
- (7) نستخدم معامل `Between` مع جملة `Where` لاستعادة السجلات التي تقع نطاق معين من القيم.
- (8) يمكنك معامل `LIKE` القيام بعمليات البحث عن بنود داخل إحدى القوائم، بينما يمكنك استخدام `IN` استعادة واسترجاع السجلات التي قيم عمود فيها تشابه سلسلة أحرف نبحث عنها

9) تستخدم معامل LIKE مع الأعمدة التي يكون نوع بياناتها رقمي.

10) يعني معامل AND أنه لاستعادة السجلات لابد من تحقق كافة الشروط

المجموعة الثانية:

أسئلة لتدريب الطلاب على أسئلة الاختيار المتعدد:

(1) المصطلح DDL هو:.

1. اختصار data definition language.
2. بناء SQL المستخدم لتعريف البيانات في قاعدة البيانات.
3. بناء SQL المستخدم لمعالجة البيانات في قاعدة البيانات.
4. 1&2 .

(2) المصطلح DML هو:.

1. بناء SQL المستخدم لمعالجة البيانات في قاعدة البيانات .
2. اختصار ل Data Manipulation Language .
3. بناء SQL المستخدم لتعريف البيانات .
4. 2&1 .

(3) الأمر update:.

1. من أوامر . DML

2. يستخدم لتحديث البيانات في جدول.

3. من أوامر . DQL

4. 1&2 .

قواعد البيانات (2)

4) الأمر Insert ::

- 1) من أوامر DML.
- 2) من أوامر DDL.
- 3) يستخدم في إدراج البيانات في جدول.
- 4) 1 & 3 .

5) الأمر drop ::

1. من أوامر DML.
2. من أوامر DDL.
3. يستخدم لحذف حقول من جدول.
4. 2 & 3 .

7) الأمر Revoke من:

- 1- أوامر DDL.
- 2- أوامر DML.
- 3- أوامر DQL.
- 4- أوامر PL/SQL.

8) الأمر GRANT من:

1. أوامر DDL.
2. أوامر DML.
3. أوامر DQL.
4. أوامر PL/SQL.

قواعد البيانات (2)

9) الأمر insert من :

1. أوامر DML.
2. أوامر DQL .
3. أوامر DDL .
4. أوامر PL/SQL.

10) الأمر Truncate يستخدم في :

- 1) حذف الأعمدة في الجدول.
- 2) حذف كافة الصفوف الموجودة في الجدول.
- 3) حذف عمود واحد.
- 4) حذف صف واحد.

11) بفرض أن لديك الجدول emp التالي :

empno	Name	Sal
1	Ali	1000
2	Hani	2000
3	Mohammed	1500
4	Ayman	1200

يكون ناتج الأوامر التالية هي :

➤ `select * from emp where sal > 1500;`

- 1) سجل 1
- 2) سجل 2
- 3) كل السجلات
- 4) سجل 4

➤ `select * from state where empno not in (1, 2);`

(1) سجل 3 و 4

(2) سجل 1 و 4

(3) سجل 3

(4) سجل 4

➤ `select * form state where sal between 1000 and 1300;`

(1) سجل 2 و 4

(2) سجل 1 و 4

(3) سجل 3

(4) سجل 4

➤ `select * from state where name like ' A% ';`

(1) سجل 2 و 4

(2) سجل 1 و 4

(3) سجل 2 و 3

(4) سجل 4

قواعد البيانات (2)

➤ `select*from state where sal not between 1000 and 1500;`

(1) سجل 2

(2) سجل 1 و 4

(3) سجل 3

(4) سجل 4

الفصل الثالث

تقنيات لغة SQL

الأهداف:

إعطاء الطالب صورة موضحة عن أساسيات الربط بين الجداول في SQL، استخدام الدوال Functions في SQL، كيفية إنشاء مجموعات من السجلات باستخدام كلا من having, group by مع جملة SELECT، بالإضافة إلى إمكانية إنشاء استعلام فرعي sub query

المحتويات:

1. الربط بين جدولين أو أكثر
2. استخدام multi-row functions
3. استخدام Single Row Functions
4. استخدام Group by clause
5. استخدام Having clause
6. استخدام وعمل استعلام فرعي sub query

ماذا سنتعلم في هذا الفصل:

في نهاية هذا الفصل يكون الطالب قد اكتسب المهارات والمعارف التالية:

1. الربط بين جدولين أو أكثر
2. التعرف علي أنواع العلاقات بين الجداول
3. استعادة بيانات من جدولين أو أكثر
4. استخدام group functions
5. استخدام Single Row Functions
6. إنشاء مجموعات من السجلات باستخدام Having,group by
7. استخدام وعمل استعلام فرعي Sub query

1- الربط بين جدولين أو أكثر

كثيراً ما نحتاج إلي استرجاع بيانات من جدولين أو أكثر في استعلام واحد Query، لإجراء مثل ذلك يتطلب منا إنشاء علاقة بين الجدولين أو الجداول، حيث تكمن القوة الحقيقية لقاعدة بيانات علائقية في قدرتها علي استرجاع سجلات من جداول مختلفة في استعلام واحد، وتوجد لدينا العديد من الأسباب التي تدعونا إلي إنشاء علاقة بين الجداول منها:

- (1) دمج الأعمدة من جدولين أو أكثر.
- (2) اختيار أعمدة موجودة في جدول واحد بناء علي شرط يطبق علي عمود آخر (self join).

1-1 استرجاع بيانات من جدولين أو أكثر (Displaying Data from Multiple Tables)

عند استرجاع بيانات من جدولين أو أكثر هناك عدة اعتبارات يجب مراعاتها

- (1) وضع أسماء الأعمدة في جملة Select.
 - (2) وضع أسماء الجداول في جملة From، علي أن يتم الفصل بينها بالفاصلة.
 - (3) إضافة شرط الربط في جملة Where.
 - (4) الصيغة العامة لاسترجاع بيانات من جدولين
- ```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

مثال 1

```
SQL>SELECT emp.empno, emp.ename, dept.dname
2 FROM emp, dept
3 WHERE emp. deptno=dept.deptno;
```

في هذا المثال تم استرجاع بيانات من جدولي emp, dept بينهما علاقة، حيث تم كتابة

إسم الجدول قبل إسم العمود مع الفصل بينهما بنقطة في جملة select، كتابة الجداول

مصدر البيانات في جملة from، مع كتابة شرط الربط في جملة Where.

| EMPNO | ENAME  | DNAME      |
|-------|--------|------------|
| 7369  | SMITH  | RESEARCH   |
| 7499  | ALLEN  | SALES      |
| 7521  | WARD   | SALES      |
| 7566  | JONES  | RESEARCH   |
| 7654  | MARTIN | SALES      |
| 7698  | BLAKE  | SALES      |
| 7782  | CLARK  | ACCOUNTING |
| 7788  | SCOTT  | RESEARCH   |
| 7839  | KING   | ACCOUNTING |
| 7844  | TURNER | SALES      |
| 7876  | ADAMS  | RESEARCH   |
| 7900  | JAMES  | SALES      |
| 7902  | FORD   | RESEARCH   |
| 7934  | MILLER | ACCOUNTING |

14 row selected

تلميح

في المثال السابق نجد أن أعمدة الربط لها نفس الاسم deptno، وهذا يسبب

مشاكل مع أوراكل حيث لا يعرف لأي جدول تنتمي هذه الأعمدة ولذلك تم وضع اسم الجدول

قبل اسم العمود، مع الفصل بينهما بنقطة.

## 2-1 أنواع العلاقات

1. **الرابطة المتكافئة equijoin**: لابد أن كل سجل في الجدول الرئيس الأب

يقابله سجل أو سجلات في الجدول المرتبط به الابن

2. **الرابطة غير المتكافئة nonequi join:** لا تمتلك الجداول أعمدة وقيماً

مشتركة

3. **الرابطة الخارجية outer join:** تمتلك الجداول المرتبطة أعمدة مشتركة، ولكن

السجلات المرتبطة يمكن أن تحتوي قيم مشتركة ويمكن أن لا تحتوي، بمعنى أن الجدول

الرئيس الأب يحتوي سجل أو سجلات لا يقابلها سجل أو سجلات في الجدول المرتبط

به الابن

4. **الروابط الذاتية Self join:** يمكننا عمل رابطة ذاتية عند ربط الجدول مع

نفسه

5. **الرابطة الديكارتية Cartesian joins:**

للحصول علي الرابطة الديكارتية قم بحذف شرط الربط بين الجدولين في جملة Where،

سيقوم أوراكل بربط كل سجل في الجدول الأول مع كل سجل في الجدول الثاني مع نفسه

### 1-2-1 الرابطة المتكافئة

فعلي سبيل المثال إذا أردت تحديد اسم الإدارة لكل موظف فما عليك سوي المقارنة بين قيم فإذا

عمود deptno في جدول الموظفين وقيم deptno في جدول dept كانت

متساوية بمعنى أن جدول dept لا يوجد به إدارة ليس بها عاملين، بمعنى أن كل الإدارات

يتم اختيارها في استعلام يربط الجدولين، إن مثل تلك العلاقة تسمي بالرابطة المتكافئة.

```
SQL>SELECT emp.empno, emp.ename, emp.deptno,
 Dept.deptno, dept.loc
 FROM emp, dept
 WHERE emp.deptno=dept.deptno;
```

| EMPNO           | ENAME  | DEPTNO | DEPTNO | LOC      |
|-----------------|--------|--------|--------|----------|
| 7369            | SMITH  | 20     | 20     | DALLAS   |
| 7499            | ALLEN  | 30     | 30     | CHICAGO  |
| 7521            | WARD   | 30     | 30     | CHICAGO  |
| 7566            | JONES  | 20     | 20     | DALLAS   |
| 7654            | MARTIN | 30     | 30     | CHICAGO  |
| 7698            | BLAKE  | 30     | 30     | CHICAGO  |
| 7782            | CLARK  | 10     | 10     | NEW YORK |
| 7788            | SCOTT  | 20     | 20     | DALLAS   |
| 7839            | KING   | 10     | 10     | NEW YORK |
| 7844            | TURNER | 30     | 30     | CHICAGO  |
| 7876            | ADAMS  | 20     | 20     | DALLAS   |
| 7900            | JAMES  | 30     | 30     | CHICAGO  |
| 7902            | FORD   | 20     | 20     | DALLAS   |
| 7934            | MILLER | 10     | 10     | NEW YORK |
| 14 row selected |        |        |        |          |

تلميح

1. في هذا المثال نجد أن جدول dept لا يوجد به إدارة ليس بها عاملين في جدول

emp، حيث في هذا الاستعلام تم استرجاع بيانات لجميع الإدارات.

2. حيث أن عمود deptno له نفس الاسم في كلا الجدولين، تم وضع اسم الجدول قبل

اسم العمود، مع الفصل بينهما بنقطة.

3. يطلق علي العلاقة equijoin أيضاً simple join أو inner join

**1-1-2-1 استخدام أكثر من شرط في جملة WHERE**

يمكننا عند استرجاع بيانات من أكثر من جدول، يمكننا إضافة أكثر من شرط في جملة الشرط

بعد شرط الربط، كما بالمثل التالي، فمثلاً لاسترجاع بيانات محلي النظم من الرقم، الاسم،

الوظيفة، رقم الإدارة، موقع الإدارة، لابد من إضافة شرط في where كما بالمثل التالي:

```
SQL>SELECT empno, ename, emp.job,emp.deptno, loc
FROM emp, dept
WHERE emp.deptno=dept.deptno
And job='ANALYST';
```

| EMPNO | ENAME | JOB     | DEPTNO | LOC    |
|-------|-------|---------|--------|--------|
| 7788  | SCOTT | ANALYST | 20     | DALLAS |
| 7902  | FORD  | ANALYST | 20     | DALLAS |

**2-1-2-1 استخدام TABLE Aliases في جملة Where**

كما سبق وتعلمنا أنه يمكن استخدام اسم اعتباري للعمود، يمكننا أيضا اسم اعتباري للجدول

Table Aliases في جملة WHERE ، فعلي سبيل المثال يمكننا إجراء مثال 1 السابق

باستخدام الأسماء الاعتبارية للجدول

```
SQL>SELECT e.empno, e.ename, e.deptno,
d.deptno, d.loc
FROM emp e, dept d
WHERE e.deptno=d.deptno;
```

تلميح

هناك عدة اعتبارات يجب مراعاتها عند استخدام الأسماء الاعتبارية

(1) الاسم الاعتباري للجدول لا يزيد عن 30 حرفاً.

(2) لابد أن يستخدم الاسم الاعتباري أولاً في جملة WHERE

(3) يفضل أن يكون الاسم الاعتباري معبراً



| EMPNO | ENAME  | DEPTNO | DEPTNO | LOC      |
|-------|--------|--------|--------|----------|
| 7369  | SMITH  | 20     | 20     | DALLAS   |
| 7499  | ALLEN  | 30     | 30     | CHICAGO  |
| 7521  | WARD   | 30     | 30     | CHICAGO  |
| 7566  | JONES  | 20     | 20     | DALLAS   |
| 7654  | MARTIN | 30     | 30     | CHICAGO  |
| 7698  | BLAKE  | 30     | 30     | CHICAGO  |
| 7782  | CLARK  | 10     | 10     | NEW YORK |
| 7788  | SCOTT  | 20     | 20     | DALLAS   |
| 7839  | KING   | 10     | 10     | NEW YORK |
| 7844  | TURNER | 30     | 30     | CHICAGO  |
| 7876  | ADAMS  | 20     | 20     | DALLAS   |
| 7900  | JAMES  | 30     | 30     | CHICAGO  |
| 7902  | FORD   | 20     | 20     | DALLAS   |
| 7934  | MILLER | 10     | 10     | NEW YORK |

14 row selected

### 3-1-2-1 استرجاع بيانات من عدة جداول (More than two tables)

أحيانا ما نريد استرجاع بيانات من أكثر من جدولين، فعلي سبيل المثال نريد استرجاع

بيانات الاسم ورقم الطلبية والأصناف وإجمالي الأصناف، إجمالي الطلبيات للعميل

Tarek، يمكن استرجاع تلك البيانات من ثلاث جداول هي customer, ord,

item، كما بالمثال التالي

```
SQL>SELECT c.name, o.ord,i.item,i.itemtot,o.total
FROM customer c,ord o,item i
WHERE c.custid=o.custid
And o.ordid=i.ordid
And c.name='Tarek';
```

### 2-2-1 الرابطة غير المتكافئة nonequijoin

نادراً ما نحتاج إلي ربط سجلات من جدولين لا يمتلكان أعمدة ولاقيما مشتركة. إن مثل تلك الرابطة

تسمي رابطة غير متكافئة، كما بالمثال التالي

| emp               |         |      | salgrade                                                                          |       |       |
|-------------------|---------|------|-----------------------------------------------------------------------------------|-------|-------|
| EMPNO             | ENAME   | SAL  | GRADE                                                                             | LOSAL | HISAL |
| 1                 | tarek   | 4000 | 1                                                                                 | 800   | 1300  |
| 2                 | ayman   | 2850 | 2                                                                                 | 1301  | 1400  |
| 3                 | said    | 2450 | 3                                                                                 | 1401  | 2500  |
| 4                 | ali     | 2998 | 4                                                                                 | 2500  | 9999  |
| 5                 | ahmed   | 1350 |                                                                                   |       |       |
| 6                 | helal   | 1700 |                                                                                   |       |       |
| 7                 | ebrahim | 1600 |                                                                                   |       |       |
| ...               |         |      |                                                                                   |       |       |
| 25 rows selected. |         |      | <p>المرتبة في جدول emp يقع بين<br/>أقل مرتبة وأكبر مرتبة في جدول<br/>SALGRADE</p> |       |       |

تلميح

✓ العلاقة بين الجدولين في عمود sal في جدول emp ينحصر بين عمودي

.Salgrade في Losal, hisal

✓ في الروابط غير المتكافئة لا تستخدم معامل المساواة

مثال

```
SQL>SELECT emp.ename, emp.sal, salgrade.grade
FROM emp, salgrade
WHERE emp.sal
BETWEEN salgrade.losal AND salgrade.hisal;
```

|        |      |   |
|--------|------|---|
| SMITH  | 800  | 1 |
| ADAMS  | 1100 | 1 |
| JAMES  | 950  | 1 |
| WARD   | 1250 | 2 |
| MARTIN | 1250 | 2 |
| MILLER | 1300 | 2 |
| ALLEN  | 1600 | 3 |
| TURNER | 1500 | 3 |
| JONES  | 2975 | 4 |
| BLAKE  | 2850 | 4 |
| CLARK  | 2450 | 4 |
| SCOTT  | 3000 | 4 |
| FORD   | 3000 | 4 |
| KING   | 5000 | 5 |

14row selected

### 3-2-1 الرابطة الخارجية outerjoin

تمتلك الجداول المرتبطة أعمدة مشتركة، ولكن السجلات في الجداول المرتبطة يمكن أن تحتوي قيم مشتركة ويمكن أن لا تحتوي، فعلي سبيل المثال إذا احتوي جدول dept علي إدارة لا يوجد بها عاملين، معني ذلك أن تلك الإدارة لن يتم اختيارها في استعلام يربط بين الجدولين، فعلي سبيل المثال في الشكل التوضيحي التالي نجد أن لا يوجد موظفين في .

الإدارة ENGINEERING

| emp   |        | dept   |             |
|-------|--------|--------|-------------|
| ENAME | DEPTNO | DEPTNO | DNAME       |
| ----- | -----  | -----  | -----       |
| Tarek | 1      | 1      | ACCOUNTING  |
| Ayman | 1      | 2      | SALES       |
| Said  | 3      | 3      | RESEARCH    |
| Ali   | 2      | ...    |             |
| ...   |        | 10     | ENGINEERING |

لا يوجد موظفين في الإدارة  
ENGINEERING

## قواعد البيانات (2)

فإذا قمنا بكتابة الكود التالي:

```
SQL> SELECT e.ename, d.deptno, d.dname
2 FROM emp e, dept d
3 WHERE e.deptno = d.deptno
```

| ENAME  | DEPTNO | DNAME      |
|--------|--------|------------|
| SMITH  | 20     | RESEARCH   |
| ALLEN  | 30     | SALES      |
| WARD   | 30     | SALES      |
| JONES  | 20     | RESEARCH   |
| MARTIN | 30     | SALES      |
| BLAKE  | 30     | SALES      |
| CLARK  | 10     | ACCOUNTING |
| SCOTT  | 20     | RESEARCH   |
| KING   | 10     | ACCOUNTING |
| TURNER | 30     | SALES      |
| ADAMS  | 20     | RESEARCH   |
| JAMES  | 30     | SALES      |
| FORD   | 20     | RESEARCH   |
| MILLER | 10     | ACCOUNTING |

14 row selected

نجد أن هناك سجلات لا تحقق شرط الربط (WHERE e.Deptno = d.deptno)

وبالتالي لن يتم استرجاعها في نتيجة الاستعلام، وحيث لا يوجد موظفين بالإدارة

ENGINEERING، وبالتالي لن تظهر في نتيجة الاستعلام طبقاً لل Equijoin، لكي نعالج

ذلك لابد من إنشاء رابطة خارجية وذلك بوضع إشارة الجمع (+) محاطة بقوسين وذلك بعد اسم

محدد شرط الربط في جملة Where وذلك للجدول الذي يحتوي السجلات التي لا يقابلها سجل

أو سجلات في الجدول المرتبط.

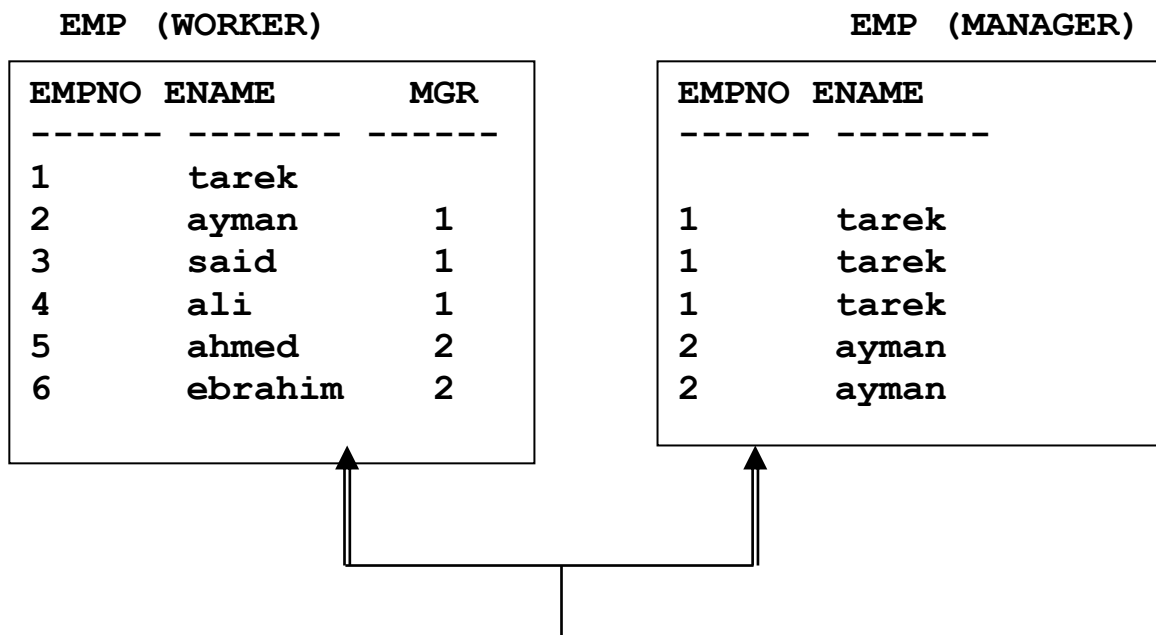
مثال

```
SQL> SELECT e.ename, d.deptno, d.dname, d.loc
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno;
```

في هذا المثال الإدارة ENGINEERING سيتم عرضها في نتيجة هذا الاستعلام، وذلك من خلال إنشاء رابطة خارجية وذلك بوضع إشارة الجمع (+) محاطة بقوسين بعد اسم محدد شرط الربط في جملة Where وذلك للجدول الذي يحتوي السجلات التي لا يقابلها سجل أو سجلات في الجدول المرتبط وهو في مثالنا جدول emp.

### 4-2-1 Selfjoin الرابطة الذاتية

يمكننا عمل الرابطة الذاتية عند ربط جدول ما مع نفسه، فعندما تربط جدولاً مع نفسه سوف تكتب اسم نفس الجدول عدة مرات في جملة Where، فيجب علينا استخدام أسماء اعتبارية Aliases، وفي المثال الموضح بالشكل التالي حالة جدول emp الذي يحتوي علي عمود (mgr) حيث يحدد رقم المدير الرقم في عمود المفتاح الرئيسي empno بجدول emp، لذلك توجد علاقة من Selfjoin بين عمود mgr وعمود empno.



" MGR في جدول WORKER تساوي EMPNO في جدول MANAGER "

مثال

```
SQL>SELECT worker.ename,manager.ename
FROM emp worker, emp manager
WHERE worker.mgr = manager.empno
```

| ENAME           | ENAME |
|-----------------|-------|
| SMITH           | FORD  |
| ALLEN           | BLAKE |
| WARD            | BLAKE |
| JONES           | KING  |
| MARTIN          | BLAKE |
| BLAKE           | KING  |
| CLARK           | KING  |
| SCOTT           | JONES |
| TURNER          | BLAKE |
| ADAMS           | SCOTT |
| JAMES           | BLAKE |
| FORD            | JONES |
| MILLER          | CLARK |
| 13 row selected |       |

### 1-2-4 الـرابطـة الديكـارتية Cartesian joins

للحصول علي الرابطـة الديكـارتية قم بحذف شرط الربط بين الجدولين في جملة Where، سيقوم

أوراكل برط كل سجل في الجدول الأول مع كل سجل في الجدول الثاني، هذا يجعل أوراكل يقوم

بإنشاء ومعالجة كمية كبيرة جداً من البيانات لا قيمة لها، فعلي سبيل المثال إذا كان جدول dept

يحتوي 10 سجلات و جدول emp يحتوي 30 سجل ،فإذا تم حذف شرط الربط من جملة

Where في نتيجة الاستعلام تكون (30\*10) وهي 300 سجل ،ولحساب السجلات المسترجعة

من رابطـة ديكـارتية للجدولين dept , emp يتم كتابة الاستعلام التالي

```
SQL>SELECT count(*) from dept,emp;
```

| COUNT(*) |
|----------|
| 300      |

**2 - استخدام الدوال Functions****1-2 استخدام Multi row Functions**

تزودنا SQL بعدة دوال تجميع Group Functions يمكن أن تنفذ علي مجموعة من السجلات وهي:

| الدالة   | الاستخدام                    |
|----------|------------------------------|
| COUNT    | عدد السجلات في جدول ما       |
| MAX      | تحديد القيمة العظمي لعمود ما |
| MIN      | تحديد القيمة الدنيا لعمود ما |
| AVG      | المتوسط الحسابي لعمود ما     |
| STDDEV   | الانحراف المعياري لعمود ما   |
| SUM      | المجموع الكلي لعمود ما       |
| VARIANCE | حساب التباين لعمود ما        |

تقوم الدوال التجميعية Group Functions بمعالجة قيم العمود المختار من الجدول وتقدم

النتيجة في شكل قيمة وحيدة تخص العمود المختار، ويجب عليك إتباع التعليمات التالية عند

إستخدام Group function

(1) وضع إسم العمود بين قوسين ( ) بعد الدالة مباشرة

```
Select MAX(sal) From emp;
```

(2) يمكن لجميع الأعمدة في جملة Select أن تكون كلها دوال تجميعية

```
Select MAX(sal), MIN(SAL), SUM(sal);
```

(3) إستخدام دالة تجميعية أخرى غير مسموح به في SQL

```
Select MAX(avg(sal)) from emp;
```

(4) الدوال التجميعية تتجاهل القيم الفارغة NULLVALUES

(5) يمكن إستخدام الدوال التجميعية ضمن التعابير الحسابية

### 1-1-2 استخدام count

نستخدم الدالة count لحساب عدد السجلات التي تحتوي قيمة غير فارغة (not

null) في العمود المحدد في المجموعة، فمثلاً لحساب عدد السجلات في جدول emp الذي

يملك قيمة غير فارغة في العمود ename يتم كتابة الكود التالي:

```
SQL>SELECT count (ename) from emp;
```

| COUNT (ENAME) |
|---------------|
| 14            |

تلميح

✓ تم وضع العمود المراد حساب عدد سجلاته بين قوسين بعد الدالة count.

✓ لا يتم عد السجلات التي تحتوي قيم فارغة null.

### 1-1-1-2 استخدام count (\*)

عندما نستبدل اسم العمود بالنجمة (\*) فعندئذ تقوم الدالة count بحساب عدد السجلات بما

فيها السجلات التي تحتوي علي قيم فارغة، فعلي سبيل المثال:

```
SQL> SELECT COUNT (*)
2 FROM emp
3 WHERE deptno = 10;
```

| COUNT (*) |
|-----------|
| 3         |



### 2-1-1-2 استخدام distinct مع count

يمكننا استخدام كلمة distinct مع count، حيث نقوم باستبعاد السجلات المكررة، فعلي

سبيل المثال لحساب عدد السجلات في جدول emp مع عدم استبعاد القيم الفارغة في العمود

ename يتم كتابة الكود التالي:

```
SQL>SELECT count (distinct (ename)) from emp;
```

### 2-1-2 استخدام max

نستخدم الدالة MAX لحساب أكبر قيمة في مجموعة ما، يمكن استخدام الدالة MAX مع القيم

العديدية أو النصية أو التاريخ والوقت، فعلي سبيل المثال:

```
SQL> SELECT MAX(SAL), MAX(hiredate), MAX(ename)
2 FROM emp;
```

تلميح

✓ يمكن استخدام max مع أي نوع من البيانات.

✓ عند استخدام MAX مع البيانات الحرفية تعيد أعلى قيمة ASCII، وعند

استخدامها مع الأعمدة العددية تعيد أعلى قيمة جبرية، وعند استخدامها مع

التاريخ والوقت تعيد القيمة الأحدث في العمود.

✓ تهمل الدالة MAX القيم NULL.

### 3-1-2 استخدام MIN

تعيد الدالة MIN أصغر قيمة لا تساوي NULL، كما أنها تستخدم مع أي نوع من البيانات، فعلي

سبيل المثال نستخدم الدالة MAX لتحديد أقل مرتب في الإدارة رقم 10

```
SQL> SELECT MIN(SAL)
FROM emp
WHERE deptno=10;
```

| MIN(SAL) |
|----------|
| 1300     |

**4-1-2 استخدام AVG**

نستخدم AVG لحساب المتوسط الحسابي لعمود ما، حيث يعيد AVG القيمة الوسطي لجميع القيم

التي لا تساوي NULL في العمود، وهناك عدة اعتبارات يجب مراعاتها عند استخدام AVG

✓ يستخدم AVG مع القيم العددية فقط.

✓ يهمل AVG القيمة NULL عند حسابه للقيمة الوسطي.

✓ يهمل AVG القيم المتكررة في العمود عند حسابه للقيمة الوسطي وذلك إذا صرحنا

بكتابة DISTINCT قبل إسم العمود.

مثال 1

يمكننا حساب المتوسط الحسابي لعمود SAL في جدول EMP، وذلك بكتابة الكود التالي:

```
SQL>SELECT AVG (SAL)
FROM emp ;
```

| AVG(SAL)  |
|-----------|
| 2073.2143 |

مثال 2

يمكننا حساب كلا أعلى قيمة وأدنى قيمة والمتوسط الحسابي لعمود SAL بالكود التالي

```
SQL> SELECT MAX (SAL) , MIN (SAL) , AVG (SAL)
FROM emp ;
```

| MAX(SAL) | MIN(SAL) | AVG(SAL)  |
|----------|----------|-----------|
| 5000     | 800      | 2073.2143 |

مثال 3

يمكننا حساب المتوسط الحسابي لعمود COMM في جدول EMP، وذلك بكتابة الكود التالي

```
SQL>SELECT AVG (SAL)
FROM emp ;
```

| AVG (COMM) |
|------------|
| 550        |

نلاحظ في هذا المثال أن avg تتجاهل القيم null.

#### 4-1-2 استخدام SUM

نستخدم sum لحساب المجموع الكلي لقيم عمود ما، وهناك عدة اعتبارات يجب مراعاتها

✓ يستخدم sum مع القيم العددية فقط.

✓ تهمل sum القيمة NULL عند حسابه المجموع الكلي.

✓ تهمل sum القيم المتكررة في العمود عند حسابه المجموع الكلي وذلك إذا صرحنا

بكتابة DISTINCT قبل إسم العمود.

مثال

يمكننا حساب المجموع الكلي لعمود SAL في جدول EMP، وذلك بكتابة الكود التالي:

```
SQL>SELECT sum (SAL)
FROM emp;
```

| SUM(SAL) |
|----------|
| 29025    |

### 2-1-2 استخدام الدالة nvl مع الدوال التجميعية group Functions

نستخدم الدالة NVL مع الأعمدة التي تحتوي القيم الفارغة NULL، حيث تقوم بمعالجة مشاكل القيم الفارغة باستبدالها بالقيمة الصفرية وبالتالي يمكننا استخدام الدوال التجميعية مع الأعمدة التي تحتوي قيم فارغة لكي تشمل جميع السجلات.

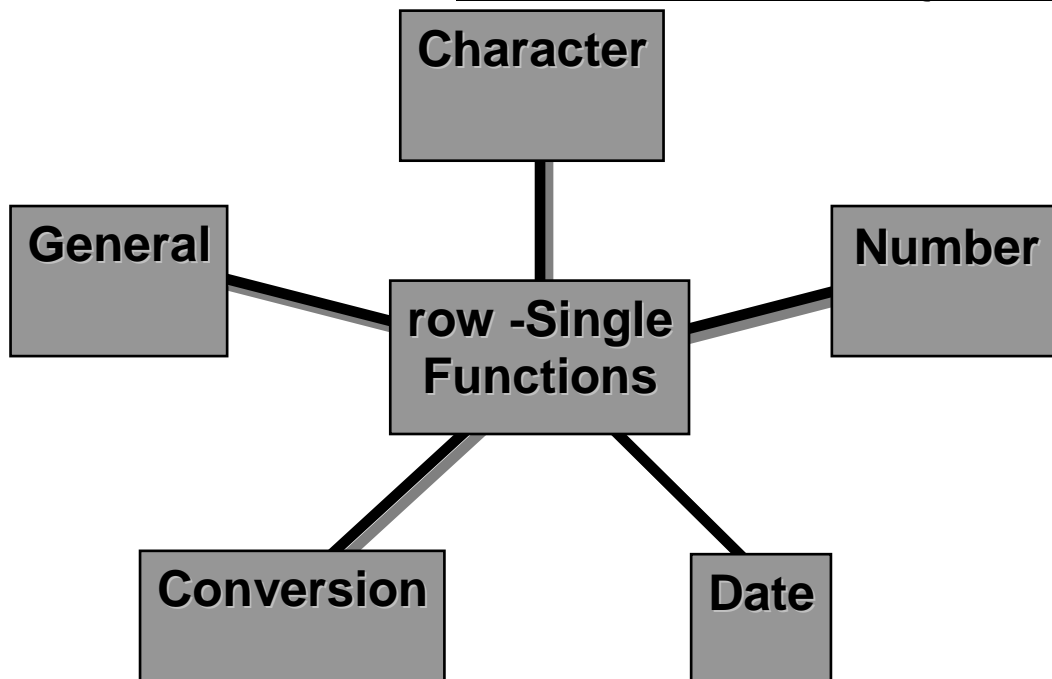
مثال

```
SQL> SELECT AVG (NVL (comm, 0))
2 FROM emp;
```

### 2-2 استخدام single row functions

تعالج single row functions قيم العمود المختار من الجدول وتقدم النتيجة علي شكل قيمة وحيدة لكل سجل ، هذا في حين تستخدم group function في معالجة قيم العمود المختار من الجدول وتقدم النتيجة علي شكل قيمة وحيدة تخص العمود المختار

### 1-2-2 أنواع single row functions



1-1-2-2 الدوال النصية character functions

يمتلك أوراكل عدد من الدوال التي تستخدم لتعديل أنماط البيانات النصية (char,

varchar)، وسوف نتعرض لأهم هذه الدوال النصية في هذا الفصل وهي:

| الدالة النصية | الاستخدام                                                                                                              |
|---------------|------------------------------------------------------------------------------------------------------------------------|
| INITCAP       | تغيير حالة الحرف الأول من كل سلسلة نصية إلى حرف كبير و تحويل حالة الأحرف التالية للحرف الكبير إلى حالة الأحرف الصغيرة. |
| LENGTH        | يعيد عدد الأحرف المكونة لسلسلة حرفية                                                                                   |
| LOWER         | تحويل كل حرف ضمن سلسلة حرفية إلى أحرف صغيرة                                                                            |
| UPPER         | تحويل كل حرف ضمن سلسلة حرفية إلى أحرف كبيرة                                                                            |
| CONCAT        | دمج سلسلة حرفية مع سلسلة حرفية أخرى                                                                                    |
| REPLACE       | استبدال سلسلة حرفية بسلسلة حرفية أخرى                                                                                  |
| SOUNDEX       | إيجاد الكلمات المتشابهة من حيث اللفظ الصوتي                                                                            |
| SUBSTR        | نسخ الجزء المحدد من الكلمة ابتداء من حرف محدد                                                                          |
| INSTR         | يعيد قيمة عددية تمثل موضع مجموعة أحرف ضمن سلسلة حرفية                                                                  |

## مثال 1

سنقوم في المثال باستعراض أمثلة علي الدوال النصية كما هو موضح بالجدول التالي

| الدالة FUNCTION            | النتيجة RESULT  |
|----------------------------|-----------------|
| LOWER ('JICC Center')      | ('jicc center') |
| UPPER ('jicc Center')      | ('JICC CENTER') |
| INITCAP ('JICC Center')    | ('Jicc Center') |
| CONCAT('JICC'), ('CENTER') | JICCCENTER      |
| SUBSTR('String',1,3)       | Str             |
| Length('String')           | 6               |
| INSTR('String','t')        | 2               |

```
SQL>SELECT ename, CONCAT (ename, job),
 LENGTH (ename), INSTR (ename, 'A')
FROM emp;
```

| ENAME  | CONCAT (ENAME, JOB) | LENGTH (ENAME) | INSTR (ENAME, 'A') |
|--------|---------------------|----------------|--------------------|
| SMITH  | SMITHCLERK          | 5              | 0                  |
| ALLEN  | ALLENSALESMAN       | 5              | 1                  |
| WARD   | WARDSALESMAN        | 4              | 2                  |
| JONES  | JONESMANAGER        | 5              | 0                  |
| MARTIN | MARTINSALESMAN      | 6              | 2                  |
| BLAKE  | BLAKEMANAGER        | 5              | 3                  |
| CLARK  | CLARKMANAGER        | 5              | 3                  |
| SCOTT  | SCOTTANALYST        | 5              | 0                  |
| KING   | KINGPRESIDENT       | 4              | 0                  |
| TURNER | TURNERSALESMAN      | 6              | 0                  |
| ADAMS  | ADAMSCLERK          | 5              | 1                  |
| JAMES  | JAMESCLERK          | 5              | 2                  |
| FORD   | FORDANALYST         | 4              | 0                  |
| MILLER | MILLERCLERK         | 6              | 0                  |

## 2-1-2-2 الدوال الرقمية Number functions

يملك أوراكل عدد من الدوال الرقمية التي تستخدم مع أنواع البيانات الرقمية Numeric

Data، وسوف نتعرض لأهم هذه الدوال الرقمية في هذا الفصل وهي:

| الدالة الرقمية | الاستخدام                                               |
|----------------|---------------------------------------------------------|
| ROUND          | تستخدم لتقريب قيمة رقمية إلى عدد أرقام محدد بعد الفاصلة |
| TRUNC          | تستخدم لقطع أو قص القيم إلى دقة محددة                   |
| MOD            | يستخدم لتحديد باقي القسمة الناتج عن عملية ما            |

سنقوم في المثال باستعراض أمثلة علي الدوال الرقمية كما هو موضح كالتالي

|          |             |       |
|----------|-------------|-------|
| 1) ROUND | (42.926, 2) | 42.93 |
| 2) TRUNC | (44.926, 2) | 44.92 |
| 3) MOD   | (1600, 300) | 100   |

مثال 2

```
SQL> SELECT ROUND(42.923,2), ROUND(43.923,0),
2 ROUND (44.923,-1)
3 FROM DUAL;
```

| ROUND(42.923,2) | ROUND(43.923,0) | ROUND(44.923,-1) |
|-----------------|-----------------|------------------|
| 42.92           | 44              | 40               |

مثال 3

```
SQL>SELECT TRUNC(35.832,2), TRUNC(65.323),
2 TRUNC (65.843,-1)
3 FROM DUAL;
```

| TRUNC(35.832,2) | TRUNC(65.323) | TRUNC(65.843,-1) |
|-----------------|---------------|------------------|
| 35.83           | 65            | 60               |

مثال 4

```
SQL>SELECT ename, sal, comm, MOD(sal,comm)
2 FROM emp;
```

| ENAME           | SAL  | COMM | MOD (SAL , COMM) |
|-----------------|------|------|------------------|
| SMITH           | 800  |      |                  |
| ALLEN           | 1600 | 300  | 100              |
| WARD            | 1250 | 500  | 250              |
| JONES           | 2975 |      |                  |
| MARTIN          | 1250 | 1400 | 1250             |
| BLAKE           | 2850 |      |                  |
| CLARK           | 2450 |      |                  |
| SCOTT           | 3000 |      |                  |
| KING            | 5000 |      |                  |
| TURNER          | 1500 | 0    | 1500             |
| ADAMS           | 1100 |      |                  |
| JAMES           | 950  |      |                  |
| FORD            | 3000 |      |                  |
| MILLER          | 1300 |      |                  |
| 14 row selected |      |      |                  |

### 2-2-1-3 التاريخ ودوال التاريخ Date and Date functions

يملك أوراكل عدداً من الدوال التي تستخدم مع أعمدة التاريخ، حيث يمكن إظهار التاريخ بشكل حرفي، كما يمكننا استخدام التاريخ في عمليات الجمع والطرح، كما يمكن إضافة قيمة ما تاريخ ما ونحصل بذلك علي قيمة تاريخ جديدة، وسوف نتعرض لأهم هذه الدوال وهي موضحة في الجدول

التالي

| الاستخدام                                            | الدالة                             |
|------------------------------------------------------|------------------------------------|
| يحدد عدد الأشهر بين تاريخين                          | MONTHS_BETWEEN (date1, date2)      |
| يحدد اليوم التالي من الأسبوع الذي يلي التاريخ المحدد | NEXT_DAY (date, 'day')             |
| يستخدم لحساب تاريخ جديد بناء علي عدد محدد من الأشهر  | ADD_MONTHS (date, numberof months) |
| يحدد آخر يوم في الشهر الحالي أو لتاريخ محدد          | LAST_DAY (date1)                   |



مثال 1

سنقوم في المثال باستعراض أمثلة علي الدوال التاريخية كما هو موضح كالتالي

MONTHS\_BETWEEN ('1-SEP-93','11-JAN-44')  
19.6774194

ADD\_MONTHS ('11-JAN-96',6) '11-jul-96'

NEXT\_DAY ('01-SEP-95','friday') '8-sep-95'

LAST\_DAY ('01-sep-95') '30-sep-95'

مثال 2

SQL> SELECT SYSDATE  
2 FROM dual;

SYSDATE  
-----  
20/01/04

✓ Sysdate دالة SQL Function تستخدم لاسترجاع التاريخ والوقت الحاليين.

## 4-1-2-2 الدوال التحويلية conversion functions

يستخدم أوراكل عدة دوال تحويلية من أهم هذه الدوال التحويلية

✓ To\_number ويستخدم لتحويل نوع البيانات إلي نوع البيانات الرقمية. فمثلاً يمكننا

استخدام to\_number لتحويل السلسلة الحرفية السنة إلي حقل رقمي وذلك لاستخدام

أحد الدوال الرقمية علي هذا الحقل مثل MOD.

✓ To\_char ويستخدم لتحويل نوع البيانات إلي نوع البيانات النصية، فمثلاً يمكننا

استخدامه لتغيير تاريخ الميلاد للموظف إلي حقل حرفي وذلك لتنسيقه ليظهر بتنسيق

## قواعد البيانات (2)

خاص ، ويعتبر من أهم الدوال التحويلية في أوراكل،ومن أهم استخداماته هو تحويل نوع البيانات الرقمية إلي نوع البيانات النصية وذلك لإضفاء عليها تنسيق خاص مثل إظهار علامة العلامة بجوار الرقم والتحكم في رمز العلامة العشرية وعدد الأرقام بعد العلامة العشرية وهناك عدة رموز يمكننا استخدامها عند تحويل البيانات الرقمية إلي نوع البيانات النصية ، كما هو موضح بالجدول التالي :

| الرمز | الاستخدام                                           |
|-------|-----------------------------------------------------|
| 9     | يمثل رقم number                                     |
| 0     | يمثل الصفر zero                                     |
| \$    | يمثل علامة الدولار dollar sign                      |
| L     | يمثل علامة العلامة علي النظام local currency symbol |
| .     | يمثل رمز العلامة العشرية Decimal point              |
| '     | رمز تجميع الأرقام A thousand indicator              |

✓ To\_date يستخدم لتحويل نوع البيانات إلي نوع البيانات التاريخية.

مثال 1

في هذا المثال سنقوم بتحويل نوع بيانات عمود hiredate من نوع date إلي نوع البيانات الحرفية وذلك بإسخدام To\_char وذلك لعمل تنسيق خاص لحقل التاريخ

```
SQL>SELECT ename,TO_CHAR(hiredate,'fmDD Month YYYY')HIREDATE
2 FROM emp;
```

| ENAME  | HIREDATE             |
|--------|----------------------|
| SMITH  | 17 1980 كانون الأول  |
| ALLEN  | 20 1981 شباط         |
| WARD   | 22 1981 شباط         |
| JONES  | 2 1981 نيسان         |
| MARTIN | 28 1981 أيلول        |
| BLAKE  | 1 1981 أيار          |
| CLARK  | 9 1981 حزيران        |
| SCOTT  | 19 1987 نيسان        |
| KING   | 17 1981 تشرين الثاني |
| TURNER | 8 1981 أيلول         |
| ADAMS  | 23 1987 أيار         |
| JAMES  | 3 1981 كانون الأول   |
| FORD   | 3 1981 كانون الأول   |
| MILLER | 23 1982 كانون الثاني |

تم اختيار 14 صف

SQL>

تلميح

✓ تنسيق التاريخ لابد أن يوضع بين علامتي تنصيص مفردة ' ' .

✓ تفصل بين التاريخ والتنسيق بفاصلة

مثال 2

في هذا المثال سنقوم بتحويل نوع بيانات عمود sal من نوع number إلى نوع البيانات

الحرفية وذلك بإسخدام To\_char وذلك لعمل تنسيق خاص لحقل المرتب مع استخدام الرموز

```
SQL>SELECT TO_CHAR(sal,'$99,999') SALARY
2 FROM emp;
```

| SALARY  |
|---------|
| \$800   |
| \$1,600 |
| \$1,250 |
| \$2,975 |
| \$1,250 |
| \$2,850 |
| \$2,450 |
| \$3,000 |
| \$5,000 |
| \$1,500 |
| \$1,100 |
| \$950   |
| \$3,000 |
| \$1,300 |

14 row selected

**5-1-2-2 الدوال العامة General functions**

من أهم الدوال العامة في single row function دالتي nvl، decode وفيما

يلي توضيح لاستخدامات كلتا الدالتين في SQL.

**1-5-1-2-2 الدالة NVL**

تستخدم الدالة NVL لاستبدال القيمة الفارغة بقيمة أخرى، ويمكن استخدام NVL مع نوع البيانات

الرقمية وغير الرقمية، ويعتبر من أهم استخدامات تلك الدالة هو التغلب علي مشكلة القيم الصفرية

في البيانات الرقمية إذا تقوم NVL باستبدال القيم الفارغة بقيم صفرية، وهناك عدة اعتبارات يجب

مراعاتها عند استخدام NVL

✓ تستخدم الدالة NVL في تحويل القيم الفارغة إلي قيم فعلية.

✓ يمكنها معالجة أنواع البيانات الرقمية NUMERIC DATA، والنصية

CHARACTER STRING، والتاريخية DATE.

✓ عند تحويل القيم الفارغة عليك بالالتزام بنفس نوع البيانات Data types في

العمود الذي يحتوي القيم الفارغة فعلي سبيل المثال:

- Nvl(comm,0)
- Nvl(job,'no job yet')
- Nvl(hiredate,'01-jan-98')

مثال

```
SQL> SELECT ename,sal,comm,(sal*12)+NVL(comm,0)
2 FROM emp;
```

| ENAME  | SAL  | COMM | (SAL*12)+NVL(COMM,0) |
|--------|------|------|----------------------|
| SMITH  | 800  |      | 9600                 |
| ALLEN  | 1600 | 300  | 19500                |
| WARD   | 1250 | 500  | 15500                |
| JONES  | 2975 |      | 35700                |
| MARTIN | 1250 | 1400 | 16400                |
| BLAKE  | 2850 |      | 34200                |
| CLARK  | 2450 |      | 29400                |
| SCOTT  | 3000 |      | 36000                |
| KING   | 5000 |      | 60000                |
| TURNER | 1500 | 0    | 18000                |
| ADAMS  | 1100 |      | 13200                |
| JAMES  | 950  |      | 11400                |
| FORD   | 3000 |      | 36000                |
| MILLER | 1300 |      | 15600                |

14 row selected

**Decode الدالة 2-5-1-2-2**

تسهل الدالة Decode إجراء الاستعلام الشرطي في SQL، وتمثل عمل كلاً من CASE،

Decode، والصيغة العامة لاستخدام الدالة Decode

مثال

```
DECODE (col/expression, search1, result1
[, search2, result2,...,][, default])
```

في المثال التالي نستخدم Decode لإجراء الاستعلام الشرطي التالي، فإذا كانت

الوظيفة ANALYST يتم حساب الحوافز علي أساس 5% من المرتب، وإذا كانت الوظيفة

SALESMAN يتم حساب الحوافز علي أساس 6% من المرتب، وإذا كانت الوظيفة MANAGER

يتم حساب الحوافز علي أساس 25% من المرتب، فيما عدا ذلك لا يتم احتساب حوافز.

```
SQL> SELECT job,sal,
2 DECODE(job,'ANALYST',SAL*.05
3 'salesman', SAL*.06
4 'MANAGER', SAL*.25,
5 SAL)
6 BONUS
7 FROM emp;
```

| JOB             | SAL  | BONUS  |
|-----------------|------|--------|
| CLERK           | 800  | 800    |
| SALESMAN        | 1600 | 1600   |
| SALESMAN        | 1250 | 1250   |
| MANAGER         | 2975 | 743.75 |
| SALESMAN        | 1250 | 1250   |
| MANAGER         | 2850 | 712.5  |
| MANAGER         | 2450 | 612.5  |
| ANALYST         | 3000 | 150    |
| PRESIDENT       | 5000 | 5000   |
| SALESMAN        | 1500 | 1500   |
| CLERK           | 1100 | 1100   |
| CLERK           | 950  | 950    |
| ANALYST         | 3000 | 150    |
| CLERK           | 1300 | 1300   |
| 14 row selected |      |        |

**3- استخدام GROUP BY**

كنا فيما سبق كانت SQL تعالج الجدول أو الجداول بالكامل أو السجلات المحققة لشرط معين في جملة WHERE، إلا أننا في بعض الأحيان نريد إجراء عمليات منطقية علي سجلات معينة مثل إظهار القيمة العظمي والدنيا والمتوسط الحسابي لكل إدارة من الإدارات علي حدة، لإجراء مثل هذا نحتاج إلي استخدام Group by، حيث تستخدم جملة group by لتحديد الأعمدة التي يتم التجميع بناء عليها أو عندما نريد تطبيق الدوال التجميعية مثل avg, sum, count, max, min علي سجلات من الجدول معزولة علي شكل مجموعات فرعية، حيث تميز كل مجموعة بنفس قيم المعطيات، بحيث تقسم group by سجلات الجدول إلي مجموعات صغيرة متشابهة، ثم يمكنك استخدام group functions لاسترجاع معلومات تلخيصية لكل مجموعة من هذه المجموعات علي حدة، والصيغة العامة لاستخدام group by تأخذ الشكل التالي:

```
SQL>SELECT column, group_function(column)
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```


 تلميح

✓ تقع جملة group by بعد جملة where أو بعد جملة From في حالة عدم

وجود جملة where.

✓ تقسم group by سجلات الجدول إلي مجموعات صغيرة متشابهة، ثم يمكنك

استخدام group functions لاسترجاع معلومات تلخيصية لكل مجموعة من

هذه المجموعات علي حدة.

## قواعد البيانات (2)

✓ تتعامل GROUP BY مع القيم الفارغة NULL كمجموعة مميزة عن غيرها.

✓ يمكننا GROUP BY بإنشاء مجموعات فرعية ضمن مجموعات فرعية أخرى مثل:

GROUP BY deptno, sal

وهناك عدة اعتبارات يجب مراعاتها عند استخدام Group by

✓ يمكن أن تحتوي عدة أعمدة وتعابير.

✓ يتم وضع فواصل بين أسماء الأعمدة.

✓ يجب أن تحتوي علي كل الأعمدة المكتوبة بجملة select والتي لا

تستخدمها دوال التجميع.

مثال 1

في هذا المثال نريد حساب المجموع الكلي لمرتبات كل إدارة علي حدة، لفعل ذلك يجب علينا

استخدام عبارة group by في جملة select

```
SQL> SELECT deptno, sum(sal)
2 FROM emp
3 GROUP BY deptno;
```

| DEPTNO | SUM(SAL) |
|--------|----------|
| 10     | 8750     |
| 20     | 10875    |
| 30     | 9400     |

تلميح

✓ يجب أن تحتوي جملة group by علي كل الأعمدة المكتوبة بجملة select

والتي لا تستخدمها دوال التجميع (وهو في المثال السابق حقل deptno الذي يراد يتم

التجميع بناء عليه.

مثال 2

يمكننا إجراء المثال السابق و حساب المجموع الكلي لمرتبات كل إدارة علي حدة

باستخدام عمود deptno بدون كتابة عمود deptno في جملة select

```
SQL> SELECT sum(sal)
2 FROM emp
3 GROUP BY deptno;
```

| SUM(SAL) |
|----------|
| 8750     |
| 10875    |
| 9400     |

تلميح

✓ يمكننا إنشاء مجموعات من السجلات باستخدام group by بناء علي أعمدة ليست

موجودة بجملة Select

مثال 3

في هذا المثال سنوضح إمكانية قيام GROUP BY بإنشاء مجموعات فرعية ضمن

مجموعات فرعية أخرى

```
SQL> SELECT deptno, job, sum(sal)
2 FROM emp
3 GROUP BY deptno, job;
```



تلميح

✓ أولاً يتم إنشاء مجموعات متشابهة بناءً على رقم الإدارة deptno

(First, the rows are grouped by department number)

✓ ثانياً في مجموعات الإدارة يتم عمل مجموعات متشابهة من للسجلات بناءً على الوظيفة.

(Second, within the department number groups, the rows are grouped by job title.)

| DEPTNO         | JOB       | SUM(SAL) |
|----------------|-----------|----------|
| 10             | CLERK     | 1300     |
| 10             | MANAGER   | 2450     |
| 10             | PRESIDENT | 5000     |
| 20             | ANALYST   | 6000     |
| 20             | CLERK     | 1900     |
| 20             | MANAGER   | 2975     |
| 30             | CLERK     | 950      |
| 30             | MANAGER   | 2850     |
| 30             | SALESMAN  | 5600     |
| 9 row selected |           |          |

مثال 4

عند إجراء المثال التالي تظهر لنا رسالة الخطأ التالية (ORA-00934 : وظيفة المجموعة غير

مسموحة هنا) والتي تعني أننا لا نستطيع استخدام الدوال التجميعية Group Function مع

جملة Where

```
SQL> SELECT deptno, MAX(sal)
2 FROM emp
3 WHERE MAX (sal) > 2000
4 GROUP BY deptno;
```

```
WHERE MAX (sal) > 3000
*
ERROR at line 3:
ORA-00934: group function is not allowed here
```

#### 4- استخدام Having (تقييد الاختيار ضمن المجموعة)

##### (Restriction group selection)

قد لا نريد استرجاع دائماً جميع المجموعات الفرعية المختلفة في الجدول ضمن استعلام واحد، فنقدم لنا SQL جملة Having لحذف المجموعات الفرعية من التي لا نريد استرجاعها في الاستعلام أو التقرير، وتعتبر جملة Having مشابهة لجملة Where من حيث أنها تحدد فيما إذا كانت السجلات قابلة للاختيار أو لا، وتحتوي جملة Having علي محددات تستخدم لتقييم السجلات، حيث يمكننا وضع عدة شروط في جملة Having، كما يمكننا استخدام المعاملات المنطقية معها مثل and, or، وقد يتبادر إلي الذهن لماذا لا نستخدم عوضاً عنها جملة where، لا يمكننا ذلك حيث يكمن الاختلاف بين جملة Having وجملة Where في أن محددات الجملة Having يجب أن تكون تابع لجميع، ويتم استخدام Having فقط عند يتم حساب قيمة مجمعة بواسطة Select، بينما عمل Where هو ترشيح السجلات التي سيطبق عليها لاحقاً جملة Group by، وليس ترشيح المجموعات، والصيغة العامة لاستخدام

having موضحة كالتالي

```
SQL>SELECT column, group_function(column)
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

## قواعد البيانات (2)

وهناك عدة اعتبارات يجب مراعاتها عند استخدام HAVING

✓ تقع جملة Having بعد فقرة Group by.

✓ تستخدم Having فقط مع Group by ولا يمكن استخدامها بدونها.

✓ نستخدم Having مع الدوال التجميعية التالية فقط

(MAX, MIN, AVG, COUNT, SUM)

✓ يمكن تحديد عدة شروط في جملة Having وذلك باستخدام المعاملات المنطقية

.and, or

✓ يمكننا استخدام كلاً من Where، Having في جملة Select.

مثال

```
SQL>SELECT job, SUM(sal) PAYROLL
2 FROM emp
3 WHERE job NOT LIKE 'SALES%'
4 GROUP BY job
5 HAVING SUM (sal)<6000
6 ORDER BY SUM (sal);
```

في المثال السابق نقوم باسترجاع الوظيفة والمجموع الكلي للمرتبات الشهرية لكل وظيفة علي حده

بشرط أن يكون المجموع الكلي للمرتبات الشهرية لكل وظيفة أقل من 6000 ريال باستثناء وظيفة

salesman، ثم الترتيب التصاعدي بناء علي المجموع الكلي للمرتبات الشهرية.

| JOB       | PAYROLL |
|-----------|---------|
| CLERK     | 4150    |
| PRESIDENT | 5000    |

## 5- استخدام وعمل استعلام فرعي (Sub query)

تمكننا sql من جعل الاستعلامات متداخلة مع بعضها البعض بشكل نموذجي، بحيث ينتج عن

الاستعلام الداخلي (inner query) قيمة تفحص في قسم الشرط للاستعلام

الخارجي (outer query) لمعرفة فيما إذا كان الشرط سيتحقق عندها أم لا

### 5-1 كيف تعمل الاستعلامات الفرعية

نفترض أنك تريد كتابة استعلاماً لاسترجاع من يأخذ مرتباً أكثر من محمد ؟ نجد أننا نحتاج إلي

استعلامين

الأول: ماهو مرتب محمد؟

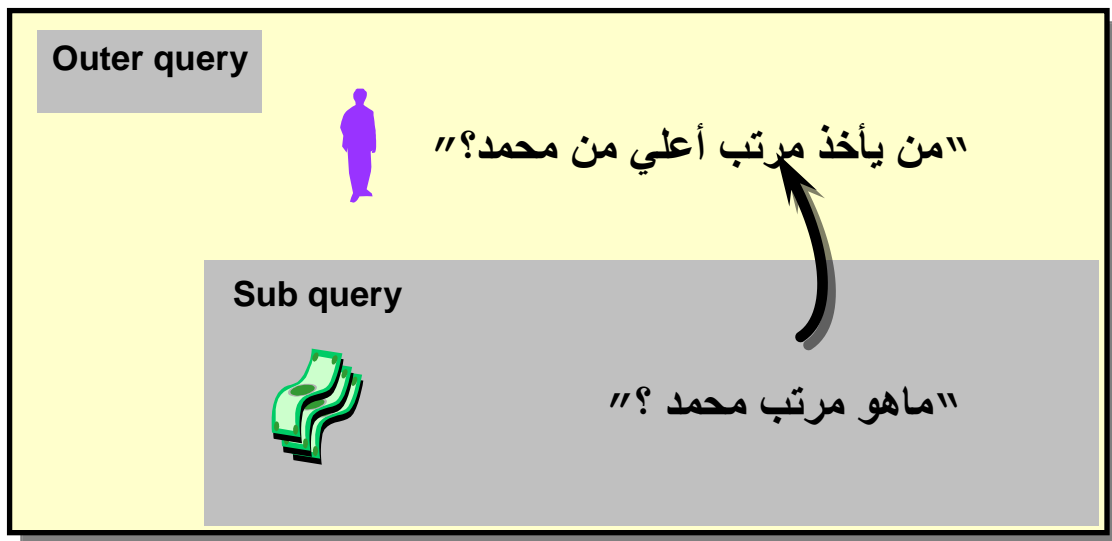
الثاني: البحث عن من يأخذ مرتباً أعلى من محمد؟

و لحل مثل تلك المشكلة لابد من دمج الاستعلامين معا بمعنى استخدام استعلام داخل استعلام

آخر، حيث يعيد الاستعلام الداخلي أو الفرعي قيمة يستخدمها الاستعلام الرئيسي أو الخارجي

لمعرفة فيما إذا كان سيتحقق الشرط عندها أم لا

من يأخذ مرتباً أعلى من محمد



- ✓ ينفذ الاستعلام الفرعي أو الداخلي أولاً ثم ينفذ الاستعلام الرئيسي أو الخارجي.
- ✓ يولد أو يعيد الاستعلام الداخلي نتيجة يستخدمها الاستعلام الخارجي لمعرفة ما إذا سيتحقق الشرط عنها أم لا.
- ✓ الصيغة العامة لعمل واستخدام الاستعلامات الفرعية Sub queries كالتالي

```
SQL>SELECT select_list
FROM table
WHERE expr operator
 (SELECT select_list
 FROM table);
```

مثال

```
SQL> SELECT ename
2 FROM emp 5000
3 WHERE sal >
4 (SELECT sal
5 FROM emp
6 WHERE empno=1) ;
```

في المثال السابق نجد أن الاستعلام الداخلي يحدد مرتب الموظف الذي رقمه هو 1، يأخذ الاستعلام الخارجي أو الرئيسي هذه النتيجة ويستخدمها لاسترجاع الموظفين الذين يتقاضون مرتباً أعلى من هذا المرتب

## 2-5 أنواع الاستعلامات الفرعية Types Of Sub queries

يوجد لدينا نوعين من الاستعلامات الفرعية وهي كالتالي:

### 1-2-5 الاستعلامات الفرعية التي تعيد سجل واحد

(single-row sub queries)

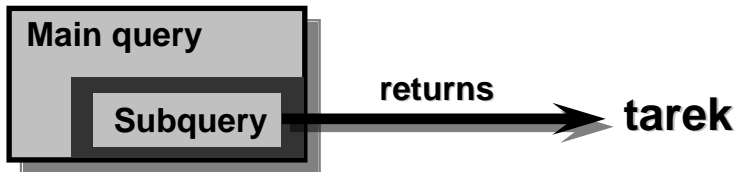
2-2-5 الاستعلامات الفرعية التي تعيد أكثر من سجل واحد

(Multiple-row sub queries)

3-2-5 الاستعلامات الفرعية التي تعيد أكثر من عمود واحد

(Multiple-column sub queries)

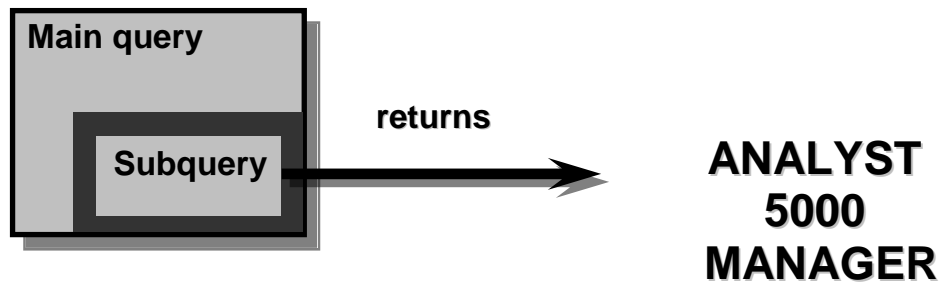
• Single-row subquery



• Multiple-row subquery



• Multiple-column subquery



1-2-5 الاستعلامات الفرعية التي تعيد سجل واحد

كما سبق وتعلمنا أن (single-row sub queries) تعيد سجل واحد فقط ،

ويستخدم ذلك النوع من الاستعلامات مع عوامل المقارنة الموضحة بالجدول التالي :

| المعامل Operator | المعني Meaning   |
|------------------|------------------|
| =                | يساوي            |
| >                | أكبر من          |
| >=               | أكبر من أو يساوي |
| <                | أقل من           |
| <=               | أقل من أو يساوي  |
| <>Or! =          | لا يساوي         |

مثال 1

في هذا المثال سنقوم باسترجاع بيانات الموظفين (الاسم، الوظيفة) للموظفين الذين لهم نفس

الوظيفة للموظف رقم 555

```
SQL>SELECT ename, job
FROM emp
WHERE job=
 (SELECT job
 FROM emp
 Where empno=555) ;
```

| ENAME  | JOB   |
|--------|-------|
| SMITH  | CLERK |
| ADAMS  | CLERK |
| JAMES  | CLERK |
| MILLER | CLERK |

في هذا المثال سنقوم باسترجاع بيانات الموظفين (الاسم، الوظيفة) للموظفين الذين لهم نفس

الوظيفة للموظف رقم 7698 ومرتباتهم أكبر من راتب الموظف رقم 7934

```
SQL> SELECT ename, job
2 FROM emp
3 WHERE job =
4 (SELECT job
5 FROM
6 WHERE empno = 7698)
7 AND sal >
8 (SELECT sal
9 FROM emp
10 WHERE empno = 7934);
```

| ENAME | JOB     |
|-------|---------|
| JONES | MANAGER |
| BLAKE | MANAGER |
| CLARK | MANAGER |

تلميح

✓ لابد من وضع الاستعلام الفرعي بين قوسين.

✓ لا تستخدم جملة Order By داخل الاستعلام الفرعي.

### 5-2-1-1 استخدام الدوال التجميعية في Sub query

تتميز الدوال التجميعية بأنها تولد أو تعيد قيمة وحيدة مهما كان عدد السجلات المختارة، وينبغي

علينا الانتباه إلى أن الدوال التجميعية التي هي دوال تجميع معرفة في جملة Group By ستولد

وتعيد عدة قيم، ولهذا لا يسمح باستخدامها مع هذا النوع من الاستعلامات الفرعية التي تعيد سجل

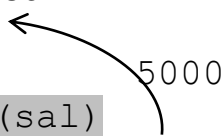


## قواعد البيانات (2)

واحد، حتى لو استخدمنا Group by و Having لتوليد مجموعة وحيدة لخرج الاستعلام الفرعي، ولهذا يجب علينا استخدام الدوال التجميعية المرغوبة في جملة where التي تحذف المجموعات غير المرغوبة.

في هذا المثال سنقوم باسترجاع بيانات الموظف (الاسم، الوظيفة، المرتب) الذي يتقاضى أعلى مرتب، حيث يمكننا تنفيذ هذا الاستعلام باستخدام group function داخل الاستعلام الفرعي كما بالتالي:

```
SQL> SELECT ename, job, sal
2 FROM emp
3 WHERE sal =
4 (SELECT MAX(sal)
5 FROM emp);
```



| ENAME | JOB       | SAL  |
|-------|-----------|------|
| KING  | PRESIDENT | 5000 |

### 2-1-2-5 استخدام (group by و having) مع Sub query

يمكن للاستعلامات الفرعية أن تستخدم في جملة having، ويمكنها استخدام الدوال التجميعية

أو تستخدم group و having كما بالمثال التالي:

مثال في هذا المثال نريد إنشاء مجموعات أصغر مرتبات بناء علي عمود رقم الإدارة ، مع تقييد ذلك بشرط وهو أن يكون أقل مرتب في تلك الإدارات أكبر من أقل مرتب في الإدارة رقم 10

```
SQL> SELECT deptno, MIN(sal)
2 FROM emp
3 GROUP BY deptno
4 HAVING MIN(sal) >
5 (SELECT MIN(sal)
6 FROM emp
7 WHERE deptno = 10);
```

| DEPTNO | MIN(SAL) |
|--------|----------|
| 10     | 1300     |
| 30     | 950      |

✓ يجب علينا التأكد عند استخدام الاستعلامات الفرعية في القسم الشرطي (الذي فيه

عامل المساواة أو عدم المساواة ) من أن الاستعلام الفرعي سيعيد سجلاً واحداً فقط،

وإلا سيصدر خطأ إذا أعاد أكثر من قيمة واحدة، ولن يفشل الأمر إذا لم يعيد

تلميح

الاستعلام الفرعي أية قيمة ولكننا عندها لن نحصل علي خرج من الاستعلام الفرعي.

#### 2-2-5 الاستعلامات الفرعية التي تعيد أكثر من سجل واحد

تعيد الاستعلامات الفرعية متعددة السجلات أكثر من سجل، وتتطلب هذه الأنواع من الاستعلامات

الفرعية معاملاً يمكن استخدامه لتقييم عدة قيم، فعلي سبيل المثال لا الحصر يمكننا استخدام

المعامل IN مع الاستعلامات الفرعية متعددة الأسطر لأنه يقيم مصفوفة من القيم، والجدول

التالي يوضح المعاملات التي يمكننا استخدامها لتقييم الاستعلامات متعددة السجلات.

| المعامل Operator | الوصف                                                                                                            |
|------------------|------------------------------------------------------------------------------------------------------------------|
| IN               | يكون المحدد مساوياً لأي من القيم التي يعيدها الإستعلام الفرعي لهذا الشرط حتي يصبح محققاً (True)                  |
| ANY              | يقارن المحدد بكل قيمة يعيدها الإستعلام الفرعي ، ويكون الشرط محققاً (True) إذا كان أي عنصر من المجموعة يحقق الشرط |
| ALL              | يقارن المحدد بكل قيمة يعيدها الإستعلام الفرعي ، ويكون الشرط محققاً                                               |

مثال

في هذا المثال نريد استرجاع بيانات الموظفين (رقم الموظف، الاسم، الوظيفة) ذوي المرتبات أقل من أي clerk والذين لا يعملون في وظيفة clerk

```
SQL> SELECT empno, ename, job
2 FROM emp
3 WHERE sal < ANY
4 (SELECT sal
5 FROM emp
6 WHERE job = 'CLERK')
7 AND job <> 'CLERK';
```

تلميح

any <: تعني أقل من minimum ✓

any >: تعني أقل من maximum ✓

any =: تعادل تكافئ معامل IN ✓

### 3-2-5 الاستعلامات الفرعية التي تعيد أكثر من عمود واحد

#### (Multiple-column sub queries)

رأينا في الأمثلة السابقة استعلامات تستخدم أعمدة وحيدة كمحددات تقييم، وأحياناً لتنفيذ نوع من الشروط يجب علينا استعراض عدة أعمدة وكأنها محدد واحد، ويجب إحاطة تلك الأعمدة بين قوسين ووضع فواصل بين الأعمدة.

مثال

```
SQL> SELECT ename, deptno, sal, comm
2 FROM emp
3 WHERE (sal, NVL (comm,0)) IN
4 (SELECT sal, NVL (comm,0)
5 FROM emp
6 WHERE deptno = 30);
```

في المثال السابق نريد استرجاع بيانات الموظف (الاسم، رقم الإدارة، المرتب، الحوافز)، لأي موظف يتماثل في المرتب والحوافز مع أي موظف في الإدارة رقم 30.

| ENAME  | DEPTNO | SAL  | COMM |
|--------|--------|------|------|
| JAMES  | 30     | 950  |      |
| WARD   | 30     | 1250 | 500  |
| MARTIN | 30     | 1250 | 1400 |
| TURNER | 30     | 1500 | 0    |
| ALLEN  | 30     | 1600 | 300  |
| BLAKE  | 30     | 2850 |      |

6 row selected

## الأسئلة

### المجموعة الأولى:

ضع علامة ( / ) أمام العبارة الصحيحة وعلامة ( x ) أمام العبارة الخاطئة مع تصحيح الخطأ.

- 1) ينفذ الاستعلام الفرعي أو الداخلي أولاً ثم ينفذ الاستعلام الرئيسي أو الخارجي
- 2) في الرابطة المتكافئة لابد أن كل سجل في الجدول الرئيس الأب يقابله سجل أو سجلات في الجدول المرتبط به الابن.
- 3) الرابطة غير المتكافئة تمتلك الجداول أعمدة وقيماً مشتركة.
- 4) في الرابطة الخارجية تكون السجلات المرتبطة يمكن أن تحتوي قيم مشتركة ويمكن أن لا تحتوي.
- 5) الروابط الذاتية Self join تمكننا من عمل رابطة ذاتية عند ربط الجدول مع نفسه.
- 6) يطلق علي العلاقة inner join simple join equi join أو inner join
- 7) في جملة Where يمكننا استخدام TABLE Aliases.
- 8) تقوم الدوال التجميعية Group Functions بمعالجة قيم العمود المختار من الجدول وتقدم النتيجة في شكل قيمة وحيدة تخص العمود المختار.
- 9) `Select MAX(avg(sal)) from emp;`
- 10) (\*) Count تقوم بحساب عدد السجلات بما فيها السجلات التي تحتوي علي قيم فارغة.
- 11) يمكن لجميع الأعمدة في جملة select أن تكون كلها دوال تجميعية.
- 12) يمكن استخدام الدوال التجميعية ضمن التعابير الحسابية.

13) يستخدم AVG مع القيم العددية فقط ويهمل AVG القيمة NULL عند حسابه للقيمة الوسطي.

14) To\_number يستخدم لتحويل نوع البيانات إلى نوع البيانات الحرفية.

15) To\_char يستخدم لتحويل نوع البيانات الحرفية إلى نوع البيانات الرقمية.

### المجموعة الثانية:

#### أسئلة لتدريب الطلاب على أسئلة الاختيار المتعدد:

1) من الأسباب التي تدعونا إلى إنشاء العلاقات

(a) دمج الأعمدة من جدولين أو أكثر.

(b) إختيار أعمدة موجودة في جدول واحد بناء علي شرط يطبق علي عمود آخر.

(c) 1&2.

(d) ليس شيء مما سبق.

2) عند استرجاع بيانات من جدولين أو أكثر يتم إضافة شرط الربط في عبارة Where.

(a) العبارة صحيحة.

(b) العبارة خاطئة.

3) من أنواع العلاقات بين الجداول

(a) الرابطة المتكافئة.

(b) الرابطة الذاتية.

(c) الرابطة الخارجية .

(d) جميع ما سبق.

4) في الرابطة الخارجية Outer join:

(a) تمتلك الجداول أعمدة مشتركة.

(b) السجلات يمكن أن تحتوي قيم مشتركة أولا تحتوي.

(c) 2&1

(d) ليس شيء مما سبق .

5) الرابطة الديكارتية تقوم :-

- a) بحذف شرط الربط بين الجدولين في جملة Where.
- b) يقوم أوراكل بربط كل سجل في الجدول الأول مع كل سجل في الجدول الثاني.
- c) يقوم أوراكل بإنشاء ومعالجة كمية كبيرة جداً من البيانات
- d) جميع ما سبق

6) هناك عدة اعتبارات يجب إتباعها عند استخدام group functions من:

- a) وضع إسم العمود بين قوسين ( ) بعد الدالة مباشرة
- b) يمكن لجميع الأعمدة في جملة Select أن تكون كلها دوال تجميعية
- c) استخدام دالة تجميعية أخرى غير مسموح به في SQL
- d) الدوال التجميعية تتجاهل القيم الفارغة NULLVALUES
- e) يمكن استخدام الدوال التجميعية ضمن التعابير الحسابية
- f) جميع ما سبق

7) الدالة MAX:

- a) يمكن استخدام max مع أي نوع من البيانات.
- b) تهمل الدالة MAX القيم NULL.
- c) 2&1
- d) يمكن استخدام max مع نوع البيانات الحرفية فقط.

8) الدالة AVG:

- a) يستخدم AVG مع القيم العددية فقط.
- b) يهمل AVG القيمة NULL عند حسابه للقيمة الوسطي.
- c) 2&1
- d) يستخدم AVG مع أي نوع من البيانات.

9) الدالة MONTH\_BETWEEN:

- (a) تحدد عدد الأشهر بين تاريخين.
- (b) يحدد اليوم التالي من الشهر القادم الذي يلي التاريخ المحدد.
- (c) يحدد تاريخ جديد بناء علي عدد محدد من الأشهر
- (d) 2&1

10) الدالة LAST\_DAY:

- (a) تحدد عدد الأشهر بين تاريخين.
- (b) يحدد اليوم التالي من الشهر القادم الذي يلي التاريخ المحدد.
- (c) يحدد تاريخ جديد بناء علي عدد محدد من الأشهر
- (d) يحدد آخر يوم في الشهر الحالي أو التاريخ المحدد.

11) هناك عدة اعتبارات يجب مراعاتها عن استخدام الدالة nvl:

- (a) تستخدم الدالة NVL في تحويل القيم الفارغة إلي قيم فعلية.
- (b) يمكنها معالجة أنواع البيانات الرقمية NUMERIC DATA، والنصية CHARACTER STRING، والتاريخية DATE.
- (c) عند تحويل القيم الفارغة عليك بالالتزام بنفس نوع البيانات Data types في العمود الذي يحتوي القيم الفارغة.
- (d) A&b
- (e) A&b&c



## 12 group by:

- (a) تتعامل GROUP BY مع القيم الفارغة NULL كمجموعة مميزة عن غيرها.
- (b) يمكننا GROUP BY بإنشاء مجموعات فرعية ضمن مجموعات فرعية أخرى
- (c) A&b
- (d) ليس شيء مما سبق.

## 13 HAVING:

- (a) نستخدم Having مع الدوال التجميعية التالية فقط
- (b) يمكننا استخدام كلاً من Where، Having في جملة Select.
- (c) تقع جملة Having بعد فقرة Group by.
- (d) A&B&C
- (e) A&C

## 14 HAVING:

- (f) نستخدم Having مع الدوال التجميعية التالية فقط
- (g) يمكننا استخدام كلاً من Where، Having في جملة Select.
- (h) تقع جملة Having بعد فقرة Group by.
- (i) A&B&C
- (j) A&C

# الفصل الرابع

## معالجة البيانات باستخدام

# SQL

## الأهداف:

إعطاء الطالب صورة واضحة عن كيفية إضافة سجلات جديدة إلي الجداول، وكيفية حذف وتعديل بيانات موجودة في قاعدة البيانات، وكيفية إنشاء وإضافة محددات من نوع

Primary key, foreign key

## المحتويات:

1. مقدمة

2. كيفية إضافة سجلات جديدة للجداول.

3. كيفية حذف بيانات موجودة في قاعدة البيانات .

4. كيفية تعديل البيانات الموجودة في قاعدة البيانات .

5. كيفية إنشاء جدول وإضافة محددات من نوع

(primary key, foreign key)

## ماذا سنتعلم في هذا الفصل :

في نهاية هذا الفصل يكون الطالب قد اكتسب المهارات والمعارف التالية :

1. إنشاء وتعديل وإسقاط الجداول
2. كيفية إضافة سجلات جديدة للجداول.
3. كيفية حذف بيانات موجودة في قاعدة البيانات .
4. كيفية تعديل البيانات الموجودة في قاعدة البيانات .
5. كيفية إنشاء جدول وإضافة محددات من نوع  
(primary key, foreign key)

## مقدمة

كما تعلمنا سابقاً أن عبارات SQL تنقسم إلى ثلاث فئات رئيسية وهي كالتالي:

### 1- لغة تعريف البيانات DDL

هي مجموعة من أوامر SQL تستطيع من خلالها إنشاء وتعريف الكائنات في قاعدة البيانات، وتقوم هذه الأوامر بإنشاء أو إسقاط أو تغيير كائن قاعدة البيانات، وهي اختصاراً للكلمات (Data Definition Language)، ومن أهم أوامرها Create, Alter, Drop, Rename, Truncate

### 2- لغة معالجة البيانات DML

هي مجموعة من العبارات التي تستطيع من خلالها معالجة البيانات الموجودة في قاعدة البيانات من إدراج البيانات وتحديثها أو تحديد أو حذف البيانات، وهي اختصاراً للكلمات (Data Manipulation Language)، ومن أهم أوامرها Insert, Update, Delete

### 3- لغة التحكم والصلاحيات DCL

هي مجموعة من أوامر SQL تستطيع من خلالها منح أو سحب إمتيازات استخدام كائن قاعدة البيانات، وهي اختصاراً للكلمات (Data Control Language)، ومن أهم أوامرها Grant, Revoke

سنتعلم في هذا الفصل كيفية التعامل مع البيانات المخزنة في قاعدة البيانات من إضافة سجلات أو حذف سجلات أو تغيير قيمة مخزنة في سجل، بالإضافة إلى كيفية إنشاء الجداول وإضافة

محددات من نوع primary key, foreign key

## 1- كيفية إضافة سجلات جديدة

تمتلك معظم برامج أنظمة قواعد البيانات أدوات ممتازة لإدخال وتعديل وحذف البيانات من الجداول، وهي بلا شك أفضل وأسرع بكثير من إدخال البيانات بعبارات sql، إلا أن ذلك يمكن المبرمج من معالجة البيانات من ضمن برامجه التي يكتبها، ونستطيع إضافة سجلات جديدة إلي الجدول باستخدام عبارة insert، ويوجد لدينا صيغتين من عبارة insert، الأولى تسمح بإضافة سجل واحد في نفس الوقت إلي الجدول، والثانية تسمح باختيار مجموعة سجلات من جدول وإدراجها في جدول آخر.

### 1-1 إضافة سجل واحد إلي الجدول

كما وضعنا سابقا أن عبارة insert لها صيغتين الأولى والتي نحن بصددتها تمكننا من إضافة سجل واحد إلي الجدول في نفس الوقت، والصيغة العامة لها تأخذ الشكل التالي:

```
INSERT INTO table [(column [, column...])]
VALUES (value [, value...]);
```

مما سبق نجد أن الصيغة العامة لعبارة insert تتألف مما يلي:

1. insert, into, values هي كلمات محجوزة في sql

2. Table ويشير إلي اسم جدول وحيد في قاعدة البيانات ، وبالتحديد الجدول الذي نريد

إدراج السجلات به.

3. Column: وتشير إلي أعمدة الجدول.

4. لا ئحة من القيم محاطة بين قوسين تقوم sql بإدراجها في الأعمدة المقابلة لها علي

الترتيب.

## مثال 1

لإدراج سجل من البيانات إلى جدول dept، نقوم بكتابة الكود التالي:

```
SQL> INSERT INTO dept (deptno, dname, loc)
 2 VALUES (50, 'DEVELOPMENT', 'cairo');
1 row created.
```

## مثال 2

لإدراج سجل من البيانات إلى جدول emp، نقوم بكتابة الكود التالي:

```
SQL> INSERT INTO EMP (empno, ename, job,
 2 mgr, hiredate, sal, comm,
 3 deptno)
 4 VALUES (7369, 'ayman', 'programmer',
 5 7372, SYSDATE, 4000, NULL,
 6 50);
1 row created.
```

## تلميح

✓ يجب أن تتطابق الأعمدة والقيم المقابلة لها في نوع البيانات data types.

✓ لائحة القيم لابد أن تأخذ نفس ترتيب الأعمدة في الجدول المراد إضافة السجلات

إليه.

✓ لابد من وجود علاقة واحد إلى واحد بين الأعمدة ولائحة القيم، بمعنى أن توجد قيمة

واحدة من أجل كل عمود وبنفس الترتيب.

✓ قد نجهل قيمة أحد الحقول، وذلك بكتابة كلمة NULL، بشرط ألا يكون العمود

المقابل للقيمة معرّفاً بالخاصية NOT NULL.

✓ قد تسبب INSERT مشكلة في التجانس والتكامل المرجعي (وسيتناول ذلك عند

إنشاء العلاقات بين الجداول لاحقاً في هذا الفصل).

✓ لابد من وضع الأحرف النصية والتاريخ بين علامتي تنصيص مفردة ' ' .

✓ للتأكد من تنفيذ العبارة السابقة وإضافة السجل إلي جدول emp قم بتنفيذ العبارة

التالية:

```
SQL>Select * From emp;
```

### 2-1 إضافة عدة سجلات إلي الجدول:

نلاحظ أنه ليس من المنطقي أن تقوم بإدخال 20,000 سجلاً مثلاً في أحد الجداول سجلاً

سجلاً باستخدام الصيغة الأولى من عبارة insert، ولإضافة عدة سجلات في نفس الوقت

تمكنا الصيغة الثانية من عبارة insert بإضافة عدة سجلات إلي جدولنا مختارة من

جدول آخر ، والصيغة العامة لها هي:

```
INSERT INTO table([(column [, column...])
Select ([(column [, column...])])
[where conditions]);
```

مما سبق

نجد أن الصيغة العامة لعبارة insert تتألف مما يلي:

1. insert, into هي كلمات محجوزة في Sql

2. Table ويشير إلي اسم جدول وحيد في قاعدة البيانات ، وبالتحديد الجدول الذي نريد

إدراج السجلات به.

3. Column: وتشير إلي أعمدة الجدول.

4. عبارة select: هي عبارة select عادية تستخدم لانتخاب الأعمدة التي ستستخ

محتوياتها إلي الجدول المراد إضافة السجلات إليه ، فعلي سبيل المثال نريد إضافة



سجلات أمان إلي الجدول secemp من أجل كل سجل من سجلات جدول emp

فعلي سبيل المثال

```
SQL>INSERT INTO secemp
(empno,name,job,mgr,sal,comm,deptno)
(Select (mpno,name,job,mgr,sal,comm,deptno)
From emp;
```

## 2-حذف بيانات موجودة في قاعدة البيانات (Deleting Record)

نستخدم عبارة DELETE لحذف سجلات من جدول ما، وتأخذ الصيغة العامة لعبارة

DELETE الشكل التالي:

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

مما سبق نجد أن الصيغة العامة لعبارة Delete تتألف مما يلي:

1. Delete: هي أحد الكلمات المحجوزة في Sql.

2. From : هي أحد الكلمات المحجوزة في Sql ويليه اسم الجدول المراد حذف السجلات

منه

3. WHERE: هي أحد الكلمات المحجوزة في Sql ويليه شرط حذف السجلات ، وهي عبارة

اختيارية للحد من عدد السجلات المحذوفة ، فإذا تم حذف عبارة where من جملة

delete فإن هذا يعني حذف كافة السجلات

## 2-1 حذف سجلات محددة (Deleting SEPECIFIC Record)

مثال 1

لحذف السجلات الموظفين ذوي المرتبات أكبر من 5000 من جدول emp يتم كتابة

العبارة التالية

```
SQL> DELETE FROM emp
Where sal > 5000;
```

**2-2 حذف كافة السجلات (Deleting ALL Record)**

يمكننا حذف كافة السجلات من جدول ما بحذف عبارة where من جملة delete فإن هذا يعني حذف كافة السجلات ، والصيغة العامة لها هي:

```
SQL>DELETE [FROM] table;
```

مثال

لحذف كافة السجلات من جدول emp يتم كتابة العبارة التالية

```
SQL> DELETE FROM emp;
```

**3-2 استخدام الاستعلام الفرعي في جملة DELETE****(sub queries in DELETE statements )**

يمكننا استخدام الاستعلام الفرعي في جملة Delete، وذلك لحذف سجلات من جدول

بناء علي قيم من جدول آخر، فعلي سبيل المثال إذا أردنا حذف سجلات الموظفين التابعين لإدارة

المبيعات يتم ذلك علي النحو التالي

مثال

```
SQL> DELETE FROM employee
2 WHERE deptno =
3 (SELECT deptno
4 FROM dept
5 WHERE dname='sales');
5 rows deleted.
```

تلميح

باستخدام Delete نستطيع حذف بيانات جدول واحد فقط في نفس الوقت.

✓ تحذف Delete جميع السجلات التي تحقق الشرط في جملة Where.

✓ لا تحذف Delete الجدول.

✓ تحذف السجلات من الجدول بشكل مستمر، إلا أن الحذف الحقيقي للسجلات

يعتمد علي معالجة commit المستخدمة التي سيتم تناولها في هذا الفصل

✓ قد تسبب delete مشكلة في التجانس والتكامل المرجعي (وسيتم تناول ذلك

عند إنشاء العلاقات بين الجداول لاحقاً في هذا الفصل).

✓ تحذف delete السجل أو السجلات بالكامل فلا حاجة للإشارة إلى أي عمود

إلا في جملة Where عند استخدامها.

## 2- تعديل البيانات الموجودة في قاعدة البيانات

### (Using the update command)

يمكننا استخدام update لتعديل القيم الموجودة في جدول ما، والصيغة العامة لعبارة update تأخذ الشكل التالي:

```
SQL> UPDATE table
 SET column = value [, column = value]
 [WHERE condition];
```

مما سبق نجد أن الصيغة العامة لعبارة update تتألف مما يلي:

1. Update: من الكلمات المحجوزة في، وهو أمر بتحديث البيانات في Sql.

2. Table: ويشير إلى اسم الجدول المراد التعديل في بياناته.

3. Set : وتحتوي على سلسلة من عمليات التحديث ، حيث نقوم بوضع اسم العمود

الذي نقوم بتعديل بياناته كمعامل أول ، ويحتوي المعامل الثاني على القيمة الجديدة ،

وفصل بين المعاملين إشارة المساواة.

4. Where: وهي عبارة اختيارية يتم استخدامها لتحديد السجلات التي يراد التعديل بها ،

فإذا تم حذف عبارة Where فإن التعديل سيتم على كل سجلات الجدول.

### 3-1 تحديث كافة السجلات (Update ALL Record)

يمكننا تحديث كافة السجلات من جدول ما بحذف عبارة where من جملة Update فإن هذا يعني تحديث كافة السجلات، والصيغة العامة لها هي:

```
SQL> UPDATE table
SET column = value [, column = value];
```

مثال

لتحديث كافة السجلات من جدول emp وذلك بوضع المرتب 3000 لكل الموظفين يتم كتابة العبارة التالية:

```
SQL> UPDATE employee
 2 SET sal = 3000;
14 rows updated.
```

### 3-2 تحديث سجلات محددة (Update Specific Record)

يمكننا تحديث سجلات محددة والتعديل فيها بناء علي شرط أو عدة شروط، وذلك بتضمين عبارة Update فقرة where، فعلي سبيل المثال لتعديل مرتب الموظف رقم 555 من 3000 إلي 6000 يتم ذلك علي النحو التالي:

```
SQL> UPDATE emp
 2 SET sal = 6000
 3 WHERE empno = 555;
1 row updated.
```

### 3-3 استخدام الاستعلام الفرعي في جملة update

( sub queries in update statements )

يمكننا استخدام الاستعلام الفرعي في جملة update فعلي سبيل المثال إذا أردنا تعديل أو تحديث كلا من الوظيفة والإدارة للموظف رقم 555 لتمثيل الوظيفة والإدارة للموظف رقم 7489، سوف نحتاج إلي استخدام استعلام فرعي، ويتم كتابة عبارة update علي النحو التالي:

```
SQL> UPDATE emp
2 SET (job, deptno) =
3 (SELECT job, deptno
4 FROM emp
5 WHERE empno = 7489)
6 WHERE empno = 555;
1 row updated.
```

#### 4- استخدام تعليمات التثبيت commit والتراجع Rollback

يتطلب تنفيذ التغييرات في بيانات الجداول خطوتين:

(1) تنفيذ إحدى العبارات التالية Insert, Update, Delete.

(2) تأكيد التغيير باستخدام commit.

عندما يتم تنفيذ عبارة من عبارات DML، يقوم أوراكل بتغيير الجدول وحفظ نسخة من السجلات قبل إجراء التعديل، وسيبدو الأمر للمستخدم وكأن التغييرات قد تمت بشكل نهائي ودائم، فإذا قام المستخدم بالاستعلام في قاعدة البيانات، فإنه سيري التعديلات موجودة، فإذا قام مستخدم آخر بالاستعلام في قاعدة البيانات فلن يري تلك التعديلات، وذلك لأن التعديلات علي البيانات لا تصبح مرئية من قبل المستخدمين الذين يسمح لهم بالوصول إلي تلك الجداول، ولا تصبح دائمة حتى يتم تنفيذ تعليمة COMMIT، بمعنى أن تلك الخاصية تسمح للمستخدم بالتراجع عن أي تغيير قام به. ويتم التراجع عن التعديلات باستخدام تعليمة Rollback. حيث تعيد هذه التعليمة قاعدة البيانات إلي الحالة التي كانت عليها بعد تنفيذ آخر تعليمة commit.

✓ تلميح يتم تنفيذ تعليمة commit بشكل ضمني إذا قطع المطور الاتصال مع نظام

أوراكل بشكل نظامي أو نفذ تعليمة من تعليمات DDL، حيث يؤدي ذلك إلي حفظ

التغييرات.

## قواعد البيانات (2)

✓ إذا قطع الاتصال مع نظام أوراكل بشكل غير نظامي قبل تنفيذ تعليمة

COMMIT لن يتم حفظ التغييرات، وستعود قاعدة البيانات إلى النقطة التي تم

عندها تنفيذ آخر تعليمة COMMIT.

✓ يمكن التراجع عن التغييرات بشكل جزئي، حيث تسمح تعليمة SAVEPOINT

للمستخدم حفظ التغييرات حتى جزء محدد من التغييرات، فعلي سبيل المثال

نفترض أنك قمت بمجموعة من التعديلات علي مجموعة من السجلات، ولكنك

غير متأكد من أنك أجريت التعديلات بشكل صحيح، وترغب في رؤية السجلات

قبل تثبيت هذه التغييرات، ففي هذه الحالة يمكنك استخدام عدة تعليمات

Savepoint، وبالتالي يمكنك التراجع عن التغييرات حتى أي نقطة من

النقاط العلام الموضوعية كما هو موضح بالشكل التالي ، والصيغة العامة لكلاً من

نقطة العلام Savepoint والتراجع Rollback إلى نقطة علام تأخذ

الشكل التالي :

```
SAVEPOINT save_point_name
ROLLBACK save_point_name
```

مثال 1

لتحديث كافة السجلات من جدول emp وذلك بوضع المرتب 3000 لكل

الموظفين يتم كتابة العبارة التالية:

```
SQL> UPDATE employee
2 SET sal = 3000;
14 rows updated.
```

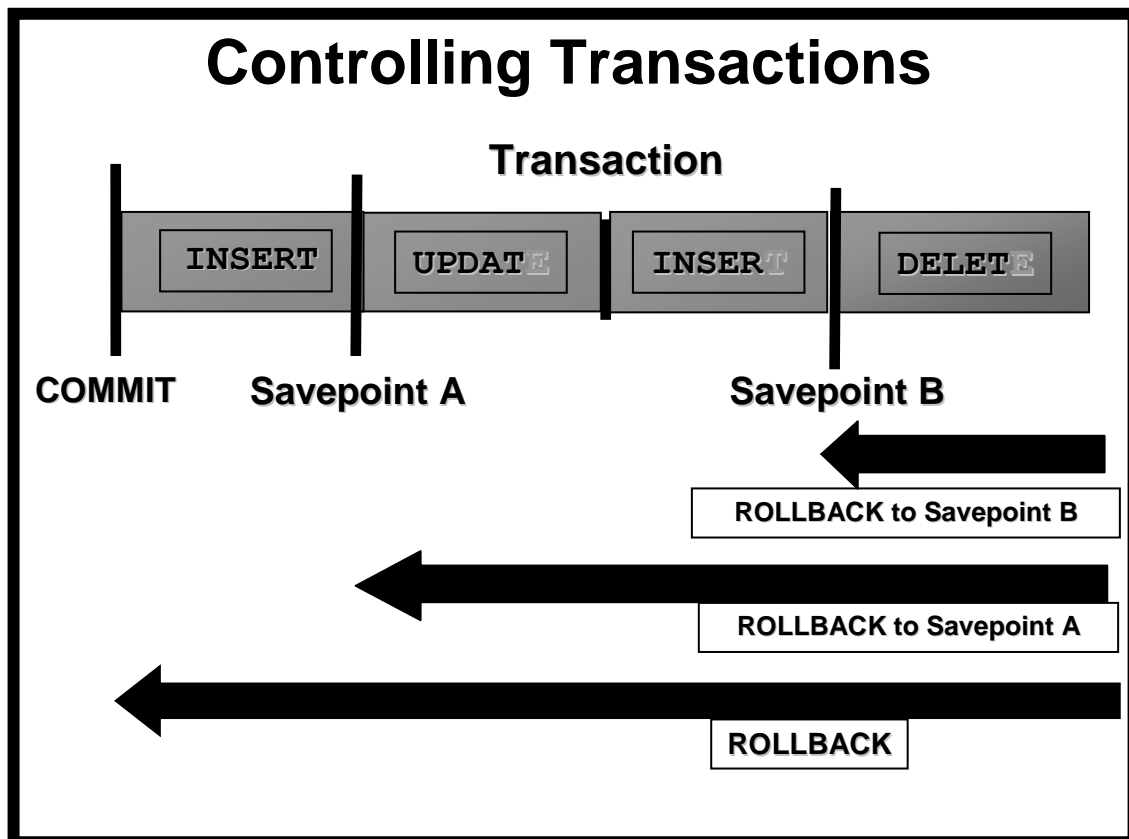
ولحفظ التعديلات في قاعدة البيانات يتم كتابة العبارة التالية:

```
SQL> COMMIT;
Commit complete.
```

لحذف كافة السجلات من جدول emp ثم التراجع عن ذلك يتم بكتابة العبارات التالية:

```
SQL> DELETE FROM emp;
14 rows deleted.
SQL> ROLLBACK;
Rollback complete.
```

مثال 2



مثال 3

```
SQL> UPDATE...
SQL> SAVEPOINT jicc_ update;
Savepoint created.
SQL> INSERT...
SQL> ROLLBACK TO jicc_ update;
Rollback complete.
```

## 5- كيفية إنشاء جداول وإضافة محددات من نوع

### **(primary key, foreign key)**

مما سبق نجد أنه يوجد في لغة SQL ثلاث عبارات تسمى بعبارات تعريف البيانات وهي

تسمح لنا بتعريف ومحي وتعديل الجداول في قاعدة البيانات، وهذه العبارات هي:

1. CREATE: تستخدم لتعريف جدول جديد في قاعدة البيانات.

2. DROP : تستخدم لمحي جدول موجود من قاعدة البيانات.

3. ALTER: تستخدم لتغيير بنية الجدول الموجود.

قواعد البيانات العلائقية قادرة علي تنفيذ كافة عبارات SQL في أي وقت بشرط أن يكون

له الحق في تنفيذ تلك الأوامر والعبارات، فلكي تقوم بإنشاء ومحي وتعديل الجداول لابد

من أن تتأكد من مدير قاعدة البيانات أنه يمنحك حق تعريف ومحي وتعديل الجداول

## 5-1 إنشاء وتعديل وإسقاط الجداول

### 5-1-1 إنشاء الجداول (create table)

تمكننا عبارة create من إنشاء الجداول في SQL، وتأخذ عبارة Create الشكل التالي:

```
SQL> CREATE TABLE [schema.]Table
 (column datatype [DEFAULT expr]);
```

مما سبق نجد أن الصيغة العامة لعبارة Create تتألف مما يلي:

1-create: من الكلمات المحجوزة في SQL، وتخير DBMS بإنشاء كائن في قاعدة

البيانات

2-Table: من الكلمات المحجوزة في SQL، وتخير DBMS بنوع الكائن المراد إنشائه

في قاعدة البيانات



3-Table name: ويستخدم لتحديد اسم الجدول ، ويجب أن يبدأ اسم الجدول

بحرف نصي ويمكن أن يحتوي أحرف وأرقام والشرطة ،ويجب ألا يتجاوز طوله 30

حرفاً،ولايسمح بالفراغات ،ولكن يسمح بالشرطة السفلية (underscore).

4-Column names: وتستخدم لتحديد أسماء الأعمدة المكونة للجدول المراد

إنشائه، وتنطبق نفس القواعد الخاصة بالتسمية علي اسم الجدول علي أسماء الأعمدة

، ويجب أن تحاط مجموعة تعريف الأعمدة بين قوسين.

5-Column data type: يحتوي تعريف الأعمدة علي نوع البيانات ،وطولها

ودقتها كما هي موضح بالجدول التالي

6-Constraints: وهي إعدادات اختيارية يمكن استخدامها لتمثل قيود للمحافظة

علي تكامل قاعدة البيانات.

وهناك عدة اعتبارات يجب مراعاتها عند إنشاء الجداول وهي كالتالي:

1-يجب وضع أقواس تضم أسماء الأعمدة ونوع بياناتها.

2-يتم وضع فاصلة ( , ) بين تعريف أي عمود وآخر.

3-يجب أن يكون اسم العمود فريدا داخل الجدول.

4-لايمكن استخدام الكلمات المحجوزة كأسماء أعمدة في الجدول.

5-يتم استخدام الفاصلة المنقوطة لإخبار SQL أن الجملة انتهت وجاهزة للتنفيذ.

6-الجدول التالي يوضح لنا أهم أنواع البيانات التي يمكن أن نستخدمها عند إنشاء

الجدول في SQL

| نوع البيانات  | الوصف                                                                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHAR (n)      | يعرف هذا النوع من البيانات عمود من نوع البيانات الحرفية بطول (n) حرف، وجميع الحقول متساوية في الطول، $n \leq 255$                                                                                                         |
| VARCHAR (n)   | يعرف هذا النوع من البيانات عمود من نوع البيانات الحرفية بطول (n) حرف، وتختلف الحقول في الطول وتتجاوز 255                                                                                                                  |
| NUMBER (n)    | يعرف عمود من نوع بيانات من نوع رقم، مع حد أقصى لعدد الأرقام (n)، ويمكن أن تأخذ (n) كحد أقصى 105                                                                                                                           |
| NUMBER (n, d) | يعرف عمود من نوع البيانات number، ويحتوي عدد أرقام قدره (n)، والتي تتضمن القيمة (d) التي توضح عدد الأرقام العشرية بعد الفاصلة.                                                                                            |
| INTEGER       | يشبه نوع البيانات NUMBER، ولكنه لا يأخذ غير القيم الرقمية الصحيحة، ولا يقبل أي أرقام عشرية بعد الفاصلة.                                                                                                                   |
| LONG          | يعرف عمود من نوع البيانات الحرفية، وأقصى طول له 65.535، ولا يمكن تعريف سوي عمود واحد من هذا النوع في الجدول، ولا يمكن استخدام هذا العمود ضمن عبارات Where أو في الاستعلامات الفرعية أو الدوال أو التعابير أو ضمن الفهارس. |
| RAW (n)       | يعرف هذا النوع عمود يحتوي سجل بيانات بطول قدره n                                                                                                                                                                          |
| SMALLINT      | يعرف عمود يخزن أرقاماً موجبة وسالبة صغيرة تقع في النطاق (-32, 767, +32, 767)                                                                                                                                              |
| DATE          | يعرف عمود نوع بياناته تاريخ.                                                                                                                                                                                              |

## مثال 1

نقوم بإنشاء جدول department، الموضح بالكود التالي أسماء أعمدته ونوع بياناتها

```
SQL> CREATE TABLE department
2 (deptno NUMBER (2) not null,
3 dname VARCHAR2 (15),
4 loc VARCHAR2 (20));
Table created.
```

ولعرض مواصفات وتعريفات الأعمدة نستخدم عبارة describe أو اختصارا desc وتأخذ الشكل التالي:

```
SQL> describe Table name;
```

```
SQL> desc department;
```

| Name   | Null     | Type         |
|--------|----------|--------------|
| DEPTNO | NOT NULL | NUMBER(2)    |
| DNAME  |          | VARCHAR2(15) |
| LOC    |          | VARCHAR2(20) |

### 5-1-1-1 إنشاء الجداول باستخدام الاستعلام الفرعي

يمكنك إنشاء جدول باستخدام الاستعلام الفرعي كما بالمثل التالي:

```
SQL> CREATE TABLE dept10
2 AS
3 SELECT empno, ename, sal*12 ANNSAL,hiredate
4 FROM emp
5 WHERE deptno = 10;
Table created.
```

نلاحظ وجود كلمة as بين اسم الجدول الجديد المراد إنشائه والاستعلام الفرعي، للتأكد من

مواصفات وتعريفات الأعمدة column definitions نستخدم الأمر describe

```
SQL> describe dept10;
```

```
SQL> describe dept10;
```

| Name     | Null | Type         |
|----------|------|--------------|
| EMPNO    |      | NUMBER(4)    |
| ENAME    |      | VARCHAR2(10) |
| ANNSAL   |      | NUMBER       |
| HIREDATE |      | DATE         |

**2-1-5 تعديل الجدول (altering the table)**

يستخدم `alter table` في تعديل بنية جدول موجود وتغيير محددات الجدول بعد إنشائه

وفيما يلي الخيارات التي يمكن استخدامها مع الأمر `alter`:

(1) `Add`: يستخدم هذا الخيار لإضافة أعمدة جديدة أو شروط إلى الجدول ويأخذ الشكل

التالي:

```
SQL> ALTER TABLE table
 ADD (column datatype [DEFAULT expr]
 [, column datatype]...);
```

(2) `Modify`: يستخدم لتغيير محددات موجودة ويأخذ الشكل التالي:

```
SQL> ALTER TABLE table
 MODIFY (column datatype [DEFAULT expr]
 [, column datatype]...);
```

(3) `Disable`: يستخدم لإلغاء تفعيل شرط ما في الجدول.

(4) `Enable`: يستخدم لإعادة تفعيل شرط كنا قد قمنا بإلغاء تفعيله.

(5) `Drop`: لإزالة شرط ما بشكل دائم من الجدول.

(6) `Set unused`: يستخدم لتحديد عمود أو أكثر كأعمدة غير مستخدمة ، بحيث

يمكن حذف تلك الأعمدة من الجدول عندما تصبح مصادر النظام ملائمة لذلك.

(7) `Drop column`: يستخدم لإزالة عمود ما من الجدول

مثال 1

لإضافة عمود `job` إلى الجدول `dept10` نقوم بكتابة الكود التالي:

```
SQL> ALTER TABLE dept10
 2 ADD (job VARCHAR2(9));
Table altered.
```

للتعديل في نوع البيانات dala type الخاصة بعمود الename في جدول dept10  
نقوم بكتابة الكود التالي:

```
SQL>ALTER TABLE dept30
MODIFY (ename VARCHAR2(15));
Table altered.
```

### 3-1-5 حذف الجداول (Dropping table)

تستخدم Sql عبارة drop لحذف الجداول من النظام، فكل ما تحتاجه لحذف جدول هو أن  
تكتب اسمه بعد drop table فيتم حذف الجدول نهائياً بجميع سجلاته وفهارسه، ولا يمكن  
النظام فرصة لاستعادة الجدول، وتأخذ جملة Drop الشكل التالي:

```
SQL> DROP TABLE table_name;
```

لحذف جدول dept10 نقوم بكتابة الكود التالي

```
SQL> DROP TABLE dept10;
```

### 2-5 إضافة محددات إلى الجدول من نوع

#### **(Primary key, foreign key)**

#### 1-2-5 المحافظة على تكامل قاعدة البيانات

يوجد في أوراكل بعض الأدوات لحماية قاعدة البيانات، تسمى هذه الأدوات بالشروط  
Constraints، وتقوم هذه الشروط بعدة وظائف هامة منها:

(1) التأكد من المفتاح الأساسي أو الرئيسي unique.

(2) فرض التكامل المرجعي بمعنى التأكد من أن السجلات الأبناء الموجودة في جداول

مرتبطة، تمتلك سجل أب.

3) التأكد من أن الأعمدة المرتبطة بالشروط تحتوي دائماً قيمة بداخلها NOT NULL.

4) التأكد من وضع القيمة الافتراضية DEFAULT VALUE في عمود ما.

5) التأكد من عموداً ما يحتوي علي قيمة ما، وأن هذه القيمة موجودة ضمن نطاق محدد

من القيم The Check Constraint

#### 5-2-1-1 الشرط CHECK (فحص وجود قيمة)

يستخدم هذا الشرط للتأكد من أن قمة ما لعمود ما تقع بين قيم مجموعة محددة، عندما تستخدم

الشرط CHECK سيقوم أوراكل بمقارنة أية قيمة يتم إدخالها مع مجموعة القيم المحددة في

المجموعة.

مثال

نستخدم الشرط CHECK في تعريف جدول employee التالي، تم إعطاء إسم للشرط

check وهو (gender\_validation)، وتسمية الشرط يكون اختياريًا، فإذا لم يتم

تسمية الشرط فإن أوراكل سيولد إسمًا خاصاً للشرط.

```
SQL> CREATE TABLE employee
2 (emp_id NUMBER (2) not null,
3 name VARCHAR2 (15),
4 gender char (1)
5 Constraint gender_validation check (gender
6 in('M', 'F')));
```

نلاحظ أن أوراكل سيقوم بمقارنة أية قيمة يتم إدخالها مع مجموعة القيم المحددة في

المجموعة ('M', 'F') فقط في العمود gender بمعنى ألا يسمح ألا بهاتين القيمتين فقط.

**2-1-2-5 خيار القيمة الافتراضية (default option)**

يستخدم هذا الخيار default لإعطاء قيمة افتراضية لعمود ما عندما لا يتم وضع قيمة في هذا

العمود، ولا يمنع هذا الخيار من جعل القيمة null في هذا العمود

مثال

في هذا المثال تم استخدام الخيار default للتأكد من العمود comm تم إعطائه القيمة

صفر عند عدم وضع أي قيمة به، ويكون ذلك مفيداً في معالجة القيم الفارغة null

عند إجراء العمليات الحسابية.

```
SQL> CREATE TABLE employee
2 (emp_id NUMBER (2) not null,
3 name VARCHAR2 (15),
4 gender char (1)
5 sal number
6 comm default (0),
Constraint gender_validation check (gender
in('M','F')));
```

**3-1-2-5 الشرط not null (ليس فارغاً)**

نستخدم الشرط not null للتأكد من أن العمود دوماً سيحتوي قيمة بداخله، وخاصة ما يتم

وضع هذا الشرط علي عمود أو أعمدة المفتاح الأساسي للجدول ، لكي تقوم بوضع الشرط not

null علي عمود ما ، قم بوضع العبارة not null بعد نوع بيانات العمود كما بالمثال

التالي:

```
SQL> CREATE TABLE employee
2 (emp_id NUMBER (2) not null,
3 name VARCHAR2 (15),
4 gender char (1)
5 sal number
6 comm default (0),
Constraint gender_validation check (gender
in('M','F')));
```

**5-1-2-4 unique الشرط (وحيد)**

يستخدم الشرط unique للتأكيد من أن القيمة الموجودة بالعمود المشروط هي قيمة وحيدة في جميع سجلات الجدول، بمعنى عدم السماح بتكرار القيم، فيقوم هذا الشرط بعمله عبر إنشاء فهرس وحيد (unique index) علي هذا العمود، يعتبر الشرط unique أداة جيدة للمحافظة علي هذه الخاصية في قاعدة البيانات، فعلي سبيل المثل يمكن اعتبار عمود "number\_social\_security" الضمان الاجتماعي " الموجود في جدول employee لا يحتوي سوي قيمة وحيدة، مع العلم بأن العمود " emp\_id " هو المفتاح الأساسي في الجدول بمعنى أنه لابد أن يحتوي قيما فريدة ووحيدة ولايسمح بالتكرار كما هو موضح كالتالي:

```
SQL> CREATE TABLE employee
2 (emp_id NUMBER (2) not null
3 constraint unique_emp_id unique,
4 name VARCHAR2 (15),
5 gender char (1)
Constraint gender_validation check (gender
```

**5-1-2-5 primary key الشرط (المفتاح الأساسي)**

يستخدم الشرط primary key للمحافظة علي تكامل عمود أو أعمدة المفتاح الأولي، حيث يجعل هذا الشرط القيم الموجودة في المفتاح في عمود مشروط به تتمتع بالشرطين Not null, unique معاً، فعند تعريف هذا الشرط يتم إنشاء فهرس وحيد unique index ضمنى وإنشاء شرط not null ضمنى أيضاً علي العمود أو الأعمدة المشروطة بشرط المفتاح الأساسي، ويمكن اعتبار تعريف هذا الشرط كجزء من تعريف العمود أو جزء من



تعريف الجدول ، حيث إذا تم تعريف هذا الشرط علي عدة أعمدة (حالة مفتاح أساسي مركب) فإن تعريف هذا الشرط سيكون جزءاً من تعريف الجدول.

مثال

لكي تعرف الشرط primary key كجزء من تعريف عمود ما، قم بكتابة العبارة

primary key بعد نوع بيانات الجدول كما بالكود التالي:

```
SQL> CREATE TABLE employee
 (emp_id NUMBER primary key,
 name VARCHAR2 (15),
 Gender char (1)
 Constraint gender_validation check (gender
in ('M','F')));
```

مثال

لكي تعرف الشرط primary key كجزء من تعريف الجدول، قم بكتابة العبارة

primary key في أسفل تعريف الجدول، ويتم تعريفه بعد تعريف آخر عمود، ويجب وضع

فاصلة بعد تعريف آخر عمود وقبل تعريف شرط المفتاح الأساسي كما بالكود التالي:

```
SQL> CREATE TABLE employee
 (emp_id NUMBER not null,
 F_name VARCHAR2 (25),
 L_name VARCHAR2 (15),
 Primary key(emp_id));
```

**6-1-2-5 foreign key (المفتاح الخارجي)**

يستخدم الشرط foreign key كأداة للتأكيد من الجداول المرتبطة لا تحتوي علي سجلات غير منتمية إلي سجل أب، ويحتوي الشرط foreign key علي خيار يسمح بحذف كل السجلات الأبناء المرتبطة مع سجل الأب عند حذف سجل الأب وهذا الخيار هو ON DELETE CASCADE، يمكننا إنشاء مفتاح الربط الخارجي كجزء من تعريف العمود عند إنشاء الجدول ، حيث يتم استخدام كلمة REFERENCES ويليه اسم الجدول الذي يحتوي السجل الأب وذلك يكون بعد تعريف العمود ولسنا بحاجة لذكر اسم العمود من الجدول الأب ، حيث إذا لم نقوم بكتابة اسم العمود الموافق من الجدول الأب يفترض أوراكل أنه هو المفتاح الأساسي

مثال 1

في هذا المثال سنقوم بتعرف الشرط Foreign key كجزء من تعريف عمود ما كما هو موضح بالتالي:

```
SQL> CREATE TABLE consultant_projects
2 (emp_id number references consultant,
3 project_name varchar (25),
4 complete_date date);
Table created
```

مثال 2

لكي تعرف الشرط foreign key كجزء من تعريف الجدول، قم بكتابة العبارة foreign key في أسفل تعريف الجدول متبوعة باسم العمود المطبق عليه الشرط ثم كلمة reference ثم عمود المفتاح الرئيسي بالجدول الأب، ويتم تعريفه بعد تعريف آخر عمود، ويجب وضع فاصلة بعد تعريف آخر عمود وقبل تعريف شرط المفتاح الخارجي كما بالكود التالي:

```

SQL> CREATE TABLE emp (
2 empno NUMBER(4),
3 ename VARCHAR2(10) NOT NULL,
4 job VARCHAR2(9),
5 mgr NUMBER(4),
6 hiredate DATE,
7 sal NUMBER(7,2),
8 comm NUMBER(7,2),
9 deptno NUMBER(7,2) NOT NULL,
10 CONSTRAINT emp_deptno_fk FOREIGN KEY(deptno)
11 REFERENCES dept (deptno));

```

### 7-1-2-5 تعديل تعريف شرط ما (Modifying Constraints)

يمكننا إضافة شرط للجدول وذلك بعد إنشائه، كما يمكننا إزالة تلك الشروط أو إلغاء تفعيلها أو تفعيلها وذلك باستخدام `alter table` ويمتلك هذا الأمر عدة خيارات يمكن استخدامها مع الشروط وهي موضحة كالتالي:

- (1) `ADD` (إضافة): يستخدم لإضافة شرط جديد إلي الجدول.
- (2) `MODIFY` (تعديل): يستخدم لإضافة شرط إلي عمود موجود.
- (3) `DROP` (حذف) : يستخدم لحذف الشرط من الجدول.
- (4) `DISABLE` (إلغاء تفعيل): يسمح بإدخال البيانات دون النظر في فيما إذا كانت تحقق الشرط أم لا ، ولكن مع بقاء الشرط كما هو في قاموس البيانات.
- (5) `ENABLE` (تفعيل): يقوم بالتحقق من أن جميع البيانات التي يتم إدخالها والموجودة مسبقاً تحقق الشرط.

مثال 1

```

SQL> alter table consultant_projects
2 add primary key (emp_id,project_name);

```

مثال 2

```
SQL> ALTER TABLE EMP
 2 ADD CONSTRAINT emp_mgr_fk
 3 FOREIGN KEY (mgr) REFERENCES emp (empno);
Table altered.
```

مثال 3

```
SQL> ALTER TABLE emp
 2 disable primary key;
Table altered.
```

مثال 4

```
SQL> ALTER TABLE emp
 2 enable primary key;
Table altered.
```

مثال 5

```
SQL> ALTER TABLE emp
 2 drop primary key;
Table altered.
```

## الأسئلة

### المجموعة الأولى:

ضع علامة ( / ) أمام العبارة الصحيحة وعلامة ( x ) أمام العبارة الخاطئة مع تصحيح الخطأ.

1. عند استخدام Insert into لائحة القيم لابد أن تأخذ نفس ترتيب الأعمدة في

الجدول المراد إضافة السجلات إليه.

2. للتأكد من تنفيذ عبارة insert وإضافة السجل إلي جدول emp نقوم بتنفيذ العبارة

التالية:

```
SQL>Select * From emp;
```

3. عند استخدام عبارة insert يجب أن تتطابق الأعمدة والقيم المقابلة لها في نوع

البيانات data types.

4. عند استخدام عبارة insert لا يتم وضع الأحرف النصية والتاريخ بين علامتي

تنصيص مفردة ' ' .

5. لحذف كافة السجلات من جدول emp يتم كتابة العبارة التالية

```
SQL> DELETE * FROM emp;
```

6. لا يمكننا استخدام الاستعلام الفرعي في عبارة delete.

7. تحذف Delete الجدول والبيانات التي بداخله.

8. تحذف delete السجل أو السجلات بالكامل فلا حاجة للإشارة إلي أي عمود إلا في

جملة Where عند استخدامها.

9. لتحديث كافة السجلات من جدول emp وذلك بوضع المرتب 3000 لكل الموظفين

يتم كتابة العبارة التالية:

```
Sql> Update table emp set sal =3000;
```

10. نستخدم commit لحفظ وتأكيـد التغييرات، بينما rollback للترجع عن التغييرات.

### المجموعة الثانية:

#### أسئلة لتدريب الطلاب على أسئلة الاختيار المتعدد:

##### 1. عند استخدام insert.:

- (a) يجب أن تتطابق الأعمدة والقيم المقابلة لها في نوع البيانات data types.
- (b) لائحة القيم لابد أن تأخذ نفس ترتيب الأعمدة في الجدول المراد إضافة السجلات إليه.
- (c) لابد من وجود علاقة واحد إلي واحد بين الأعمدة ولائحة القيم، بمعنى أن توجد قيمة واحدة من أجل كل عمود وبـنفس الترتيب.
- (d) جميع ما سبق.

##### 2. عند استخدام insert.:

- (a) قد نجهل قيمة أحد الحقول، وذلك بكتابة كلمة NULL
- (b) قد تسبب INSERT مشكلة في التجانس والتكامل المرجعي
- (c) لابد من وضع الأحرف النصية والتاريخ بين علامتي تنصيص مفردة ' ' .
- (d) A&b&c
- (e) A&c

3. عند استخدام delete:.

- (a) يمكننا استخدام الاستعلام الفرعي في جملة Delete.
- (b) تحذف Delete جميع السجلات التي تحقق الشرط في جملة Where.
- (c) تحذف Delete الجدول
- (d) a&b

4. عند استخدام update:.

- (a) يمكننا استخدام update لتعديل القيم الموجودة في جدول ما.
- (b) إذا تم حذف عبارة Where فإن التعديل سيتم علي كل سجلات الجدول.
- (c) لا يمكننا تحديث سجلات محددة والتعديل فيها بناء علي شرط أو عدة شروط.
- (d) لا يمكننا استخدام الاستعلام الفرعي في حملة update فعلي سبيل المثال إذا أردنا.
- (e) A&b

5. عند استخدام تعليمة commit:

- a. تأكيد التغيير باستخدام commit.
- b. لا يتم تنفيذ تعليمة commit إذا قطع المطور الاتصال مع نظام أوراكل بشكل نظامي

c. إذا قطع الاتصال مع نظام أوراكل بشكل غير نظامي قبل تنفيذ تعليمة

COMMIT لن يتم حفظ التغييرات

d. يمكن التراجع عن التغييرات بشكل جزئي باستخدام SAVEPOINT حيث

تمكن المستخدم حفظ التغييرات حتى جزء

e. A&c&d

6. عند استخدام **create**:-

- (a) يمكننا عبارة **create** من إنشاء الجداول في SQL
- (b) ليس من الضروري وضع أقواس تضم أسماء الأعمدة ونوع بياناتها.
- (c) يتم وضع فاصلة ( , ) بين تعريف أي عمود وآخر.
- (d) يجب أن يكون اسم العمود فريدا داخل الجدول.
- (e) يمكن استخدام الكلمات المحجوزة كأسماء أعمدة في الجدول.
- (f)  $A \& c \& d$

7. عند استخدام **create**:-

- (a) يمكنك إنشاء جدول باستخدام الاستعلام الفرعي
- (b) وضع كلمة **as** بين اسم الجدول الجديد المراد إنشائه والاستعلام الفرعي
- (c) يجب وضع أقواس تضم أسماء الأعمدة ونوع بياناتها
- (d)  $A \& b \& c$

8. من أهم الأسباب التي تدعونا إلي إنشاء **constraints**:

- (a) التأكد من المفتاح الأساسي أو الرئيسي **unique**.
- (b) فرض التكامل المرجعي بمعنى التأكد من أن السجلات الأبناء الموجودة في جداول مرتبطة، تمتلك سجل أب.
- (c) التأكد من أن الأعمدة المرتبطة بالشروط تحتوي دائما قيمة بداخلها **NOT NULL**.
- (d) التأكد من وضع القيمة الافتراضية **DEFAULT VALUE** في عمود ما.
- (e) جميع ما سبق.



**9. عند استخدام الشرط check:-**

- (a) يستخدم هذا الشرط للتأكد من أن قمة ما لعمود ما تقع بين قيم مجموعة محددة
- (b) يقوم أوراكل بمقارنة أية قيمة يتم إدخالها مع مجموعة القيم المحددة في المجموعة
- (c)  $A \& b$
- (d) ليس شيء مما سبق.

**10. الشرط not null:**

- (a) نستخدم الشرط not null للتأكد من أن العمود دوماً سيحتوي قيمة بداخله
- (b) يتم وضع هذا الشرط علي عمود أو أعمدة المفتاح الأساسي للجدول
- (c)  $A \& b$
- (d) ليس شيء مما سبق.

**11. الشرط default:**

- (a) يستخدم هذا الخيار default لإعطاء قيمة افتراضية لعمود ما
- (b) يمنع هذا الخيار من جعل القيمة null في هذا العمود
- (c)  $A \& b$
- (d) ليس شيء مما سبق.

**12. الشرط unique:**

- (a) يستخدم الشرط unique للتأكد من أن القيمة الموجودة بالعمود المشروط هي قيمة وحيدة في جميع سجلات الجدول.
- (b) فيقوم هذا الشرط بعمله عبر إنشاء فهرس وحيد (unique index) علي العمود المشروط.
- (c)  $A \& b$
- (d) ليس شيء مما سبق.

### 13. الشرط `primary key`:

(a) يستخدم الشرط `primary key` للمحافظة علي تكامل عمود أو أعمدة المفتاح الأولي.

(b) يجعل هذا الشرط القيم الموجودة في المفتاح في عمود مشروط به تتمتع بالشرطين

(c) `Not null, unique` معاً.

(d) عند تعريف هذا الشرط يتم إنشاء فهرس وحيد.

(e) يمكن اعتبار تعريف هذا الشرط كجزء من تعريف العمود أو جزء من تعريف الجدول

(f) جميع ما سبق.

### 14. الشرط `foreign key`:-

(a) يستخدم الشرط `foreign key` كأداة للتأكيد من الجداول المرتبطة لا تحتوي علي سجلات غير منتمية إلي سجل أب.

(b) يحتوي الشرط `foreign key` علي خيار يسمح بحذف كل السجلات الأبناء

المرتبطة مع سجل الأب

(c) يمكننا إنشاء مفتاح الربط الخارجي كجزء من تعريف الجدول

(d) جميع ما سبق

### 15. باستخدام `alter table`:

(a) يمكننا إضافة شرط للجدول وذلك بعد إنشائه

(b) كما يمكننا إزالة الشروط أو إلغاء تفعيلها أو تفعيلها

(c) `A&b`

(d) ليس شيء مما سبق

# الفصل الخامس

## ورشة عمل

## الأهداف:

إعطاء الطالب ترسيخ المفاهيم الأساسية للغة SQL في ذهن الطالب من خلال قيام الطالب بإنشاء عدة جداول، وإضافة محددات من نوع Primary key, foreign key، والقيام بعمليات إضافة البيانات إلى الجداول وحذف وتعديل البيانات في قاعدة البيانات، وإجراء الاستعلامات المختلفة علي هذه الجداول

## المحتويات:

- 1) إعطاء الطالب تعريفات لعدة جداول ويطلب منه القيام بإنشائها.
- 2) إضافة سجلات جديدة للجداول.
- 3) حذف بيانات موجودة في قاعدة البيانات.
- 4) تعديل البيانات الموجودة في قاعدة البيانات.
- 5) إضافة محددات من نوع (primary key, foreign key)
- 6) إنشاء العديد من الاستعلامات

## ماذا سنتعلم في هذا الفصل:

في نهاية هذا الفصل يكون الطالب قد اكتسب المهارات والمعارف التالية:

- (1) إنشاء وتعديل وإسقاط الجداول
- (2) إضافة سجلات جديدة للجداول.
- (3) حذف بيانات موجودة في قاعدة البيانات .
- (4) كيفية تعديل البيانات الموجودة في قاعدة البيانات.
- (5) إضافة محددات من نوع (primary key, foreign
- (6) التدريب علي إنشاء الاستعلامات المختلفة.

ورشة عمل1- قم بإنشاء الجداول التالية:الجدول الأول: department

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

الجدول الثاني: employee

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

2- قم بتنفيذ التعليمات التالية:

(1) قم بإنشاء كلا الجدولين بناءً على التعريفات الموضحة سابقاً.

(2) في الجدول الأول department قم بإضافة الشرط Primary Key لحقل

deptno، وإضافة الشرط Primary Key لحقل empno في جدول

employee.

(3) إضافة الشرط Foreign key لحقل deptno في جدول employee كجزء من

تعريف الجدول.

(4) إنشاء الرابطة الذاتية في جدول employee حيث يشير عمود mgr إلى عمود المفتاح

الأساسي في الجدول وهو empno.

## قواعد البيانات (2)

(5) قم بإدخال البيانات التالية في جدول department

| Dept no | dname      | Location |
|---------|------------|----------|
| 1       | accounting | New York |
| 2       | sales      | Boston   |
| 3       | operations | Chicago  |

(6) قم بإدخال البيانات التالية في جدول employee

| empno | Ename  | Job        | mgr | hiredate | sal  | Comm | deptno |
|-------|--------|------------|-----|----------|------|------|--------|
| 1     | ayman  | accountant | 4   | 1999/8/5 | 3000 | 500  | 1      |
| 2     | Said   | Sales man  |     | 2001/9/5 | 2500 | 250  | 2      |
| 2     | Tamer  | Sales man  | 2   | 1999/5/2 | 4000 | 530  | 2      |
| 4     | ali    | accountant |     | 1999/6/4 | 5000 |      | 1      |
| 5     | ahamed | accountant | 4   | 1998/8/2 | 3500 | 120  | 1      |

(7) قم باسترجاع بيانات الموظفين ( empno, ename, sal) الذين يتقاضون مرتباً

ينحصر بين 1000، 3000.

(8) قم باسترجاع جميع بيانات الموظفين من جدول employee الذين تبدأ أسمائهم بحرف

a.

(9) قم بإنشاء إستعلام لاسترجاع بيانات الاسم والوظيفة والمرتب للموظفين الذي تم تعيينهم

خلال عام 1999 وذلك من شهر يناير إلي شهر ديسمبر.

(10) قم بإنشاء إستعلام لاستعادة بيانات جميع الموظفين مرتبة تنازلياً علي أساس تاريخ

التعيين.

(11) قم بتحديث بيانات المرتب للموظفين الذين يعملون بالإدارة رقم 2 وذلك بزيادة المرتب

بنسبة 5%.

12) قم بكتابة العبارة التي تمكننا من حذف جميع سجلات جدول emp.

13) قم بإضافة عمود address إلى جدول emp علماً بأن نوع بياناته هي  
varchar(20).

14) قم بإنشاء إستعلام يقوم بحذف سجلات الموظفين التابعين لإدارة المبيعات.

15) قم بإضافة الشرط check للعمود address علي ألا يقبل سوي سكان

new york, cairo, alex, jeddah.