قواعد البيانات باستخدام SQL

INTRODUCTION TO DATABASE USING SQL

ORACLE®

MICROSOFT SQL SERVER 2008

إعداد المهندس: عمار محمد هلال

٢٠١٨/٢٠١٧

فهرس المحتويات

مقدمة :

يجمع العاملون في المعلوماتية على اعتبار قواعد البيانات أحد اوسع علوم المعلوماتية انتشاراً وأكثرها فائدة في الحياة العملية لمهندسي المعلوماتية. ويتأكد ذلك من خلال ملاحظة النسبة المرتفعة من التطبيقات المعلوماتية، التي تؤول إلى تصميم وتطوير قاعدة معطيات، تتضمن معلومات المؤسسة التي تسعى لاستخدام الوسائل المعلوماتية في أعمالها، والبرامج التي تحقق الوظائف المتوقعة من النظام المعلوماتية.

لقد تولد لدى اوائل العاملين في التطبيقات الإدارية والمالية المعلوماتية، العدد من الأنكار التي نجلت في البداية كمنهاجيات ومبادئ عامة، تفيد في جعل العمل البرمجي يتحول من عمل يعتمد إلى حد بعيد على المهارات الفردية للمبرمجين وفهمهم للمسألة المطروحة، إلى عمل هندسي يمتلك معايير ومنهجيات تحاكي ما توفر للمهن الهندسية العريقة كالمهندس المعلوماتية او المهندس الكهربائية.

وقد كان من اهم هذه الأنكار عزل البيانات عن البرامج التي تعالجها وعن بنى ووسائط التخزين التي تسجل عليها.

ولاحظ العاملون في المعلوماتية تكراراً في الوظائف البرمجية التي يجدون انفسهم مضطرين إلى إعادة كتابتها واختبارها في كل التطبيقات التي ينفذونها، فوجدوا أنه لابد تطوير نظام برمجي عام يعني بتحقيق مجموعة من الوظائف العامة التي تفيد في تعريف بنى المعطيات وتتيح التعامل مع البيانات على مستوى عالٍ من التجرد. ومن هنا تولدت فكرة إنشاء انظمة إدارة قواعد البيانات.

لقد ساهم العاملون في قواعد البيانات في إرساء عدد من المبادئ والمفاهيم التي جرى تطويرها في مراحل لاحقة لتكون المراحل الاولى في علم هندسة البرمجيات. واستفاد العاملون في تطوير أنظمة إدارة قواعد البيانات من منهجيات وأدوات البرمجة التي أنتجها علم هندسة البرمجيات، وضمنوا جزءاً كبيراً منها في منتجاتهم البرمجية التي كرسوها لإدارة قواعد البيانات.

لقد كان عقد السبعينات حافلاً بالعديد من الافكار والنظريات التي مهدت لظهور انظمة قواعد البيانات في بداية الثمانينات. ومنذ ذلك الوقت مازال هذا النوع من الأنظمة يزداد تطويراً وتزداد بذلك أهمية ويتعمق دور كأداة لا بد من إتقان استخدامها لتطوير الأنظمة المعلوماتية.

وساهم التنافس التجاري بين الشركات التي عملت على تطوير وتسويق انظمة إدارة قواعد البيانات في إغناء هذه الانظمة. فقد سعت كل منها إلى إضافة مكونات ووظائف جديدة لنظم إدارة قواعد البيانات التي تنتجها لتجعلها متفوقة على سواها، وهذا ما جعل هذه الأنظمة تتحول تدريجياً إلى محيط تطوير متكامل يوفر كافة الوظائف التي يحتاج إليها العاملون في تطوير انظمة المعلومات خلال معظم مراحل المشروع المعلوماتي، بل أن بعض هذه الشركات ضمنت منتجاتها أدوات مساعدة في هندسة البرمجيات.

وقد كان لهذا التنافس بعض الآثار السلبية التي تجلت في عدم التوافق بين الانظمة المتعددة، وهذا ما استدعى بذل جهود كبيرة في بداية التسعينيات للاتفاق على صيغ معيارية، خاصة فيما يتعلق بلغات قواعد البيانات. كما اجتمعت هذه الشركات في نهاية التسعينات لتضع في متناول مستخدمي أنظمتها أدوات للربط بين مختلف هذه الانظمة.

يتناول هذا الكتاب مادة قواعد البيانات من جهة النظر الاكاديمي متجنباً إلى بعيد الخصوصيات التي تقدمها انظمة إدارة قواعد البيانات التجارية.

تمهيد

أهمية قواعد البيانات:

تعتبر جمع البيانات ودقتها والتعامل معها من اهم العمليات التي يُعتد عليها في معرفة معلومة معينة أو استنتاج واستنباط فكرة ما، وهي ضرورية جداً لصاحب القرار في أي مجال لاتخاذ القرار المناسب في الوقت المناسب، فمثلاً لو أن وزارة الصحة تريد معرفة عدد المرضى الموجودين في مدينة معينة وذلك لتحديد عدد الأطباء المفروض توفيرهم في كل مدينة مثلاً، فذلك يتطلب أن يكون هناك بيانات دقيقة حول المرضى والسكان في كل مدينة لتزويد الوزارة بهذه المعلومة وعلى هذا الاساس تتخذ الوزارة التدابير والقرارات المناسبة لتوفير عدد الأطباء وباقي الاحتياجات لديها معلومات حول المدن التي ينتشر فيها المرض ونسبة انتشاره وعليه يتم اتخاذ القرار المناسب لمحاربة هذا المرض قبل انتشاره. فلو فرضنا أن هذه البيانات غير متوفرة او أنها متوفرة بشكل غير منظم وغير دقيق فإن عملية اتخاذ القرار سوف تتأخر بشكل كبير ومن الممكن أن يُتخذ قرار غير مناسب مما يؤثر على عملية تسيير وتشغيل امور المواطنين وما يترتب عليه من أضرار. ومن هذا نستنتج ان عملية جمع البيانات الدقيقة والتعامل معها بشكل صحيح من أهم العمليات التي تؤثر بشكل مباشر في الحياة اليومية وبخاصة ونحن في عصر المعلومات.

وفي المواقع إن البيانات الحقيقة الدقيقة تؤدي إلى معلومات صحيحة والبيانات غير الحقيقة تؤدي إلى معلومات غير صحيحة وعليه فلا بد دراسة وتحليل البيانات واكتشاف أنظمة لتسهيل هذه المهمة وضمان سلامة البيانات وسريتها لضمان الاستفادة القصوى من المعلومات. يوضح الشكل التالي العلاقة بين البيانات والمعلومات.

يعتبر جمع البيانات ودقتها والتعامل معها من العمليات الضرورية التي يعتمد عليها في كثير مجالات الحياة، فمثلاً يوجد مكتبة كبيرة توفر بيانات الكتب والمجلدات بأنواعها وفي مجالات كثيرة يساعد في التحليل والحصول على

نتائج مفيدة مثلاً الحاجة لمجموعات جديدة من الكتب في مجالات معينة كما أن توفر بيانات عن نسب إقبال القرّاء على أنواع من الكتب في مجالات معينة تمكننا زيادة هذه المجموعات وتوسيع القاعات لأولئك القارئين، وأذكر مثالاً آخر لأهمية البيانات كمعلومات الموظفين في شركة ما وأقسامهم ورواتبهم وجداول دوامهم ونسب الصادر والوارد الاقتصادي كل ذلك يساعد في اتخاذ القرارات كعدد الموظفين وخبراتهم والشراكة مع شركات أخرى والأرباح والجدوى الاقتصادية..... .

وبالنتيجة البيانات الدقيقة تؤدي إلى قرارات دقيقة بعد تحليلها وذلك باستخدام انظمة تسهل هذه الوظيفة.

الفصل الاول

عموميات حول البيانات

➢ الأهداف :

١. الملفات Files .
٢. تخزين البيانات.
٣. طرق تخزين الملفات البيانات.
   - التخزين المباشر.
   - التخزين التسلسلي.
   - التخزين التسلسلي المباشر.
٤. قواعد البيانات التراتبية.
٥. قواعد البيانات الترابطية أو العلائقية .

عموميات حول البيانات :

الملفات Files :

الملف File هو مجموعة من البيانات التي تنتمي إلى نفس النوع، وتنقسم الملفات إلى نوعين:

- الملفات النصية Text File : ويكون محتوي الملف عبارة عن بيانات نصية.
- الملفات الثنائية Binary File : تكون على شكل بيانات ثنائية Binary Data، وهذا النوع من الملفات يستخدم غالباً من قبل لغات البرمجة.

تخزين البيانات :

يسعى الإنسان دائماً إلى تسهيل المهام عليه وتيسير كل عقبات الحياة، فلو نظرنا إلى أول إصدارات الحواسيب لوجدنا مساحات التخزين لديها صغيرة جداً، ناهي عن البطء الوصل إلى البيانات بسبب ضعف أداء الحواسيب من جهة، وبسبب رداءة نظام التشغيل من جانب آخر، ولكن الإنسان بسبب ملكته الإبداعية فإنه طور ومازال يطور اداء الحاسوب آلياً وبرمجياً، حتى حصلنا على حواسيب بكفاءات عالية وبطرق فعالة وسريعة لحفظ البيانات ولاستغلالها.

كانت بداية عهد الانسان بتخزين البيانات في سنة 1956، حينما قام باختراع القرص الصلب Hard Disk، منذ ذلك العهد والإنسان يطور وسائل تخزين البيانات إلى يومنا هذا.

بالنسبة لطرق تخزين البيانات فإن أشهرها:

التخزين المباشر :

ويكون حفظ البيانات على شكل أسطر متتالية في ملفات، ويتميز هذا النوع من التخزين ببساطة وسهولته، ولكنه يبقى ضعيفاً بسبب صعوبة استخراج البيانات منه لأنه ليست هنالك طريقة لجلب البيانات منه من خلال رتبة السكر، إضافة إلى عيب آخر وهو انه يأخذ حجماً كبيراً.

التخزين التسلسلي:

تتم عملية التخزين بشكل متسلسل، بحيث كل سطر ينتهي بفاصل (غالباً الفاصلة العادية) ثم بعد ذلك يليه السطر الثاني، هذا النوع من التخزين يتميز عن التخزين المباشر بكونه لا يأخذ حجماً كبيراً، ولكنه لا يختلف عنه في طريقة البحث عن البيانات بحيث يجب المرور على كل الأسطر من البداية إلى غاية العثور على السطر المنشود.

التخزين التسلسلي المفهرس:

نفس طريقة التخزين السابقة، ولكننا نقوم بفهرسة البيانات المخزنة في الملف، مثلاً لو عندنا ملف لحفظ بيانات العمال(نقوم بحفظ الاسم، العمر، العنوان،... إلخ)، فكل عامل يأخذ رقماً ترتيبياً، وذلك بغرض تسريع وثيرة الوصول إلى العامل المبحوث عنه، لأن البحث لا يشمل البيانات وإنما يخص فقط فهارسها Index، لكن تبقى مسالة مراجعة فهارس البيانات صعبة لأنه من الواجب تحديثها عند كل عملية إضافة أو تعديل أو حذف.

عيوب طرق التخزين السابقة :

من عيوب التخزين المباشرة والتسلسلي، أنه ليس هنالك ترابط وعلاقات بين الملفات، مثلاً لو عندنا ملف يخزن بيانات الأستاذ، وملف يخزن قائمة التخصصات فمن المستحيل التواصل بينهما لأنهما ملفان منعزلان.

ومن جهة أخرى مسالة حماية البيانات فهي غائبة، فقد بحذف تخصص معين من جدول التخصصات، ولهذا التخصص بيانات في ملف الأساتذة فتكون هنالك بعثرة وخلط البيانات، اما إذا كان الملف مشتركاً في شبكة محلية فهنالك مشكلة كبيرة وهي تحديث البيانات فقد يعملوا مجموعة من المستخدمين على نفس البيانات مما يؤدي إلى خلل في حفظها، لهذا سنجد استعمال هذا النوع من التخزين البيانات مقتصراً على التطبيقات الصغيرة.

قواعد البيانات التراتبية :

في هذا النوع من انواع التخزين البيانات، نتخلص من مشاكل الحماية وأيضاً من مشاكل الربط بين الملفات، ولكن هنالك مشاكل أخرى...

قامت كل شركة منتجة لبرنامج لإدارة قواعد البيانات بتخزين البيانات على شكل قواعد بيانات تراتبية بنمط يخصها، وبالتالي أصبح من الصعب الإحاطة بكل برامج إدارة قواعد البيانات، لأن كل برنامج له طريقته الخاصة.

للإشارة فتاريخ ظهور هذا النوع من التخزين كان سنة 1950 حسب موسوعة ويكيبيديا.

قواعد البيانات الترابطية أو العلائقية :

أتى هذا النوع من انواع تخزين البيانات لحل كل المشاكل السابقة، بحيث يتوفر على حماية عالية البيانات، بالإضافة إلى إمكانيات ربط البيانات فيما بينهما على شكل علاقات سنفصلها فيما بعد، والميزة الباهرة التي أتى بها هذا النوع من التخزين هو اعتماد كل انظمة إدارة قواعد البيانات العلائقية على لغة موحدة. وهي لغة SQL .

في قواعد البيانات العلائقية يتم تخزين البيانات في جداول ثنائية البعد(تتكون من أسطر وأعمدة).

الفصل الثاني

مفهوم قواعد البيانات ولغة الاستعلام

الأهداف:

١. فهم قواعد البيانات وتطويرها.

٢. أنواع انظمة قواعد البيانات.

٣. قواعد البيانات العلائقية.

٤. لغة التعامل مع قواعد البيانات SQL.

٥. التعرف على بيئة SQL PLUS.

➢ مفهوم قواعد البيانات:

نفرض أننا نريد جمع البيانات للطلاب في كلية ما فسيكون لكل طالب حقل رقم الطالب، وحقل اسم الطالب، وحقل كلية، وحقل القسم التابع له، وحقل العنوان ، وحقل رقم الهاتف الارضي، وحقل رقم الجوال، وفي النتيجة يكون لكل طالب سجله كما في الشكل التالي:

➢ تعريف قاعدة البيانات DATABASE :

هي مجموعة من البيانات المخزنة بشكل منظم، وهي من اهم الدعائم التي تقوم عليها المعلومات، حيث من خلال قواعد البيانات نستطيع حفظ وتعديل وحذف المعلومات بطرق سلسة، وكذلك تتيح لنا استخراج البيانات المحفوظة كما نريد.

➢ الهدف من قاعدة البيانات DATABASE:

الهدف من قواعد البيانات هو تنظيم البيانات، لتسهيل عملية استرجاعها والبحث فيها واستخلاص المعلومات منها، واختصار الوقت والجهد اثناء التعامل مع البيانات.

➢ ميزات قاعدة البيانات :

الميزات التي جعلتها من اهم لغات الوصول إلى البيانات:

- تسمية قاعدة البيانات : تخضع تسمية الأغراض في الـ SQL SERVER إلى ما يسمى بقواعد التسمية:

- التسمية يمكن أن تبدا بأحد المحارف التالية:

a, b, c, d, e, f, g, h, I, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D,) E, F,G, H, I, J, K, L, M, N, O, P, Q, or Z)، أو أن تكون البداية رقمية (0, 1, 2, 3, 4, 5, 6, 7, 8, or 9)، أو (_) وحتى المحارف غير مقروءة مثل (@ و # و % و .....إلخ)

فيمكن ان يكون اسم القاعدة مثلاً :

DB أو 2T أو %@DB أو _DB .....Etc

- الامان : نظم الامان عالي جداً.
- الحجم : يتحمل حتى 1 تيرا بايت.
- عدد المستخدمين: لا يوجد عدد محدد.
- المرونة والسهولة في التعامل.
- يدعم خاصية الوظائف المعرفة مسبقاً Stored Procedures و Triggers.

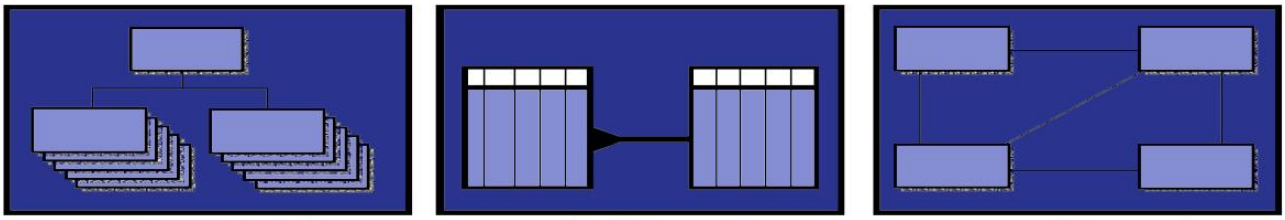
➢ مراحل تطور قواعد البيانات:

مرت عملية التعامل مع البيانات وطريقة تخزينها ومعالجتها بمراحل نذكر منها:

- حفظ البيانات في بطاقات الكترونية Electronic Sheets وتعتبر من أقدم الطرق.
- حفظ البيانات في ملفات.
- حفظ البيانات في قواعد البيانات Database وهي الطريقة المستخدمة حالياً حيث تسهل هذه

المنظومة عملية إدخال البيانات وتعديلها وحذفها ومعالجتها واسترجاعها وتحليلها وكل ذلك بسرعة جيدة وتسمى هذه الأنظمة انظمة إدارة قواعد البيانات Database Management System (DBMS) ومن هذه الأنظمة ما هو موضح في الشكل:

Type of Database Management Systems أنواع أنظمة إدارة قواعد البيانات



أنواع أنظمة قواعد البيانات:

١. نظام إدارة قواعد البيانات الهرمية Hierarchical Database Management System

استخدم هذا النظام في الماضي مع أجهزة الحواسيب الكبيرة.

٢. نظام إدارة قواعد البيانات الشبكية Network Database Management System

ظهرت بعد نظام الهرمي عند انتشار الشبكات ولكن كان يوجد صعوبة في التعامل مع البيانات.

٣. نظام إدارة قواعد البيانات العلائقية Relational Database Management System

وهو نظام الذي تعتمد عليه معظم قواعد البيانات مثل الأوراكل وهو من أفضل قواعد البيانات لقدرتها على استيعاب بيانات ضخمة دون التأثير على الأداء من حيث السرعة والدقة كما يتميز نظام الأوراكل بالسرية والامان من حيث الصلاحيات للمستخدمين والنفاذية إلى قواعد البيانات كما يتميز بسهولة الاستخدام والفهم والقدرة على برمجة تطبيقاته.

EMPNO	ENAME	JOB	DEPTNO
٧٨٣٩	KING	PRESIDENT	١٠
٧٦٩٨	BLAKE	MANAGER	٣٠
٧٧٨٢	CLARK	MANAGER	١٠
٧٥٦٦	JONES	MANAGER	٢٠
٧٣٦٩	SMITH	CLARK	٢٠
٧٤٩٩	ALLEN	SALESMAN	٣٠
٧٥٢١	WARD	SALESMAN	٣٠
٧٦٩٨	MARTN	SALESMAN	٣٠
٧٦٥٤	SCOTT	ANALYST	٢٠

(جدول الموظفين)

علاقة
RELATION

DEPTNO	DNAME	LOC
١٠	ACCOUNT	NEW YORK
٢٠	RESEARCH	DALLAS
٣٠	SALES	CHICAGO
٤٠	OPRATIONS	BOSTON

(جدول الأقسام)

➢ انواع قواعد البيانات DATABASE :

١. قواعد معطيات تحفظ في مخدم قواعد البيانات : لا يمكن الدخول إلى قاعدة البيانات إلا عن طريق الاتصال مع المخدم باستخدام اسم المستخدم وكلمة مرور وكمثال على قواعد البيانات هذه :

- **SQL SERVER:** يكون امتداد قاعدة البيانات عند الحفظ mdf .
- **Oracle :** يكون امتداد قاعدة البيانات عند الحفظ dmp .

٢. قواعد معطيات تحفظ كملف على قرص الذاكرة : لا تحفظ قاعدة البيانات على المخدم، وإنما تحفظ كملف يتم فتحه مباشرة واستخلاص البيانات من قاعدة البيانات(إذا كان الملف محمياً لا يمكن الدخول إليه بإدخال كلمة المرور) ، وكمثال على قواعد البيانات هذه Access حيث يكون امتداد قاعدة البيانات عند الحفظ mdf .

➢ قواعد البيانات العلائقية RELATIONAL DATABASE :

تعتمد قواعد البيانات العلائقية على البيانات في جداول بسيطة ثنائية الأبعاد يسهل فهمها تتكون من صفوف وأعمدة وكل عمود (Column) يسمى حقل (Field) وكل صف (Row) يسمى السجل (Record) وتم ربط هذه الجداول ببعضها بروابط تسمى (Relations) ومن هنا جاءت تسمية قواعد البيانات العلائقية.

والشكل السابق يبين العلاقة بين جدول الموظفين وجدول الاقسام من خلال عمود (DEPTNO) المشترك بين الجدولين.

➢ التعامل مع قواعد البيانات العلائقية Msnipulate With Relational Database :

قامت شركة أوراكل باعتماد لغة SQL (Structured Language Query) للتعامل مع قواعد البيانات العلائقية وهي لغة تقوم بإنشاء العناصر (Objects) الخاصة بقواعد البيانات مثل الجداول وتتعامل هذه اللغة مع الجداول لتحقيق استخلاص البيانات وتعديلها وحذفها ..... وتسمى SQL*PLUS كما يوجد برامج مختلفة يمكن ربطها بقواعد البيانات تكون كبيئة لكتابة أوامر SQL و PL/SQL وكتابة البرمجيات المختلفة.

الفرق بين قواعد البيانات العلائقية و انظمة إدارة قواعد البيانات العلائقية:

من الاخطاء الشائعة، إطلاق اسم قواعد البيانات على انظمة إدارة قواعد البيانات العلائقية. فأنظمة SQL SERVER ، Oracle، MS Access هي انظمة إدارة قواعد البيانات وليست قواعد بيانات.

تتكون هذه الأنظمة مما يلي :

- محرك قاعدة البيانات ويعتبر العنصر الأهم المسؤول عن تخزين البيانات ومعالجتها.
- دليل قاعدة البيانات الذي يحتوي كافة المعلومات التي تخص قاعدة البيانات والجداول والحقول وانواعها.
- واجهات مرئية لإدارة البيانات وتقديم نماذج وتقارير واستعلامات.
- أدوات خاصة بقواعد البيانات تشمل التوليد التلقائي لمخططات قواعد البيانات.
- أدوات تطوير التطبيقات.

➢ نظام إدارة قواعد البيانات Data Management System(DBMS) :

يقوم نظام إدارة قواعد البيانات بعملية إدارة قواعد البيانات وضمان الامن والسلامة لها، حيث يتم من خلاله إنشاء عدد من المستخدمين، وإعطاء كل منهم صلاحيات محددة، للسماح بالوصول إلى البيانات والاستفادة منها، أو إجراء العمليات عليها(كإضافة أو حذف أو التعديل) ، وذلك بإنشاء ملفات فيزيائية (في الذاكرة RAM او في الذاكرة الظاهرة على القرص الصلب) للتعامل مع البيانات المحفوظة في قاعدة البيانات.

يعد نظام غدارة قواعد البيانات صلة وصل وواجهة تخاطب بين المستخدم وقاعدة البيانات.

➢ لغة الاستفسارات Structured Language Query :

تستخدم هذه اللغة لإصدار الاوامر التي تتعلق بقواعد البيانات وتنقسم إلى خمسة أقسام رئيسة والجدول التالي يوضح هذه الأوامر وتوصيفها:

القسم	الأمر	وصف الاوامر
Data Retrieveal	SELECT	أمر استرجاع البيانات من جدول او كائن
(DML) Data Manipulation Language	INSERT	أمر إضافة بيانات إلى جدول أو كائن
	UPDATE	أمر التعديل في بيانات جدول أو كائن
	DELETE	أمر الحذف بيانات جدول أو كائن
(DDL) Data Definition Language	CREATE	أمر إنشاء جدول أو كائن
	ALTER	أمر التعديل في جدول أو كائن
	DROP	أمر إلغاء جدول او كائن
	RENAME	امر تغيير الاسم جدول او كائن
	TRUNCATE	إلغاء جزء أو بتر جزء من جدول أو كائن
Transaction Control	COMMIT	تثبيت البيانات في الجدول
	ROLLBACK	الرجوع عن تثبيت البيانات

➢ الفرق بين قواعد البيانات MYSQL و SQL SERVER :

- **MYSQL :** هي قواعد بيانات، تشبه MS-Access لكن بدون واجهة استخدام، كما أنها عادةً ما تذكر لغة PHP مع قواعد البيانات MS-Access على الرغم من ان لها استخدامات اخرى، وكما انها مجانية في الانتاج.
- **SQL Server :** هو برنامج لعمل قواعد بيانات من شركة مايكروسوفت وشديد التفاعل مع لغات البرمجة المصممة من قبل هذه الشكرة اكثر من غيرها من البرامج الأخرى.

الفصل الثالث

لغة الاستعلامات البنيوية SQL

الأهداف:

١. مفهوم SQL .

٢. تطبيق برنامج SQL.

٣. الاستفادة من تعليم SQL .

٤. أقسام قواعد البيانات في SQL .

٥. أقسام SQL.

٦. محرر بيئة SQL*PLUS.

➢ مفهوم الـ SQL :

- **أولا :** هي اختصار لكلمة Structured Query Language
- **ثانيا :** هي لغة غير إجرائية أي لا يوجد بها If , Select case , Loop , for Next
- **ثالثا :** SQL لغة قياسية ANSI

➢ ماذا يعني أن لغة SQL هي لغة قياسية ANSI:

ANSI هي اختصار لـ (American Natioml Standars Institute)، اعتمد هذا المعهد لغى الـ SQL لجعلها قياسية في التعامل مع جميع قواعد البيانات.

- **رابعا :** نقوم عن طريق هذه اللغة بتحديد العمليات التي نريد أن ننفذها على قواعد البيانات و تتولى DBMS تنفيذ هذه العمليات.
- **DBMS :** هي اختصار لـ (**D**ata**b**ase **M**anagement **S**ystem)، نظم إدارة قواعد البيانات ويقصد بها البرامج التي تتعامل مع قواعد البيانات مثل **MS Access**.

➢ أين تطبق SQL :

تعمل مع جميع برامج قواعد البيانات مثل :

MS Access, MS SQL Server، DB2, Informix, Oracle, Sybase, MySQL, PostgerSQL

➢ ما هي الاستفادة من تعلم SQL :

إدارة قواعد بياناتك بصورة أفضل ، أقوى .. و بشكل احترافي

➢ **تنقسم قواعد البيانات في SQL إلى نوعين وهما :**

١. قواعد البيانات النظام.

٢. قواعد البيانات المستخدمين.

أ. قواعد البيانات النظام : وهي أربعة :

- **Master :**

تعمل هذ القاعدة على تخزين كافة معلومات نظام المخدم كما تقوم بتخزين كافة حسابات المستخدمين وجميع الإعدادات.. كما انها الدليل الحقيقي على وجود بقية قواعد البيانات وأماكن تواجد ملفات كل منها.. وبالتالي فإن من أولويات إدارة المخدم هي حفظ نسخة احتياطية عن هذه القاعدة.

- **Msdb :**

هذه القاعدة لها مهمة رئيسية وهي جدولة المهام وتخزين operators أو ما يسمى بالمشغلات.

- **Model :**

يعتمد مخدم الـ SQL على قاعة البيانات هذه في إنشاء قاعدة معطيات جديدة في كل مرة، حيث تعد قالب لكل قواعد البيانات في النظام.

- **Tempdb :**

هذه القاعدة اسمها يدل عليها قاعدة البيانات تخزين اغراض قواعد البيانات مؤقتة كالجداول المؤقتة والإجراءات المخزنة كما ويستخدمها الـ SQL Server وراء الكواليس في التخزين المؤقت عند الحاجة، أي يمكننا القول أنها مصدر عام للمعطيات.

ولهذه القاعدة خصوصية رائعة وهي انها تعيد إنشاء نفسها عند كل مرة يتم فيها تشغيل المخدم من جديد..

حيث أن جميع الجداول المؤقتة والإجراءات المؤقتة تحذف عند :

- إغلاق الاتصال بالـ Server .
- إغلاق جميع الجلسات (Sessions) مع الـ Server التابعة لعدة مستخدمين.

➢ اقسام لغة SQL :

١-لغة التعريف البياناتDDL

"DATA AEFINITION LANGUAGE"

٢-لغة التعامل مع البياناتDML :

"DATA MANIPULATION LANGUAGE"

وتستخدم بغرض الاستعلام وتحديث البيانات

٣-لغة التحكم في البياناتDCL :

"DATA CONTROL LANGUAGE"

وتستخدم بغرض التحكم في البيانات TRANSACTION وحقوق المستخدمين.

محرر بيئة SQL*PLUS :

وهي بيئة مناسبة لتنفيذ أوامر SQL من خلالها واستخلاص البيانات وتعديلها..... وعند تشغيل التطبيق الخاص بها يطلب منك ادخال المستخدم وكلمة المرور لتستطيع الدخول لقاعدة البيانات وحسب صلاحيات المستخدم ينفذ الاوامر فيمكن للمستخدم أن يكون مستخدماً عادياً "NORMAL" أو مستخدم "مدير قاعدة بيانات /DBA/"، ولنأخذ بعض الأوامر الممكن تطبيقها في هذه البيئة :

- **SQL > EDIT**

ويستخدم لتحرير آخر امر تم تنفيذه في محرر SQL*PLUS وبالتالي يمكن تعديل هذا الامر وحفظه من جديد ويمكن اختصار الأمر كالتالي: SQL > ED.

- **SQL > RUN**

يستخدم هذا الأمر لإعادة تنفيذ آخر أمر تم تطبيقه في محرر SQL*PLUS ويمكن اختصاره كالتالي

. SQL> R

- **SQL > SPOOL Filename**

يستخدم لحفظ كل ما تم عمله داخل محرر SQL *PLUS في ملف نصي بامتداد (LST) وذلك بغرض المراجعة ومن الممكن الحصول على نسخة مطبوعة بواسطة الأمر التالي SQL > SPOOL OUT .

- **SQL > SAVE Filename**

يستخدم هذا الأمر لحفظ الأوامر في ملف وذلك لاسترجاعها مرة أخرى وتنفيذها وهنا لا بد من حفظ الملف بامتداد (SQL) وذلك لنتمكن من تشغيله مرة أخرى فإذا أردنا حفظ امر في ملف اسمه TEST.SQL نكتب الأمر SQL>SAVE TEST.SQL .

- **SQL > GET Filename**

يستخدم لاسترجاع الأوامر التي تم حفظها من خلال الأمر السابق وذلك لتنفيذها مرة أخرى فإذا أردنا استرجاع الاوامر من ملف TEST.SQL نكتب الامر SQL > GET TEST.SQL.

- **SQL > START Filename**

يستخدم هذا الأمر لتنفيذ التعليمات الموجودة في ملف تم حفظه امتداده SQL فإذا أردنا تنفيذ التعليمات الموجودة في الملف TEST.SQL نقوم بكتابة الأمر التالي : SQL > START TEST.SQL .

- **SQL > @ Filename**

هذه التعليمة تكافئ التعليمة SQL > START Filename .

- **SQL > LIST**

تستخدم لاستعراض سطور آخر أمر تمت كتابته، ويمكن استعراض سطور معينة فمثلاً لو أردت استعراض السطور من ١إلى ٣ تنفذ التعليمة SQL > L 1 3 .

وسيتم التعرف على خيارات هذه التعليمات وتعليمات أخرى في الفصول القادمة.

الفصل الرابع

تعليمة SELECT الأساسية

➢ الأهداف :

١. فهم الصيغة العامة ومتطلبات كتابتها.
٢. استرجاع البيانات من الجدول بواسطة SELECT .
٣. استرجاع الحقول بأسماء مستعارة Aliases .
٤. استخدام أداة الربط || Concatenation .
٥. استخدام عبارة DISTINCT لمنع تكرار السجلات.
٦. توصيف الجدول باستخدام التعليمة Desc .
٧. التعامل مع القيمة NULL .

➢ الصيغة العامة للتعليمة SELECT :

- **SELECT** * or Cokumns [alias]
- **From** Table
- **WHERE** Condition or conditions
- **ORDER BY** Column or Alias [ASC or DESC] ;

➢ ارشادات لكتابة عبارة SQL :

١) يمكن كتابة جملة SQL بالمحارف الكبيرة والصغيرة فذلك لا يؤثر على سلامة الجملة.

٢) يفصل بين أسماء الحقول الفاصلة (,).

٣) يمكن كتابة جملة SQL على عدة سطور.

٤) لا يمكن فصل الكلمات المحجوزة للغة أو اختصارها وتسمى الكلمات المحجوزة Keywords مثل (ORDER BY, FROM , WHERE , SELECT) .

➢ لتنفيذ جملة SQL :

١) نضع فاصلة منقوطة في نهاية الجملة.

٢) أو اشارة (/) من بداية السطر عند مؤشر SQL > .

٣) نكتب الامر RUN عند مؤشر SQL > .

➢ **لنعرض بعض الأمثلة :**

المثال الأول : عرض جميع الحقول من جدول DEPT :

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

الناتج

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

وهنا استعرضنا جميع الحقول باستخدام (*) ولاحظ أن أسماء الحقول تظهر بالمحارف الكبيرة.

المثال الثاني : عرض حقول معينة من جدول DEPT :

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

الناتج

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

المثال يستعرض حقلي ارقام الأقسام وأسمائها لجميع الأقسام ونلاحظ وجود الفاصلة بين أسماء الحقلين.

➢ **استرجاع الحقول بأسماء مستعارة Aliases :**

ونستخدم ذلك عندما نريد أن نظهر أسماء الحقول بأسماء ليست كما هي موجودة في الجدول وذلك بغاية التوضيح

وهناك ثلاث طرق لإظهار أسماء الحقول المستعارة :

١) استخدام كلمة (AS) بين اسم الحقل والاسم المستعار.

٢) ترك مسافة بين اسم الحقل والاسم المستعار.

٣) استخدام علامة التنصيص المزدوجة (" ") وذلك عندما يكون الاسم المستعار أكثر من كلمة.

المثال الثالث : عرض أسماء مستعارة لحقول من جدول الموظفين:

```
SQL > SELECT ename AS name , salary , job "employee job"
2 FROM  emp ;
```

NAME	SALARY	Employee job
SMITH	800	CLARK
ALLEN	1600	SALESMAN
WARD	1250	SALESMAN
JONES	2975	MANAGER
MARTIN	1250	SALESMAN
BLAKE	2850	MANAGER
CLARK	2450	MANAGER
SCOTT	3000	ANALYST
KING	5000	PRESIDENT
TURNER	1500	SALESMAN
ADAMS	1100	CLARK

نلاحظ كيف ظهر ENAME باسم (NAME) كمحارف كبيرة وكذلك SAL ظهر (SALARY) إلا أن job ظهرت job (employee) كمحارف صغيرة.

استخدام العمليات الحسابية في جملة SQL :

من الممكن استخدام العمليات الحسابية على الحقول الرقمية وذلك للحصول على معلومات معينة مثلاً لمعرفة راتب موظف خلال عام نضرب الراتب بـ 12 (SAL * 12) أو مثلاً إظهار إجمالي راتب الموظف بعد إضافة 500 ليرة سورية (SAL + 500) وهذه العمليات لا تؤثر على القيم في الجدول.

المعاملات المستخدمة في العمليات الحسابية:

١. الجمع (+).
٢. الطرح (-).
٣. الضرب (*).
٤. القسمة (/).

المثال الرابع : عرض رواتب الموظفين السنوية من جدول الموظفين:

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

الناتج

ENAME	SAL	Annual salary
SMITH	800	9600
ALLEN	1600	19200
WARD	1250	15000
JONES	2975	35700

المثال الخامس : عرض رواتب الموظفين السنوية بعد إضافة مئة ليرة على إجمالي الراتب من جدول الموظفين:

```
SQL > SELECT ename , sal , 12*sal +100
2       FROM  emp ;
```

الناتج

ENAME	SAL	12*sal+100
SMITH	800	9700
ALLEN	1600	19300

المثال السادس : عرض رواتب الموظفين السنوية بعد إضافة مئة ليرة على الراتب من جدول الموظفين:

```
SQL > SELECT ename , sal , 12 * (sal+100)
2       FROM  emp ;
```

الناتج

ENAME	SAL	12*(sal+100)
SMITH	800	10800
ALLEN	1600	20400

➢ استخدام أداة الربط | | Concatenation:

لعمل سلسلة من الحقول يمكن ربط حقلين أو أكثر باستخدام الأداة || ويكون ناتج الربط حقل وحيد ومن الممكن ربط هذه الحقول مع نص معين نضعه بين علامتي تنصيص فردية ' ' والمثالين التاليين يوضحان ذلك :

```
SQL > SELECT ename , job , ename  | | job as "employees"
  2       FROM emp;
```

الناتج		
ENAME	**JOB**	**employees**
SMITH	CLERK	SMITHCLERK
ALLEN	SALESMAN	ALLENSALESMAN
WARD	SALESMAN	WARDSALESMAN
JONES	MANAGER	JONESMANAGER
MARTIN	SALESMAN	MARTINSALESMAN

```
SQL > SELECT ename , job , ename  |' is a' | job as "employees"
```

الناتج		
ENAME	**JOB**	**employees**
SMITH	CLERK	SMITH is a CLERK
ALLEN	SALESMAN	ALLEN is a SALESMAN
WARD	SALESMAN	WARD is a SALESMAN
JONES	MANAGER	JONES is a MANAGER
MARTIN	SALESMAN	MARTIN is a SALESMAN

➢ استخدام عبارة DISTINCT لمنع تكرار السجلات :

عند إظهار محتويات حقل ما وقيمته مكررة او إظهار محتويات حقول مكررة بدون فائدة يمكن إظهار قيم الحقول بدون تكرار باستخدام عبارة DISTINCT .

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

الناتج
DEPTNO
20
30
30
20
30
30

لاحظ التكرار في أرقام الإدارات

```
SQL > SELECT DISTINCT deptno
  2       FROM emp;
```

الناتج
DEPTNO
10
20
30

➢ إظهار البنية الداخلية للجداول باستخدام تعليمة DESC أو DESCRIBR :

```
SQL > DESC emp ;
```

الناتج

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)

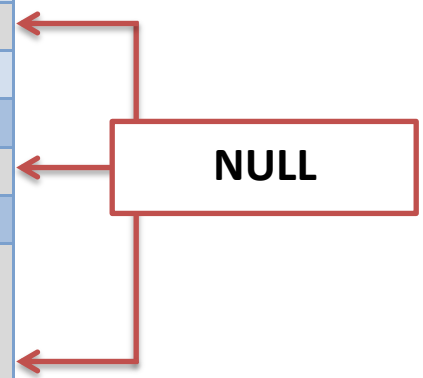
➢ التعامل مع NULL :

وهي لا تساوي الصفر ولا رقم ولا محرف فهي قيمة فارغة فنلاحظ مثلا في جدول EMP في حقل COMM أن:

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

الناتج

ENAME	JOB	SAL	COMM
----------	----------	800	----------
SMITH	CLERK	1600	
ALLEN	SALESMAN	1250	300
WARD	SALESMAN	2975	500
JONES	MANAGER	1250	
MARTDN	SALESMAN	2850	1400
BLAKE	MANAGER	2450	
CLARK	MANAGER	3000	
SCOTT	ANALYST	5000	
KING	PRESIDENT	1500	
TURNER	SALESMAN	1500	0



بعض الموظفين لا يكلفون بعمل إضافي وبالتالي قيمة الحقل NULL... .

وعندما يدخل حقل قيمته NULL في أي عملية حسابية يكون الناتج NULL كما في المثال التالي :

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

الناتج			
ENAME	JOB	SAL	12*SAL+COMM
SMITH	CLERK	800	
ALLEN	SALESMAN	1600	19500

NULL

ويمكن تجاوز هذه المشكلة باستخدام التابع NVL سنتحدث عنه لاحقاً.

الفصل الخامس

استرجاع البيانات بشروط وترتيبها

➢ الأهداف :

١) فهم الجلمة WHERE .

٢) معرفة واستخدام معاملات المقارنة Comparison Operators .

٣) معرفة واستخدام معاملات المقارنة الخاصة (IN , Between , Like , is , NULL).

٤) معرفة واستخدام المعاملات المنطقية (AND , OR , NOT).

٥) ترتيب البيانات التصاعدي والتنازلي وذلك حسب حقل أو أكثر.

في الفصل السابق تعرفنا على جملة SELECT بشكل عام ومفصل وعند استخدمها كنا نسترجع كل البيانات في الجدول فلو اردنا استرجاع بيانات فئة محددة من الموظفين وذلك باستخدام WHERE وأردنا ترتيب رواتبهم بشكل تنازلي أو تصاعدي باستخدام Order By .

➢ جملة WHERE :

وتكتب كلمة WHERE بعد اسم الجدول وتستخدم لاسترجاع بيانات محددة وذلك بالاستعانة بمعاملات المقارنة المختلفة حسب المطلوب فإذا الشرط تحقق تسترجع بيانات وإذا لم يتحقق لا تسترجع البيانات.

➢ مكونات الجملة WHERE :

أسماء الحقول ، معاملات المقارنة ، عمليات حسابية ، قيم ثابتة نصية أو رقمية.

متطلبات كتابة الجملة WHERE: يجب مراعاة ذلك عند الجملة الشرطية :

- عند استخدام قيم نصية أو قيم تعبر عن تاريخ لا بد من وضعها داخل علامتي تنصيص (' ').
- في حال استخدام قيم ثابتة نصية لا بد من مراعاة المحارف الصغيرة والكبيرة.
- في حال استخدام قيم تعبر عن تاريخ لا بد مراعاة التنسيق علماً أن لغة التنسيق الأساسية في لغة SQL هي (DD-MON-YY).

➢ جملة Order By :

تستخدم لترتيب البيانات تصاعدياً أو تنازلياً وتكتب في نهاية جملة SELECT دائماً.

متطلبات كتابة جملة Order By : يجب مراعاة:

- يجب أن تكتب في آخر جملة SELECT .
- تحتوي على أسماء الحقول الأساسية او أسماء المستعارة او رقم الحقل.
- للترتيب التصاعدي نكتب ASC من (ASCENDING)، وللترتيب التنازلي نكتب DESC من (DESCENDIG).

مثال (١) : عرض أسماء و وظائف وأرقام الموظفين الذين يعملون بوظيفة CLERK مع ترتيب الناتج تصاعدياً حسب رقم القسم :

```
SQL > SELECT ename , job , deptno
  2       FROM emp ;
  3       WHERE   job = 'CLERK'
  4       ORDER BY deptno
```

الناتج

ENAME	JOB	DEPTNO
MILLER	CLERK	10
SMITH	CLERK	20
ADAMS	CLERK	20
JAMES	CLERK	30

معاملات المقارنة المستخدمة في جملة WHERE :

#	المعامل	المعنى
١	=	يساوي
٢	>	اكبر من
٣	>=	أكبر من أو يساوي
٤	<	أصفر من
٥	<=	أصغر من أو يساوي
٦	<> أو !=	لا يساوي

أمثلة :

- WHERE hiredate = '01–JAN– 95'
- WHERE SAL > = 1500
- WHERE ENAME = 'SMITH'

مثال (2) : عرض أسماء و وظائف ورواتب الموظفين الذين رواتبهم أكبر أو يساوي (3000) :

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

الناتج

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

مثال (3) : عرض أسماء و رواتب وعمولة الموظفين الذين رواتبهم أقل أو تساوي العمولة الخاصة :

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

الناتج		
ENAME	SAL	COMM
MARTIN	1250	1400

معاملات مقارنة أخرى :

#	المعامل	المعنى
١	BETWEEN قيمة ، AND قيمة	حصر البيانات بين رقمين
٢	IN (مجموعة من قيم)	حصر البيانات ضمن مجموعة من القيم
٣	LIKE { % , _ }	حصر البيانات حسب مطابقة النص أو الحروف
٤	IS NULL	حصر البيانات الخالية Null

مثال (4) : عرض اسماء و رواتب الموظفين الذين تنحصر رواتبهم بين 1500 و 2500 :

```
SQL > SELECT ename , sal
  2       FROM   emp ;
  3       WHERE SAL  BETWEEN 1500   AND   2500 ;
```

الناتج	
ENAME	SAL
ALLEN	1600
CLARK	2450
TURNER	1500

مثال (5) : عرض رقم واسم وراتب ورقم المدير للموظفين الذين لديهم مدراء بأرقام محددة :

```
SQL > SELECT empno , ename , sal , mgr
  2       FROM  emp ;
  3       WHERE  mgr  IN (7902, 7566 ,7788 , 7839);
```

الناتج			
EMPNO	ENAME	SAL	MGR
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788
7902	FORD	3000	7566

المعامل { _ , % } LIKE :

يستخدم للبحث عن نص معين داخل ثابت أو حقل نصي حيث يتم مطابقة النص المطلوب ضمن جملة الشرط :

- الرمز (%)، تعني أي حرف أو احرف مثلاً التعبير (' A% ')، يعني مطابقة النصوص التي تبدأ بحرف (A) مهما كانت باقي الحروف.

  والتعبير (' %A ')، يعني مطابقة النصوص التي تنتهي بمحرف (A) مهما كانت الحروف التي تسبقه

  اما التعبير ('%A%')،يعني البحث عن النصوص التي تحتوي الحرف A.
- الرمز (_)، يعني مطابقة حرق واحد فمثلاً التعبير (' A% _')، يعني البحث عن النص الذي يكون فيه المحرف الثاني A ، أما التعبير ('A% _ _') يستخدم للبحث عن النص الذي فيه المحرف الثالث A.

ملاحظة: يجب مراعاة المحارف الصغيرة أما الكبيرة عند استخدام المعامل LIKE.

مثال (6) : عرض أسماء الموظفين التي تبدأ أسماءهم بحرف S :

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

الناتج
ENAME
SMITH
SCOTT

مثال (7) : عرض اسم وتاريخ تعيين الموظفين الذين تم تعينهم في عام 1981 :

```
SQL > SELECT ename , hiredate
  2       FROM   emp;
  3       WHERE  ename LIKE '%81' ;
```

الناتج

ENAME	HIREDATE
ALLEN	20/02/81
WARD	22/02/81
JONES	02/04/81
MARTIN	28/09/81
BLARK	01/05/81
CLARK	09/06/81
KING	17/11/81
TURNER	08/09/81
JAMES	03/12/81
FORD	03/12/81

مثال (8) عرض أسماء الموظفين الذين يكون الحرف الثاني في أسمائهم A :

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

الناتج

ENAME
WARD
MARTIN
JAMES

مثال (9) عرض اسم ورقم الموظفين الذين ليس لهم مدير :

```
SQL > SELECT ename , mgr
  2       FROM   emp;
  3       WHERE mgr IS NULL ;
```

ENAME	MGR
KING	

ملاحظة : لا يمكن استخدام المعامل (=) مع قيمة NULL بل نستخدم الصيغة IS NULL :

```
SQL > SELECT ename , mgr
2       FROM   emp;
3       WHERE mgr =  NULL ;
```

المعاملات المنطقية في جملة الشرط WHERE :

#	المعامل	المعنى
١	AND	ترجع النتيجة TRUE إذا كانت جملتا الشرط TRUE
٢	OR	ترجع النتيجة TRUE إذا كانت إحدى جملتي الشرط TRUE
٣	NOT	تنفي النتيجة، أي ترجع النتيجة TRUE إذا كانت جملة الشرط FALSE

تستخدم المعاملات المنطقية لتكوين اكثر من شرط داخل عبارة **WHERE** وهي معاملات تربط بين جملتين شرطيتين أو أكثر وتكون النتيجة **TRUE** أي تحقق الشرط، **FALSE** أي لم يتحقق الشرط.

- المعامل **AND** : يربط بين جملتين شرطيتين ويكون الناتج **TRUE** في حال كلتا الجملتين **TRUE** والجدول التالي يوضح نتائج المعامل **AND** مع الحالات المختلفة لجملتي الشرط .

جملة الشرط الاولى	جملة الشرط الثانية	ناتج المعامل AND
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	FALSE	FALSE
TRUE	NULL	NULL
FALSE	NULL	FALSE
NULL	NULL	NULL

بالتدقيق في الجدول السابق نستخلص :

- ناتج المعامل AND دائماً FALSE إلا في حال إذا كانت الجملتين TRUE .
- عند استخدام القيمة NULL مع المعامل AND الناتج يكون NULL إلا في حال إحدى الجملتين NULL والأخرى FALSE .

مثال (10) عرض رقم واسم ووظيفة وراتب الموظفين الذين رواتبهم أكبر من أو تساوي 1100 وبنفس الوقت CLERK :

```
SQL > SELECT empno , ename , job , sal
  2       FROM   emp ;
  3       WHERE  sal >= 1100  AND  job = 'CLERK' ;
```

النتائج

EMPNO	ENAME	JOB	SAL
7876	ADMS	CLERK	1100
7934	MILLER	CLERK	1300

مثال (11) عرض اسم وراتب وعمولة الموظفين الذين يزيد راتبهم عن 1100 وعمولتهم أقل من 500 :

```
SQL > SELECT ename , sal , comm
  2       FROM  emp ;
  3       WHERE  sal> 1100   AND  comm< 500 ;
```

النتائج

ENAME	SAL	COMM
ALLEN	1600	300
TURNER	1500	0

- **المعامل OR** : يربط بين جملتين شرطيتين ويكون الناتج TRUE إذا كانت إحدى الجملتين أو كلاهما TRUE والجدول التالي يوضح نتائج المعامل OR في الحالات المختلفة للجملتين الشرطيتين.

جملة الشرط الاولى	جملة الشرط الثانية	ناتج المعامل AND
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE
TRUE	NULL	TRUE
FALSE	NULL	NULL
NULL	NULL	NULL

نستخلص من الجدول السابق :

- ناتج المعامل **OR** دائماً **TRUE** إلا في حال الجملتين **FALSE** فقط .
- عند استخدام القيمة **NULL** مع المعامل **OR** يكون الناتج دائماً **NULL** إلا في حال كانت إحدى الجملتين **NULL** الأخرى **TRUE** يكون الناتج **TRUE** .

مثال (12) عرض رقم واسم ووظيفة وراتب الموظفين الذين رواتبهم أكبر من 2500 أو تكون وظيفتهم MANAGER .

```
SQL > SELECT empno , ename , job , sal
  2       FROM  emp
  3       WHERE  sal> 2500   OR  job = 'MANAGER' ;
```

الناتج			
EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

مثال (13) عرض اسم وراتب ورقم القسم للموظفين الذين رواتبهم أقل من 1000 أو يكون رقم ادارتهم 10 .

```
SQL > SELECT ename , sal , deptno
  2       FROM  emp
  3       WHERE  sal> 1000   OR  deptno =10;
```

الناتج		
ENAME	SAL	DEPTNO
SMITH	800	20
CLARK	2450	10
KING	5000	10
JAMES	950	30
MILLER	1300	10

- **المعامل NOT** : يقوم بعكس نتيجة الشرط أي إذا كانت نتيجة الشرط **TRUE** فإن ناتج المعامل **NOT** يكون **FALSE** والعكس صحيح والجدول التالي يبين تأثير هذا المعامل على جملة الشرط:

جملة الشرط الاولى	ناتج المعامل AND
TRUE	**FALSE**
FALSE	**TRUE**
NULL	**NULL**

كما يستخدم المعامل NOT لنفي المعاملات الموضحة في الجدول التالي :

#	المعامل	نفي المعامل	المعنى
١	**BETWEEN...AND...**	NOT BETWEEN...AND...	ليست بين رقمي .... و ....
٢	**IN (....)**	NOT IN (....)	ليست ضمن القائمة .....
٣	**LIKE { % , _}**	NOT LIKE { % , _}	ليست مطابقة
٤	**IS NULL**	NOT IS NULL	ليست قيمة خالية

مثال على استخدام المعامل NOT لعكس المعاملات السابقة الذكر :

- WHERE JOB NOT IN ('CLERK' , 'MANAGER')
- WHERE SAL NOT BETWEEN 1000 AND 1500
- WHERE ENAME NOT LIKE '%A%'
- WHERE COMM IS NOT NULL

مثال (14) عرض اسم و وظيفة الموظفين الذين ليست لهم وظائفهم من ضمن التالي (CLERK, MANAGER , ANALYST) :

```
SQL > SELECT ename , job
  2   FROM   emp ;
  3   WHERE  JOB  NOT IN ('CLERK' , 'MANAGER' , 'ANALYST');
```

الناتج

ENAME	JOB
ALLEN	SALESMAN
WARD	SALESMAN
MARTIN	SALESMAN
KING	PRESIDENT
TURNER	SALESMAN

مثال (15) عرض اسم ووظيفة وراتب الموظفين الذين لا تنحصر رواتبهم بين (1000 و3000):

```
SQL > SELECT ename , job , sal
  2     FROM   emp
  3     WHERE  sal  NOT BETWEEN  1000 AND  3000;
```

الناتج

ENAME	JOB	SAL
SMITH	CLERK	800
KING	PRESIDENT	5000
JAMES	CLERK	950

مثال (١٦) عرض اسم ووظيفة وراتب وعمولة الموظفين الذين يأخذون عمولة :

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

النتائج

ENAME	JOB	SAL	COMM
ALLEN	SALESMAN	1600	300
WARD	SALESMAN	1250	500
MARTIN	SALESMAN	1250	1400
TURNER	SALESMAN	1500	0

الفصل السادس

استخدام توابع الصف الواحد في جملة الاستعلام SELECT

➢ الأهداف :

١. معرفة التوابع المستخدمة في لغة SQL .

٢. معرفة انواع توابع الصف الاول Single Row Function .

٣. معرفة التوابع المحرفية Character Function .

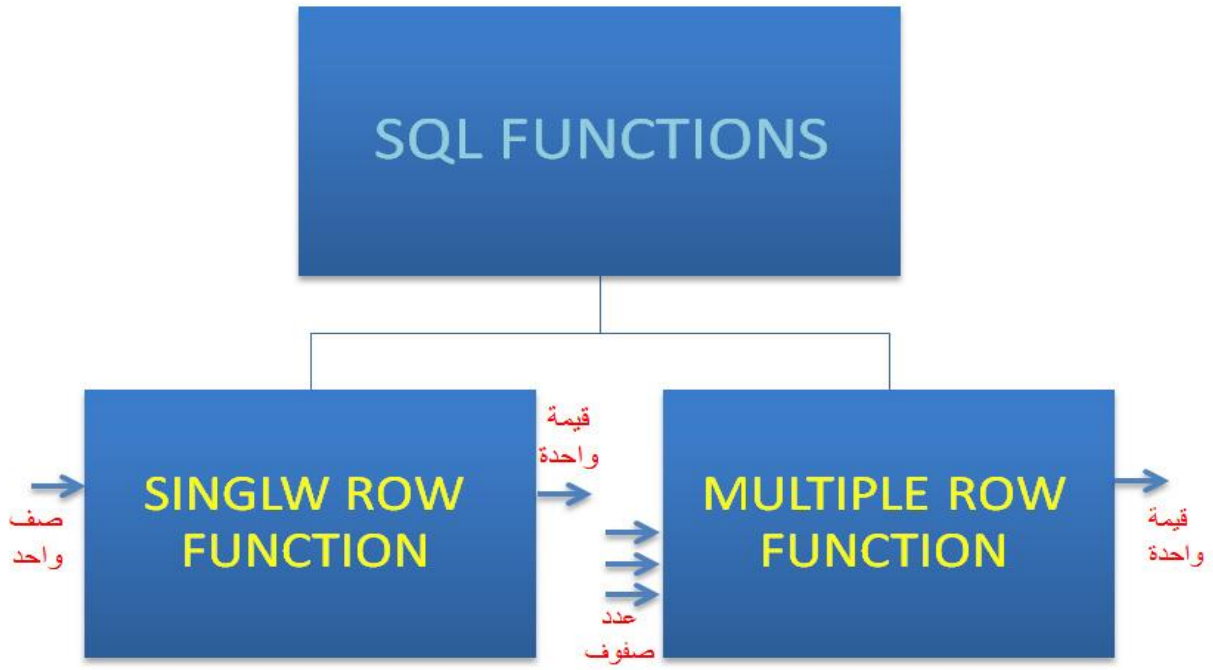
٤. معرفة التوابع الرقمية Number Function .

٥. معرفة توابع التاريخ Date Function.

٦. معرفة توابع التحويل Conversion Function .

تستخدم التوابع في لغة SQL وهي أداة قوية مفيدة عند استخدام SELECT وتنقسم هذه التوابع إلى نوعين كما في الشكل التالي :

أنواع الدوال



➢ توابع الصف الواحد Single Row Function .

حيث تتعامل مع البيانات صف وحيد وتكون قيمتها وحيدة وتنقسم لعدة أنواع:

- توابع محرفية.
- توابع رقمية.
- توابع التاريخ.
- توابع التحويل.

➢ التوابع المحرفية Character Function:

وتتعامل مع البيانات المحرفية وتكون نتيجتها حروفاً أو أرقاماً والجدول التالي يبين هذه التوابع :

	وظيفتها	الدالة FUNCTION
١	دالة تستخدم لتحويل جميع الحروف (عمود او سلسلة) إلى حروف صغيرة Small	**LOWER** (Column \ Expression)
٢	دالة تستخدم لتحويل حروف (عمود أو سلسلة) إلى حروف كبيرة Capital	**UPPER** (Column\ Expression)
٣	دالة لتحويل الحرف الاول فقط من (عمود أو سلسلة)إلى حرف كبيرCapital وباقي الحروف تحول الى حروف صغيرة	**INITCAP** (Column\ Expression)
٤	دالة ربط عمودين أو سلسلتين معاً وهي تماماً مثل أداة الربط (||)	**CONCAT**(Column1\ Expression1, Column2\ Expression2)
٥	دالة تستخدم لقطع جزء من عمود أو سلسلة بدايةً من الحرف رقك M وعدد الحروف المقطوعة هي n	**SUBSTR** (Column\ Expression,M,N)
٦	دالة تستخدم لإيجاد عدد حروف السلسلة أو العمود (الناتج عدد)	**LENGTH** (Column\ Expression)
٧	دالة تستخدم لتحديد مكان حرف معين داخل سلسلة عمود (الناتج عدد)والحرف M يعبر عن الحرف المراد تحديد مكانه	**INSTR** (Column\ Expression,M)
٨	دالة تستخدم لضبط بيانات سلسلة أو عمود ناحية اليمين وذلك بملء حرف معين من اليسار والحرف n لتحديد الطول بعد الضبط	**LPAD** (Column\ Expression,N,'string')
٩	دالة تستخدم لضبط بيانات عمود أو سلسلة ناحية اليسار وذلك بملء حرف معين من اليمين والحرف n لتحديد الطول بعد الضبط	**RPAD** (Column\ Expression,N,'string')
١٠	دالة تستخدم لقطع حرف معين من بداية او نهاية الكلمة فقط	**TRIM** ('character' FROM Column\ Expression,M)

➢ والجدول التالي يوضح بعض الأمثلة عن التوابع المحرفية:

#	النتيجة	المثال
١	good by	Select LOWER ('GOOD BY') from dual;
٢	GOOD BY	Select UPPER ('GOOD BY') from dual;
٣	Good	Select INITCAP ('GOOD) from dual;
٤	GOODBY	Select CONCAT ('GOOD' , 'BY') from dual;
٥	OOD	Select STBSTR ('GOOD BY',2,3) from dual;
٦	4	Select LENGTH ('GOOD') from dual;
٧	4	Select INSTR ('GOOD' , 'D') from dual;
٨	*****AHMED	Select LPAD ('AHMED',10,'*') from dual;
٩	AHMED*****	Select RPAD ('AHMED',10,'*') from dual;
١٠	AMI	Select TRIM ('S' FROM 'SAMI') from dual;

ملاحظة : الجدول DUAL هو جدول وهمي ضمن أوراكل يستخدم لإجراء العمليات التي لا يدخل فيها أي جدول.

➢ بعض الامثلة عن التوابع المحرفية :

المثال (1) :

```
SQL > SELECT LOWER (ename), UPPER (job), INITCAP(job),
CONCAT (ename,job)
2       FROM emp
3       WHERE SAL= 3000;
```

الناتج			
LOWER (ENAME)	UPPER (JOB)	INITCAP(JOB)	CONCAT (ENAME, JOB)
scott	ANALYST	Analyst	SCOTTANALYST
ford	ANALYST	Analyst	FORDANALYST

مثال (2) :

```
SQL > SELECT ename, SUBSTR (ename,2,3), LENGTH(ename),
INSTR (ename,'K')
  2      FROM emp
  3      WHERE LOWER(job)= 'manager';
```

النتائج

ENAME	SUBSTR (ename,2,3)	LENGTH(ename)	INSTR (ename,'K')
JONES	ONE	5	0
BLAKE	LAK	5	4
CLARK	LAR	5	5

مثال (3):

```
SQL > SELECT ename, TRIM ('S' FROM  ename),
LPAD(ename,10,'*'), RPAD (ename,10,'#')
  2      FROM  emp
  3      WHERE  SAL > 2500;
```

النتائج

ENAME	TRIM ('S' FROM ename)	LPAD(ename,10,'*')	RPAD (ename,10,'#')
JONES	JONES	*****JONES	JONES#####
BLAKE	BLAKE	***** BLAKE	BLAKE#####
SCOTT	SCOTT	***** SCOTT	SCOTT#####
KING	KING	***** KING	KING#####
FORD	FORD	***** FORD	FORD#####

➢ التوابع الرقمية NUMBER Function :

وهي توابع تتعامل مع البيانات رقمية والنتيجة أرقاماً والجدول التالي يبين هذه التوابع وعملها :

	وظيفتها الدالة	Function
١	دالة تستخدم لقص عدد معين من الجزء العشري مع تقريب الاعداد إلى أقرب عدد العشري أو إلى عدد صحيح والحرف n يبين عدد الأرقام بعد العلامة العشرية، وتوجد حالات للعرف n. • إذا كان (n=0) فإن التقريب يكون إلى أقرب عدد صحيح. • إذا كان (n>0) أي عدد موجب فإن التقريب يكون في الجزء بعد العلامة العشرية (الجزء العشري). • إذا كان (n<0) أي عدد سالب فإن التقريب يكون في الجزء قبل العلامة العشرية (الجزء الصحيح)	ROUNF (Column\ Expression,N)
٢	دالة تستخدم لقص عدد معين من الجزء العشري بدون تقريب، وأيضاً توجد حالات للحرف N • إذا كان (n=0) فإنه يتم قص الجزء العشري كله ويكون الناتج عدد صحيح . • إذا كان (n>0) أي عدد موجب فغن القص يكون في الجزء بعد العلامة العشرية (الجزء العشري). • إذا كان (n<0) أي عدد سالب فإن القص يكون في الجزء قبل العلامة العشرية (الجزء الصحيح)	TRUNC (Column\ Expression,N)
٣	دالة تستخدم لإيجاد باقي قسمة العدد M على العدد N	MOD(m,n)

مثال (4) :

```
SQL > SELECT ROUND(45.923,0), ROUND(45.923,2),
      ROUND(45.923,-1), ROUND(45.923,-2)
  2   FROM dual ;
```

الناتج			
ROUND(45.923,0)	ROUND(45.923,2)	ROUND(45.923,-1)	ROUND(45.923,-2)
46	45.92	50	0

مثال (5) :

```
SQL > SELECT TRUNC(45.923,0), TRUNC (45.923,2), TRUNC (45.923,–1),
TRUNC (45.923,–2)
2       FROM dual ;
```

الناتج			
TRUNC (45.923,0)	TRUNC (45.923,2)	TRUNC (45.923,-1)	TRUNC (45.923,-2)
45	45.92	40	0

مثال (6) :

```
SQL > SELECT ename , sal , comm , MOD(SAL,COMM)
2       FROM  emp
3       WHERE sal= 1600;
```

الناتج			
ENAME	SAL	COMM	MOD(SAL,COMM)
ALLEN	1600	300	100

➢ توابع التاريخ Date Function :

وهي توابع تتعامل مع التاريخ واهمية التاريخ في حياتنا قامت أوراكل بتوفير توابع بسيطة للتعامل مع التاريخ والجدول التالي يبين هذه التوابع وطريقة عملها:

	وظيفتها	الدالة Function
١	دالة تستخدم لعرض تاريخ النظام الموجود بجهاز الحاسب الآلي (تاريخ اليوم الحالي)	SYSDATE
٢	دالة لإيجاد عدد الأشهر بين تاريخين	MONTHS_BETWEEN(date1,date2)
٣	دالة لإضافة عدد معين من الأشهر على تاريخ معطى	ADD_MONTHS(date,n)
٤	دالة لإيجاد تاريخ يوم معين بعد تاريخ معطى	NEXT_DAY(date,'day')
٥	دالة لإيجاد آخر يوم في الشهر لتاريخ معطى	LAST_DAY(date)
٦	دالة تستخدم لتقريب التاريخ لأقرب شهر أو سنة	ROUND(date)
٧	دالة تستخدم لقص التاريخ لأقرب شهر او سنة	TRUNC(date)

- والجدول التالي يوضح امثلة عن هذه التوابع :

	المعنى	النتيجة	المثال
١	تم إيجاد الأشهر بين التاريخين المعطيين	19.6774194	MONTHS_BETWEEN('01-SEP-95', '11-JAN-94')
٢	تم إضافة ٦ أشهر على التاريخ المعطى	'11-JUL-94'	ADD_MONTHS('11-JAN-94',6)
٣	إيجاد تاريخ يوم الجمعة بعد التاريخ '01-SEP-95'	'08-SEP-95'	NEXT_DAY('01-SEP-95' , 'FRIDAY')
٤	إيجاد آخر يوم في شهر ستمبر September	'30-SEP-95'	LAST_DAY('01-SEP-95')
٥	تم تقريب التاريخ المعطى إلى أقرب شهر وبما ان اليوم خو (25) أي أكبر من (15) فتم تقريب إلى أول الشهر الذي يليه	01-AUG-95	ROUND('25-JUL-95' , 'MONTH')
٦	تم تقريب التاريخ المعطى إلى أقري سنة وبما ان شهر	'01-JUL-95'	ROUND('25-JUL-95' , YEAR)

	(JULY) هو شهر (7) فإنه تم التقريب إلى أول السنة التالية		
٧	تم قص التاريخ المعطى بدون تقريب إلى أول يوم في الشهر	'01-JUL-95'	TEUNC('25-JUL-95' ,' MONTH)
٨	تم قص التاريخ المعطى بدون تقريب إلى أول السنة الحالية	'01-JUL-95'	TEUNC('25-JUL-95' ,' YEAR')

➢ **بعض الأمثلة عن توابع التاريخ :**

مثال (7) :

```
SQL > SELECT SYSDATE FROM dual ;
```

الناتج
SYSDATE
25-01-2017

مثال (8) :

```
SQL > SELECT empno, hiredate,
MONTHS_BETWEEN(sysdate,hiredate)
2       FROM emp
3       WHERE hiredate like '%1987' ;
```

الناتج		
EMPNO	HIREDATE	MONTHS_BETWEEN(sysdate,hiredate)
7788	19-04-1987	201.200404
7876	23-05-1987	200.071371

مثال (9) :

```
SQL > SELECT empno , hiredate
,ADD_MONTHS(HIREDATE,6),LAST_DAY(HIREDATE)
2       FROM emp
3       WHERE hiredate like '%1987' ;
```

الناتج			
EMPNO	HIREDATE	ADD_MONTHS(HIREDATE,6)	LAST_DAY(HIREDATE)
7788	19-04-1987	19-10-1987	30-04-1987
7876	23-05-1987	23-11-1987	31-05-1987

مثال (10) :

```
SQL > SELECT empno,hiredate,NEXT_DAY(hiredate,'FRIDAY')
2       FROM    emp
3       WHERE hiredate like '%1987';
```

الناتج		
EMPNO	HIREDATE	NEXT_DAY(hiredate,'FRIDAY')
7788	19-04-1987	24-04-1987
7876	23-05-1987	09-05-1987

➢ توابع التحويل Conversion Functions :

يوجد أنواع كثيرة من البيانات يمكن تخزينها في الجدول ومنها الرقمية number والمحرفية character والتاريخ date وتوابع التحويل تقوم بتحويل البيانات من نوع لآخر وهي عبارة عن ثلاثة أنواع، الجدول التالي يوضحها :

	وظيفتها	الدالة Functions
١	تستخدم هذه الدالة لتحويل البيانات الرقمية او بيانات التاريخ إلى بيانات حرفية بشكل معين (FORMAT) حسب الطلب fmt .	TO_CHAR(DATE / NUMBER , 'fmt')
٢	تستخدم لتحويل البيانات الحرفية الى بيانات من نوع التاريخ بشكل معين (FORMAT) حسب الطلب fmt	TO_DATE (CHAR , 'fmt')
٣	تستخدم لتحويل البيانات الحرفية الى بيانات رقيمة بشكل معين (FORMAT) حسب الطلب fmt	TO_NUMBER (CHAR , 'fmt')

- **دالة التحويل TO_CHAR :**

أولاً – استخدامها لتحويل بيانات التاريخ إلى بيانات محرفية:

الشكل العام

TO_CHAR(DATE , 'fmt')

مثال (11) :

```
SQL > SELECT sysdate , TO_CHAR(sysdate, 'DD/MM/YYYY')
  2      FROM   dual ;
```

الناتج	
SYSDATE	TO_CHAR(SYSDATE,'DD/MM/YYYY')
26-01-2004	26/01-2004

- الجدول التالي يوضح بعض التنسيقات التي يمكن استخدامها عند تحويل التاريخ الى بيانات محرفية :

١	إظهار السنة كاملة بالأرقام (القرن + السنة) مثل 2004	YYYY
٢	إظهار رقمين فقط من السنة 04	YY
٣	إظهار السنة كاملة كتابة (TWO THOUSAND FOUR) وحالة احرف الكتابة تتوقف على حالة أحرف كلمة YEAR	YEAR
٤	إظهار الشهر في شكل رقمين 01	MM
٥	إظهار الشهر كتابة (JANURY)	MONTH
٦	إظهار الثلاث حروف الاولى من اليوم (JAN)	DY
٧	إظهار اليوم كاملاً كتابتاً (FRIDAY)	DAY
٨	إظهار الوقت بنظام 12 ساعة وهل هو صباحاً أم مساءً (04:30:50 PM)	HH12:MI:SS AM

مثال (12) :

```
SQL > SELECT empno, TO_CHAR(hiredate, 'DAY "OF" MONTH
YYYY HH12:MI:SS AM')
2       FROM   emp
3       WHERE ename=upper('king') ;
```

الناتج

EMPNO	TO_CHAR(hiredate, 'DAY "OF" MONTH YYYY HH12:MI:SS AM')
7839	TUESDAY OF NOVEMBER 1981 12:00:00 AM

ثانياً : استخدام التابع TO_CHAR في تحويل البيانات الرقمية إلى بيانات محرفية :

الشكل العام

TO_CHAR(NUMBER, 'fmt')

مثال (13) :

```
SQL > SELECT empno, TO_CHAR(sal, '$99.999') salary
  2       FROM   emp
  3       WHERE sal > 2500;
```

الناتج	
EMPNO	**SALARY**
7566	$2,975
7698	$2,850
7788	$3,000
7839	$5,000
7902	$3,000

- الجدول التالي يبين بعض الرموز التي تستخدم في تحويل البيانات الرقمية إلى محرفية لتظهر بشكل معين:

١	عدد تكرار هذا الرقم يمثل عدد الخانات التي تظهر، مثلاً عندما نكتب (99) معناها ظهور رقمين وهكذا	9
٢	يعني ظهور الرقم وقبله صفر	09
٣	يعني ظهور صفراً إذا كانت القيمة معدومة	990
٤	إظهار علامة $ قبل الرقم	$99
٥	إظهار العلامة العشرية	.
٦	إظهار فواصل بين كل ثلاثة أرقام (فاصلة الالوف)	,
٧	إظهار علامة السالب (-) يمين الرقم إذا كان سالباً	MI

- **تابع التحويل TO_DATE :**

الشكل العام

TO_DATE (CHAR, 'fmt')

تقوم بتحويل البيانات المحرفية التي تعبر عن التاريخ إلى بيانات من نوع DATE بشكل معين :

مثال (14) :

```
SQL > SELECT TO_DATE('FEBRUARY 22, 1981', 'MONTH
DD, YYYY')
2       FROM   emp
```

الناتج
TO_DATE('FEBRUARY 22, 1981', 'MONTH DD, YYYY')
22-FEB-1981

- **تابع التحويل TO_NUMBER:**

الشكل العام

TO_NUMBER(CHAR , 'fmt')

يستخدم هذا التحويل البيانات المحرفية التي تعبر عن أرقام إلى بيانات رقمية بتنسيق معين وذلك لإجراء العلميات الحسابية.

الفصل السابع

التوابع التجميعية GROUP FUNCTION

➢ الأهداف :

١) فهم ومعرفة التوابع التجميعية لأكثر من صف وانواعها.

٢) إنشاء مجموعات من البيانات باستخدام Group by .

٣) فهم ومعرفة جملة الشرط المستخدمة مع التوابع التجميعية Having .

تستخدم التوابع التجميعية للتعامل مع بيانات مجموعة من الصفوف للحصول على قيمة واحدة فمثلاً لإيجاد مجموعة رواتب الموظفين نستخدم التابع SUM كما موضع في الجدول :

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

➢ **أنواع التوابع التجميعية : الجدول التالي يبين ذلك :**

	وظيفتها	الدالة Function
١	دالة تستخدم لإيجاد المجموع لعدد من القيم	SUM
٢	دالة تستخدم لإيجاد اكبر قيمة من بين مجموعة من القيم	MAX
٣	دالة تستخدم لإيجاد أقل قيمة من بين مجموعة من القيم	MIN
٤	دالة تستخدم لإيجاد المتوسط الحسابي لمجموعة من القيم	AVG
٥	دالة تستخدم لإيجاد عدد القيم او عد الصفوف وهذه الدالة تتجاهل القيم الفارغة NULL عند عملية العد	COUNT
٦	دالة تستخدم لإيجاد الانحراف المعياري لمجموعة من القيم	STDDEV DEVIATION
٧	دالة تستخدم لإيجاد مقدار التباين (التشتت) لمجموعة من القيم	VARIANCE

➢ **وجميع هذه التوابع تتجاهل القيمة NULL داخل الأعمدة.**

مثال (1) :

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

الناتج

SUM(SAL)	MAX(SAL)	MIN(SAL)	AVG(SAL)
29025	5000	800	2073.21429

ملاحظة : التابعان (MAX , MIN) يتعاملان مع جميع البيانات فمثلاً عند التعامل مع المحارف تكون النتيجة حسب الترتيب الأبجدي لنلاحظ المثال التالي :

مثال (2) :

```
SQL > SELECT MAX(ename) , min(ename)
  2      FROM emp
```

الناتج

MAX(ENAME)	MIN(ENAME)
WARD	ADAMS

مثال (3) :

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

الناتج
AVG(NVL(comm , 0))
157.14286

في المثال السابق تم حساب المتوسط الحسابي لمكافآت الموظفين ونلاحظ استخدام التابع NVL لتجاوز مشكلة القيمة NULL إذ ان التوابع التجميعية تتجاهل القيم NULL .

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

AVG(comm)
550

التعامل مع التابع COUNT :

- يوجد حالتين :

- التابع Count (*) لعد جميع الصفوف داخل الجدول بما فيها الصفوف المكررة والصفوف التي تحوي القيم NULL وإذا كانت الجملة تحوي عبارة WHERE فغنها تقوم بعد الصفوف حسب الشرط.
- التابع Count(column) لعد بيانات عمود معين مع تجاهل القيم NULL.

والأمثلة توضح :

مثال (4) :

```
SQL > SELECT COUNT(*),COUNT(comm), COUNT(DEPTNO)
  2       FROM    emp ;
```

الناتج		
COUNT(*)	COUNT(comm)	COUNT(DEPTNO)
14	4	4

مثال (5) :

```
SQL > SELECT COUNT(comm) , COUNT(*)
  2         FORM emp
  3         WHERE deptno=30;
```

النتائج

COUNT(*)	COUNT(comm)
6	4

إنشاء مجموعات من البيانات باستخدام الجزء GROUP BY :

لنفرض أننا نريد إيجاد أكبر راتب في كل قسم لذلك نقوم بتقسم الجدول إلى أقسام ونوجد أكبر راتب في كل قسم ونحقق ذبك باستخدام **GROUP BY** والتي تقسم الجدول إلى مجموعات حسب عمود معين او أكثر:

مثال (6) :

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

النتائج

Deptno	AVG(sal)
30	1566.66667
20	2175
10	2914.66667

ملاحظات لاستخدام التوابع التجميعية :

- عند كتابة اسم عمود داخل الجملة SELECT لا بد من كتابته ضمن group by وذلك لأن التوابع التجميعية تتعامل مع عدة صفوف.
- يمكن استخدام العبارة ORDER BY مع التوابع التجميعية كما مبين في المثال السابق.
- لا يمكن استخدام التوابع التجميعية ضمن العبارة WHERE ولكن نستخدم بدلاً عنها HAVING.

مثال (7) :

```
SQL > SELECT deptno , AVG(sal)
  2         FROM    emp
  3         ORDER BY AVG(SAL) ;
ERROR at Line 1 :

ORA-00937: Not A Single-Group Group Function <_____ رسالة الخطأ
```

مثال (8) :

```
SQL > SELECT deptno , AVG(SAL)
  2         FROM    emp
  3         WHERE avg(sal)>2000
  4          GROUP BY deptno ;
ERROR at Line 3 :

ORA-00934: group function is not allowed here <_____ رسالة الخطأ
```

ظهرت رسالة الخطأ هنا لاستخدامنا عبارة الشرط where مع التابع التجميعي ونصحح ذلك باستخدامنا عبارة HAVING كما في المثال التالي

مثال (9) :

```
SQL > SELECT deptno , AVG(SAL)
  2         FROM    emp
  3         WHERE  AVG (sal)>2000
  4          HAVING  AVG(SAL) > 2000 ;
```

النتائج

deptno	AVG(sal)
10	2916.66667
20	2175

مثال (10) :

```
SQL > SELECT JOB , SUM(SAL)
  2         FROM    emp
  3         WHERE JOB NOT Like 'SALES%'
  4        GROUP BY JOB
  5        HAVING SUM(SAL) > 5000
  6      ORDER BY SUM(SAL) ;
```

النتائج

JOB	SUM(SAL)
ANALYST	6000
MANAGER	8275

الفصل الثامن

عرض واستعراض البيانات من جداول متعددة

➢ **الأهداف :**

١) فهم كيفية عرض البيانات من جداول مختلفة ومفهوم ربط الجداول.

٢) معرفة انواع ربط الجداول:

- الربط بالتساوي Equijoin
- الربط بعدم التساوي Non– Equijoin
- الربط الخارجي Outer Join
- الربط الداخلي Self Join

٣) استخدام الاسماء البديلة أثناء الربط بين الجداول.

في بعض الاحيان نحتاج لعرض بيانات من أكثر من جدول لعمل تقارير مفيدة وشاملة فمثلاً لو أردنا عرض رقم الموظف ورقم القسم التابع له وموقع هذا القسم نجد انه لا بد من الحصول على هذه البيانات من جدولي الموظفين والأقسام كما موضح في الشكل التالي :

EMP table

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

DEPT table

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

EMPNO	DEPTNO	LOC
7839	10	NEW YORK
7698	20	DALLAS
7782	30	CHICAGO
7566	40	BOSTON

البيانات المراد الحصول عليها من الجدولين

➢ الربط بالتساوي Equijoin :

في هذا النوع يتم ربط جدولين او أكثر عن طريق عمودين متساويين العمود الاول مفتاح أساسي Primary key في الجدول الاول والعمود الثاني عمود ربط Foreign key في الجدول الثاني.

والشكل التالي يبين الربط بين جدول الموظفين وجدول الاقسام عن طريق عمود deptno في كل منهما .

EMP table

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

DEPT table

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

Foreign Key **Primary Key**

ونحقق المثال من خلال جملة SELECT التالية :

مثال (1)

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

شرط الربط بين الجدولين

الناتج				
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

استخدام الأسماء المستعارة في جملة SELECT :

يمكن استخدام أسماء مستعارة لجداول لتسهيل كتابة أسماء الاعمدة فمثلاً نقوم باستبدال اسم الجدول emp بـ (e) واسم الجدول dept بـ (d) لتكون الجملة كالتالي :

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

مثال (2) لنعرض البيانات كما في المثال الاول ولكن للموظف king :

```
SQL > SELECT e.empno , e.ename , e.deptno ,
2       d.deptno , d.loc
3       FROM emp  e , dept  d
4       WHERE e.deptno=d.deptno
5      AND e.ename = upper('king');
```

الأسماء المستعارة (e , d)

النتيجة				
EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7839	KING	10	10	NEW YORK

الربط بعدم التساوي Non–Equijoin :

يتم استخدام هذا النوع من الربط عند عدم وجود علاقة ربط مباشرة بين الجدولين أي أننا لا نستخدم اشارة التساوي لكن لا بد من وجود علاقة غير مباسرة مثل شرط معين ينطبق عليهما، فمثلاً لدينا جدول الموظفين وفيه عمود الراتب (SAL) ولدينا الفئات SALGRADE وفي هذا الجدول يتم وضع فئات الراتب وكل فئة تنحصر بين أدنى راتب واعلى راتب فمثلاً الموظف الذي راتبه (3000) يتبع الفئة (4) كما مبين في جدول الفئات أدناه وبذلك نجد علاقة بين الجدولين وهي ان كل راتب من جدول الموظفين يتبع فئة معينة من جدول الفئات أي انه يقع بين أعلى وأدنى قيمة من الجدول، والشكل التالي يوضح تلك العلاقة :

EMP table

EMPNO	ENAME	JOB	SAL
7839	KING	PRESIDENT	5000
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250

SALGRADE table

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

SAL	GRADE
5000	5
2850	4
2450	4
2975	4
1250	2

البيانات التي توضح فئة كل راتب من رواتب الموظفين والتي حصلنا عليها من الجدولين

المثال التالي يوضح كيفية الربط بين الجدولين :

مثال (3)

```
SQL > SELECT e.ename , e.sal, s.grade
  2       FROM   emp e , salgrade s
  3       WHERE e.sal BETWEEN s.losal  AND  s.husal ;
```

شرط الربط بين الجدولين

الناتج		
ENAME	SAL	GRADE
SMITH	800	1
ADANS	1100	1
JAMES	950	1
WARD	1250	2
MARTUN	1250	2
MILLER	1300	2
AKKEN	1600	3
TURNER	1500	3
JONES	2975	4
BLAKE	2850	4
CLARK	2450	4

الربط الخارجي Outer join :

ويتم استخدامه عندما يوجد بينات في احد الجدول ولكنها لا تظهر في حالة الربط بالتساوي بين الجدولين في هذه الحالة نربط بين الجدولين باستخدام الربط بالتساوي مع إضافة الجزء (+) بجانب العمود الفاقد البيانات، فمثلاً يوجد القسم رقم (40) في جدول الأقسام ولكن لا يوجد فيه موظفين مسجلين في الجدول الوظائف لذلك عند الربط بالتساوي فإن هذا القسم لا يظهر في المخرجات لعدم تطابق شرط التساوي عليه ولإظهار هذا القسم لا بد من استخدام الربط الخارجي.

المثال التالي يبين كيفية الربط بين جدولين لإظهار كافة البيانات الموجودة في الجدولين باستخدام الربط الخارجي سواء كانت البيانات مطابقة لشرط التساوي او غير مطابقة.

مثال (4) :

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

علامة الربط الخارجي

الناتج			
EMPNO	**ENAME**	**DEPTNO**	**DNAME**
7782	CLARK	10	ACCPUNTING
7839	KING	10	ACCPUNTING
7934	MILLER	10	ACCPUNTING
7369	SMITH	20	RESEARCH
7876	ADAMS	20	RESEARCH
7902	FORD	20	RESEARCH
7788	SCOTT	20	RESEARCH
7566	JONES	20	RESEARCH
7499	ALLEN	30	SALES
7698	BLAKE	30	SALES
7654	MARTIN	30	SALES
7900	JAMES	30	SALES
7844	TURNER	30	SALES
7521	WARD	30	SALES
		40	OPERATIONS

هذه الإدارة ظهرت لاستخدامنا الربط الخارجي

الربط الداخلي لنفس الجدول Self join :

EMP table

EMPNO	ENAME	JOB	MGR
7839	KING	PRESIDENT	
7698	BLAKE	MANAGER	7839
7782	CLARK	MANAGER	7839
7566	JONES	MANAGER	7839
7654	MARTIN	SALESMAN	7698

EMP (WORKER)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698

EMP (MANAGER)

EMPNO	ENAME
7839	KING
7698	BLAKE
7782	CLARK
7566	JONES
7654	MARTIN

عندما ندقق في جدول الموظفين نجد فيه عمود MGR ويمثل رقم المدير للموظف فنجد الموظف BLAKE مديره الموظف ذو الرقم (7839) أي الموظف KING ومن هنا يتضح وجود علاقة بين عمود المدير MGR ورقم الموظف EMPNO ونستطيع ربط الجدول بنفسه عن طريق هذين العمودين.

والمثال التالي يوضح الربط الداخلي :

مثال (5) :

```
SQL > SELECT WORKER.empno , WORKER.ename ,
MANAGER.ename manger
2        FROM emp worker , emp manager
3        WHERE worker.mgr  = manager.empno ;
```

الناتج		
EMPNO	ENAME	MANAGER
7369	SMITH	FORD
7499	ALLEN	BLAKE
7521	WARD	BLAKE
7566	JONES	KING
7654	MARTIN	BLAKE
7698	BLAKE	KING
7782	CLARK	KING
7788	ACOTT	JONES
7844	TURNER	BLAKE
7876	ADAMS	SCOTT
7900	JAMES	BLAKE
7902	FORD	JONES
7934	MILLER	CLARK

➢ الربط بين أكثر من جدولين :

لربط أكثر من جدولين لا بد من وجود علاقة ما بينهم جميعاً ويجب أم يكزن عدد جمل الشرط مساوي (لعدد الجداول المرتبطة – 1) فمثلاً إذا تم ربط ثلاثة جداول فعدد جمل الشرط اثنتان ونضع الشرط المعامل AND .

المثال التالي يبين ربط ثلاثة جداول مع بعضها لعرض بيانات من كل منها.

مثال (6) :

```
SQL > SELECT e.empno , e.ename , e.sal , d.dname , s.grade
2       FROM    emp  e , dept  d  ,  salgrade  s
3       WHERE e.deptno = d.deptno
4       AND   e.sal  BETWEEN  s.losal and s.hisal
```

شرط الربط بين جدول الموظفين وجدول الإدارة

شرط الربط بين جدول الموظفين وجدول الفئات

الناتج				
EMPNO	**ENAME**	**SAL**	**DNAME**	**GRADE**
7369	SMITH	800	RESEARCH	1
7876	ADAMS	1100	RESEARCH	1
7900	JAMES	950	SALES	1
7821	WARD	1250	SALES	2
7654	MARTIN	1250	SALES	2
7934	MILLER	1300	ACCOUNTING	2
7499	ALLEN	1600	SALES	3
7844	TURNER	1500	SALES	3
7566	JONES	2975	RESEARCH	4
7698	BLAKE	2850	SALES	4
7782	CLARK	2450	ACCOUNTING	4
7788	SCOTT	3000	RESEARCH	4
7902	FORD	3000	RESEARCH	4
7839	KING	5000	ACCOUNTING	5

الفصل التاسع

الاستعلامات الفرعية SUBQUERIES

➢ الأهداف :

١) معرفة متى وكيفية استخدام الاستعلامات الفرعية وانواعها.

٢) معرفة استخدام الاستعلام الفرعي أحادي الصف Single Row Subquery .

٣) استخدام معاملات المقارنة ذات الصف الواحد مع الاستعلام الفرعي أحادي الصف .

٤) معرفة واستخدام الاستعلام الفرعي متعدد الصفوفMultiple Row Subquery.

٥) استخدام معاملات المقارنة متعددة الصفوف مع الاستعلام الفرعي متعدد الصفوف.

➢ لنفرض اننا أن نعرف الموظفين الذين يأخذون راتب اعلى من راتب JONES ففي هذه الحالة نقوم بعمل استعلام اول لمعرفة راتب JONES :

```
SQL SELECT SAL
  2  FROM emp
  3     WHERE ename='jones';
```

SAL
2975

واستعلام الثاني لمعرفة الموظفين الذين رواتبهم اعلى من JONES

```
SQL SELECT ename
  2       FROM  emp
  3       WHERE sal > 2975
```

ENAME
SCOTT
KING
FORD

وهكذا وصلنا للنتيجة باستخدام استعلامين ولكننا نستطيع الحصول على النتيجة بدمج الاستعلامين بجملة واحدة بأن نجعل الاستعلام الثاني رئيسياً والاستعلام الاول فرعياً كالتالي :

```
SQL SELECT ename
  2    FROM emp
  3    WHERE sal > (SELECT  sal  FROM  emp  WHERE ename = 'JONES')
```

مثال (1) : عرض أرقام وأسم ووظائف الموظفين الذين يعملون بنفس وظيفة ALLEM :

```
SQL >  SELECT empno , ename , job
  2          FROM    emp
  3          WHERE job =
                    ( SELECT job
                      FROM emp
                      WHERE ename = 'ALLEN')
```

SALESMAN

الاستعلام الداخلي (الفرعي)

الناتج		
EMPNO	**ENAME**	**JOB**
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7654	MARTIN	SALESMAN
7844	TURNER	SALESMAN

➢ أنواع الاستعلامات الفرعية :

- استعلام فرعي احادي الصف ويرجع بصف واحد Single-Row Subquery.
- استعلام فرعي متعدد الصفوف ويرجع بأكثر من صف Multiple- Row Subquery.
- استعلام فرعي متعدد الاعمدة ويرجع بأكثر من عمود Multiple- Column Subquery.

➢ متطلبات الاستعلام الفرعي:

- يجب وضع الاستعلام الفرعي بين قوسين .
- يجب استخدام معاملات المقارنة ذات الصف الواحد مع الاستعلام الفرعي أحادي الصف.
- يجب استخدام معاملات المقارنة متعددة الصفوف مع الاستعلام الفرعي متعدد الصفوف .

➢ أنواع معاملات المقارنة المستخدمة في الاستعلام الفرعي :

- معاملات احادية الصف مثل (<> , >= , <= , > , < , =) وتستخدم مع الاستعلامات الفرعية احادية الصف أي التي ترجع بصف واحد (قيمة واحدة) .
- معاملات متعددة الصف مثل (IN , ANY , ALL) وتستخدم مع الاستعلامات الفرعية متعددة الصفوف أي التي ترجع بأكثر من صف .

➢ أماكن كتابة الاستعلامات الفرعية داخل جملة SELECT :

- ضمن الأجزاء التالية (FROM , HAVING , WHERER).

الاستعلامات الفرعية أحادية الصف :

وتكون نتيجتها دائماً صف واحد لذلك يستخدم معها المعاملات الأحادية(<> , >= , <= , > , < , =) .

مثال (2) :

```
SQL > SELECT ename  , sal , deptno
  2         FROM   emp
  3         WHERE  deptno  =
  4                                (select  deptno
  5                                 FROM emp
  6                                 where ename = 'KING')
```

الناتج		
ENAME	SAL	DEPTNO
CLARK	2450	10
KING	5000	10
MILLER	1300	10

مثال (٣) عرض أسماء ووظائف ورواتب الموظفين الذين رواتبهم مساوية لأصغر راتب :

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

الناتج		
ENAME	JOB	SAL
SMITH	CLEARK	800

مثال (4) عرض أرقام الأقسام واقل راتب فيها بحيث يكون أقل راتب فيها أكبر من أقل راتب في القسم رقم 20 :

```
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 = 20 )
```

الناتج	
DEPTNO	MIN(SAL)
10	1300
30	950

مثال (5) عرض أسماء ووظائف الموظفين الذين يعملون نفس وظيفة الموظف ذو الرقم (7369) ويأخذون راتب اكبر من راتب الموظف ذو الرقم (7876) :

```
SQL > SELECT ename , job
  2      FROM emp                  CLERK
  3      WHERE job =
  4                  ( select  job
  5                    from emp    1100
  6                    where empno = 7369)
  7       AND   SAL >
  8                   (select sal
  9                    from emp
 10                    where  empno = 7876 ) ;
```

الناتج	
ENAME	JOB
MILLER	CLERK

مثال (6) عرض اسماء ووظائف الموظفين الذين يعملون نفس وظيفة SMITH :

```
SQL > SELECT ename  , job
  2         FROM   emp
  3          WHERE job =
  4                              (select   job
  5                               from emp ) ;
ERROR at Line:

ORA-01427: single- row subquery return more then one row
```

رسالة الخطأ <_____

ولتصحيح ذلك نكتب :

```
SQL> SELECT ename , job
  2        FROM emp
  3         WHERE   job =
  5                              ( select    job
  6                               FROM emp
  7                               WHERE ename='SMITH')
```

➢ الاستعلامات الفرعية متعددة الصفوف Multi Row Subqeries:

وهي استعلامات ترجع دائماً أكثر من صف لذلك تستخدم فيها المعاملات المتعددة الصفوف (**in , any , all**)

والجدول التالي يبين معنى هذه المعاملات :

١	**In**	المساواة بأي قيمة داخل قائمة
٢	**Any**	مقارنة قيمة بأي من قيم داخل قائمة
٣	**<ANY**	معناها أقل من أكبر قيمة داخل قائمة
٤	**>ANY**	معناها أكبر من أقل قيمة داخل قائمة
٥	**ALL**	مقارنة قيمة بكل ما هو موجود داخل قائمة
٦	**<ALL**	معناها أقل من أقل قيمة داخل قائمة
٧	**>ALL**	معناها أكبر من أعلى قيمة داخل قائمة

مثال (7) عرض أسماء ورواتب وأرقام أقسام الموظفين الذين يأخذون رواتب مساوية لأقل راتب في كل إدارة:

هذا القسم نجد انه لا بد من الحصول على هذه البيانات من جدولي الموظفين والأقسام كما موضح في الشكل التالي : القسم التابع له وموقع

```
SQ> SELECT ename , sal , deptno
  2      FROM   EMP
  3      WHERE   SAL   IN   (select MIN (sal)
  4                          FROM emp
  5                          group   by   deptno ) ;
```

الناتج

ENAME	SAL	DEPTNO
SMITH	800	20
JAMES	950	30
MILLER	1300	10

مثال (8) عرض أرقام وأسماء ووظائف ورواتب الموظفين الذين رواتبهم أقل من راتب الموظفين أصحاب الوظيفة CLERK دون عرض موظفي الوظيفة CLERK:

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

النتائج

EMPNO	ENAME	JOB
7521	WARD	SALESMAN
7654	MARTIN	SALESMAN

مثال (9) عرض أرقام ووظائف ورواتب الموظفين الذين رواتبهم أكبر من كل المتوسطات الحسابية للرواتب في كل إدارة :

```
SQL > SELECT empno , ename , job , sal
  2       FROM   emp
  3       WHERE  SAL > ALL
  4                              ( select  AVG (sal )
  5                                from emp
  6                                group  by deptno ) ;
```

النتائج

ENAME	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

الفصل العاشر

الاستعلامات الفرعي المتعدد الاعمدة

➢ **الأهداف :**

١) فهم كيفية استعلامات الفرعية المتعددة الأعمدة.

٢) معرفة نتيجة الاستعلام الرئيسي عندما يرجع الاستعلام الفرعي بالنتيجة NULL

٣) استخدام الاستعلام الفرعي في جملة FROM

- في الفصل السابق تم معرفة نوعين من الاستعلامات الفرعية النوع الاول الاستعلام الفرعي ذو الصف الواحد والذي يرجع دائماً بصف واحد اما النوع لآخر فهو الاستعلام المتعدد الصفوف الذي يرجع بأكثر من صف ويوجد نوع ثالث من الاستعلامات وهو الاستعلام المتعدد الأعمدة والذي يرجع بأكثر من عمود واكثر من صف ولمقارنة ناتج هذا النوع لا بد من استعمال المعاملات المتعددة الصفوف والتي تم دراستها في الفصل السابق

والشكل التالي يبين الصيغة العامة لجملة الاستعلامات الفرعية المتعددة الاعمدة .

الاستعلام الخارجي (الرئيسى)

- **SQL > SELECT** مجموعة الاعمدة المطلوبة
  FROM الجدول
  WHERE (العمود ١ ، العمود ٢ ، .....) **IN**

(**select** ..... ، العمود ٢ ، العمود ١
FROM الجدول
WHERE الشرط

الاستعلام الداخلي (الفرعي)

➢ استخدام الاستعلام الفرعي في جملة FROM :

يمكن استخدام الاستعلام الفرعي في الجزء FROM من جملة الاستعلام SELECT وذلك لعمل مصدر بيانات آخر غير الجدول حيث كما نعرف انه يستخدم اسم الجدول ضمن الجزء FROM لآنه من مصادر البيانات والمثال التالي يوضح كيفية استخدام الاستعلام الفرعي ضمن الجزء FROM .

مثال (3) اعرض أسماء ورواتب وأرقام الأقسام والمتوسط الحسابي لرواتب الموظفين الذين يأخذون رواتب أعلى من المتوسط الحسابي لأقسامهم :

```
SQL > SELECT e.ename , e.sal, e.deptno, esub.salavg
2       FROM   emp  e, (select deptnom AVG(sal) salavg)
3                       FROM   emp
4                       group  by  deptno ) esub
5       WHERE  e.deptno = esub.deptno
6       AND    e.sal  >  esub.salavg
```

الاستعلام الداخلي (الفرعي) يعامل الجدول

الناتج

ENAME	SAL	DEPTNO	SALAVG
KING	5000	10	2916.66667
FORD	3000	20	2175
SCOTT	3000	20	2175
JONES	2975	20	2175
ALLEN	1600	30	1566.66667
BLAKE	2850	30	1566.66667

نلاحظ اننا استخدمنا الاستعلام الفرعي وكأنه جدول موجود باسم esub ويتكون هذا الجدول من عمودين الاول يمثل أرقام الأقسام والثاني يمثل المتوسطات الحسابية للرواتب في كل قسم وبياناته كالآتي :

```
SQL > SELECT deptno ,AVG(sal) salavg
2           FROM    emp
3           group by  deptno   ;
```

الناتج	
DEPTNO	**SALAVG**
10	2916.66667
20	2175
30	1566.66667

ولأننا نقوم بعرض البيانات من جدولين فلا بد من ربطهما أي نقوم بربط الجدول EMP بالجدول ESUB المكون بالاستعلام الفرعي.

الفصل الحادي عشر

التعامل مع البيانات(لغة معالجة البيانات)

➢ الأهداف :

١) معرفة لغة التعامل مع البيانات**Data Manipulation Language(DML)**

٢) معرفة كيفية إضافة سجل أو أكثر إلى جدول بعبارة **Insert Into**

٣) معرفة كيفية تعديل بيانات جدول معين بعبارة **UPDATE**

٤) كيفية حذف صفوف من جدول معين بعبارة **DELETE FROM**

الكلمات المفتاحية :

- تعليمة، صيغة، جدول، حقل، سجل، عمود، استعلام، استعلام فرعي.

ملخص :

تم تصنيف تعليمات لغة SQL إلى ثلاثة أصناف رئيسية ضمن ثلاث لغات فرعي: لغة معالجة البيانات، ولغة تعريف البيانات، ولغة التحكم بالبيانات. ومن بين هذه التعليمات تعتبر تعليمات معالجة البيانات الاكثر استخداماً. سنستعرض تعليمات المعالجة والتي تشمل SELECT , INSERT , DELETE , UPDATE وبعض الكلمات المفتاحية المُستخدمة معها.

سيتعرف الطالب في الفصل على قسم مهم من لغة SQL وهي لغة التعامل مع البيانات DEL حيث تمكنا من التعامل من البيانات ضمن الجدول حيث نتمكن من خلالها إضافة سجل أو أكثر أو تعديل سجل معين أو مجموعة سجل او محي سجل معين أو مجموعة سجلات.

وتتكون لغة DEL من عدة جمل كالتالي :

- جملة إضافة بيانات إلى الجدول **INSERT INTO** .
- جملة التعديل في بيانات الجدول **UPDATE** .
- جملة حذف بيانات من الجدول **DELETE FROM** .

سنقوم في الفصل بشرح هذه الجمل بالتفصيل .

إضافة سجل أو عدد من السجلات إلى جدول معين (INSERT INTO) :

الصيغة العامة :

```
SQL > INSERT INTO جدول (عامود ١ ، عامود ٢ ، عامود ٣ ، .....)
VALUES (قيمة ١ ، قيمة ٢ ، قيمة ٣ ، .....)
```

➢ **القواعد التي نلتزم بها عند إضافة سجل أو عدد من السجلات إلى جدول معين :**

١) يجب أن يكون عدد القيم التي سيتم إدخال مساوياً لعدد الاعمدة المذكورة في جملة INSERT .

٢) يجب ان تكون القيم مرتبة بنفس ترتيب الأعمدة المراد إدخال القيم بها كما يجب أن تكون القيم من نفس بيانات الاعمدة.

٣) عند إدخال قيم التاريخ والنصوص لا بد من وضعها داخل علامتي تنصيص فرديتين ' ' .

٤) يجب إدخال قيماً للأعمدة التي لا تقبل قيماً فارغة NULL كأعمدة المفتاح الأساسي على سبيل المثال.

٥) في حال عدم ذكر أسماء الاعمدة في جملة INSERT لا بد من إدخال جميع قيم الأعمدة الموجودة في الموجودة حسب ترتيب الأعمدة في الجدول مع مراعاة نوع البيانات لكل عمود .

مثال (1) إضافة سجل جديد إلى جدول الأقسام :

```
SQL> INSERT INTO dept (deptno , dname , loc )
  2  VALUES             ( 50 , 'DEVELOPMENT' , 'DETROIT') ;
```

جدول الإدارات (DEPT) قبل الإضافة

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

جدول الإدارات (DEPT) بعد الإضافة

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON
50	DEVELOPMENT	DETROIT

السجل (الصف) الذي تم إضافته

- إضافة قيمة NULL إلى عمود ويتم بطريقتين بشرط أن تقبل الأعمدة قيمة NULL :
  - الأولى عدم كتابة الأعمدة المراد تسجيل قيمة NULL فيها في الجزء INSERT .
  - أن نكتب الاعمدة ولكن نكتب فيها قيمة NULL داخل الجزء VALUES .

مثال (2) إضافة سجل جديد إلى جدول الأقسام يحتوي هذا السجل على الرقم القسم واسمه فقط :

```
SQL > INSERT INTO dept (deptno , dname )
2       VALUES                (60  , 'MIS') ;
```

الناتج

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DAKKAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	

قيمة فارغة NULL

- نلاحظ أنه لم يتم إضافة الموقع إلى جدول الاقسام وبشكل افتراضي تم إضافة قيمة NULL وممكن ان تكون الجملة السابقة كالتالي :

```
SQL > INSERT INTO dept (deptno , dname , loc )
2                       (60 , 'MIS' , NULL)
```

➢ إضافة قيم خاصة داخل الاعمدة :

إذا أردنا تاريخ اليوم SYSDATE إلى العمود HIREDATE في جدول الموظفين يمكن ذلك كالآتي :

```
SQL > INSERT INTO emp
2   (empno , ename , job , mgr , hiredate , sal , comm , deptno)
3   VALUES
4   (7196 , 'AHMAD' ,' SALESMAN' , 7782 , SYSDATE , 2000 , NULL , 10) ;
```

مثال (3) إضافة بيانات موظف جديد إلى جدول الموظفين :

في المثال السابق من الممكن عدم ذكر أسماء الأعمدة في الجملة INSERT وفي هذه الحالة لا بد من إدخال كل القيم حسب ترتيب الأعمدة في الجدول كالآتي :

```
SQL > INSERT INTO  emp
      VALUES
      (7196 , 'AHMED' , 'SALESMAN' , 7782 , SYSDATE , 2000 , NULL , 10)
```

➢ إضافة سجلات عن طريق المتغيرات البديلة Substitution Variables :

تمكننا لغة SQL من عمل متغيرات تسمى المتغيرات البديلة وهي مخزن مؤقت البيانات يتم من خلالها تخزين قيم معينة وأثناء تنفيذ جمل SQL يتم استبدال هذه المتغيرات بقيمها ويتم تعريفها أثناء كتابة جمل SQL بوضع & قبل اسم المتغير كما تظهر رسالة تسأل عن قيمة هذه المتغيرات أثناء التنفيذ ونوضح ذلك في المثال الآتي :

مثال (4) إضافة قسم إلى الأقسام باستخدام المتغيرات البديلة :

```
SQL > INSERT  INTO  dept ( deptno , dname , loc)
  2        VALUES              ( &dept_id , '&dept_name' , '&dept_loc') ;
```

```
Enter value for dept _ id  : 80

Enter value for dept_ name  : EDUCATION

Enter value for dept _loc  : ATLANTA
```

ويمكن تكرار هذا الأمر عدة مرات لإضافة أكثر من قسم دون تكرار كتابة الأمر وهكذا نكون قد أضفنا قسم جديد إلى جدول

الأقسام وكأننا كتبنا الأمر التالي :

```
SQL > INSERT  INTO dept (deptno , dname , loc )
  2        VALUES        ( 80 , 'EDUCATION' , 'ATLANTA')
```

- يمكن استخدام المتغيرات البديلة بدلاً من أسماء الجدول والأعمدة كما في المثال التالي :

مثال (5) إضافة قسم جديد إلى جدول الأقسام باستخدام المتغيرات البديلة بدلاً من أسم الجدول واسم العمود :

```
SQL > INSERT  INTO &dept_table (&dept_id  , dname , dname , loc )
  2      VALUES                    (80 , 'EDUCATION'  , 'ATLANTA')
Enter  value  for  dept_table   :  dept

Enter  value  for dept_id  : deptno
```

إضافة سجلات جديدة عن طريق نسخها من جدول آخر :

لنفرض لدينا جدولاً اسمه Managers فيه بيانات المدراء وجدولاً آخر للموظفين emp ونريد إضافة الموظفين الذين يعملون كمدراء إلى جدول المدراء كما في المثال التالي :

مثال (6) : إضافة الموظفين الذين يعملون كمدراء إلى جدول المدراء :

```
SQL > INSERT INTO managers ( id , name , salary , hiredate )
  2       select       empno , ename , sal , hiredate
  3       from          emp
  4       where  job = 'MANAGER' ;
```

التعديل في سجل أو عدد من السجلات في جدول معين (UPDATE) :

الصيغة العامة :

```
SQL > UPDATE جدول
        SET عمود ١ = قيمة ١ ، عمود ٢ = قيمة ٢.....
        WHRER N شرط ;
```

القواعد الواجب التقيد بها عند التعديل :

- يجب أن تكون البيانات الجديدة من نفس بيانات الأعمدة المراد التعديل بها.
- عند تعديل قيم النصوص والتاريخ يجب وضع القيم الجديدة بين علامتي التنصيص الفردية.
- يجب الانتباه عند كتابة الجزء WHERE في جملة التعديل لتحديد أي الصفوف التي سيتم التعيد بها .

مثال (7) تعديل القسم رقم (30) ليصبح EDUCATION بدلاً من SALES :

```
SQL > UPDATE  dept
  2       SET  dname = 'EDUCATION'
  3       WHERE deptno = 30 ;
```

القيمة الجديدة لاسم الإدارة رقم(30)

1 Row updated ← رسالة تدل على أن التعديل تم فى الإدارة رقم (30) فقط

يجب الانتباه لمضمون الشرط where في جملة update فمثلاً في المثال التالي سيتم تعديل كافة الأقسام لتصبح EDUCATION :

مثال (8) :

```
SQL > UPDATE  dept
  2       SET  dname = 'EDUCATION' ;
```

2 Row updated ← رسالة تدل على أن التعديل تم فى جميع الصفوف

التعديل في أكثر من عمود :

إذا أردنا تعديل القسم والوظيفة للموظف BLAKE ليصبح مثل الموظف WARD لنرى المثال التالي :

مثال (9) :

```
SQL > UPDATE  emp
  2    SET (job , deptno) = (select job , deptno  from emp where ename =
'WARD')
```

نتيجة الاستعلام الفرعي (30 MANAGER)

1 Row updated

➢ حذف سجل أو عدد من السجلات داخل جدول DELETE FROM :

الصيغة العامة :

```
SQL > DELETE FROM    جدول
      WHERE          شرط
```

➢ القواعد التي يجب التقيد بها عند الحذف :

يجب الانتباه لمضمون الشرط WHERE لتحديد الصفوف المراد حذفها وعند عدم كتابة الشرط سيتم حذف كافة الصفوف في الجدول.

مثال (10) : حذف القسم رقم (40) من جدول الأقسام :

```
SQL > DELETE  FROM dept
      WHERE  deptno = 40 ;
```

1 Row deleted.

مثال (11) حذف جميع الموظفين من جدول الموظفين

```
SQL > DELETE FROM  emp
```

14 Row deleted

مثال (12) حذف موظفي القسم SALES :

```
SQL > DELETE  FROM  emp
2     WHERE  deptno =(select  deptno  from  dept  where dname=
'SALES')
```

6 Row deleted

إذا أردنا حذف القسم رقم (10) من جدول الأقسام ستظهر رسالة خطأ تبين تجاوز قيد ربط جدول الأقسام بجدول الموظفين ومعنى ذلك انه لا يمكن حذف القسم (10) لأنه يوجد موظفون مسجلون في هذا القسم من الجدول EMP وذلك لوجود ربط الجدولين عن طريق DEPTNO .

➢ عمليات قواعد البيانات Database Transactions :

كل عملية من عمليات الإضافة أو الحذف أو التعديل لا تتم بشكل نهائي إلا بالأمر commit التي تعادل الامر SAVE ومن Database Transactions :

- **Commit** حيث يقوم هذا الأمر بحفظ البيانات التي جرى عليها عمليات إضافة أو حذف أو تعديل.
- **Rolback** يقوم هذا الأمر بالتراجع عن عمليات الحذف والإضافة والتعديل.

الفصل الثاني عشر

إنشاء الجداول

➢ الأهداف :

١) معرفة انواع الكائنات في قاعدة البيانات.

٢) معرفة انواع البياناتDATA TYPES .

٣) معرفة كيفية إنشاء الجداول.

٤) معرفة كيفية إنشاء الجداول باستخدام الاستعلامات الفرعية.

٥) معرفة كيفية التعديل في بناء الجدول وكيفية إلغاء جدول من قاعدة البيانات.

➢ الجداول TABLES :

تعد الجداول المكون الأساسي لقاعدة البيانات، حيث يحوي الجدول مجموعة من ا لأعمدة والسطر تخزن البيانات فيها، يسمى كل عمود (Column) بالحقل (Field)، ويحدد نوع معطيات (نمط البيانات data Type) كل السجل من قيم حقول الجدول الموجودة في سطر واحد، وبالتالي الجدول يتكون من مجموعة سجلات، ومجموعة الجداول تكون قاعدة البيانات.

➢ عند إنشاء أي جدول في قاعدة البيانات تحدد المعلومات الآتية عن كل حقل في الجدول :

١. اسم الحقل (العمود) Column Name .

٢. نوع او نمط الحقل data type ونذكر منها :

- **Date** : لتخزين البيانات بشكل تاريخ .
- **Float** : لتخزين عدد حقيقي .
- **Image** : لتخزين صورة بشكل ثنائي.
- **Int** : لتخزين عدد صحيح.
- **Money** : لتخزين القيمة المدخلة بصيغة عملة.
- **Varchar(50)** : لتخزين معطيات بشكل نص، طوله الأعظمي 50 محرفاً.
- **Text** : لتخزين البيانات بشكل نص، يحوي عدد كبير من المحارف.
- **Nvarchar(50)** : لتخزين معطيات بشكل نص طوله الاعظمي 50 محرفاً مع إمكانية إدخال قيم بلغات اخرى كالعربية .
- **NTeaxt** : لتخزين البيانات بشكل نص، يحوي عدداً كبيراً من المحارف مع إمكانية إدخال نصوص بلغات اخرى كالعربية.

٣. تحديد إمكانية/ عدم إمكانية أن تكون قيمة الحقل فارغة Allow Nulls .

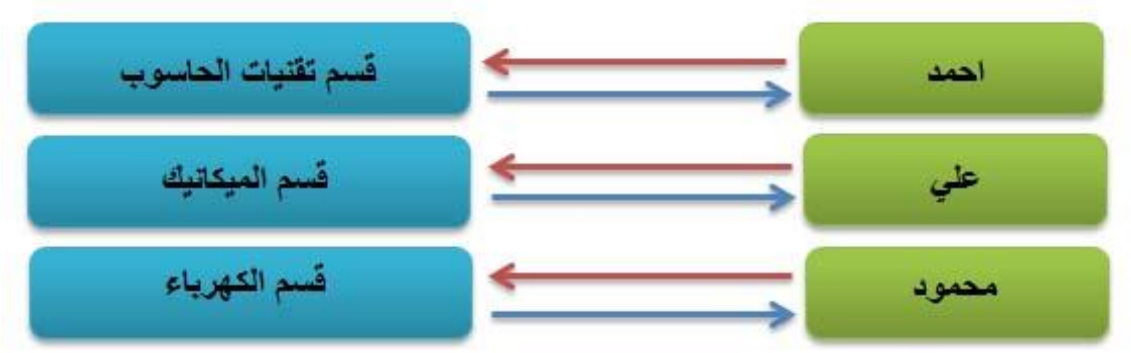
العلاقات بين الجداول :

تستخدم العلاقات او الرابط بين الجداول للحصول على جداول أخرى تحوي معطيات مستخلصة من الجداول المرتبطة مع بعضها بهدف العلاقات، وتكون العلاقة التي تربط بين جدولين من جداول قاعدة البيانات بإحدى الأشكال الآتية:

١. علاقة واحد إلى واحد (One To One) :

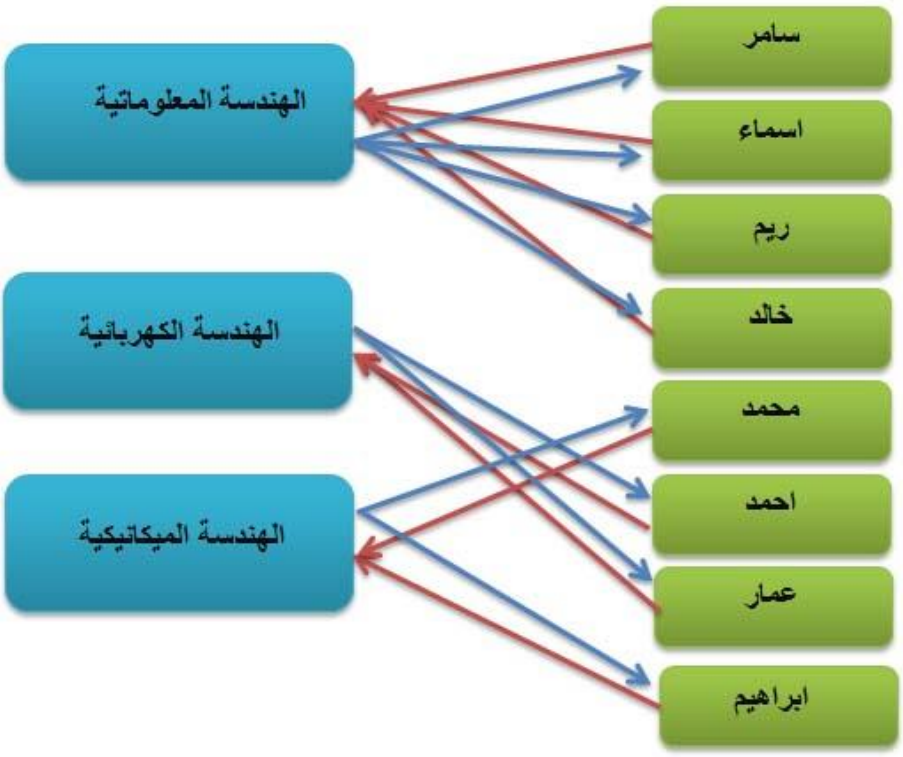
يكون كل سجل من الجداول الاول بسجل واحد فقط من الجدول الثاني، وكل سجل من الجدول الثاني مرتبط بسجل واحد فقط من الجدول الاول، مثل العلاقة بين أسماء المدراء وجدول الاقسام حيث أن لكل مدير قسم واحد فقط، ولكل قسم مدير واحد فقط.

وفي هذه الحالة يجب ان يحوي الجدول الثاني مفتاحاً يمثل المفتاح الرئيسي للجدول الاول أو بالعكس، أي يجب ان يكون في الجدول المدراء حقل يمثل رقم القسم، او ان يكون في الجدول الأقسام حقل يمثل رقم المدير، كما هو مبين في الشكل :



٢. علاقة واحد إلى عدة (كثير) (One To Many) :

يرتبط كل سجل من الجدول الأول بسجل واحد أو أكثر من الجدول الثاني، وكل سجل من الجدول الثاني يرتبط بسجل واحد فقط من الجدول الاول، مثل العلاقة بين جدول أسماء الكليات وجدول اسماء الطلاب، فكل كلية تحوي اكثر من طالب بينما كل طالب مسجل بكلية واحدة فقط، كما هو مبين في الشكل:



في هذه الحالة يجب كسر هذه العلاقة وإنشاء جدول ثالث وسيط يحوي مفتاحاً مركباً (يتألف من مفتاحين أجنبيين هما المفتاحان الرئيسيان لجدولي الطلاب والمواد)، ويحوي معلومات مرتبطة بكل من الجدولين (مثلاً علامة الطالب في المادة).

أي يتم العلاقة Many To Many بين الجدولين إلى علاقتي One To Many بين الجدول الأول (الطلاب) والجدول الوسيط، وبين الجدول الثاني (المواد) والجدول الوسيط.

- سنتعرف في هذا الفصل على لغة تعريف البيانات DATA Definition Language والتي يرمز لها DDL وهي التي تمكننا من إنشاء وتعديل وحذف أي كائن من قاعدة البيانات ومن أهم هذه الكائنات الجداول التي سنركز على إنشائها والتعديل في بياناتها وإلغائها لنرى بعض الكائنات التي تتكون منها قاعدة البيانات:

١	**Table**	هو الوحدة الاساسية لمكونات قاعدة البيانات والتي نستخدمها في حفظ البيانات ويتكون من عدة صفوف وأعمدة.
٢	**View**	المناظير: عبارة عن جزء مؤقت من جدول معين يتكون من عدة صفوف واعمدة ويستخدم لغرض معين بشكل مؤقت.
٣	**Sequence**	سلسلة: عبارة عن سلسلة تستخدم لتوليد أرقام متتالية بشكل معين دون تكرار لذلك يفضل استخدامها لتسجيل بيانات المفتاح الأساسي داخل جدول.
٤	**Index**	فهرس : ويستخدم في عملية فهرست بعض الاعمدة لتسهيل عملية البحث فيها عن معلومة معينة، وأيضاً لتقليل وقت الاستفسارات من الجدول.
٥	**Synonym**	مرادفات : تستخدم لإعطاء أثر من أسم على كائن معين.

- **أنواع البيانات Data Types :**

يوجد أنواع متعددة البيانات التي تخزن في الجداول فمنها العددية والمحرفية وبيانات التاريخ وبيانات أخرى نبين بعضها في الجدول التالي:

	أنواع البيانات	الوصف
١	**Varchar2 (الحجم)**	تستخدم مع البيانات الحرفية المتغيرة الطول.
٢	**Char (الحجم)**	تستخدم مع البيانات الحرفية الثابتة الطول لابد من تحديد طول البيانات الحرفية.
٣	**Number (p ,s)**	تستخدم مع البيانات الرقمية ويمثل الحرف (P) الجزء الصحيح قبل العلامة العشرية، والحرف (S) يمثل الجزء العشري بعد العامة العشرية.
٤	**Date**	تستخدم مع بيانات التاريخ والوقت
٥	**Long**	تستخدم لتمثيل البيانات الكبيرة الحجم التي تصل إلى (2) جيجا بايت.
٦	**CLOB – BLOB**	تستخدم لتمثيل البيانات الكبيرة مثل الصور والرسومات والتي تصل حجمها إلى أكثر من (4) جيجا بايت.
٧	**Bfile**	تستخدم لتخزين الملفات الكبيرة والخارجية والتي يصل حجمها إلى أكثر من (4) جيجا بايت.

الشروط الواجب توافرها عند اختيار أسماء الجداول وأسماء الأعمدة:

١- يجب ان يبدأ اسم الجدول أو اسم العمود بعرف.

٢- يجب أن لا يطول الاسم عن (30) محرف وممكن أن يحتوي الاسم على حروف كبيرة وصغيرة ورموز.

٣- يجب أن لا يتكرر اسم الجدول داخل قاعدة البيانات ولا اسم العمود داخل الجدول.

٤- يجب أن لا يكون الاسم من الأسماء المحجوزة لأوراكل أو **SQL** مثلاً (Select , From , ...)

٥- يفضل أن يكون الاسم له معنى بحيث يعبر عن نوع البيانات داخله.

إنشاء الجداول :

الصيغة العامة :

```
SQL > CREATE Table  أسم الجدول (
                        نوع البياناتالعمود 1 ,
                        نوع البياناتالعمود 2 ,
                        نوع البياناتالعمود 3 )
```

مثال (1) إنشاء جدول الأقسام dept2:

```
SQL > CREATE TABLE DEPT2 (
2        deptno   NUMBER (2) ,
3        dname    VARCHr2 (14) ,
4        loc      VARCHr2 (13)  ) ;
```

اسماء الأعمدة

نوع البيانات

إنشاء الجداول باستخدام الإنشاءات الفرعية :

يمكن إنشاء الجداول باستخدام جدول موجود مسبقاً في قاعدة البيانات كإنشاء جدول يحتوي على بعض الأعمدة من جدول آخر دون الحاجة إلى بيانات الموظفين للقسم رقم (30) عن طريق الجدول emp فنقوم بكتابة الامر كالتالي :

مثال (2) : إنشاء جدول للقسم رقم (30) باستخدام الجدول EMP :

```
SQL > CREATE TABLE dept 30
2       AS
3       SELECT empno , ename , sal*12 annsal , hiredate
4       FRPM emp
5       WHERE deptni = 30 ;
```

TABLE CREATED

وعندما تستعرض البيانات من جدول dept30 تكون النتيجة كالتالي :

```
SQL > SELECT * FROM dept 30 ;
```

الناتج

EMPNO	ENAME	ANNSAL	HIREDATE
7499	ALLEN	19200	20-FEB-81
751	WARD	15000	22-FEB-81
7654	MARTIN	15000	28-SEP-81
7698	BLAKE	34200	01-MAY-81
7844	TURNER	18000	08-SEP-81
7900	JANES	11400	03-DEC-81

وعندما نريد إنشاء الجدول بأسماء أعمدة غير الاسماء الاساسية نقوم بما يلي :

مثال (3) إنشاء جدول للقسم رقم (20) باستخدام جدول الموظفين يحتوي على اسماء اعمدة تختلف عن اسماء الاعمدة الاساسية :

```
SQL > CREATE  TABLE  dept20
2       (emp_id , emp_name , salay  , start_date)
3        AS
4        SELECT   empno  ,  ename  , sal , hireda
5        FROM    emp
6        WHERE   deptno = 20  ;
```

الاستعلام الفرعي

Table created

التعديل في الجداول باستخدام ALTER :

توفر لغة SQL إمكانية لتعديل البنية الداخلية للجداول باستخدام الامر ALTER TABLE وتشمل ثلاثة تعديلات إما إضافة اعمدة أو التعديل في نوع البيانات أو حذف اعمدة كما موضح في الجدول التالي :

أوجه التعديل في الجدول باستخدام الأمر ALTER TABLE	
ADD	تستخدم لإضافة اعمدة جديدة إلى الجدول
MODIFY	تستخدم للتعديل في نوع البيانات للجدول
DROP	تستخدم لإلغاء عمود معين من الجدول

مثال (4) إضافة عمود اسمه REGION إلى جدول القسم DEPT20 :

```
SQL > ALTER   TABLE  DEPT2
2        ADD      (region   VARCHAR2(20))
```

Table altered .

مثال (5) التعديل في طول بيانات العمود DNAME ليصبح بطول (20) بدلاً من (14) :

```
SQL > ALTER   TABLE   DEPT2
  2       MODIFY     (dname   VARCHAR2(20))
```

Table altered .

بعض الاعتبارات عند التعديل في أعمدة الجداول :

- يمكن زيادة حجم البيانات للأعمدة.
- يمكن تغيير البيانات من نوع لآخر دون التأثير على البيانات الموجودة.
- لا يمكن إنقاص حجم الأعمدة إذا كانت تحتوي على بيانات.

مثال (6) إلغاء العمود REGION من جدول القسم DEPT2 :

```
SQL > ALTER   TABLE   dept2
  2       drop  COLUMN   REGION
```

Table altered.

بعض الاعتبارات عند حذف اعمدة من جدول :

- يجب ان يكون العمود المراد إلغاؤه فارغاً من البيانات.
- لا يمكن إلغاء أكثر من عمود في الأمر الواحد.
- يجب أن يبقى عمود واحد على الأقل في الجدول بعد عملية الإلغاء.
- لا يمكن استعادة العمود بعد إلغائه.

إلغاء جدول باستخدام الامر DROP :

وهي عملية إلغاء الجدول من قاعدة البيانات والبيانات ضمنه والقيود المتعلقة به.

مثال (7) إلغاء الجدول المسمى DEPT 30 :

```
SQL > DROP TABLE dept30
```

TABLE Dropped .

- تغيير اسم الجدول غلى اسم آخر استخدام الأمر RENAME

مثال (8) تغير اسم الجدول DEPT2 الى DEPARTMENT:

```
SQL > RENAME DEPT2 TO Department
```

Table RENAMED

أنواع الجداول في بيئة قواعد البيانات:

- جدول ينشئها المستخدمين USER TABLES ويتم التعامل معها من خلال المستخدم.
- جدول ينشئها مخدم الاوراكل ORACLE SERVER وهي مجموعة من الجداول تسمى DATA DICTIONARY تُنشأ من أوراكل وتحتوي على معلومات عن قاعدة البيانات وتقسم لفئات:
  - الفئة (USER_) وتحتوي معلومات عن كائنات المستخدمين كالجداول.
  - الفئة (ALL_) وتحتوي معلومات عن كل الجداول والعلاقات التي يمكن للمستخدمين الدخول إليها.
  - الفئة (DBA_) وتحتوي معلومات خاصة بمدير قاعدة البيانات DBA ولا يمكن لأحد الدخول إليها واستخدامها.

مثال (9) عرض معلومات الجداول التي يملكها المستخدمين باستخدام DATA DICTIONARY :

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

مثال (10) عرض اسم ونوع الكائنات التي يمتلكها المستخدمين :

```
SQL > SELECT   object_name  , object_type
  2         FROM      user_objects  ;
```

مثال (11) عرض أسماء ونوع الجداول والكائنات التي أنشاها المستخدم :

```
SQL > SELECT    *
  2         FROM      user_ catalog ;
```

الفصل الثالث عشر

القيود على الجداول

الأهداف :

عندما يكتمل هذا الفصل يكون لديك القدرة على :

١) فهم معنى القيود (Constraints) التي تطبق على الجداول.

٢) معرفة أنواع القيود Constraints التي يمكن تطبيقها على الجداول.

٣) معرفة كيفية تطبيق القيود على مستوى العامود.

٤) معرفة كيفية تطبيق القيود على مستوى الجداول.

٥) معرفة كيفية إضافة القيود على جداول تم إنشاؤها.

٦) معرفة كيفية إلغاء القيود من الجداول.

٧) عرض القيود المطبقة على الجداول والاعمدة.

القيود عبارة عن شروط معينة توضع على الجداول لتنظيم العمليات التي تطبق على الجداول مثل الإضافة والتعديل الحذف فمثلاً نريد إضافة رقم موظف إلى جدول الموظفين بقيمة (100) ولكن هذه القيمة موجودة ولا يصح لموظفين نفس الرقم فنضع قيد Constraint بحيث لا يتكرر رقم الموظف.

وفي هذا الفصل سنشرح القيود المختلفة وكيفية تطبيقها.

أنواع القيود Constraints الجدول التالي يبينها :

	القيد (Constraint)	معنى القيد (Description)
١	NOT NULL	• يمنع هذا القيد ترك عمود معين فارغ (لابد أن يدخل قيمة للعمود) • يطبق على مستوى العمود فقط
٢	UNIQUE	• يمنع هذا القيد تكرار القيم داخل العمود (القيم داخل العمود وحيدة) • يطبق على مستوى العمود أو الجدول .
٣	PRIMARY KEY	• يستخدم لعمل مفتاح أساسي داخل الجدول، والمفتاح الأساسي يتميز بعدم تكرار القيم، وعدم ترك القيم فارغة أي إنه عبارة عن القيدين السابقين. • يطبق على المستوى العمود أو الجدول
٤	FOREIGN KEY	• يستخدم لعمل مفتاح ربط بين جدولين. • يطبق على المستوى العمود أو الجدول
٥	CHECK	• يستخدم لاختبار قيمة عمود بحيث لا يقبل هذا العمود إلا قيم حسب شرط معين. • يطبق على المستوى العمود أو الجدول.

إنشاء القيود Create Constraint :

بطريقتين

- عمل القيود أثناء إنشاء الجداول.
- عمل القيود بعد إنشاء الجداول .

- القيد PRIMARY KEY :

ويتم إنشاؤه على مستوى الجدول أو العمود ومعناه المفتاح الأساسي داخل الجدول وذلك لتمييز عمود معين بحيث يكون له خاصيتان :

١- عدم قبول التكرار للقيم داخله.

٢- عدم السماح لقيمة NULL داخله.

مثال (1) إنشاء قيود على جدول الأقسام وتطبيقها على مستوى الأعمدة :

```
SQL > CREATE TABLE dept (
  2                deptno  NUMBER(2),   PRIMARY KEY ,
  3                dname   VARCHAR2(14), NOT NULL ,
  4                loc     VARCHAR2(13)
  5                );
```

القيود على مستوى الأعمدة

مثال (2) إنشاء القيود أثناء إنشاء جدول الأقسام وتطبيقها على مستوى الجدول :

```
SQL > CREATE TABLE dept (
  2                deptno  NUMBER(2) ,
  3                dname   VARCHAR2(14)  NOT NULL ,
  4                loc     VARCHAR2(13) ,
  5      CONSTRAINT dept_deptno_pk  PRIMARY KEY (deptno) ;
```

القيد على مستوى الجدول

والفرق بين هذه الطريقة والمثال السابق هو أننا نستطيع حذف القيد بسهولة إذا كان على مستوى الجداول كما نلاحظ في هذا المثال اسم القيد dept_deptno_pk.

- القيد UNIQUE KEY :

يتم إنشاؤه على مستوى العمود أو الجدول ومعناه عدم السماح بكرار القيم داخله.

مثال (3) إنشاؤه القيد unique أثناء إنشاء جدول الأقسام وتطبيقه على مستوى الجدول :

```
SQL > CREATE  TABLE dept (
2                    deptno   NUMBER(2) ,
3                    dname   VARCHAR2(14) ,
4                    loc       VARCHAR2(13) ,
5     CONSTRAINT  dept_deptno_uk  UNIQUE(dname) ;
```

القيد على مستوى الجدول

Table cerated

ويمكن تطبيق هذا القيد على المستوى العمود.

- القيد FOREIGN KEY :

يتم إنشاؤه على المستوى الجدول أو العمود ونستخدم هذا القيد عندما نريد ربط جدولين مع بعض فمثلاً لربط جدول الموظفين بجدول الأقسام لمعرفة موظفي قسم معين لا بد من وجود عمود PRIMARY KEY في الجدول الأقسام ونفس هذا العمود موجود في جدول الموظفين كعمود Foreign Key كما في الشكل التالي :

EMP table

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

DEPT table

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

Foreign Key Primary Key

ولتنفيذ هذا الربط لا بد من تطبيق قيد Foreign key على العمود deptno في الجدول الموظفين كما في المثاليين التاليين :

مثال (4) إنشاء القيد foreign key على العمود deptno في الجدول الموظفين وتطبيقه على المستوى العمود :

```
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(2)  REFERENCES  dept (deptno) ) ;
```

القيد على مستوى العمود

TABLE Creatrd

في هذا المثال تم إنشاء جدول الموظفين وإنشاء قيد FOREIGN KEY على العمود deptno فالكلمة REFERENCES تشير إلى تطبيق القيد FOREIGN KEY على العمود DEPTNO الذي يشير إلى العمود DEPTNO في الجدول DEPT وهكذا تم تطبيق القيد على المستوى العمود.

- أما تطبيق القيد على المستوى الجدول سيتضح في المثال التالي :

مثال (5) إنشاء القيد FOREIGN KEY على العمود DEPTNO في الجدول الموظفين وتطبيقه على المستوى الجدول :

```
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 (2) ,
10   CONSTRAINT  EMP deptno  fk FOREIGN KEY (deptno)
11          REFERENCES  dept (deptno)  )  ;
```

القيد على مستوى الجدول

TABLE Creatrd

- القيد CHECK :

يتم إنشاؤه على مستوى العمود او الجدول ويُستخدم عندما نريد تحديد مجال قيم معينة لعمود في الجدول فمثلاً إذا اردنا تحديد القيم المدخلة في العمود DEPTNO في جدول الأقسام لتكون محصورة بين 10 و 99 ونوضح ذلك في المثال التالي :

مثال (6) إنشاء القيد CHECK على العمود DEPTNO في الجدول الأقسام وتطبيقه على المستوى الجدول

```
SQL > CREATE TABLE dept (
2                    deptno NUMBER(2) ,
3                    dname VARCHAR2(14) ,
4                    loc VARCHAR2(13) ,
5       CONSTRAINT dept_deptno_ck CHECK(deptno BETWEEN 10
AND 99 ) ;
```

القيد على مستوى الجدول

Table Created

إضافة قيود إلى الجداول Adding Constraints :

يتم إضافة قيود للجدول بعد إنشائه باستخدام الأمر Alter Table كما في الصيغة التالية :

```
SQL > ALTER TABLE أسم الجدول
2       ADD CONSTRAINT أسم القيد نوع القيد
```

مثال (7) إضافة القيد FOREIGN KEY على العمود MGR في جدول الموظفين :

```
SQL > ALTER TABLE emp
2       ADD CONSTRAINT emp mgr fk
3       foreign key (mgr) REFERENCES emp (empno) ;
```

Table altered.

مثال (8) إضافة القيد Primary Key إلى العمود deptno في جدول الأقسام :

```
SQL > ALTER TABLE dept
  2       ADD CONSTRAINT dept_deptno_pk PRIMARY KEY (deptno) ;
```

Table altered.

إزالة القيود من الجدول DROP :

وذلك باستخدام ALTER TABLE

مثال (9) إزالة القيد المسمى (emp_mgr_fk) من جدول الموظفين :

```
SQL > ALTER TABLE emp
  2       DROP CONSTRAINT emp_mgr_fk
```

Table altered.

كما نعرف أن جدول الأقسام له علاقة بجدول الموظفين وذلك بوجود مفتاح أساسي Primary key على العمود deptno في جدول الأقسام ومفتاح foreign key على العمود dept في جدول الموظفين وعندما نريد إزالة المفتاح الأساسي في جدول الأقسام والعلاقة مع جدول الموظفين أي إزالة foreign key من جدول الموظفين الوطفين نقوم بما يلي :

مثال (10) إزالة العلاقة بين جدولي الموظفين والإدارات عن طريق إزالة المفتاح الأساسي وتوابعه

```
SQL > ALTER TABLE DEPT
  2       DEOP Primary key CASCADE
```

Table Altered.

- **استعراض القيود المطبقة على جدول معين :**

كما نعلم أن أوراكل ينشئ جداول لتسجيل التغيرات التي تتم في قواعد البيانات وتسمى هذه الجداول DATA DICTIONARY ومن خلالها يمكن عرض القيود المطبقة على جدول معين

مثال (11) عرض القيود المطبقة على جدول الموظفين

```
SQL > SELECT Constraint name , Constraint _typy
  2   FROM user_constraints
  3   WHERE TABLE_NAME ='EMP'
```

CONSTRAINT_NAME	C
SYS_C00674	C
SYS_C00675	C
EMP_EMPNO_PK	P
FK_DEPTNO	R

فيما يلي معنى المحارف في الجدول

- الحرف **C** يعني ان نوع القيد هو **CHECK** .
- الحرف **P** يعني أن نوع القيد هو **Primary key** .
- الحرف **R** يعني أن وع القيد هو **Foreign key**.
- الحرف **U** يعني أن نوع اقيد هو **UNIQUE** .

أما نوع القيد NULL NOT فيظهر مثل القيد CHECK .

والمثال التالي يبين كيفية عرض القيود المطبقة على الأعمدة

مثال(12) عرض أسماء الأعمدة وأسماء القيود المطبقة عليها :

```
SQL > SELECT CONSTRAINT_NAME , COLUMN_NAME
  2         FROM   user_cons_column
  3         WHERE    Table_name  = 'EMP'   ;
```

الفصل الرابع عشر

القسم العملي

الاهداف :

١. تثبيت برنامج SQL SERVER 2008.
٢. إنشاء قاعدة بيانات باستخدام برنامج SQL Server Management Studio.
٣. إنشاء جداول في قاعدة البيانات.
٤. إنشاء العلاقات بين جدولين.
٥. حفظ قاعدة البيانات وإغلاق برنامج SQL Server Management Studio.
٦. تعديل قاعدة البيانات في برنامج SQL Server Management Studio.
٧. حذف قاعدة البيانات في برنامج SQL Server Management Studio.
٨. تعبئة معطيات في قاعدة البيانات.
٩. لغة الاستعلامات البنيوية SQL.
١٠. الاستعلامات (Queries).
١١. إنشاء استعلام باستخدام SQL Server Management Studio.
١٢. حفظ وفتح الاستعلام.

➢ تثبيت برنامج SQL Server 2008 :

المتطلبات اللازم توافرها في الحاسوب لتثبيت وتشغيل برنامج SQL Server 2008 عليه :

١. ذاكرة بحجم 1 GB على الأقل، ويفضل ان تكون الذاكرة بسعة 4 GB ليعمل برنامج SQL Server بأداء جيد، ويجب زيادة سعة الذاكرة المتاحة بزيادة سعة قواعد البيانات.

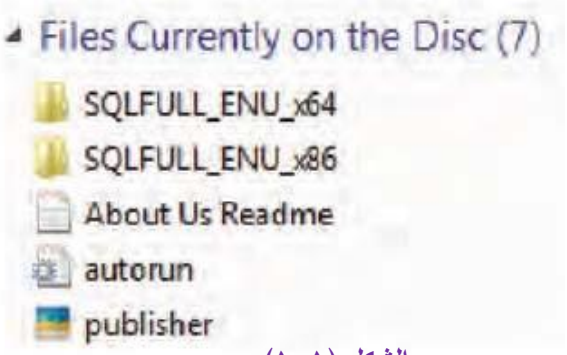
٢. سرعة وحدة المعالجة المركزية (CPU) على الاقل 1.0 GHz في حال كان المعالج x86، و 1.4 GHz في حال كان المعالج x64 ، ويفضل أن تكون سرعة المعالج مهما كان نوعه على الأقل 2 GHz ليعمل البرنامج SQL Server 2008 بأداء جيد.

➢ خطوات تثبيت برنامج SQL Server 2008 :

يجب تثبيت البرنامج SQL Server 2008 قبل تثبيت Visual Studio 2010 وإذا تم تثبيت برنامج Microsoft Visual Studio 2010 سابقاً فيجب ازالته وتثبيت البرنامج SQL Server 2008 ثم إعادة تثبيت البرنامج Microsoft Visual Studio 2010 مرة أخرى .

والخطوات هي :

١- يُختار أحد المجلدين : SQLFULL_ENU_x64 او SQLFULL_ENU_x86، وذلك حسب نظام التشغيل المثبت في الحاسوب (32 بت أو 64 بت)، كما في الشكل (١-١) الآتي :



الشكل (١-١)

٢- يُشغل الملف SETUP.exe

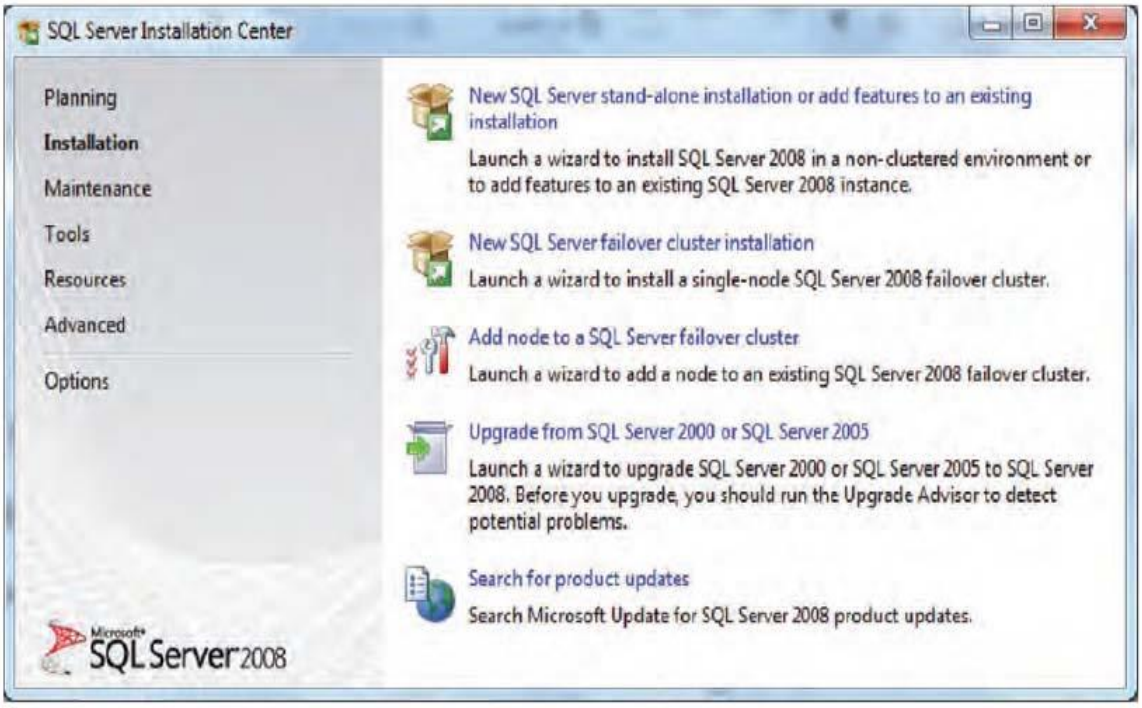
٣- تظهر النافذة كما في الشكل (١-٢) يُختار منها الأمر Installation من القائمة اليسارية.

الشكل (١-٢)

٤- تظهر النافذة كما في الشكل (١-٣) يختار منها الخيار New SQL Server stand-alone installation or add features to an existing installation وذلك في حال تثبيت برنامج SQL Server جديد، أو إضافة بعض الميزات الجديدة إلى نسخة SQL Server التي تم تثبيتها سابقاً.

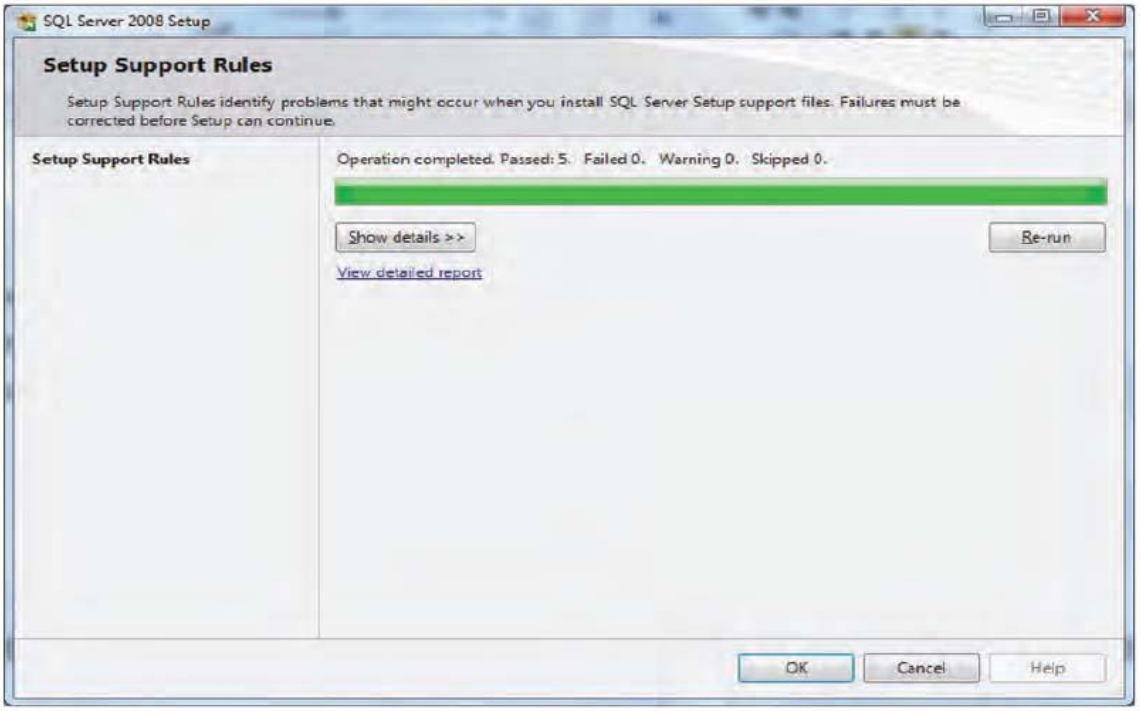
أو يختار الخيار Upgrade from SQL Server 2000 or SQL Server 2005

وذلك في حال كان المكلوب تحديث نسخة سابقة SQL SERVER .



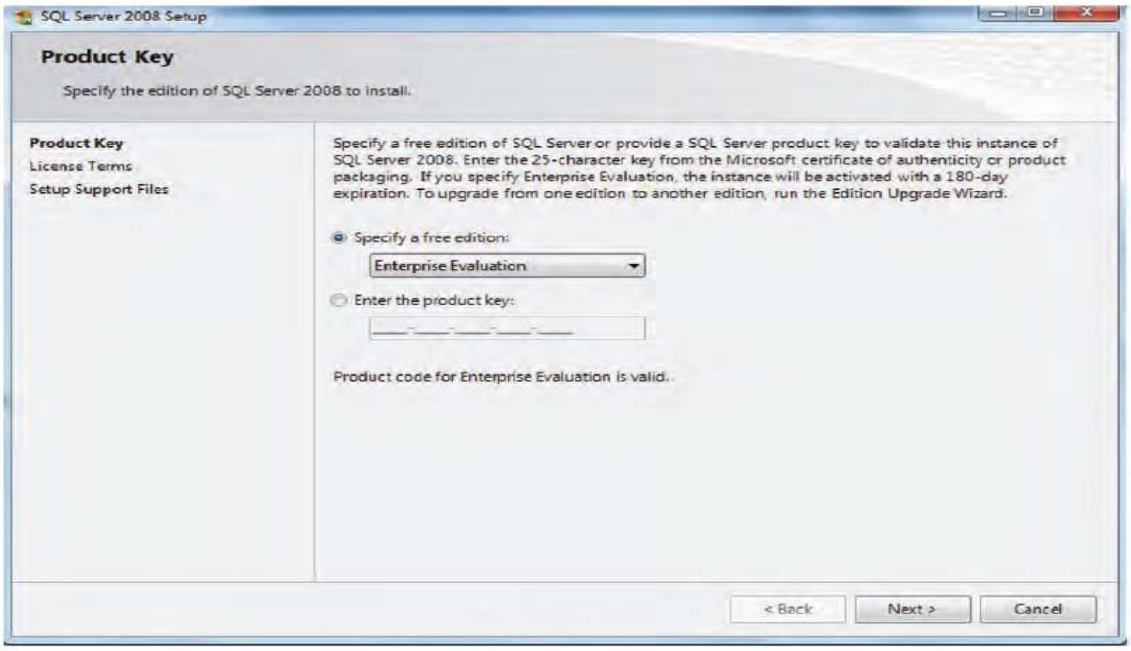
الشكل (١-٣)

٥- في حال اختيار New SQL Server stand-alone installation or add features to an existing installation كما في الشكل (١-٤) يُنقر الزر OK .



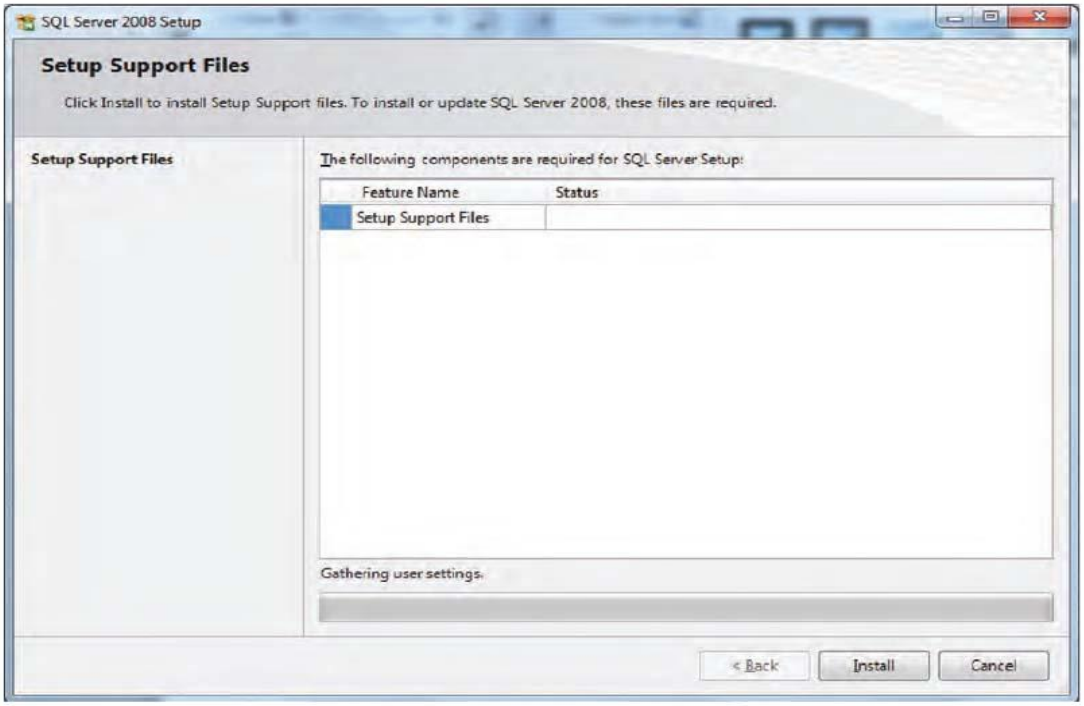
الشكل (١-٤)

٦- تظهر النافذة كما في الشكل (١-٥)، فإذا لم توافر المنتج للبرنامج يُختار Spccify a free edition ، اما إذا توافر مفتاح المنتج يُختار الخيار Enter the product key ، ويُدخل مفتاح المنتج الخاص بالتثبيت، ثم ينقر الزر Next في كلتا الحالتين.



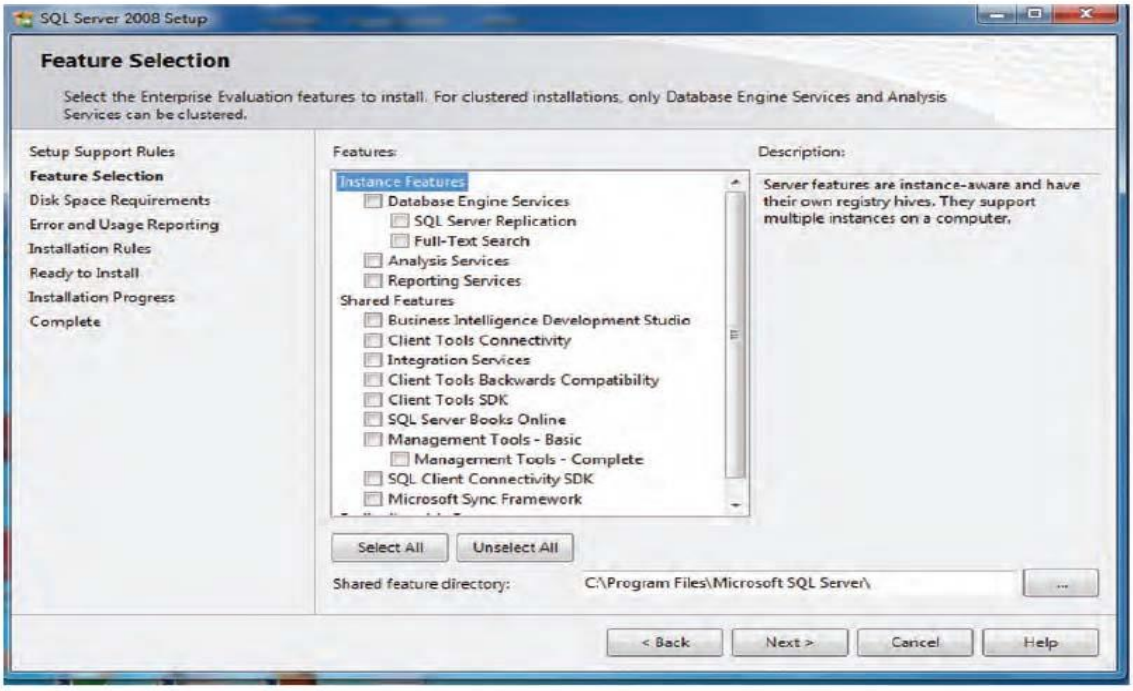
الشكل (١-٥)

٧- تظهر النافذة كما في الشكل (١-٦) فينقر الزر Install.



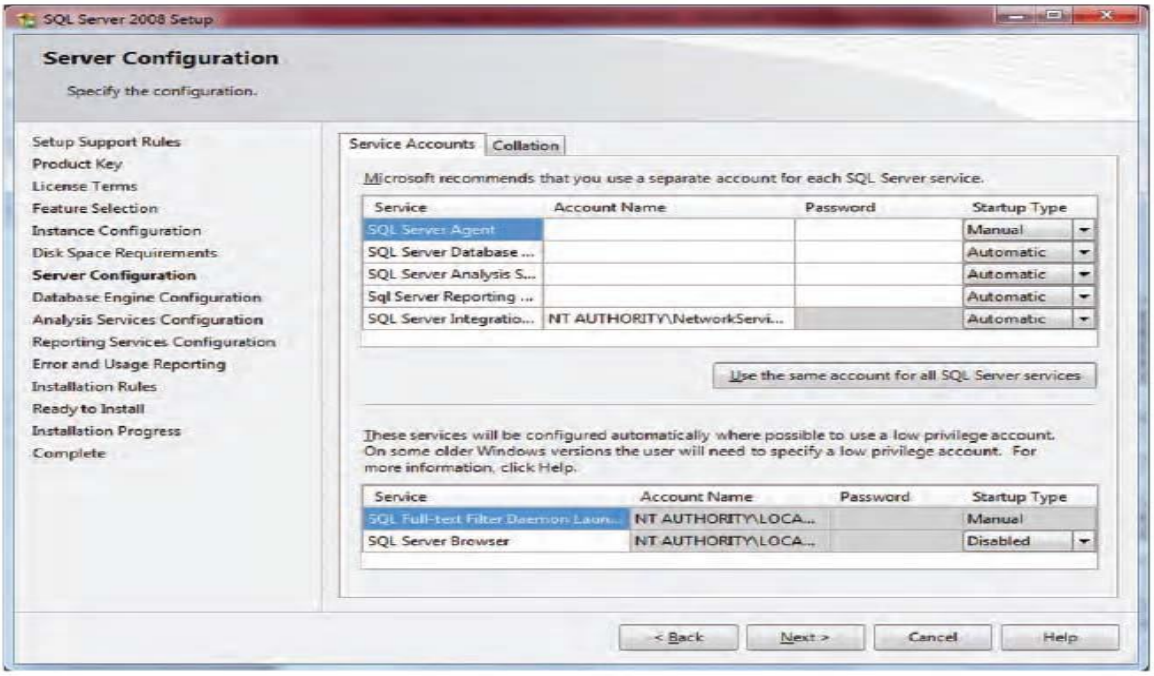
الشكل (١-٦)

٨- تظهر النافذة كما في الشكل (١-٧) ينقر الزر Select ALL لتثبيت البرنامج كاملاً وبميزاته كافة، ثم ينقر الزر Next .



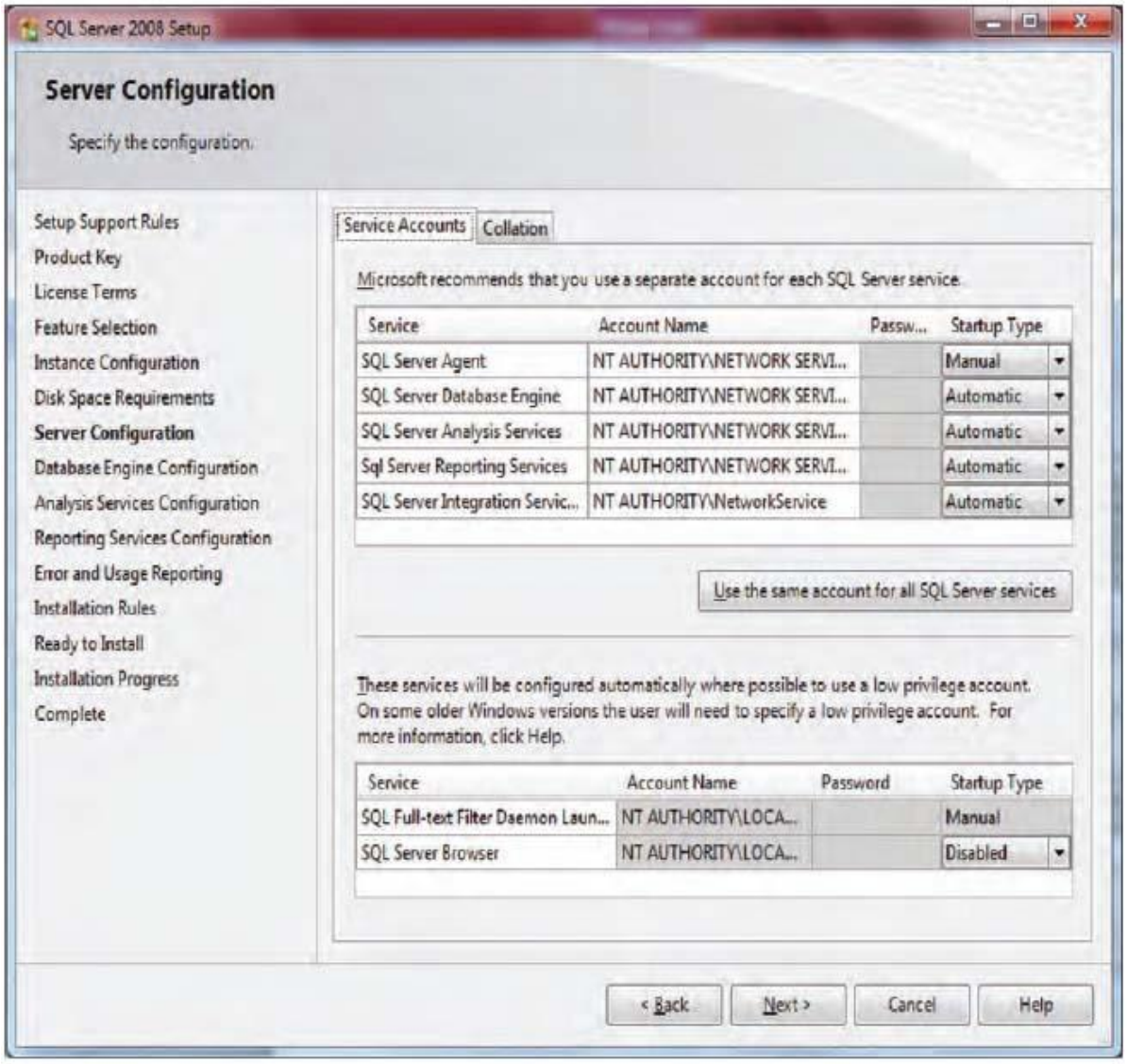
الشكل (١-٧)

٩- تظهر النافذة كما في الشكل (١-٨) :



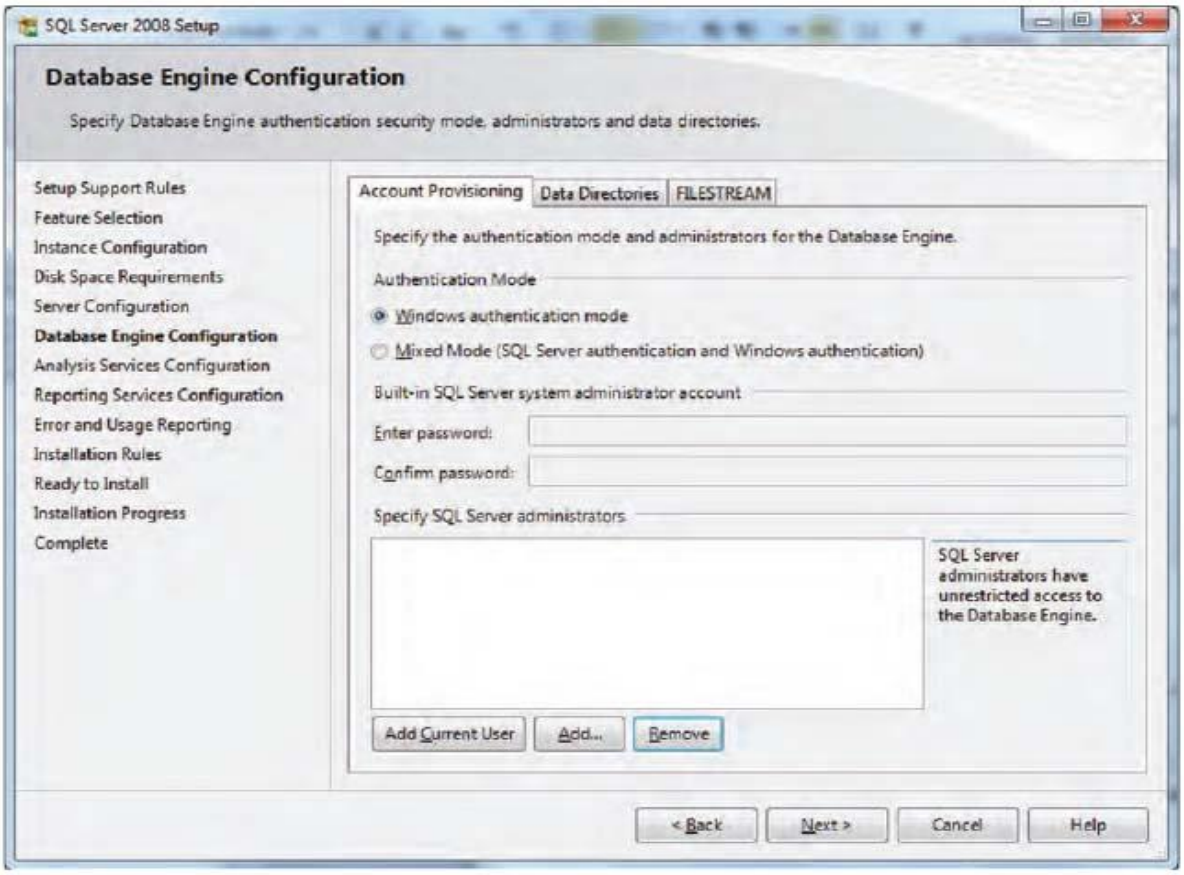
الشكل (١-٨)

١٠- يجب اختيار حسابات المستخدمين لبرنامج SQL SERVER من العمود Account Nameوتغير لجميع حسابات SQL لتصبح قيمها جميعها NT AUTHORITY\NETWORK SERVER حيث يتم اختيار هذا الخيار من القائمة المنسدلة في الحقل Account Name كل حساب من حسابات SQL، لتصبح النافذة (١-٨) كما في الشكل (١-٩) ثم ينقر الزر Next .



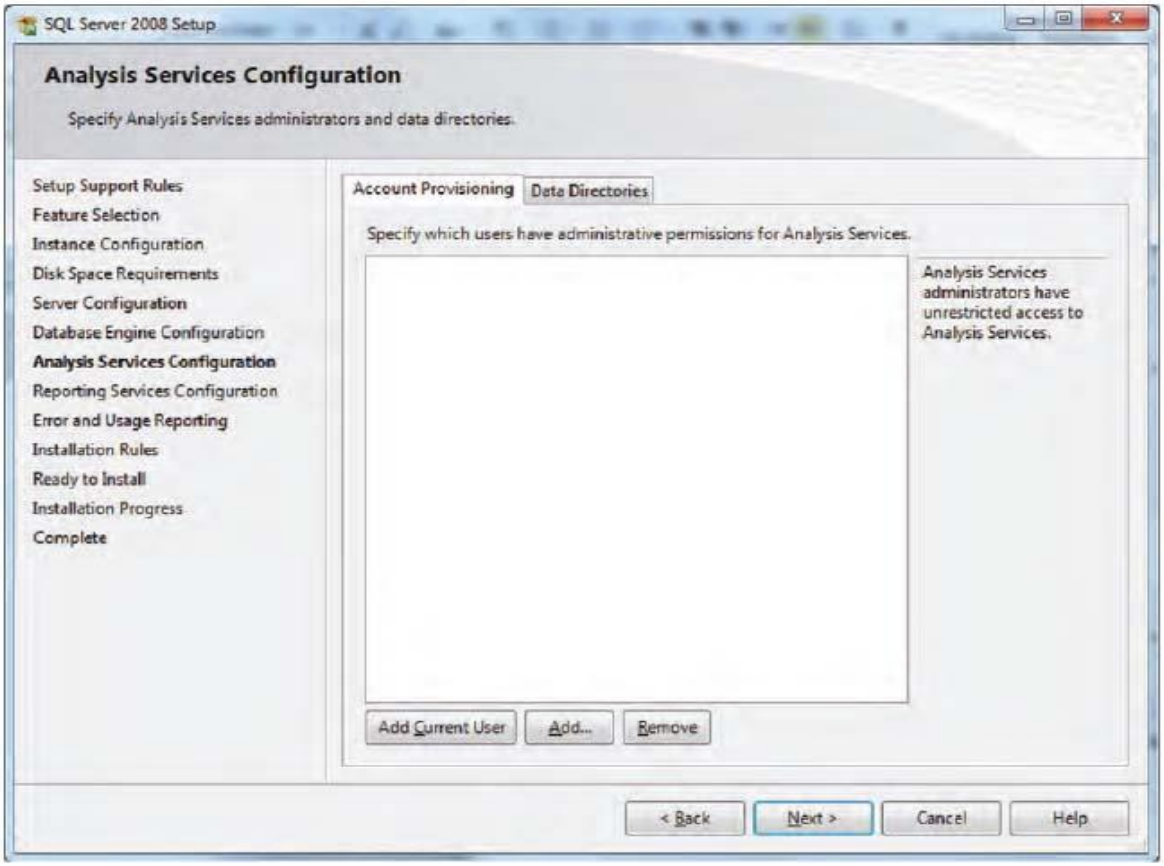
الشكل (١-٩)

١١- تظهر النافذة كما في الشكل(١-١٠) ينقر الزر Add Current User لإضافة المستخدم الحالي للنظام، ليتم اعتباره مديراً لممحرك قاعدة البيانات في برنامج SQL Server 2008 حيث يسجل باسمه عند الدخول إلى البرنامج، ثم ينقر الزر Next.



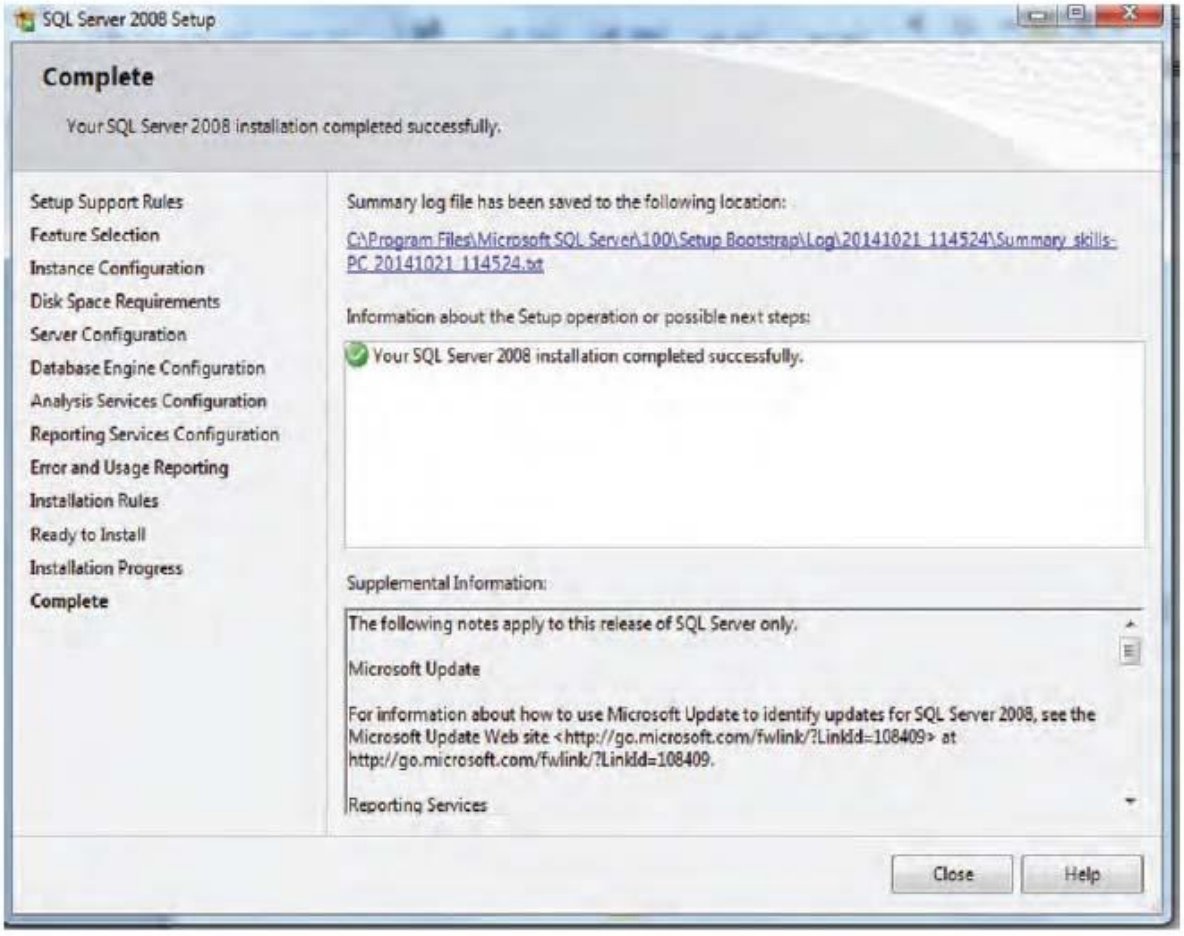
الشكل (١-١٠)

١٢- تظهر النافذة كما في الشكل (١-١١)، ينقر الزر Add Current User أيضاً، ثم ينقر الزر Next.



الشكل (١-١١)

١٣- تظهر النافذة كما في الشكل (١-١٢) ينقر الزر Close لإنهاء تثبيت البرنامج .


SQL Server 2008 Setup
Complete
Your SQL Server 2008 installation completed successfully.
Setup Support Rules
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Analysis Services Configuration
Reporting Services Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete
Summary log file has been saved to the following location:
C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log\20141021_114524\Summary_skills-PC_20141021_114524.txt
Information about the Setup operation or possible next steps:
Your SQL Server 2008 installation completed successfully.
Supplemental Information:
The following notes apply to this release of SQL Server only.
Microsoft Update
For information about how to use Microsoft Update to identify updates for SQL Server 2008, see the Microsoft Update Web site <http://go.microsoft.com/fwlink/?LinkId=108409> at http://go.microsoft.com/fwlink/?LinkId=108409.
Reporting Services
Close
Help

الشكل (١-١٢)

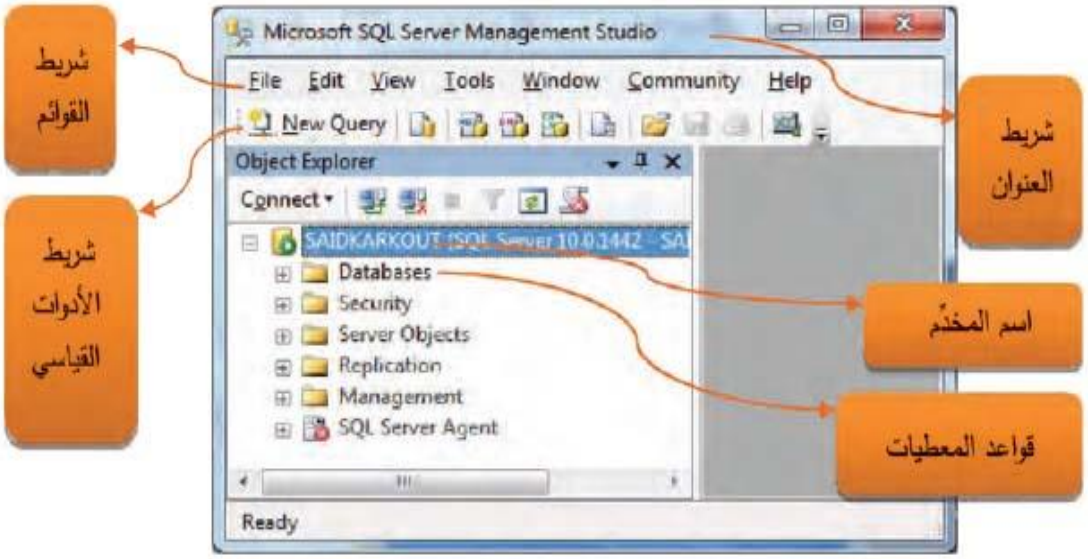
إنشاء قاعدة بيانات باستخدام برنامج SQL Server Management Studio :

١- من قائمة أبدأ – كافة البرامج – Microsoft SQL SERVER 2008 – SQL SERVER Management Studio .

٢- يفتح البرنامج، ويظهر مربع الحوار كما في الشكل (٢-١)، ينقر الزر Connect للاتصال مع مخدم قاعدة البيانات.

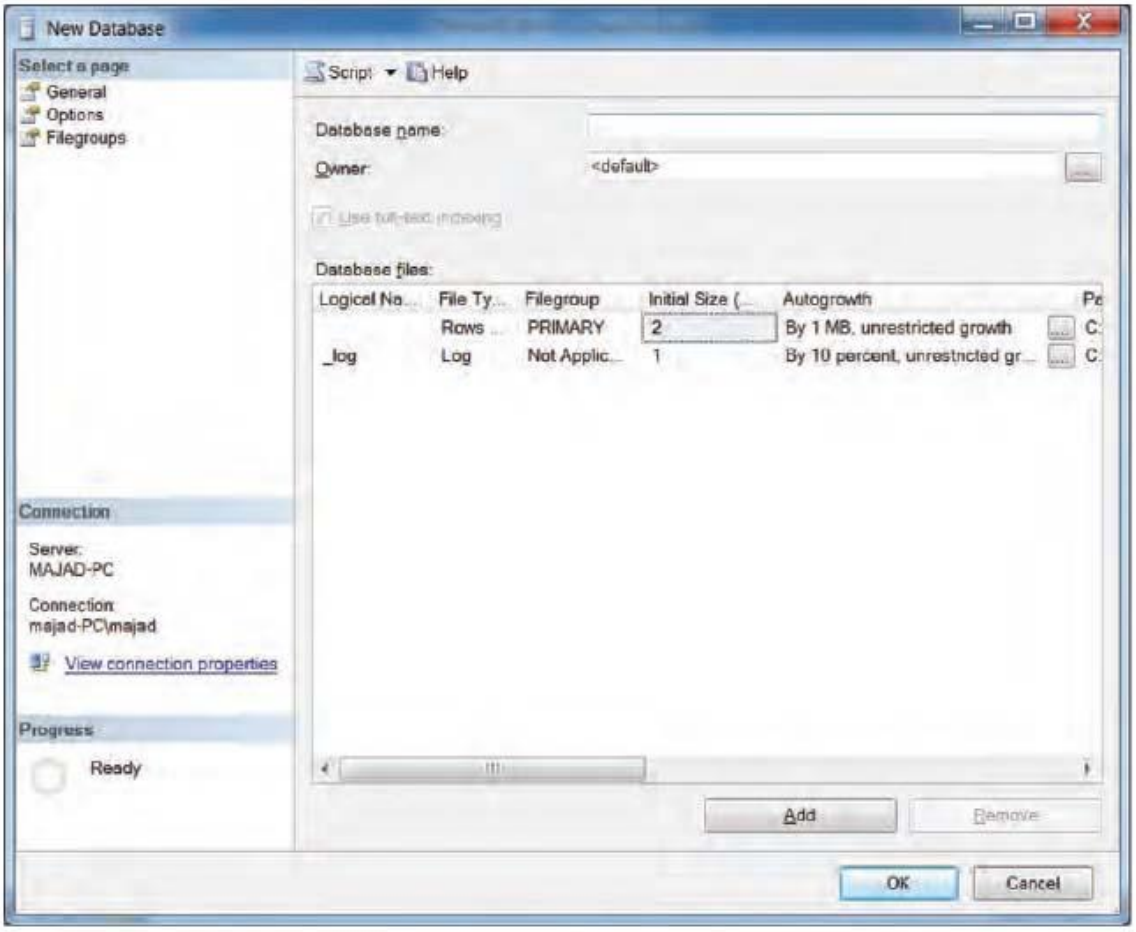
الشكل (٢-١)

٣- تظهر النافذة الرئيسية لبرنامج SQL Management Studio كما في الشكل (٢-٢) :



الشكل (٢-٢)

٤- ينقر بالزر الأيمن للفأرة على المجلد Databases، ثم ينقر الأمر New Database، فيظهر مربع الحوار New Database كما في الشكل (٢-٣):

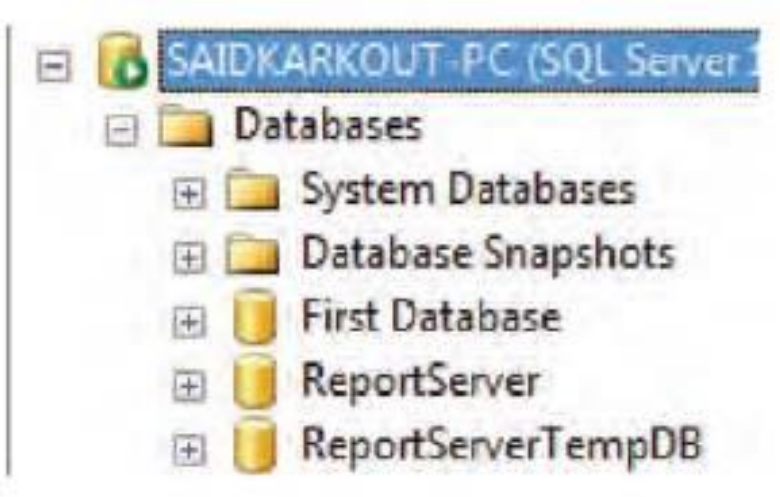


الشكل (٢-٣)

٥- يُدخل اسم قاعدة البيانات ضمن الحقل Database Name كما في الشكل (٢-٣) مثلاً الاسم First Database، ثم ينقر الزر OK فتنشأ قاعدة البيانات بالاسم First Database .

إنشاء جداول في قاعدة البيانات :

١- تنقر الإشارة (+) بجوار المجلد Database فيلاحظ ظهور قاعدة البيانات First Database ضمن مجلد كما في الشكل (٣-١)، إضافة إلى قواعد معطيات جاهزة مبنية ضمن مخدم قواعد البيانات المتصل به.



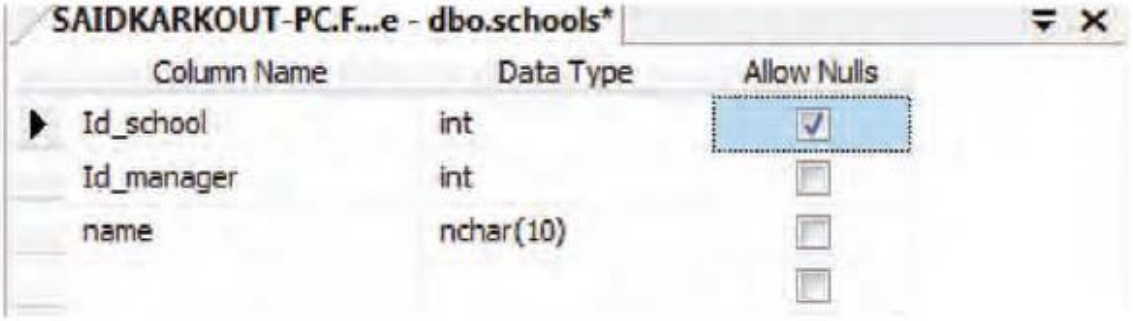
الشكل (٣-١)

٢- تنقر الاشارة (+) بجوار قاعدة البيانات First Database فتظهر القائمة المبنية بالشكل (٣-٢):

الشكل (٣-٢)

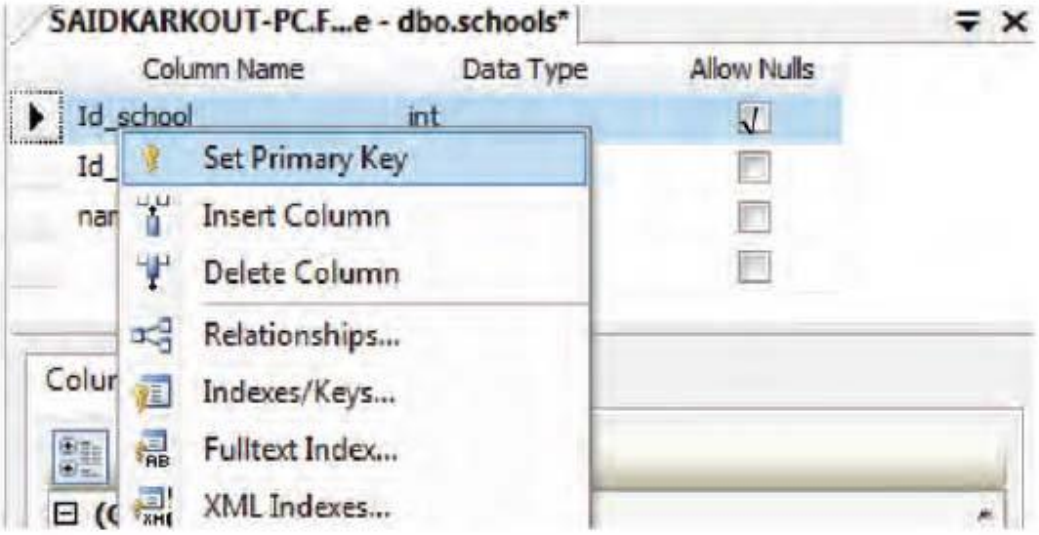
٣- ينقر بالزر الأيمن للفارة على المجلد Tables، ويختار منه الامر New Table لإنشاء جدول جديد ضمن قاعدة البيانات.

٤- تظهر النافذة المبنية في الشكل (٣-٣)، ويدخل اسم كل حقل من الحقول الجدول، ونمط كل حقل، وتحديد إمكانية أن تكون قيمته فارغة ام لا .



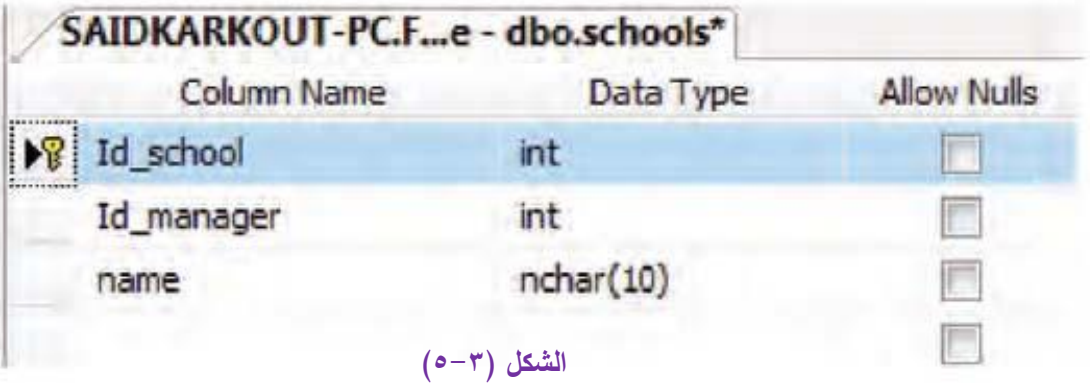
الشكل (٣-٣)

٥- بعد كتابة اسم الحقل ونمط معطياته، ولتحديد أي حقل من الجدول على انه المفتاح الرئيسي ينقر بالزر الأيمن للفأرة على السهم بجوار اسم الحقل، ويختار Set Primary Key كما في الشكل (٣-٤) التالي :



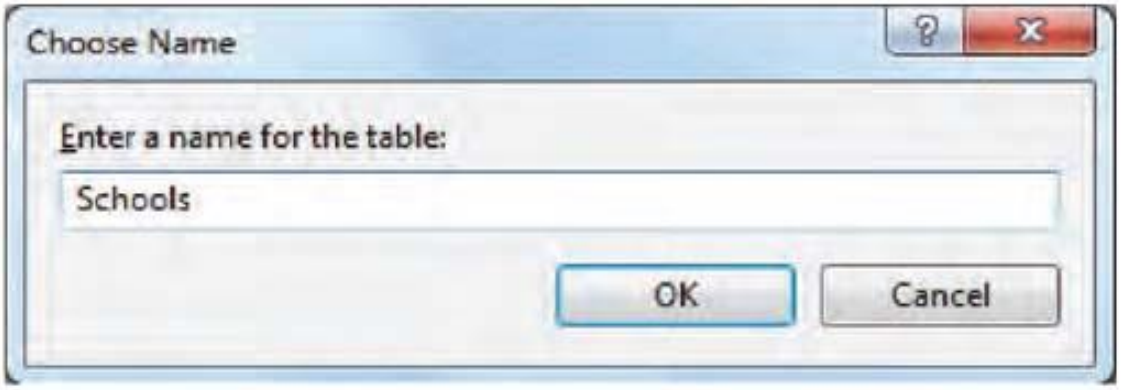
الشكل (٣-٤)

تتم تلقائياً إزالة الاختيار من Allow Nulls، لأن المفتاح الرئيسي لا يمكن أن تكون قيمته فارغة، ويلاحظ ظهور إشارة مفتاح بجوار اسم الحقل كما في الشكل (٣-٥)



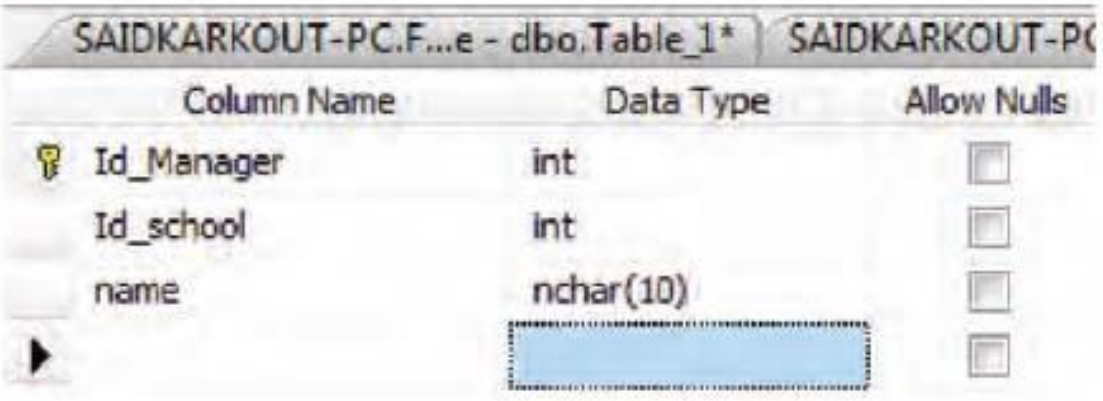
الشكل (٣-٥)

٦- يحفظ الجدول بعد إدخال أسماء وانماط حقوله بضغط المفتاحين Ctrl +S، أو من القائمة File يختار الأمر Save Table_1 فيظهر مربع الحوار لإدخال اسم الجدول المطلوب حفظه، مثلاً Schools كما في الشكل (٣-٦)



الشكل (٣-٦)

٧- تنشأ كل جدول قاعدة البيانات بالطريقة نفسها، مثلاً جدول بأسماء المدراء للمدرسة ويسمى managers، وتحديد حقوله كما في الشكل (٣-٧) الآتي :

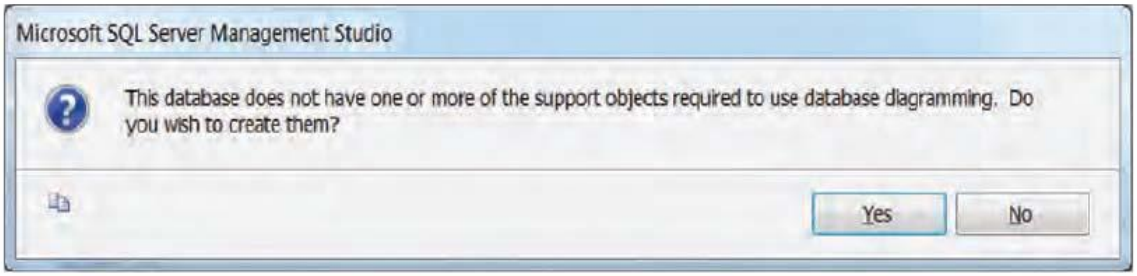


الشكل (٣-٧)

إنشاء العلاقات بين جدولين :

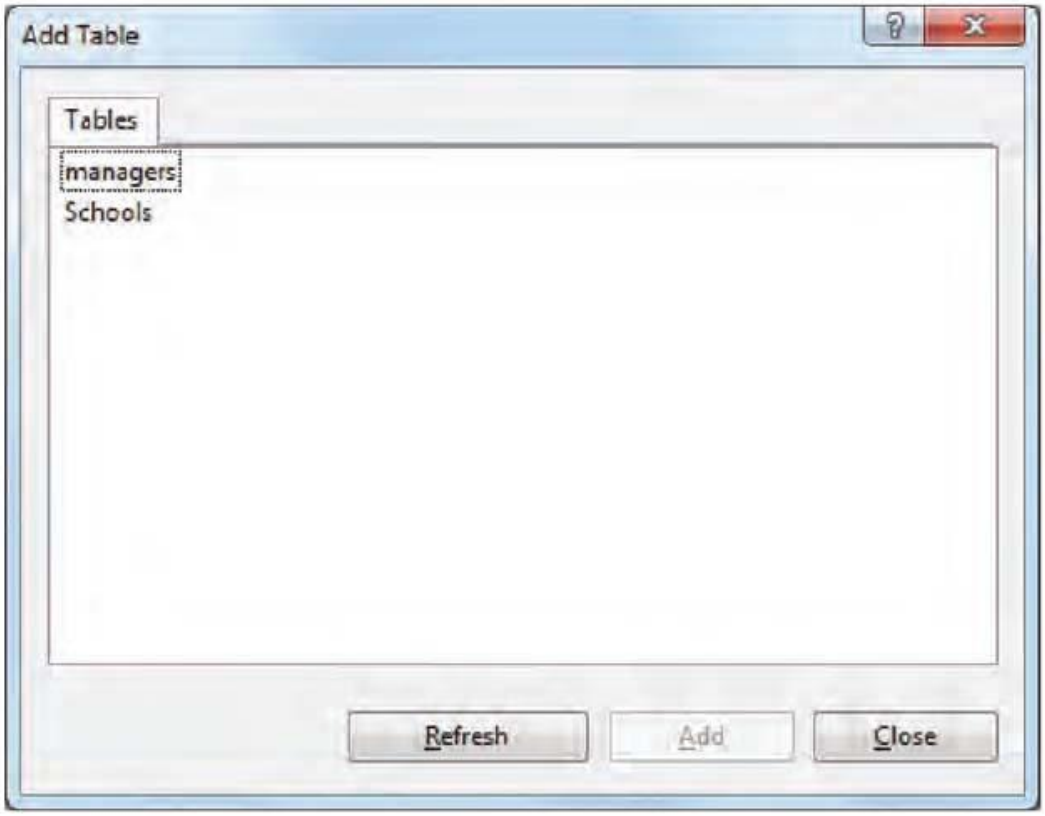
١- لتحديد العلاقات بين جدولين تنقر على الاشارة (+) بجوار مجلد البيانات المُنشأة FIRST Database، ثم ينقر بالزر الأيمن للفأرة على المجلد Database ويختار الخيار New Database Diagram .

٢- تظهر الرسالة المبنية في الشكل(٤-١)، وذلك فقط عند إنشاء أول نخطط في قاعدة البيانات، فينقر الزر "Yes" لإنشاء المكونات اللازمة لإنشاء مخطط لقاعدة البيانات.



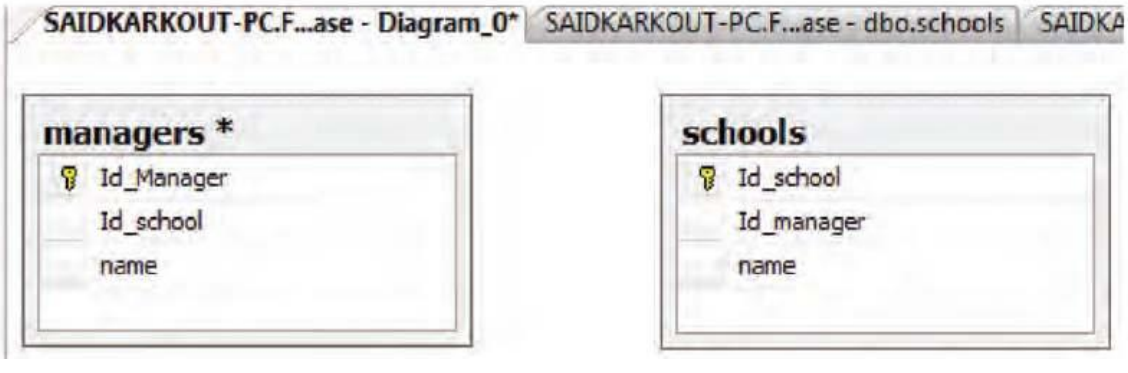
الشكل (٤-١)

٣- تظهر النافذة المبنية في الشكل (٤-٢) لتحديد الجداول المطلوب إضافتها إلى مخطط قاعدة البيانات.



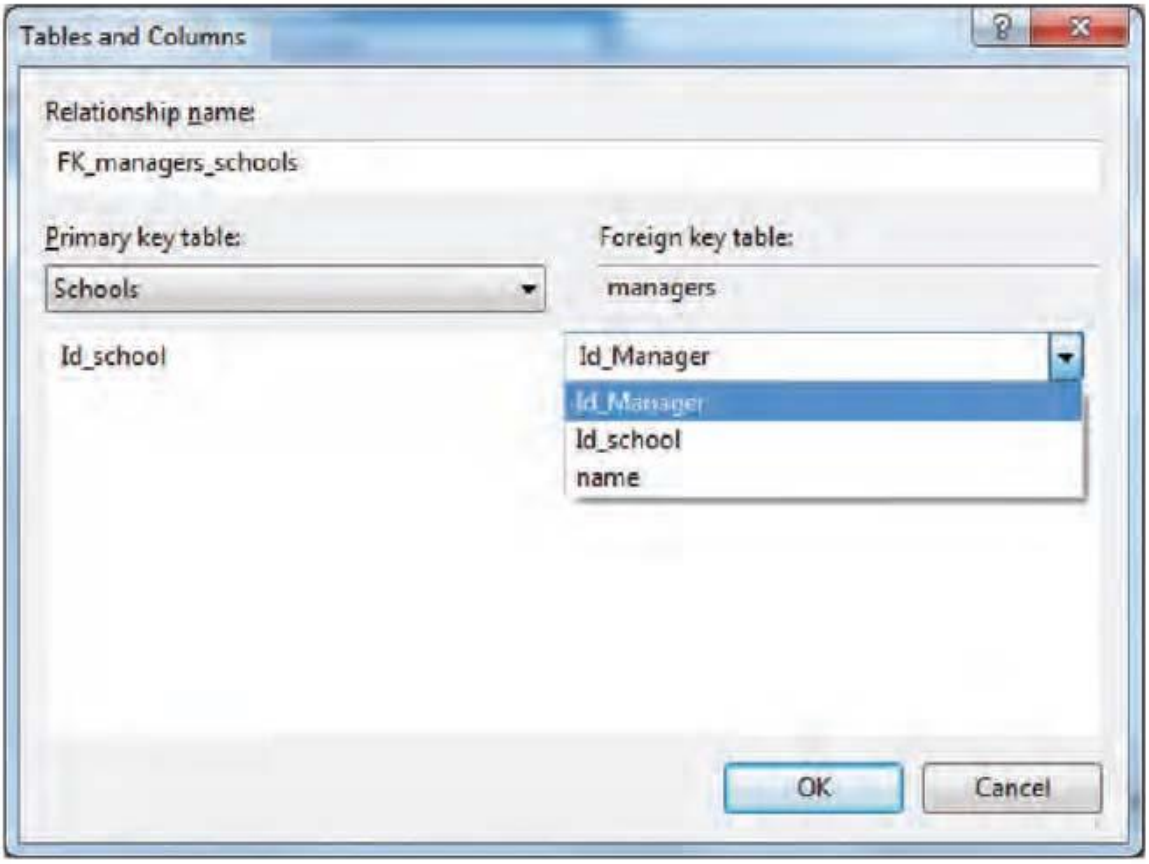
الشكل (٤-٢)

ولإضافة أي جدول يُحدد، ثم ينقر الزر Add، وفي النهاية الزر Close لإغلاق النافذة، فتظهر الجداول المضافة باستخدام الزر Add كافة على نافذة المخطط كما في الشكل (٤-٣).



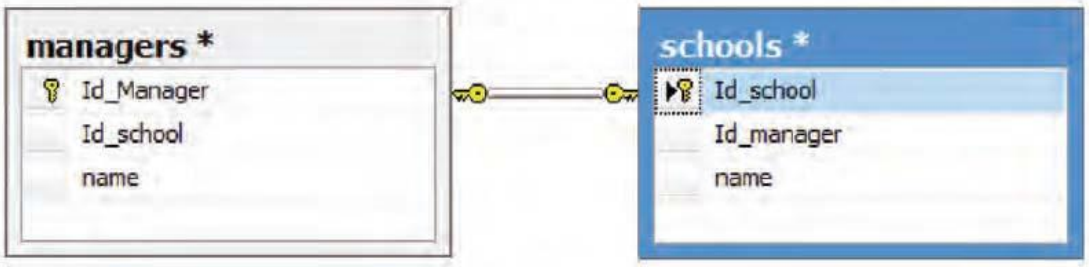
الشكل (٤-٣)

٤- لإنشاء علاقة بين الجدولين السابقين يوضع مؤشر الفأرة على المفتاح الرئيسي في الجدول Schools، ويسحب حتى الوصول إلى الجدول Managers ، فتظهر نافذة لتحديد نوع العلاقة كما في الشكل(٤-٤):



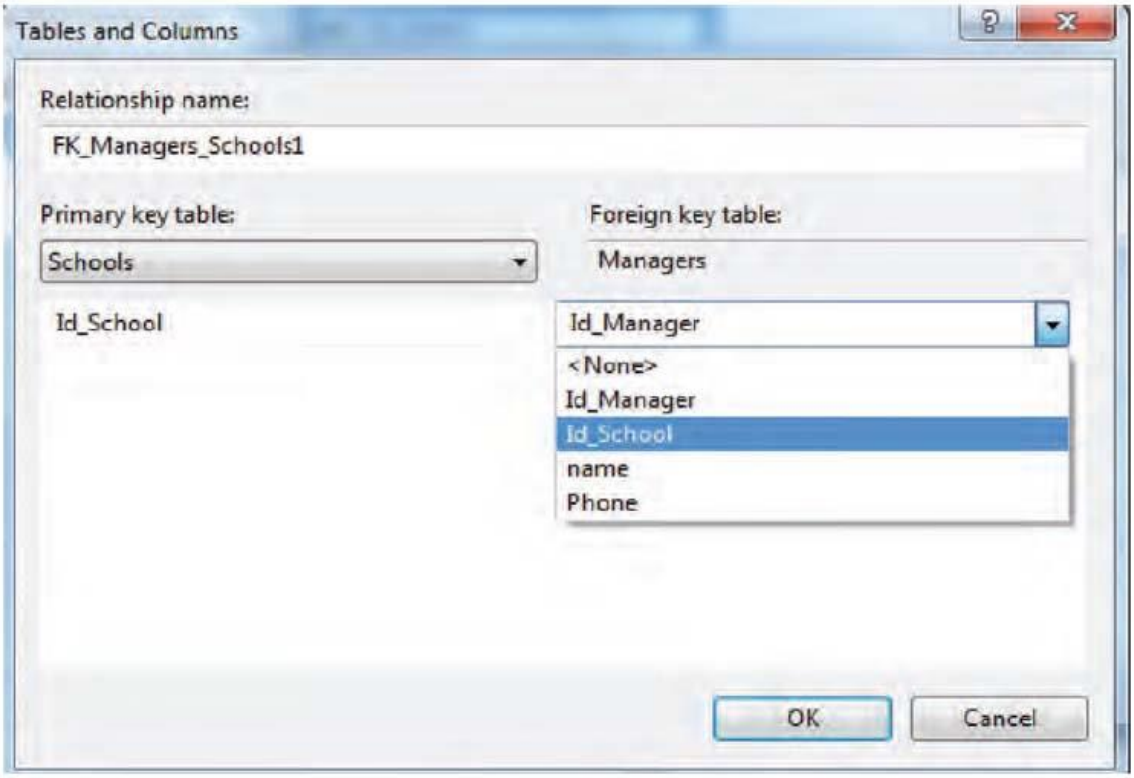
الشكل (٤-٤)

٥- يحدد نوع العلاقة من القائمة المنسدلة في الشكل السابق، بتحديد اسم المفتاح الأجنبي في الجدول Managers وهو Id_Managers ، ثم ينقر الزر OK فيلاحظ إنشاء علاقة One To One بين الجدولين بسبب ربط المفتاحين الرئيسين من الجدولين، كما في الشكل (٤-٥)



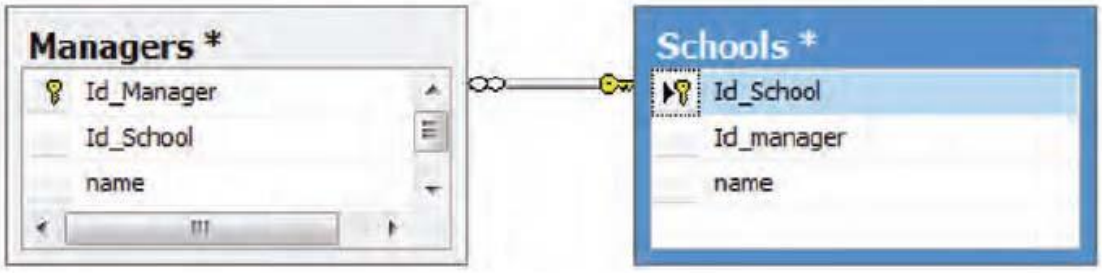
الشكل (٤-٥)

تدل إشارة المفتاح في الشكل السابق على واحد (One) وبالتالي العلاقة (One To One). ولإنشاء علاقة (One To One) يسحب الحقل Id_School من الجدول School ويربط مع الجدول Managers، وذلك باختيار الحقل Id_School كمفتاح اجنبي في الجدول Managers، كما في الشكل(٤-٦)



الشكل (٤-٦)

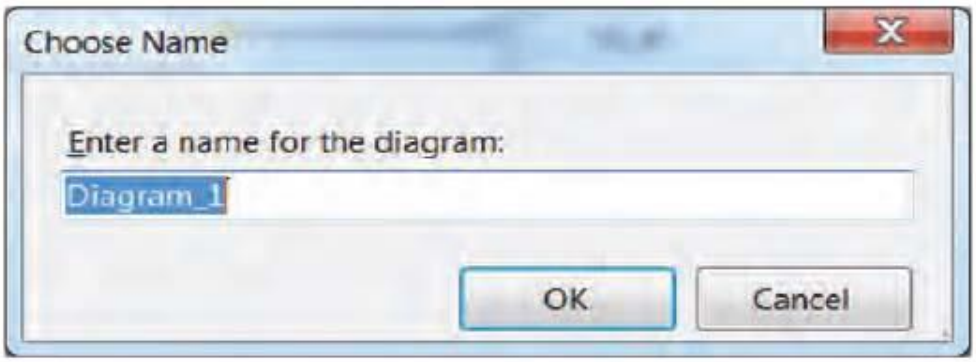
ينقر زر OK فيظهر المخطط المبين في الشكل (٤-٧) :



الشكل (٤-٧)

تدل اشارة (∞) في الشكل السابق على كثير (Many) والعلاقة هي (One To One) .

٦- يحفظ المخطط المنشأ بضغط المفتاحين Ctrl + S، أو من القائمة File يختار لأمر Save Diagram_1 فتظهر النافذة المبنية في الشكل (٤-٨)



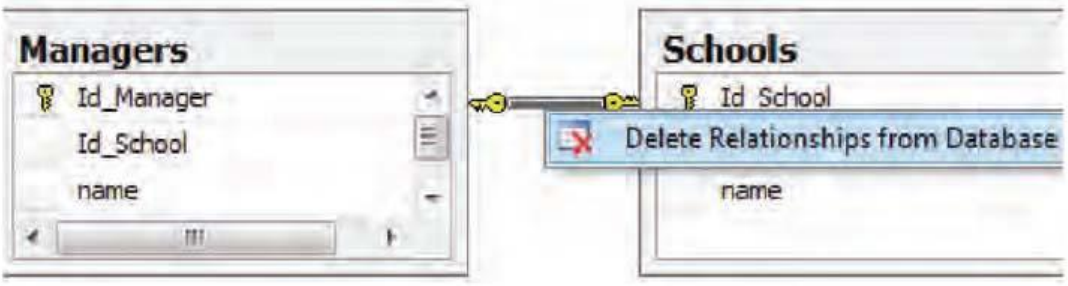
الشكل (٤-٨)

يدخل اسم للمخطط المطلوب حفظه ثم ينقر الزر OK.

٧- إظهار المخطط الذي تم إنشاؤه وحفظه، تنقر على الإشارة (+) بجوار المجلد Database Diagrams

٨- لفتح المخطط Database Diagram ينقر عليه نقراً مزدوجاً.

٩- لحذف العلاقة بين الجدولين ينقر بالزر الأيمن للفأرة على العلاقة ثم ينقر الأمر Delete Relationships from Database ، كما في الشكل (٤-٩) الآتي:



الشكل (٤-٩)

حفظ قاعدة البيانات وإغلاق برنامج SQL Server Management Studio :

تحفظ قاعدة البيانات كاملة بضغط المفتاح Ctrl + Shift+S، أو من القائمة File ينقر الأمر Save All، فيحفظ الملفان الخاصان بقاعدة البيانات، في المثال السابق مثلاً حفظ الملفان : First Database_log.ldf و First Database.mdf حيث :

- First Database.mdf : يمثل النسخة الاصلية لقاعدة البيانات التي تم حفظه، وله الامتداد mdf .
- First Database_log.ldf : يمثل النسخة الاحتياطية لقاعدة البيانات التي تم حفظها، وله الامتداد ldf.

في حال حدوث أي عطب بقاعدة البيانات الأصلية تسترجع من قاعدة البيانات الاحتياطية.

يحفظ الملفان السابقان في المسار :

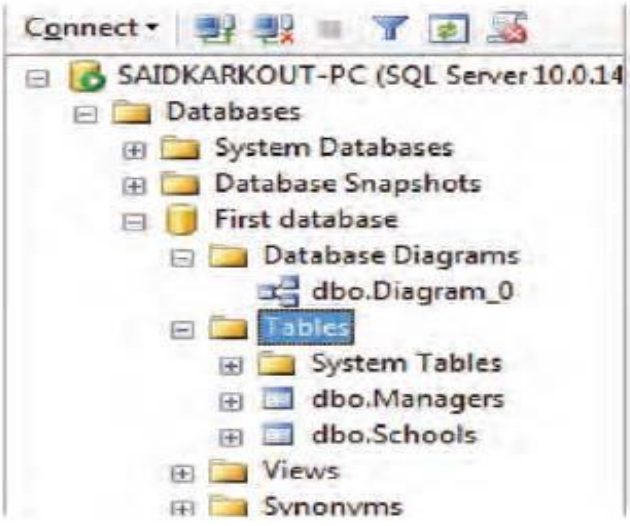
C:\program files\ Microsoft SQL server\ MSSQL10.MSSQLServer\ MSSQ\DATA

لإغلاق برنامج SQL Server Management Studio ينقر زر الإغلاق في شريط العنوان، أو من قائمة File ينقر Exit .

تعديل قاعدة البيانات في برنامج SQL Server Management Studio :

١- لتعديل أي جدول من جداول قاعدة البيانات تنقر على الإشارة (+) بجوار اسم قاعدة البيانات المطلوب تعديل جداولها، فمثلاً لتعديل جداول قاعدة البيانات First Database المُنشأة سابقاً.

٢- تنقر على الإشارة (+) بجوار المجلد Table، فتظهر جداول قاعدة البيانات التي تم إنشاؤها، كما في الشكل(٥-١):

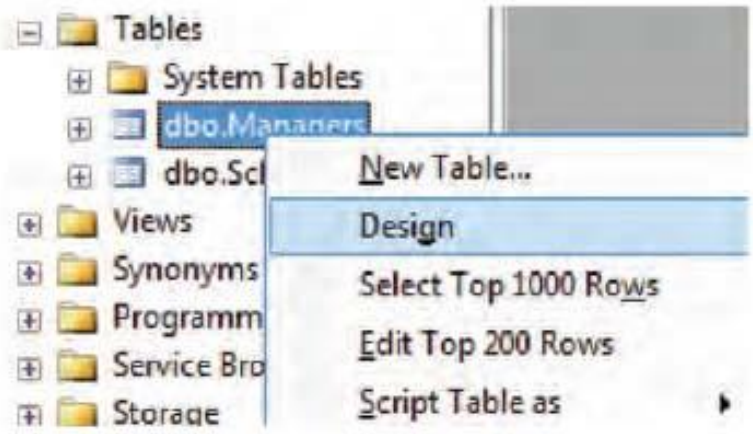


الشكل (٥-١)

٣- لتعديل اسم الجدول ينقر بالزر الأيمن للفأرة عليه، ويختار الخيار Rename ويدخل الاسم المناسب له، ثم يضغط المفتاح Enter .

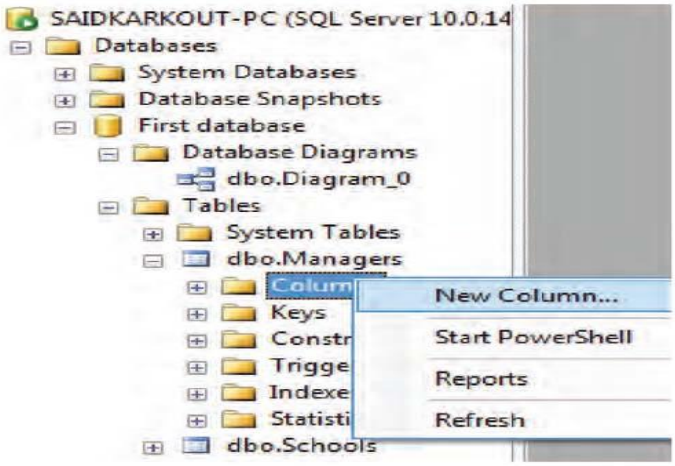
٤- لتعديل محتوى الجدول تتبع إحدى الطريقتين :

أ. ينقر بالزر الأيمن للفارة على اسن الجدول، ويختار الخيار Design كما في الشكل فيفتح تصميم الجدول وتجرى التعديلات او الإضافات اللازمة عليه.



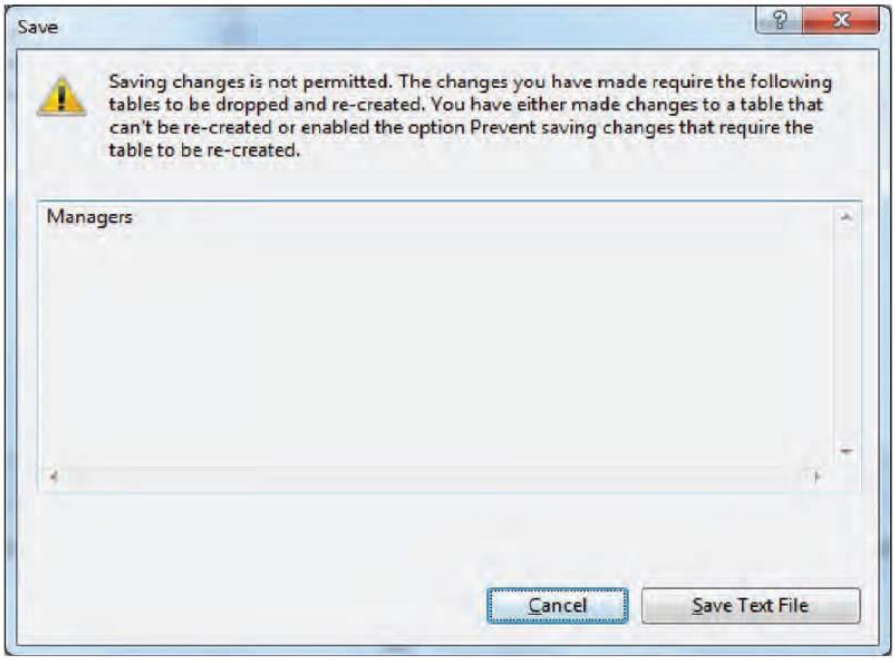
ب. تنقر على الإشارة (+) بجوار اسم الجدول المطلوب تعديله، فتظهر قائمة المجلدات لإجراء التعديلات اللازمة عليه(كإضافة حقول جديدة إلى الجداول او تغيير انماط حقول تم إنشاؤها مسبقاً او تغيير الحقول التي تم اعتبارها مفاتيح، أو حذف حقل من الجدول).

مثال : لإضافة أعمدة إلى الجدول Managers، ينقر بالزر الفأرة على المجلد Columns، ويختار الأمر New Column، كما في الشكل (٥-٢):



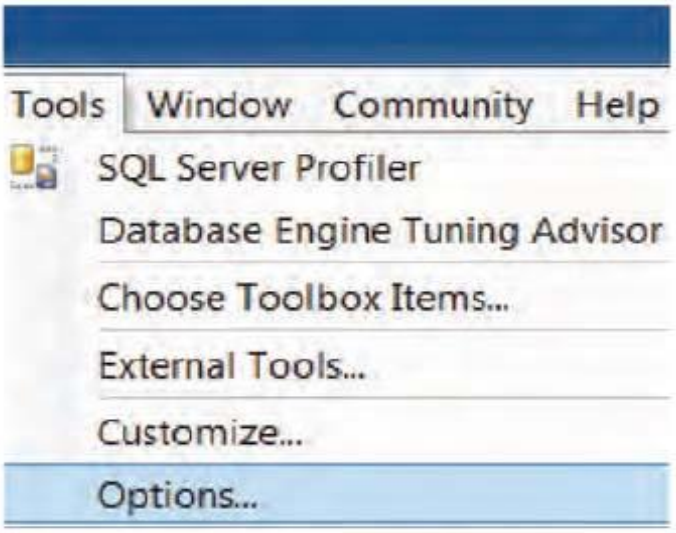
الشكل (٥-٢)

٥- بعد الانتهاء من إجراء التعديلات على الجدول يحفظ بإحدى الطرائق السابقة، فنلاحظ ظهور الرسالة المبنية في الشكل(٥-٣)، تفيد بأنه لا يمكن حفظ التعديلات على الجدول.



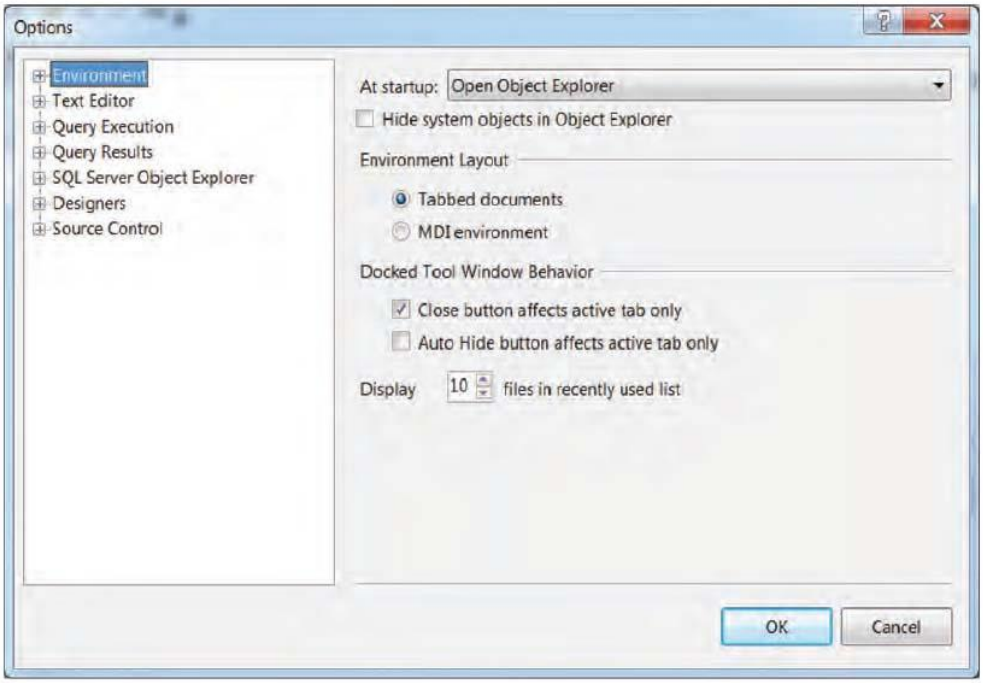
الشكل (٥-٣)

٦- ينقر على زر Cancel في النافذة السابقة لإغلاقها، ولحفظ التعديلات على الجدول تنقر على القائمة Tools في شريط القوائم العلوي، ثم يختار منها الخيار Options كما في الشكل (٥-٤) :



الشكل (٥-٤)

٧- تظهر النافذة المبنية في الشكل (٥-٥):

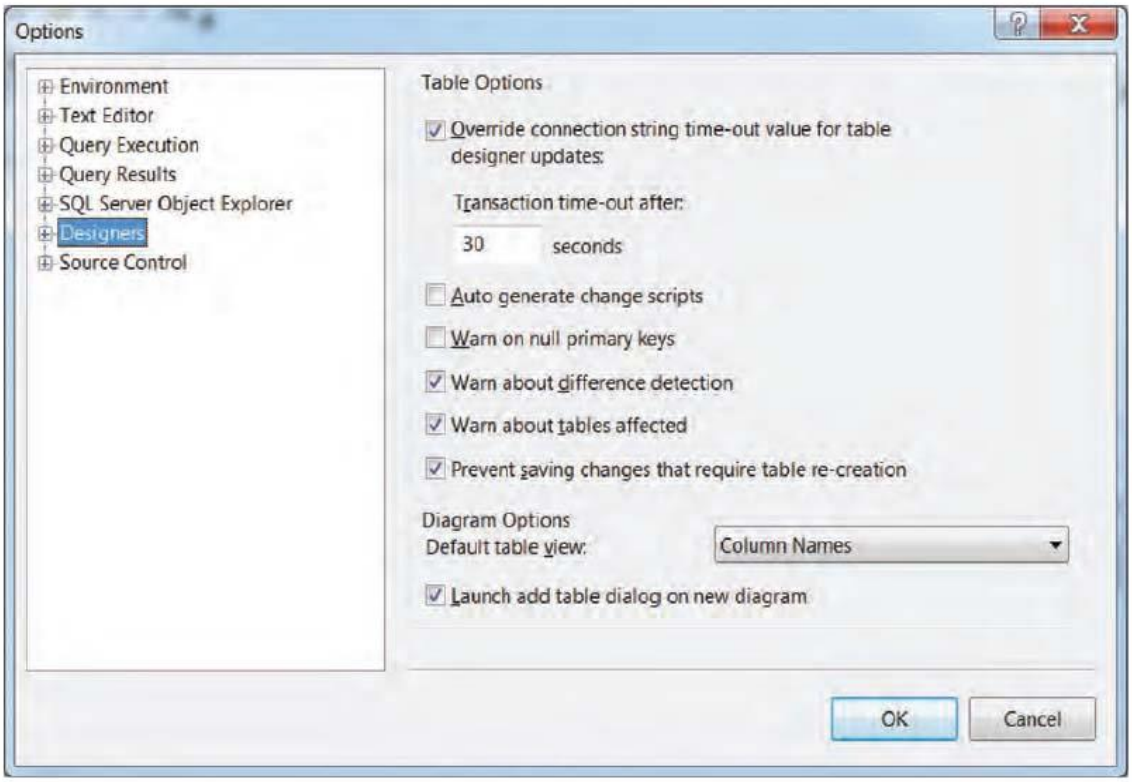


الشكل (٥-٥)

ينقر Designers، في القائمة اليسارية من النافذة السابقة، فتظهر النافذة المبنية في الشكل (٥-٦):

٨- يلغى تفعيل الخيار Prevent saving changes that require table re-creation، بإزالة الإشارة (√)من مربع الاختيار، للسماح بحفظ التعديلات على الجدول، ثم OK لإغلاق النافذة.

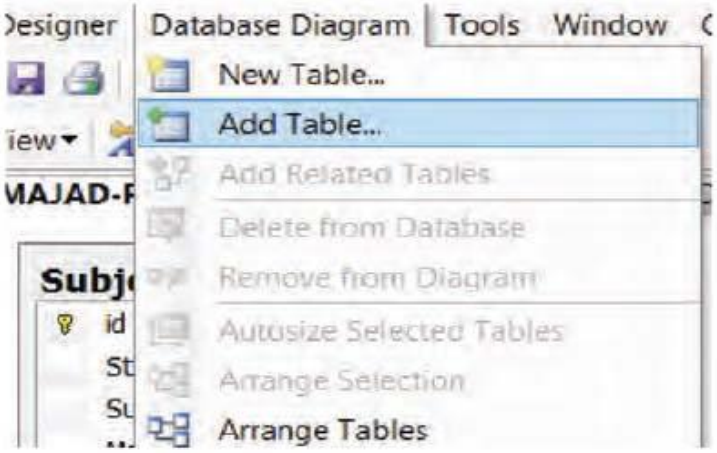
٩- يحفظ الجدول الذي أجريت التعديلات عليه، بإحدى طرائق الحفظ التي سبق ذكرها لحفظ الجدول.



الشكل (٥-٦)

١٠- لتعديل أي مخطط في قاعدة البيانات تنقر على الإشارة (+) بجوار المجلد Database Diagrams، ثم ينقر نقراً مزدوجاً على اسم المخطط لفتحه وإجراء التعديلات اللازمة عليه، فمثلاً يمكن إضافة جدول جديد مبني في قاعدة البيانات إلى المخطط باتباع إحدى الطريقتين:

أ. النقر على القائمة Database Diagrams، من شريط القوائم، ثم ينقر الامر Add Table، كما هو موضح في الشكل (٥-٧) الآتي:



الشكل (٥-٧)

ب.النقر على الأداة Add Table في شريط الأدوات  .

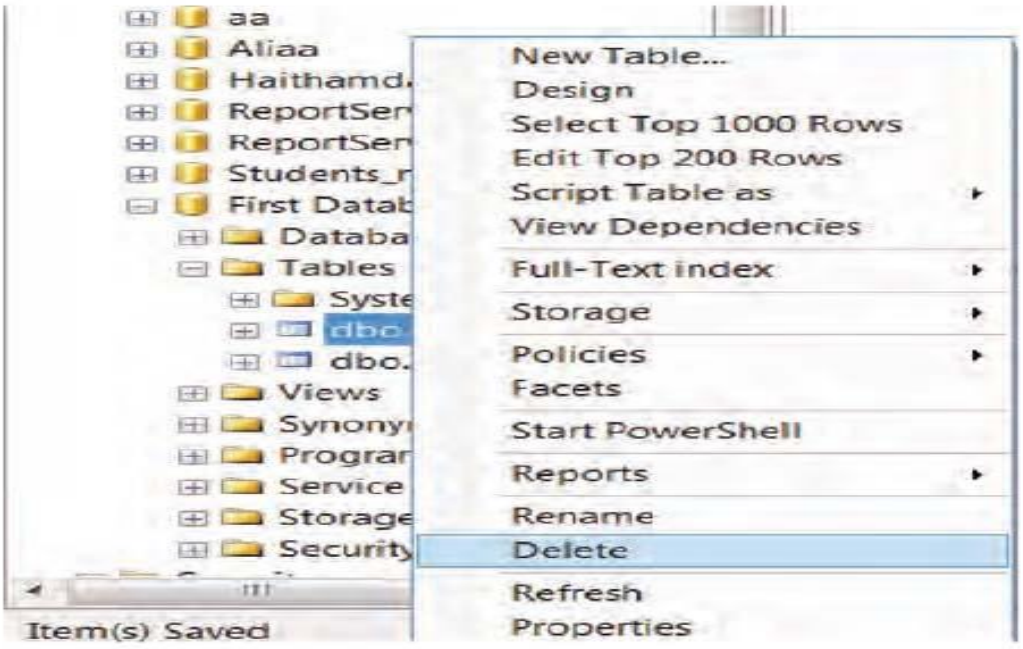
وفي كلتا الحالتين يظهر مربع حوار لإضافة الجدول المطلوب كما في الطريقة المذكورة سابقاُ لإضافة الجداول إلى المخطط، ثم يحفظ بإحدى الطرائق المذكورة سابقاً.

١١- لتعديل اسم المخطط ينقر بالزر الفأرة الأيمن عليه، ويختار الخيار Rename، ويدخل الاسم المناسب، ثم يضغط المفتاح Enter .

١٢- لتعديل اسم قاعدة البيانات بنقر بالزر الأيمن للفارة عليها، ويختار الخيار Rename ويدخل الاسم الجديد لها، ثم يضغط المفتاح Enter .

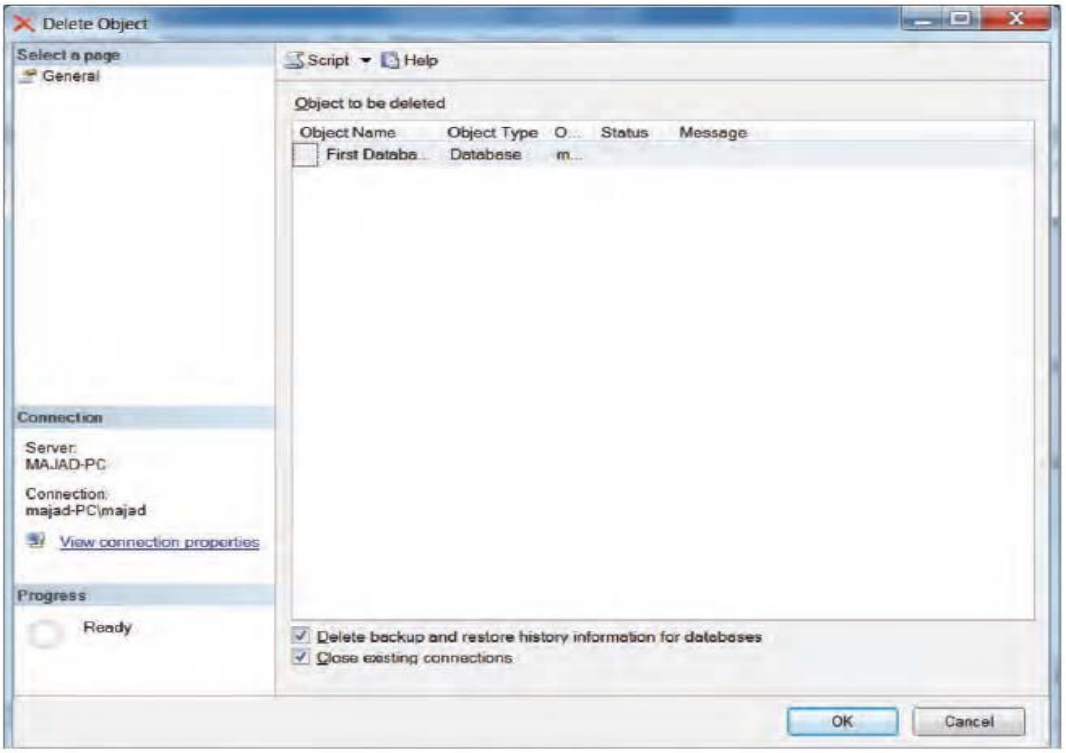
➢ حذف قاعدة البيانات في برنامج SQL Server Management Studio :

١. لحذف أي جدول من جداول قاعدة البيانات ينقر بالزر الأيمن على اسم الجدول المطلوب حذفه ويختار الخيار Delete كما في الشكل (٦-١)، فتظهر نافذة لتأكيد عملية الحذف فينقر الزرOK لحذف الجدول.



الشكل (٦-١)

٢. لحذف قاعدة البيانات بأكملها ينقر بالزر الأيمن للفأرة على اسم قاعدة البيانات، ويختار الخيار Delete، فتظهر النافذة المبنية في الشكل (٦-٢) :



الشكل (٦-٢)

ينقر الزر OK لتأكيد عملية الحذف للنسختين الأصلية والاحتياطية لقاعدة البيانات، وينتظر ريثما تتم عملية الحذف.

٣. لحذف أي مخطط من مخططات قاعدة البيانات ينقر بالزر الأيمن للفأرة على المخطط المطلوب حذفه، وينقر الأمر Delete فتظهر رسالة لتأكيد عملية الحذف، ينقر الزر OK لحذف المخطط.

➢ تعبئة معطيات في قاعدة البيانات :

يمكن تعبئة وإدخال البيانات في قاعدة البيانات بإحدى الطريقين:

١. إدخال البيانات مباشرة من خلال البرنامج الذي تم تصميم قاعدة البيانات فيه كبرنامج SQL Server Management Studio .

٢. عن طريق الربط مع برنامج صُمم بإحدى لغات البرمجة المتقدمة كلغة V.C# .

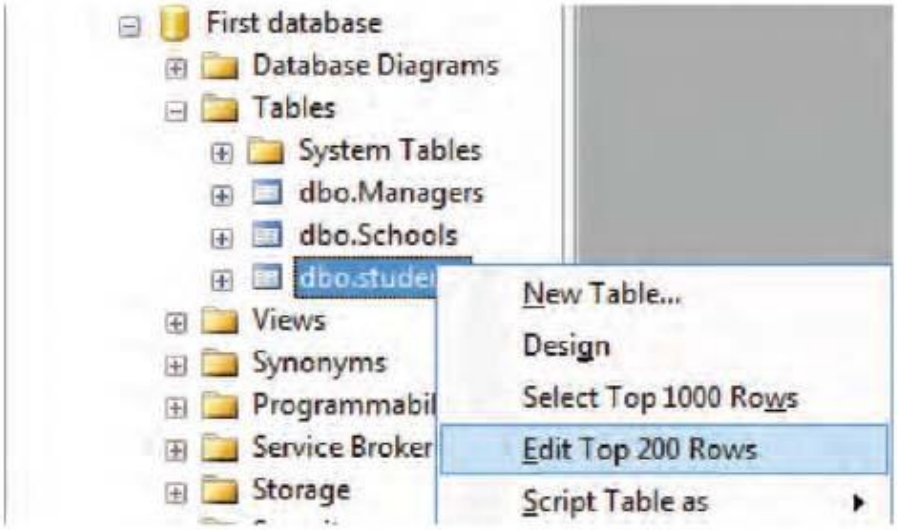
يمكن تعبئة البيانات في قاعدة البيانات باستخدام SQL Server 2008 بالطريقة المباشرة باتباع الخطوات الآتية:

أ. فتح برنامج SQL Server Management Studio .

ب. الاتصال مع مخدم قواعد البيانات.

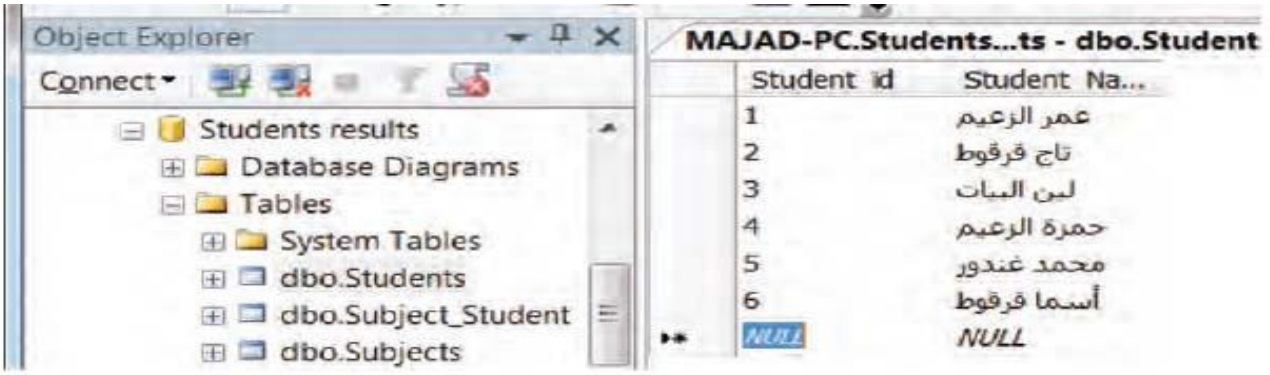
ت. نقر على الإشارة (+) بجانب المجلد Database، ثم ننقر على الإشارة (+) لإحدى قواعد البيانات، ثم ننقر على الإشارة (+) بجانب المجلد Tables .

ث. النقر بالزر الفأرة على أحدى جداول قاعدة البيانات، ثم ننقر على الأمر Edit Top 200 Rows، كما في الشكل (٧-١) الآتي :



الشكل (٧-١)

ج.تظهر نافذة إدخال البيانات في الجدول، تُدخل البيانات في حقول كل سجل، كما في الشكل(٧-٢)



الشكل (٧-٢)

➢ لغة الاستعلامات البنيوية SQL :

تحوي هذه اللغة مجموعو من التعليمات التي تهدف للبحث في قاعدة البيانات عن معطيات معينة توافق الشروط التي يضعها المبرمج في برنامجه، وكذلك تعديل قاعدة البيانات او الإضافة إليها أو الحذف منها، ويستخدم SQL Server Management Studio لكتابة تعليمات الاستعلام عن البيانات في قاعدة البيانات.

➢ الاستعلامات (Queries) :

تُستخدم الاستعلامات للتعامل مع البيانات في قاعدة البيانات بطريقة سهلة وسريعة، فالاستعلام يُستخدم لإخبار نظام قواعد البيانات DBMS عن البيانات المطلوبة للحصول عليها من قاعدة البيانات المحددة، حسب شروط يتم تحديدها ضمنه، ويستخدم الاستعلام من أجل:

١. الحصول على البيانات المطلوبة.
٢. تحديد طريقة عرض البيانات المستخلصة من الجدول.
٣. إضافة أو حذف أو تعديل البيانات في قاعدة البيانات.
٤. إضافة أو حذف الجدول من قاعدة البيانات.

ومن اللغات السهلة المستخدمة لكتابة الاستعلامات هي لغات الاستعلامات البنيوية (SQL)، لانها لغة قريبة من لغة المستخدم.

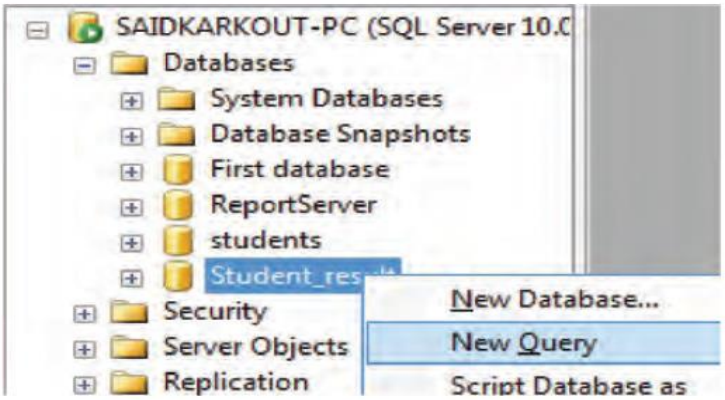
➢ إنشاء استعلام باستخدام SQL Server Management Studio:

١. يُفتح البرنامج SQL Server Management Studio.

٢. الاتصال مع مخدم البرنامج بنقر الزر Connect .

٣. ننقر على الاشارة (+) بجوار المجلد Database لعرض قاعد البيانات المُنشأة سابقاً، وينشأ الاستعلام بإحدى الطريقتين:

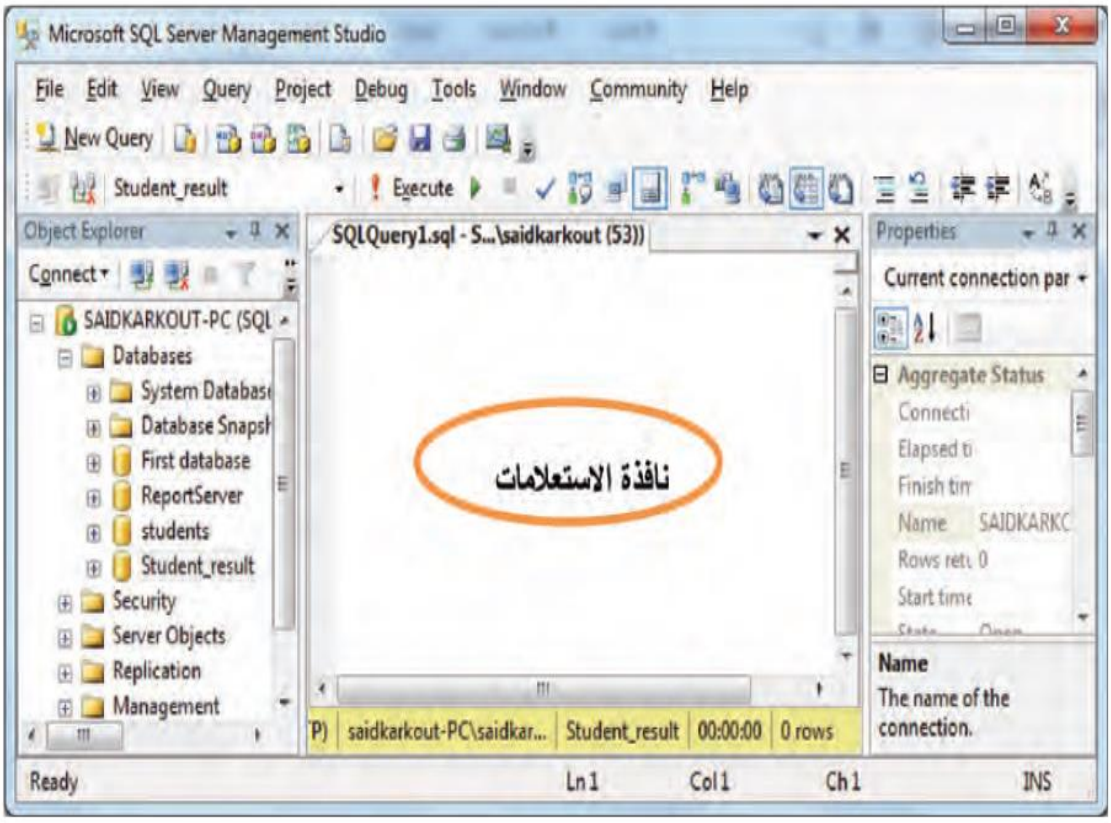
أ. ننقر بالزر الأيمن للفأرة على قاعدة البيانات المطلوب إنشاء الاستعلام عليها، وننقر على الامر New Query كما في الشكل (٨-١) :



الشكل (٨-١)

ب. تنقر على الاداة New Query من شريط الأدوات القياسي.

٤. في كلتا الطريقتين السابقتين تُفتح نافذة الاستعلامات الفارغة، وتكتب فيها تعليمات الاستعلام المطلوبة، كما في الشكل (٨-٢) :



الشكل (٨-٢)

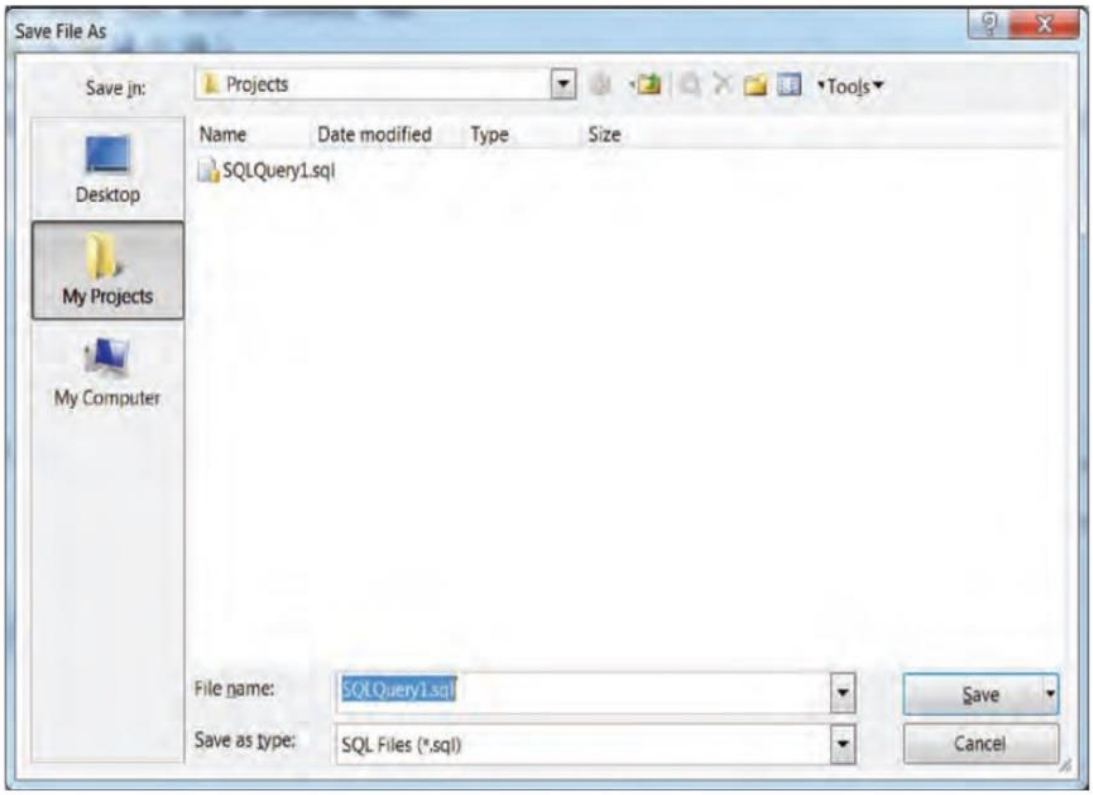
➢ حفظ وفتح الاستعلام :

بعد إنشاء الاستعلامات وكتابة تعليمات والتأكيد من تنفيذها، يمكن حفظ الاستعلامات بإتباع إحدى الطريقتين :

أ. من قائمة File يختار الأمر Save File AS .

ب. الضغط على المفتاحين Ctrl + S من لوحة المفاتيح .

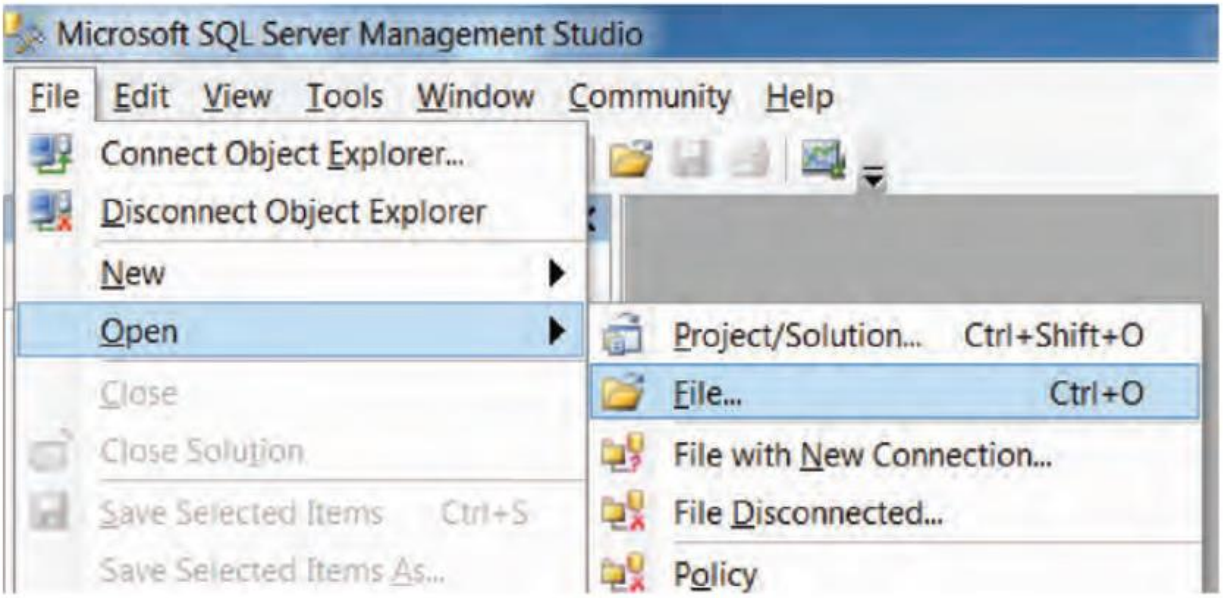
تظهر نافذة لحفظ الاستعلامات المكتوبة في نافذة الاستعلامات كما هو مبين في الشكل (٨-٣) :



الشكل (٨-٣)

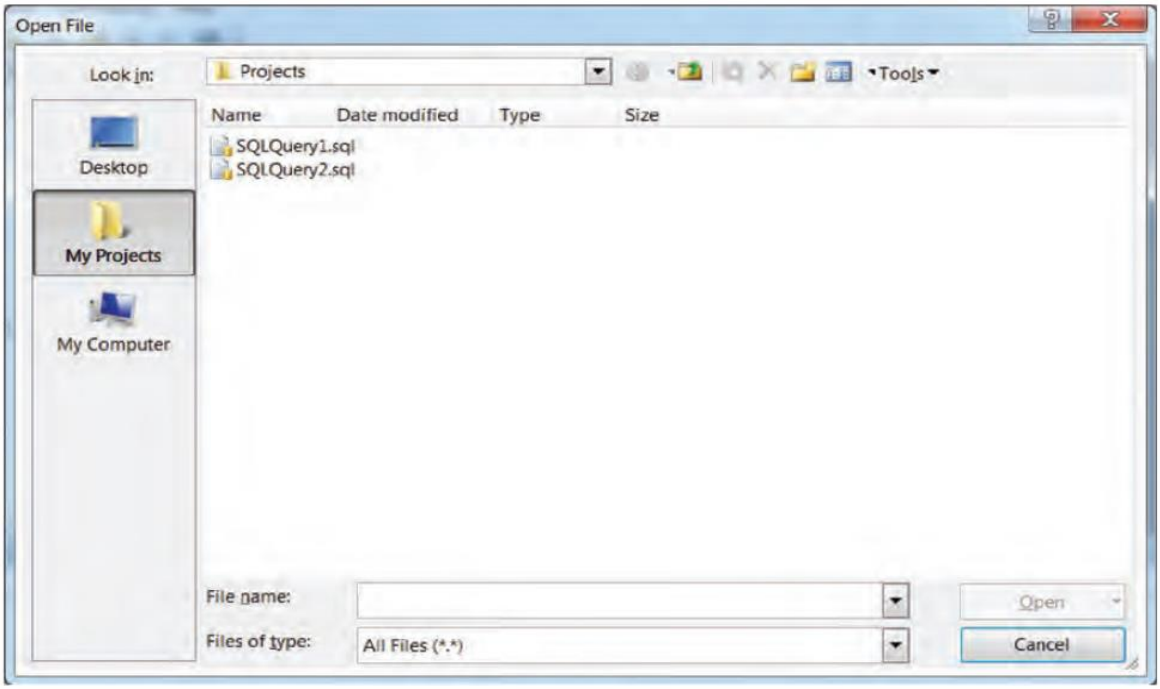
يُدخل في الحقل File name اسم الملف المطلوب حفظ الاستعلام به، مع ملاحظة ان امتداد ملف الاستعلام عند الحفظ خو SQL، ثم ننقر على الزر Save لحفظ الاستعلام كملف في المكان المحدد.

ولفتح الاستعلام مرة أخرى من قائمة File، ننقر على الامر Open، ثم ننقر على الخيار File كما في الشكل (٨-٤) الآتي :



الشكل (٨-٤)

تظهر النافذة المبينة في الشكل (٨-٥) يحدد منها المكان الذي حفظ فيه سابقاً، ثم يحدد الاستعلام وننقر على الزر Open فيفتح ملف الاستعلام في نافذة الاستعلامات.



الشكل (٨-٥)

المراجع

- جامعة دمشق، كلية المعلوماتية كتاب قواعد المعطيات (١)
- وزارة التربية، الثانوية الصناعية، حرفة تقنيات الحاسوب، كتاب لغة البرمجة
- الفريق العربي للبرمجة

http://download-internet-pdf-ebooks.com/ASP.NET/ 31-1-library-books.html

Engineer
AMMAR HILAL
0063935208805

****