الفصل الأول

نظام إدارة قواعد المعطيات

عنوان الموضوع:

أدوات وبرامج SQL Server الخدمية (1)

SQL Server Enterprise Manager.

الكلمات المفتاحية:

Enterprise Manager، مخدم، مجموعات المخدم، عميل SQL Server، سجلات SQL Server، خدمات تحويل المعطيات، لوحة مهمات المخدم، لوحة مهمات قاعدة المعطيات، مصمم الاستعلامات، خدمات تحويل المعطيات.

ملخص:

سنتناول في هذه الجلسة دراسة تفصيلية للأداة Enterprise Manager المُضمنة في نظـــام SQL Server والمـــستخدمة لإدارة مهماته ووظائفه وتسهيل العمل في بيئته.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

كيفية استخدام الأداة Enterprise Manager من أجل:

- تعريف المخدم وتحديد خصائصه وإدارته
- بناء قواعد المعطيات وتحديد خصائصها وإدارتها

SQL Server Enterprise Manager

سنتناول في هذه الجلسة دراسة تفصيلية للأداة Enterprise Manager المُضمنة مع SQL Server والمُستخدمة لإدارة مهماتــه ووظائفه وتسهيل العمل في بيئته

تعتبر SQL Server Enterprise Manager (EM) أداةً مُضمنة من قبل مايكروسوفت، مهمتها إدارة SQL Server من خلال واجهة مُستخدم تخاطبية مريحة. تبرز أهمية هذه الأداة بشكل واضح عندما نقارنها بأسلوب التعليمات السطرية الذي كان يــستخدم فيما مضى من قبل مدراء قواعد المعطيات لإدارة SQL Server

سنركز في هذه الجلسة علــى كيفيــة اســتخدام الأداة SQL Server Enterprise Manager لإدارة مخــدمات SQL Server المستخدمة.

SQL Server Enterprise Manager
تسجيل المخدمات وتشغيلها

- تأسيس مجموعات المخدمات
- تسجيل المخدمات
- تأسيس الاتصالات مع المخدمات
- تشغيل وإيقاف المخدمات
- تأسيس مجموعات المخدم:

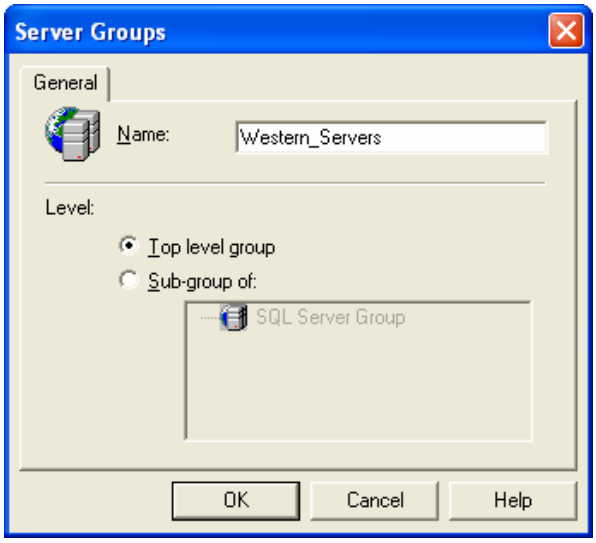
يمكن للأداة Enterprise Manager أن تقوم بإدارة عدّة مخدّمات، بحيث يتم تسجيل كل منها وتنظيمها ضمن مجموعات منطقية يطلق عليها اسم مجموعات المخدم؛

تُستخدم مجموعات المخدم لتنظيم المخدمات منطقياً، بحيث يمكن تجميع المخدمات حسب توزعها الجغرافي أو حسب وظيفتها؛

عندما يتم تنصيب الأداة EM للمرة الأولى، يتم إنشاء مجموعة افتراضية يطلق عليها اسم SQL Server Group، كما يمكن أن يتم إضافة أية مجموعة أخرى جديدة؛

لإنشاء مجموعة مخدم جديدة ينبغي الضغط بالزر اليميني للفأرة على أية مجموعة مخدمات موجودة أو علــى Microsoft SQL Servers، ثم اختيار New SQL Server Group؛

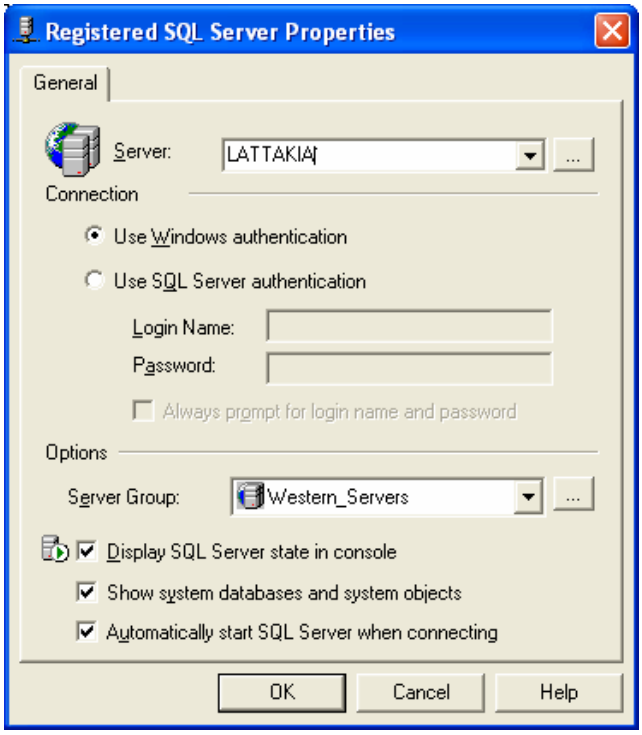
يوضح الشكل التالي كيف يمكن تسمية المجموعة الجديدة وتحديد فيما إذا كانت مجموعة جديدة مستقلة أو مجموعــة جزئيــة مــن مجموعة مخدم أخرى.



- تسجيل المخدمات:

تعتبر عملية تسجيل المخدمات الخطوة الأولى التي ينبغي تنفيذها عند تشغيل الأداة Enterprise Manager للمرة الأولى، ففي حال كانت الأداة تعمل على نفس المخدم، يتم تسجيل ذلك المخدم تلقائياً في مجموعة المخدمات الافتراضية، أما إذا كانت الأداة تعمل في مكان آخر على محطة عمل معينة، فينبغي تسجيل المخدم الذي تتعامل معه تلك المحطة لكي تتم عملية التخاطب معه، ويتم ذلك من خلال الضغط بالزر اليميني للفأرة على New SQL Server Registration، ثم تتبع الخطوات المحددة؛

يمثل الشكل التالي الواجهة التي تعبر عن خصائص مخدم جديد مسجل:



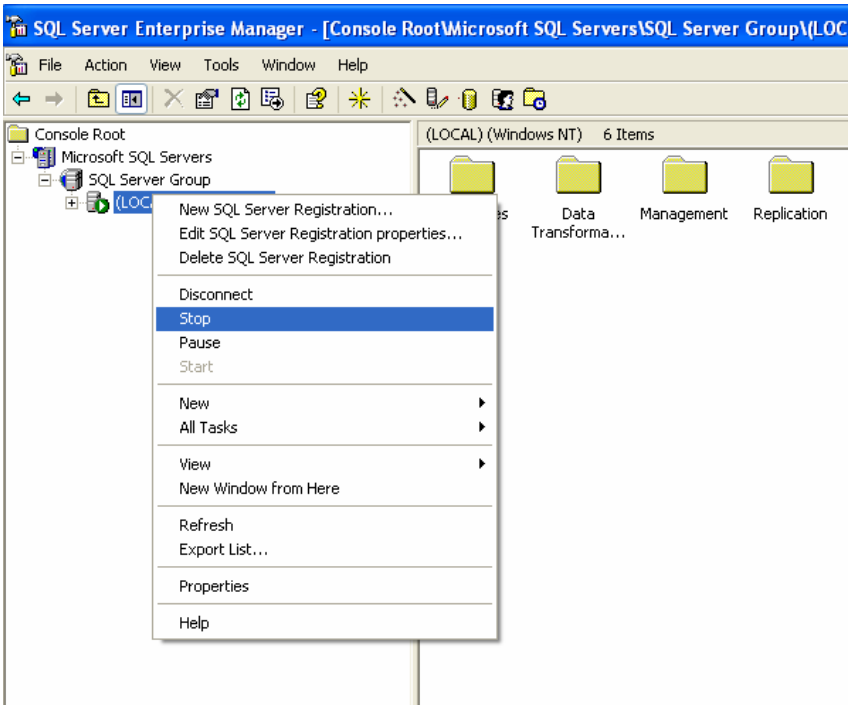
- تأسيس الاتصالات مع المخدمات:

تعتبر عملية تأسيس الاتصال أو إلغاءه مع مخدم معين عملية بسيطة تتم بعد الانتهاء من تسجيل المخدم، بحيث يمكن ببساطة اختيار "اتصال" أو "فصل الاتصال" من قائمة مهمات المخدم في الأداة Enterprise Manager؛ كما يمكن هنا الإشارة إلى الحالات التي تتطلب وجود اسم مستخدم وكلمة مرور لفتح اتصال معرّف مع المخدم.

- تشغيل وإيقاف المخدمات:

بنفس أسلوب الاتصال وفك الاتصال مع المخدم، يمكن وببساطة، من خلال الأداة Enterprise Manager تشغيل أو إيقاف تشغيل مخدم معين؛

يوضح الشكل التالي كيف يمكن إيقاف تشغيل مخدم عامل (بعد الضغط عليه بالزر اليميني):



SQL Server Enterprise Manager
إعداد المخدم

- خصائص المخدم
- خصائص الأمن
- تغيير الإعدادات

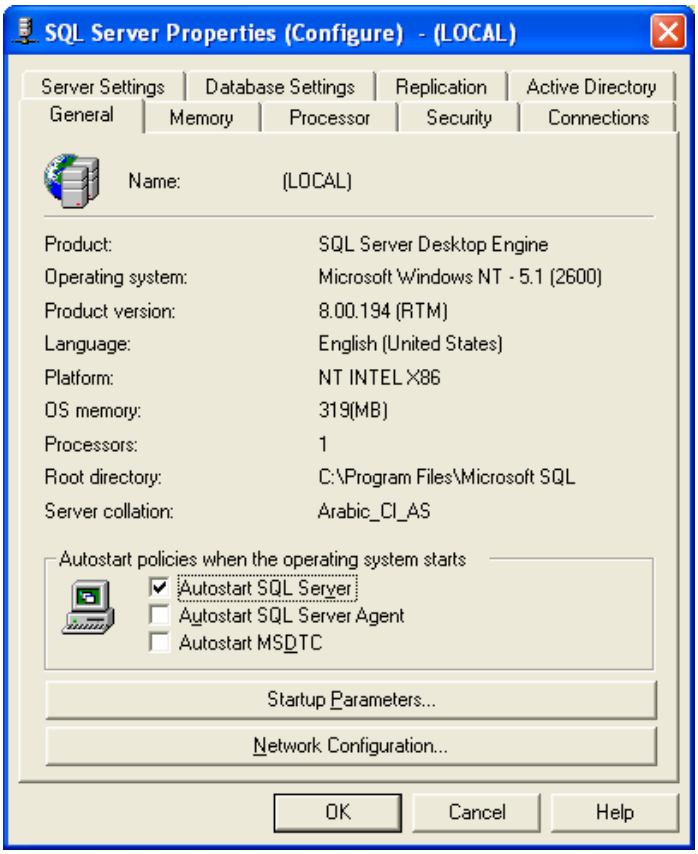
يمكن باستخدام الأداة Enterprise Manager أن يتم تغيير معظم خصائص المخدم، إلا أن التغيير في إعدادات بعض الخصائص الأخرى المتقدمة ينحصر بإجرائية النظام SP_CONFIGURE؛

تجدر الإشارة هنا إلى ضرورة إدراك أن تغيير أية خاصة من خصائص المخدم سوف يؤثر على كافة قواعد المعطيات التي يحتويها ذلك المخدم، وبالتالي ننصح بتدوين كافة التغيرات التي يتم إجراءها للتراجع عنها وقت الحاجة.

- خصائص المخدم:

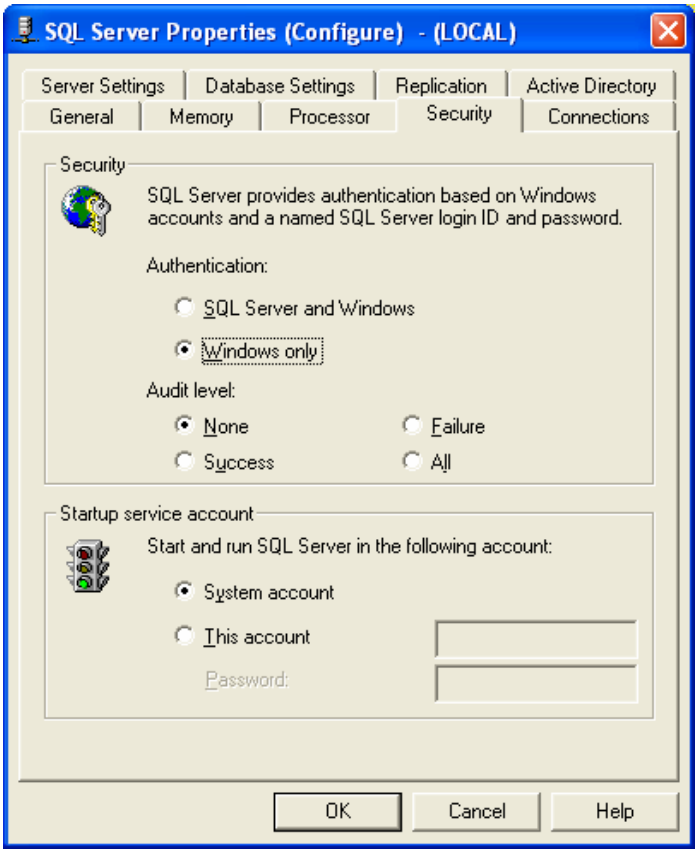
يمكن باختيار "خصائص" أو Properties من قائمة المهمات السريعة لمخدم معين أن نقوم باستعراض خصائص ذلك المخدم؛

تظهر في الشكل التالي واجهة الخصائص المذكورة، ونلاحظ فيها إمكانية التحكم بخصائص وإعدادات الذاكرة، والمعالج، والأمن، والاتصالات، وبخصائص المخدم، وبخصائص قواعد المعطيات الافتراضية بالإضافة إلى إعدادات الشبكة وخصائص الإقلاع وغيرها؛



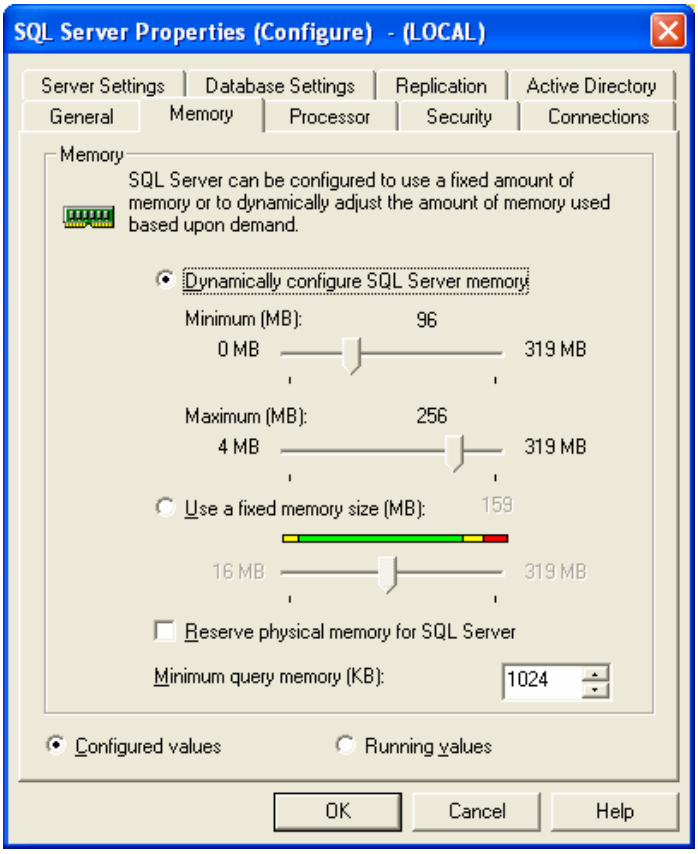
- خصائص الأمن:

يمكن من خلال صفحة الأمن، ضمن خصائص المخدم، أن يتم التحكم باستراتيجية الأمن المستخدمة، بحيث يمكن التبديل ما بين استراتيجية SQL Server مع Windows واستراتيجية Windows فقط.



- تغيير الإعدادات:

تتطلب بعض الإعدادت إعادة تشغيل SQL Server لكي يتم قبول التغيرات، تستخدم أزرار الاختيار Radio Buttons للتعبير عن هذه الحالات كما في الشكل التالي:



SQL Server Enterprise Manager

قواعد معطيات المخدم

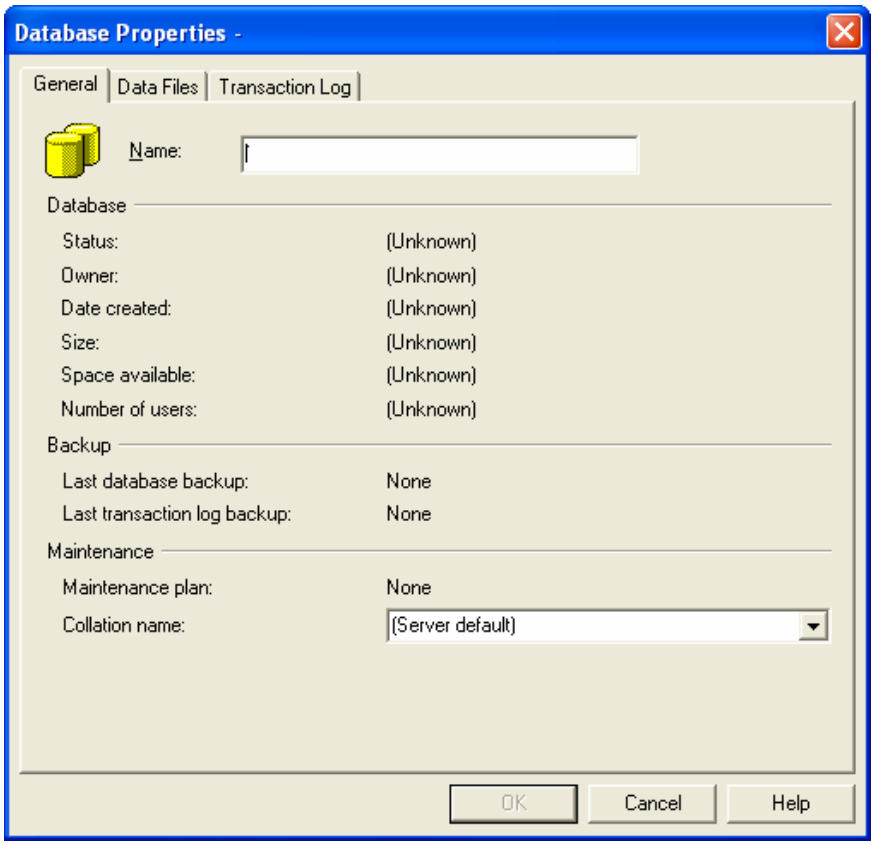
- إنشاء وتعديل قواعد المعطيات
- التخزين الاحتياطي واسترجاع المعطيات؛
- التعامل مع مخططات قواعد المعطيات.

يمثل SQL Server أداةً لاستخدام قواعد المعطيـــات، في حين تمثل Enterprise Manager الأداة المُضمنة في SQL Server والتي تُستخدم لإدارة قواعد المعطيات.

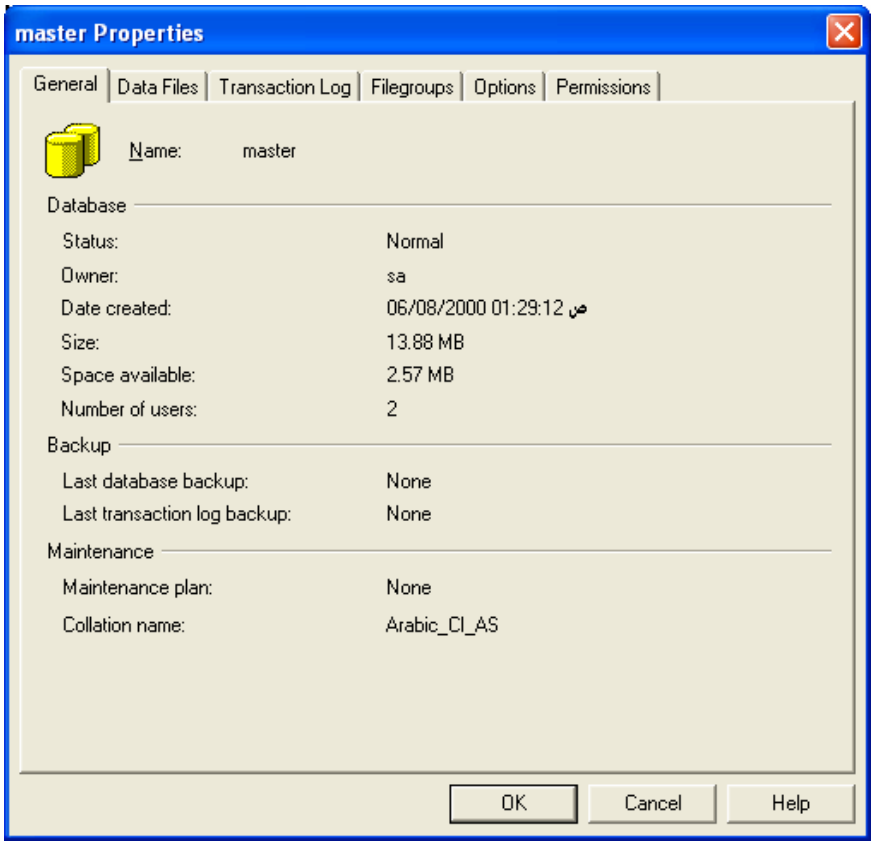
- إنشاء وتعديل قواعد المعطيات:

تعتبر عملية إنشاء قواعد المعطيات باستخدام الأداة Enterprise Manager بسيطة جداً، خاصةً مع استخدام مايكروسوفت لقيم تلقائية متعددة للتعبير عن معاملات ومتحولات قاعدة المعطيات عند إنشائها، ويمكن ملاحظة بساطة هذه العملية عند مقارنتها مع نفس العملية في أنظمة إدارة قواعد معطيات أخرى، هذا مع العلم أن إسناد قيم تلقائية لتلك المتحولات والمتغيرات قد لا يفي بالغرض في بعض الأحيان، إلا أن سهولة تعديل تلك القيم حتى بعد إنشاء القاعدة، يبقى عاملاً مميزاً لنظام SQL Server وأداة الإدارة الخاصة به Enterprise Manager.

تتم عملية إنشاء قاعدة معطيات جديدة من خلال الضغط بالزر اليميني على مجلد Databases ضمن شجرة المخدم الذي نريد إنشاء قاعدة المعطيات فيه. يتم بعدها اختيار New Database لتظهر الواجهة الموضحة بالشكل التالي والتي يمكن من خلالها تحديد اسم قاعدة المعطيات الجديدة، ونوع الترميز المستخدم (أي ترميز محارف المعطيات المستخدمة)، بالإضافة إلى إمكانية تغيير القيم التلقائية الأخرى المتعلقة بملفات المعطيات وبسجل المناقلات وبأماكن تخزينها وبحجومها وبتزايدها ... وغيرها.

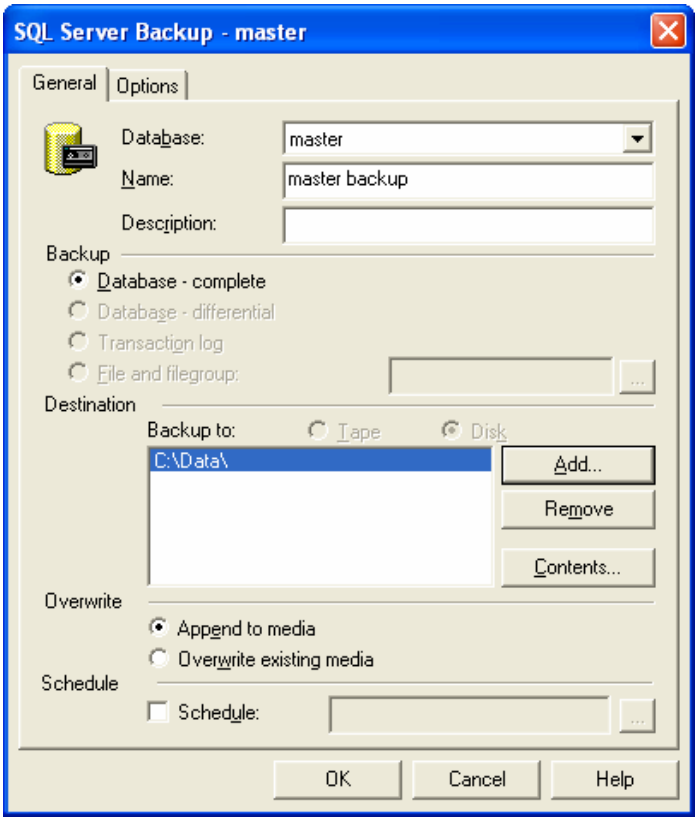


بعد إنشاء قاعدة المعطيات يمكن اختيار "خصائص" Properties بعد الضغط بالزر اليميني للفأرة عليها لتظهر الواجهة الموضحة بالشكل التالي والتي يمكننا من خلالها أن نقوم بإجراء تعديلات متعددة على خصائص قاعدة المعطيات، سواء كانت التعديلات مطبقة على ملفات المعطيات أو على السجلات أو على سماحيات استخدام القاعدة.



- التخزين الاحتياطي واسترجاع المعطيات:

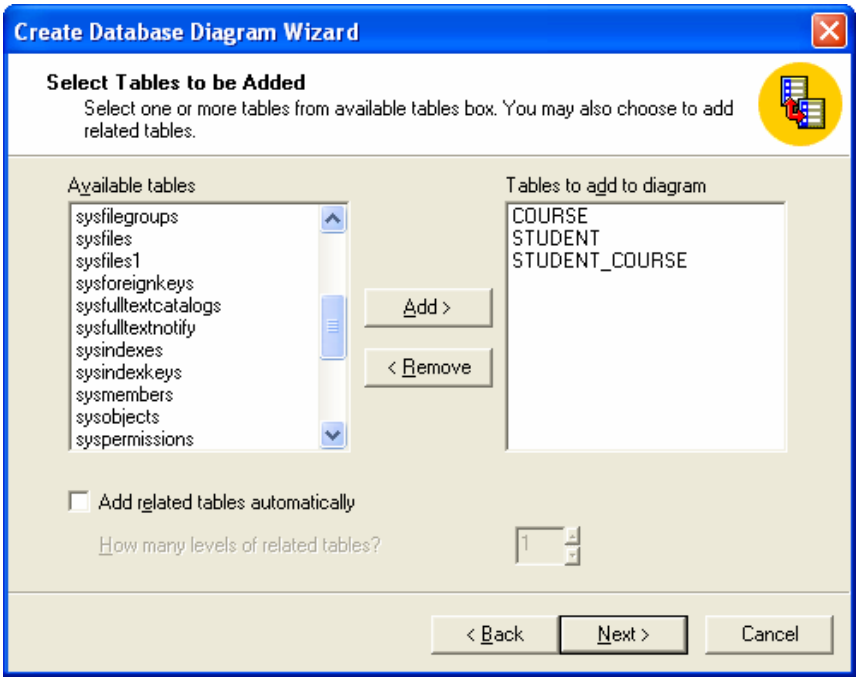
يمكن استخدام ميزات التخزين الاحتياطي واسترجاع المعطيات وذلك بالضغط بالزر اليميني على قاعدة المعطيات التـــي نرغـــب بإجراء تخزين احتياطي لمعطياتها ثم اختيار All Tasks ثم اختيار backup Database لتظهر الواجهة الموضحة بالشكل التالي والتي نستطيع من خلالها تحديد نوع عملية النسخ الاحتياطي بالإضافة إلى مكان تخزين ملف الخزن الاحتياطي الناتج:



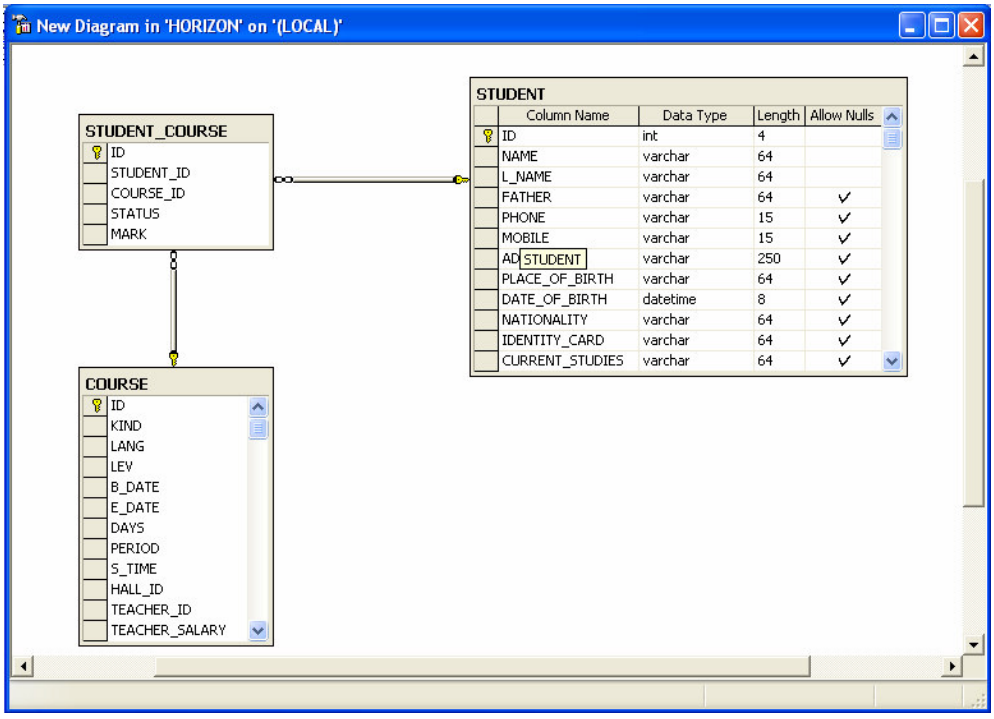
كذلك نستطيع من خلال اختيار Restore Database أن نقوم باسترجاع معطيات سبق أن تم تخزينها في مكان ما.

- التعامل مع مخططات قواعد المعطيات:

يمكننا باستخدام الأداة Enterprise Manager أن ننشئ مخطط كيانات–ارتباطات لقاعدة معطيات نقوم ببنائها، يتم ذلك من خلال اختيار New من قائمة مهمات قاعدة المعطيات التي نرغب ببناء مخطط لها ثم اختيار "Database Diagram"، بعد ذلك نقـوم بإضافة الجداول التي نرغب بتضمينها في المخطط كما يوضح الشكل التالي:



بالإضافة إلى أنه يمكننا عرض المخطط بأكثر من أسلوب كما يوضح الشكل التالي والذي نجد فيه تفصيلاً لأحد جـداول مخطـط الكيانات–ارتباطات الذي قمنا باختياره:



SQL Server Enterprise Manager

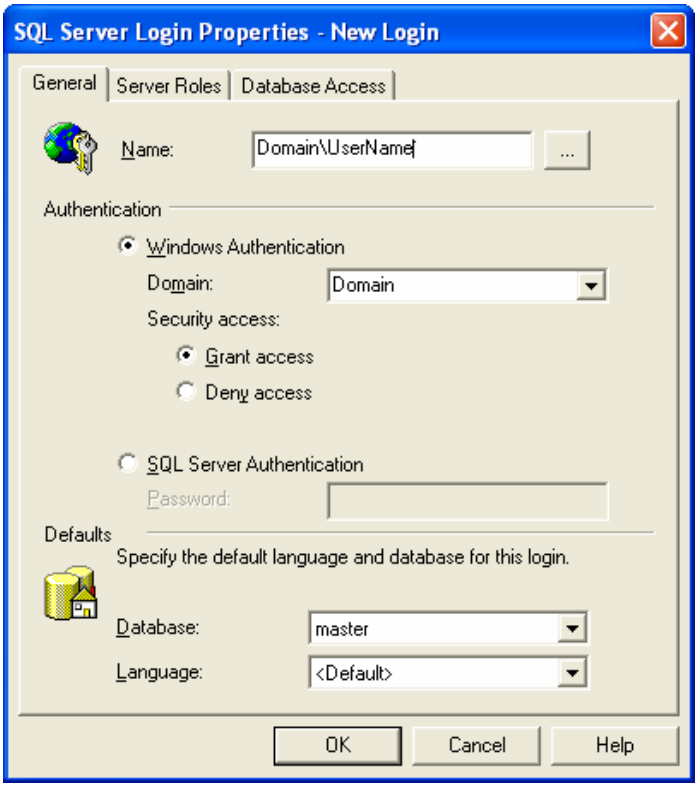
التحكم بأمن المخدم

- المستخدمين و الأدوار
- المخدمات المرتبطة بالمخدم الحالي و المخدمات البعيدة

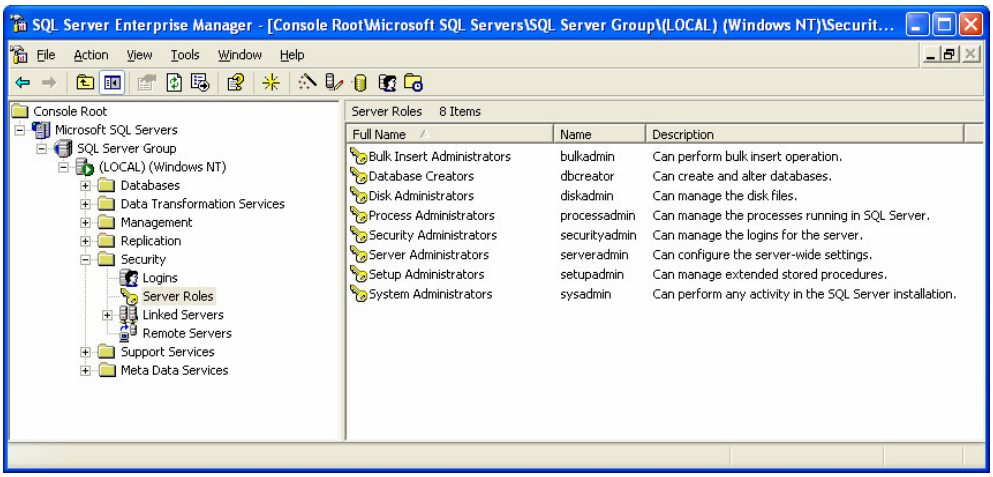
يتضمن مجلد Security الموجود ضمن شجرة المخدم في الأداة Enterprise Manager الأغراض المستخدمة لإدارة وتنظيم عمليات الولوج إلى المخدم. تُصنف هذه الأغراض ضمن صنفين أساسيين:

- المستخدمين و الأدوار:

تظهر في الشكل التالي الواجهة التي يتم عرضها عند اختيار "New Login" من قائمة مهمات الغرض Logins، بحيث يمكننا من خلالها أن نقوم بتحديد اسم المستخدم الجديد ونوع الوثوقية الممنوحة له، بالإضافة إلى قاعدة المعطيات التي يُسمح له بالولوج إليها؛



تُستخدم الأدوار لمنح سماحيات استخدام توابع النظام للمستخدمين الذين يتم تعريفهم. لا يمكن تعديل الأدوار، كما يمكننا أن نقـوم بإضافة مستخدم إلى أحد تلك الأدوار بحيث نسمح له بتنفيذ توابع النظام التي يمثلها ذلك الدور. يمثل الشكل التالي قائمـة بـأدوار المخدم المتاحة، مع العلم أننا سنتناول هذا الموضوع بشكل تفصيلي في الجلسات القادمة:



- المخدمات المرتبطة بالمخدم الحالي والمخدمات البعيدة:

يمكننا من خلال الغرضين Linked Servers و Remote Servers أن نتحكم بإعدادات الاتصال الذي نستخدمه للولوج إلى معطيات موجودة على مخدمات أخرى ترتبط بالمخدم الحالي.

SQL Server Enterprise Manager

إدارة المخدم

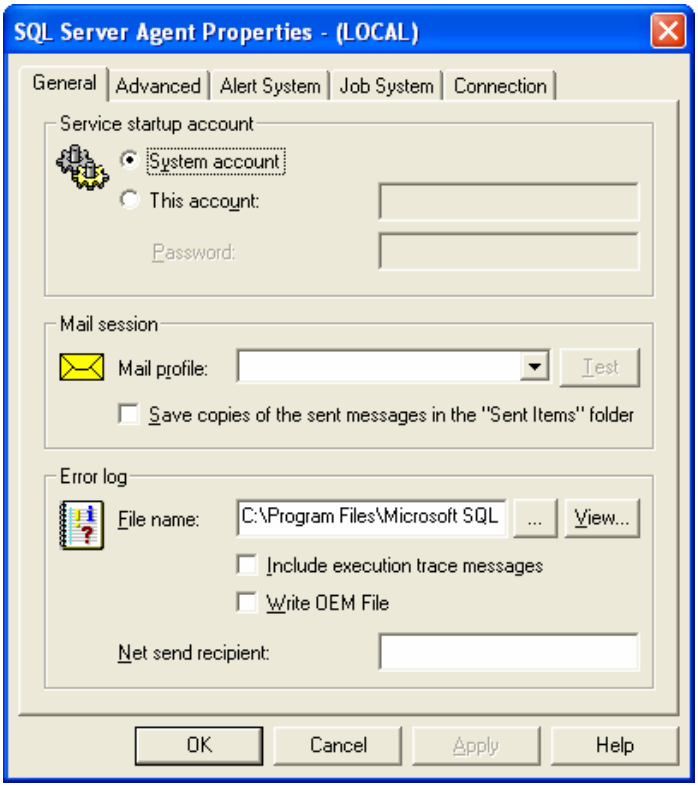
- عميل SQL Server
- الخزن الاحتياطي
- النشاط الحالي
- خطط صيانة قاعدة المعطيات
- سجلات SQL Server

يتضمن مجلد Management الموجود ضمن شجرة المخدم في الأداة Enterprise Manager الأغراض التي تلعب دوراً هاماً في

إدارة المخدم، والتي تقسم إلى:

- عميل SQL Server:

يُستخدم عميل SQL Server لجدولة بعض المهمات التي تتعلق بإدارة SQL Server، كتشغيل المخدم أو إيقافه على سبيل المثال. يمكن التحكم بتلك المهمات من خلال استعراض خصائص المجلد الذي يمثل العميل. فيما يلي عرض لتلك الواجهة:



يمكننا أيضاً من خلال عميل SQL Server أن نتحكم بمهمات أخرى كجدولة تحذيرات معينة أو أعمال خاصة أو عمليات يمكن تنفيذها في أوقات محددة، سنعرض في الجلسات القادمة تفصيلاً أدق لبعض تلك المهمات.

- الخزن الاحتياطي:

يمكن من خلال مجلد الخزن الاحتياطي الموجود ضمن مجلد Management في شجرة المخدم أن نقوم بتنفيذ وإدارة عمليات الخزن الاحتياطي على قواعد معطيات المخدم وتحديد أماكن تخزين كل نسخة احتياطية نقوم بإنشائها.

- النشاط الحالي:

يمكن من خلال هذا المجلد استعراض ومراقبة حالات إجراءات المخدم وإدارتها بقفل بعضها أو إيقاف بعضها الآخر.

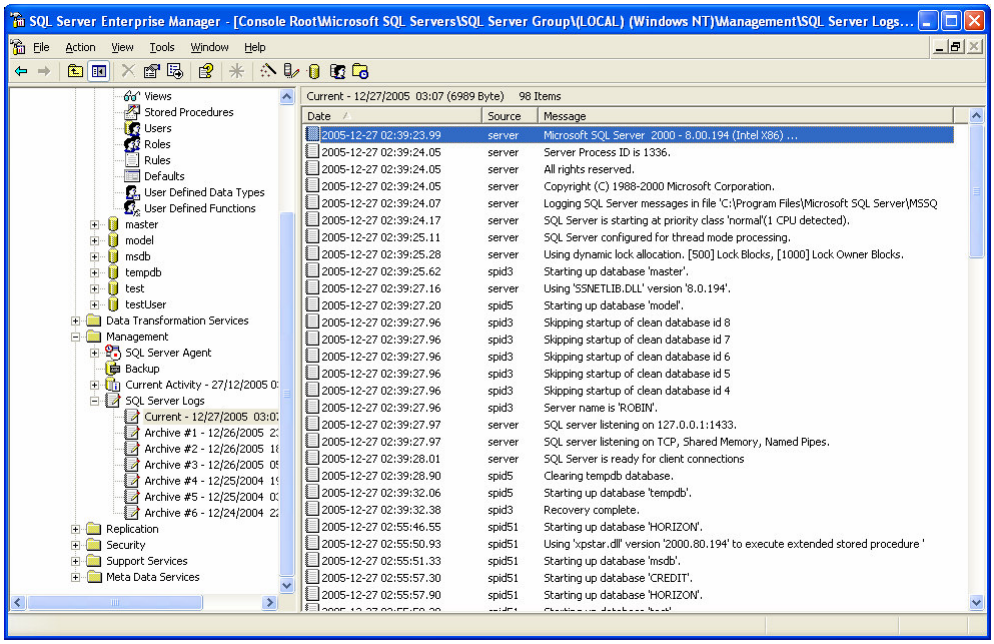
- خطط صيانة قاعدة المعطيات:

يمكن من خلال مجلد خطط صيانة قاعدة المعطيات تحديد وجدولة عدة خطط لإدارة عملية صيانة قاعدة المعطيات، بحيث يمكن من خلال الواجهات المتاحة –والتي تتميز بسهولة الاستخدام– أن تتم جدولة عدة مهمات في المخدم وعلى عدة قواعد معطيات فيه، هذا بالإضافة إلى امكانية استعراض الخطط المبنية مُسبقاًوإمكانية تعديل تلك الخطط.

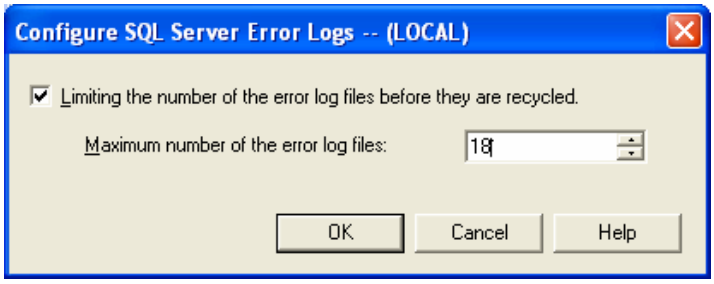
- سجلات SQL Server:

يمثل هذا المجلد كافة السجلات التي تتعلق بمعلومات المخدم وأخطاءه، وينبغي عدم الخلط ما بين هذه السجلات وبين سجلات المناقلات. إذ يحتوي هذا المجلد وبشكل تلقائي على سبعة سجلات بحيث يتم فتح سجل جديد عند كل تشغيل للمخدم وحذف أقدم سجل عند الإغلاق.

يمثل الشكل التالي عرضا لتفاصيل أحد تلك السجلات:



يمكن تغيير عدد السجلات المُخزنة وذلك باختيار "Configure" من قائمة المهمات السريعة لمجلد سجلات SQL Server بحيث تظهر الواجهة الموضحة بالشكل التالي والتي نحدد من خلالها عدد السجلات التي نرغب بالاحتفاظ بها لدواعٍ تختلف بحسب استراتيجية مدير قاعدة المعطيات:

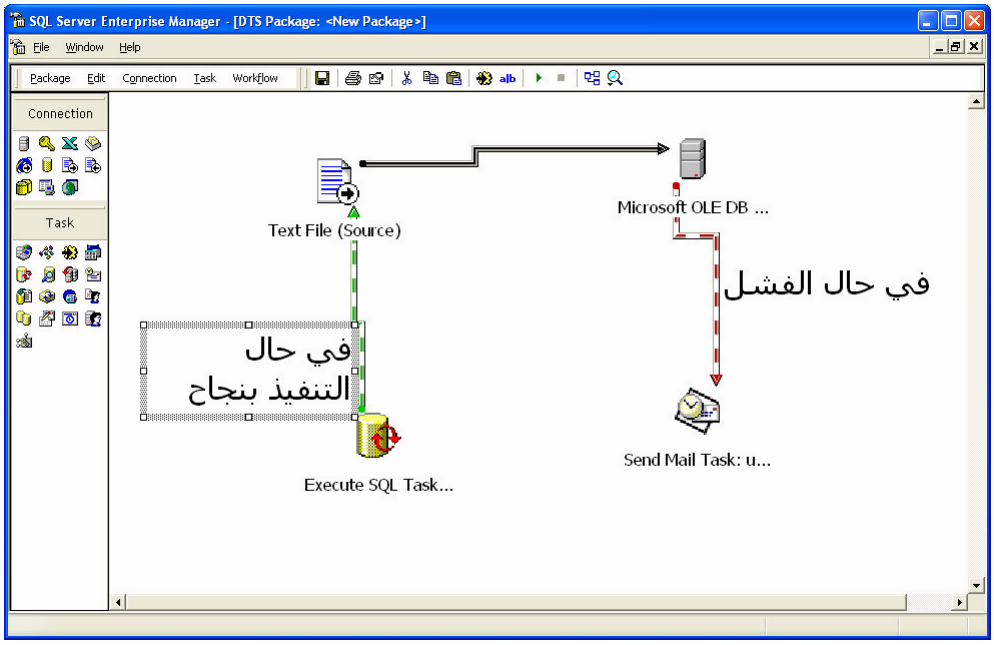


SQL Server Enterprise Manager

خدمات تحويل المعطيات

يتضمن مجلد Data Transformation Services الموجود ضمن شجرة المخدم في الأداة Enterprise Manager الأغـراض المسؤولة عن إجراء تحويلات في شكل المعطيات أوفي أسلوب تصديرها لكي يتم حفظها على وسائط خارجية؛

تجدر الإشارة إلى أنه يمكن تصدير أو استيراد المعطيات من خلال اختيار All Tasks ثم Import أو Export وذلك بالضغط بالزر اليميني على مجلد Database أو على أحد الجداول داخل قاعدة معطيات معينة. كما تجدر الإشارة إلى الأداة Enterprise Manager تدعم إمكانية بناء تصميم أكثر تعقيداً لتلك المهمة من خلال مجلد خدمات تحويل المعطيات، كما يوضح الشكل التالي:



SQL Server Enterprise Manager

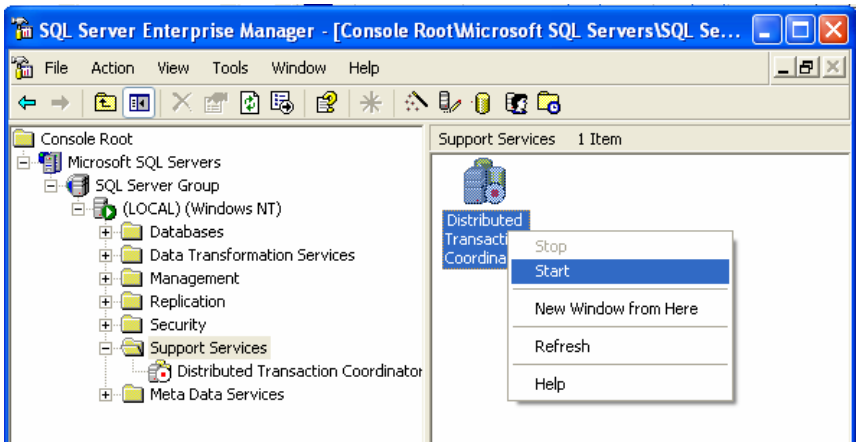
خدمات الدعم

- مشرف المناقلات الموزّعة
- البحث في النص
- بريد SQL

يتضمن مجلد Support Services الموجود ضمن شجرة المخدم في الأداة Enterprise Manager ثلاث أيقونات تحكّم تُقدم الخدمات التالية:

- مشرف المناقلات الموزّعة – Distributed Transaction Coordinator:

يمكن من خلال هذا المجلد تشغيل أو إيقاف خدمة الإشراف على تنفيذ المناقلات الموزّعة، وذلك في بيئة متعددة المخدمات، كما يوضح الشكل التالي:



- البحث في النص:

يمكن من خلال هذا المجلد تشغيل أو إيقاف خدمة البحث في النص والتي تعتمد في عملها على فهارس خاصة من أجل إجراء عمليات بحث سريعة على حجوم كبيرة من النصوص.

- بريد SQL:

يمكن من خلال هذا المجلد القيام بإرسال رسائل بريد ضمن حسابات البريد المعرّفة على المخدم.

SQL Server Enterprise Manager
أدوات النظام

- Query Analyzer
- SQL Profiler
- إدارة رسائل أخطاء SQL Server

يمكن من خلال قائمة أدوات "Tools" الموجودة في Enterprise Manager تشغيل تطبيقات أخرى تقوم بمهمات مغايرة للمهمات التي تقوم بها الأداة Enterprise Manager، هذه التطبيقات هي:

- Query Analyzer:

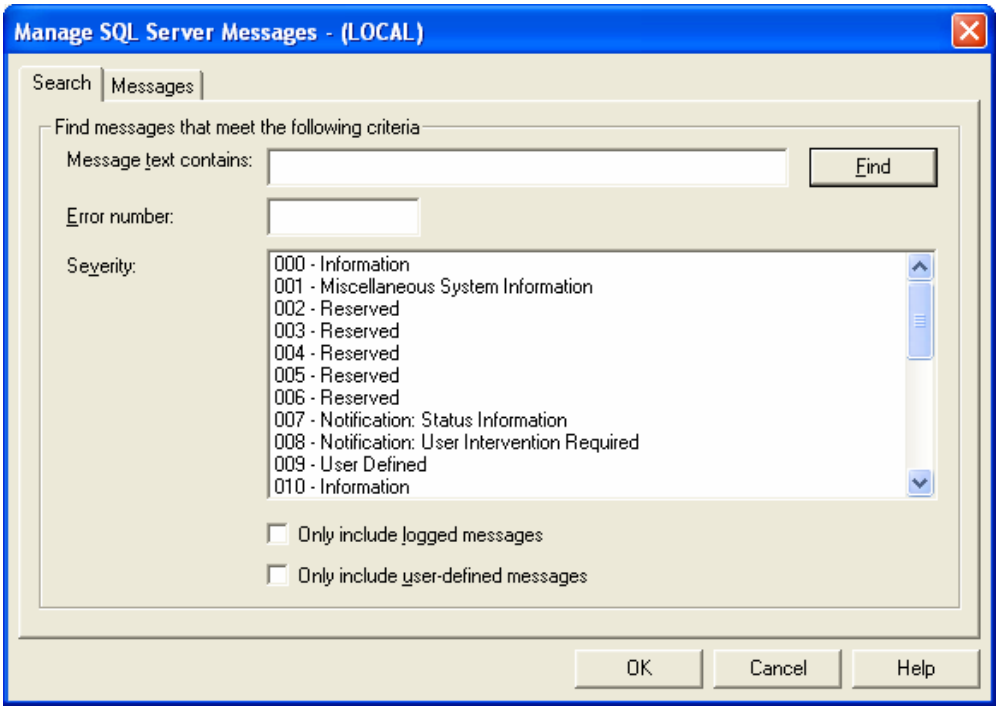
وهو عبارة عن أداة تستخدم لبناء الاستعلامات وتنفيذها وحفظها وتحليل أدائها، سيتم في الشرائح التالية استعراض مهمات هذه الأداة بالتفصيل.

- SQL Profiler:

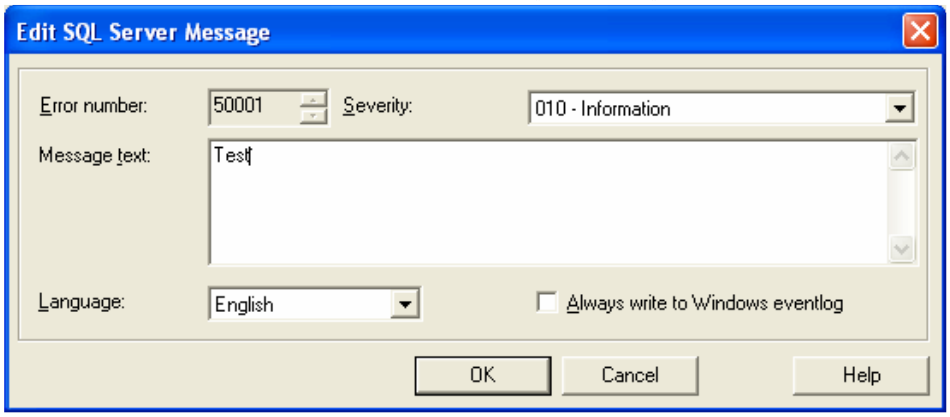
وهو عبارة عن أداة تستخدم لتسجيل الأحداث التي تقع على المخدم، كالاستعلامات على مختلف قواعد المعطيات على سبيل المثال، كما نستطيع من خلالها أن نقوم بحفظ تلك المعلومات المسجلة على ملف معين أو في جدول ما لإجراء تحليلات مناسبة، سيتم في الشرائح التالية استعراض مهمات هذه الأداة بالتفصيل.

- أداة إدارة رسائل أخطاء SQL Server:

يتم باختيار Manage SQL Server Messages من قائمة أدوات في الأداة Enterprise Manager عرض الواجهة الموضحة بالشكل التالي، والتي نستطيع من خلالها استعراض كافة رسائل الخطأ التي يعرضها SQL Server وذلك من خلال البحث عن محتوى أو رقم رسالة الخطأ:



هذا بالإضافة إلى أنه يمكننا أن نقوم بتعريف رسالة خطأ خاصة بنا وإسناد رقم إليها بحيث يمكننا استخدامها في المكان المناسب وفي الوقت الذي نرغب به. يمثل الشكل التالي الواجهة التي نُعرّف من خلالها رسالة الخطأ الجديدة:



SQL Server Enterprise Manager
تكرار المعطيات

- Publications
- Subscriptions

يمكننا من خلال المجلد "Replication" الموجود ضمن شجرة المخدم، أن نقوم بجدولة وظائف مهمتها تكرار المعطيات؛
يمكن تفعيل هذه الخاصة من خلال اختيار Configure Publishing, Subscription, and Distribution من قائمة المهمات السريعة للمجلد السابق.
تعتبر التكرارية الإجراء المسؤول عن نقل المعطيات بين المخدمات، حيث يطلق على مصدر المعطيات اسم الناشر، كما يطلق على المكان الذي ستنسخ إليه المعطيات اسم المُسجِّل.

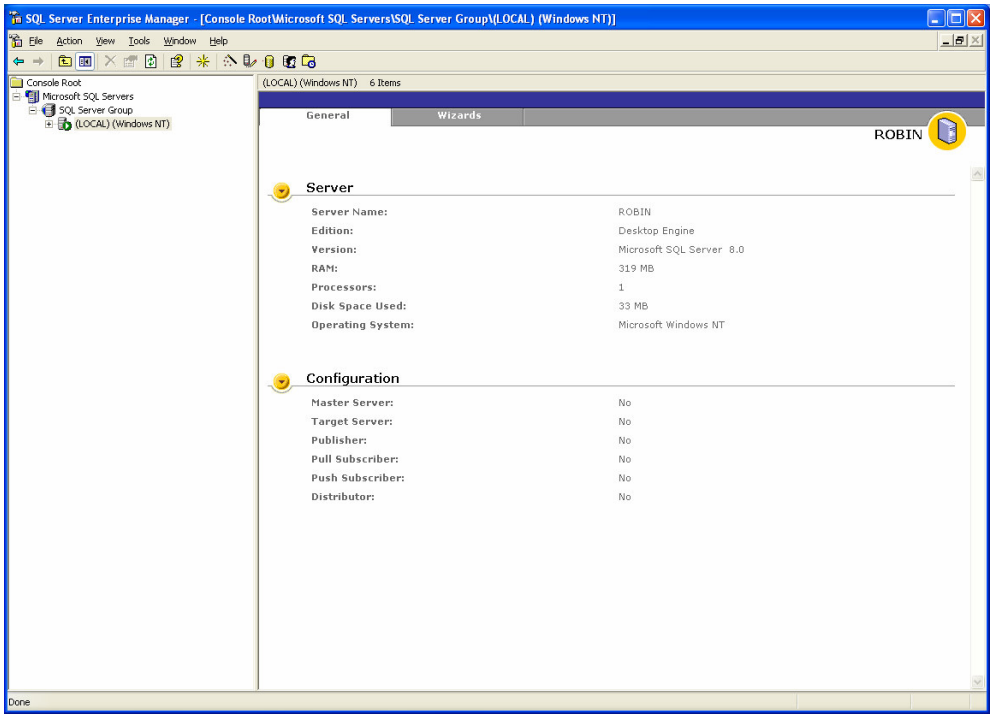
يمكن إنشاء ناشر جديد من خلال اختيار New Publication من قائمة المهمات السريعة للمجلد Publications، كما يمكن تعديل خصائص الناشر بعد إنشائه.

يمكن من خلال مجلد Subscriptions أن نقوم بإدارة عمليات استقبال وتسجيل المعطيات الواردة من مخدم ناشر مسبق التعريف.

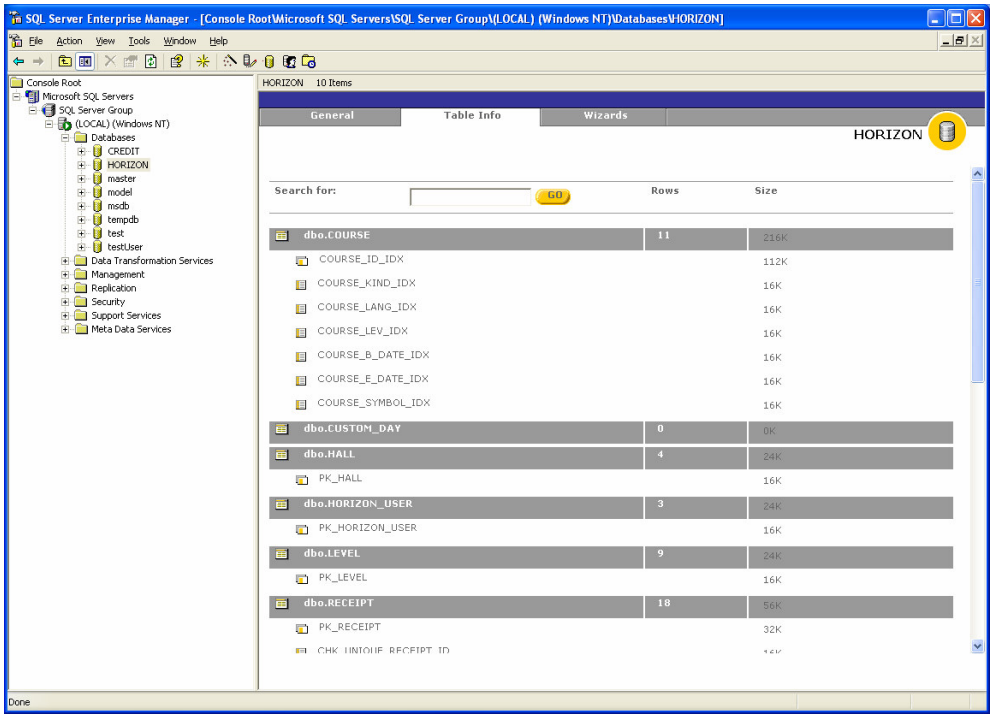
SQL Server Enterprise Manager
استخدام Database Taskpad

- لوحة مهمات المخدم
- لوحة مهمات قاعدة المعطيات

تستخدم واجهة لوحة مهمات قاعدة المعطيات Taskpad، لعرض معلومات تفصيلية وتزويد المُستخدم بالأدوات المتاحة بحسب نوع وظيفة كل منها؛
يمكن الولوج إلى لائحة المهمات تلك من خلال قائمة عرض View ثم Taskpad، وذلك بعد اختيار مخدم أو قاعدة معطيات؛
يمثل الشكل التالي لوحة مهمات مخدم، والتي تظهر فيها معلومات حول المخدم، بالإضافة إلى مجموعة الوظائف التي يمكن القيـــام بها والتي ترتبط بالمخدم:



يمثل الشكل التالي لوحة مهمات قاعدة معطيات، والتي تظهر فيها معلومات حول قاعدة المعطيات المختارة والوظائف التي يمكــن القيام بها والتي ترتبط بقاعدة المعطيات، بالإضافة إلى معلومات حول جداولها وفهارسها وحجم كل منها:

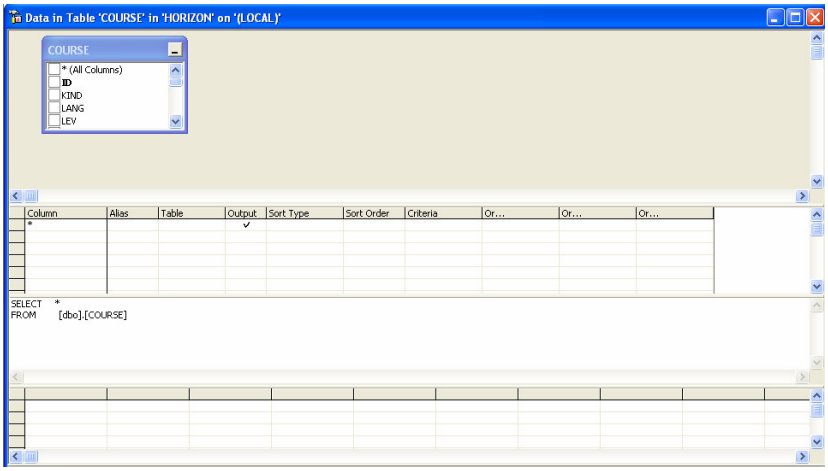


SQL Server Enterprise Manager

استخدام مصمم الاستعلامات

- كيفية عرض مصمم الاستعلامات
- مكونات الأداة:
  - لوحة المخطط
  - لوحة قائمة الأعمدة
  - لوحة SQL
  - لوحة النتيجة

يستخدم مصمم الاستعلامات من أجل بناء استعلامات على قاعدة المعطيات بشكل سهل وبسيط ومريح للمستخدم، ويمكن ضمن قاعدة معطيات معينة، كما يمكن عرضه أيضاً من خلال View من مجلد New View عرضه من خلال اختيار ليظهر الشكل التالي الذي Query ثم اختيار "Open Table" الضغط بالزر اليميني على أحد جداول قاعدة المعطيات ثم اختيار يعبر عن الأداة المسؤولة عن تصميم الاستعلامات بشكل مرئي:



تقسم هذه الأداة إلى أربعة لوحات، وهي:

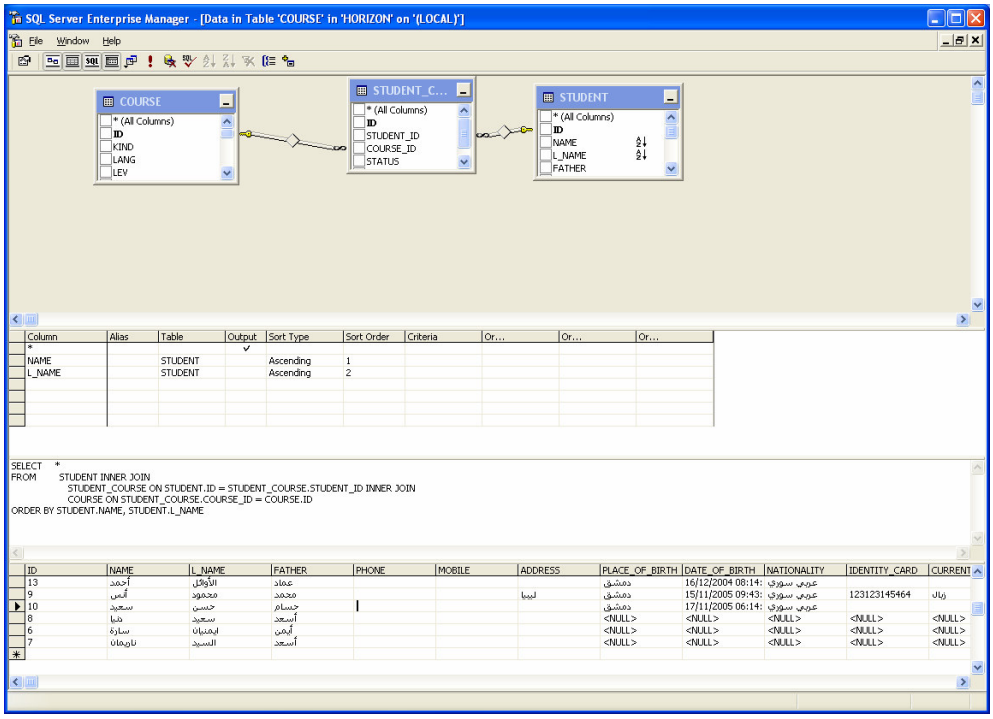
لوحة المخطط: وهو المكان الذي سنستخدمه لعرض توصيف رسومي للاستعلام الذي نقوم ببنائه، كما يمكن من خلال هذه اللوحة إضافة جداول جديدة والتأكد من وجود الأعمدة التي نرغب بعرض معطياتها في الاستعلام الذي نقوم ببنائه؛

لوحة قائمة الأعمدة: وهو المكان الذي نستطيع من خلاله اختيار الأعمدة التي نريد عرضها في الاستعلام ولكن بطريقة تختلف عن تلك في لوحة المخطط بحيث يتم هنا إضافة الأعمدة يدوياً، نلاحظ أن القيمة الافتراضية المختارة هي * والتي تعبر عن اختيار كافة الأعمدة من الجدول أو الجداول المحددة في لوحة المخطط، مع العلم أنه يمكننا تحديد الأعمدة التي نريد من أجل بناء استعلام مقتضب، كما أنن نستطيع من خلال هذه اللوحة أن نعبر عن محتويات عبارة Where في الاستعلام الذي نقوم ببنائه.

لوحة SQL: ويتم فيها عرض الاستعلام –الذي نقوم ببنائه– بلغة SQL، مع العلم أنه يمكننا أن نقوم بتعديل الاستعلام يدوياً من خلال هذه اللوحة.

لوحة النتيجة: وهو المكان الذي سيعرض نتيجة الاستعلام الذي قمنا ببنائه من خلال اللوحات السابقة ثم عملنا على تنفيذه.

فيما يلي عرض لاستعلام مبني من خلال مصمم الاستعلامات:



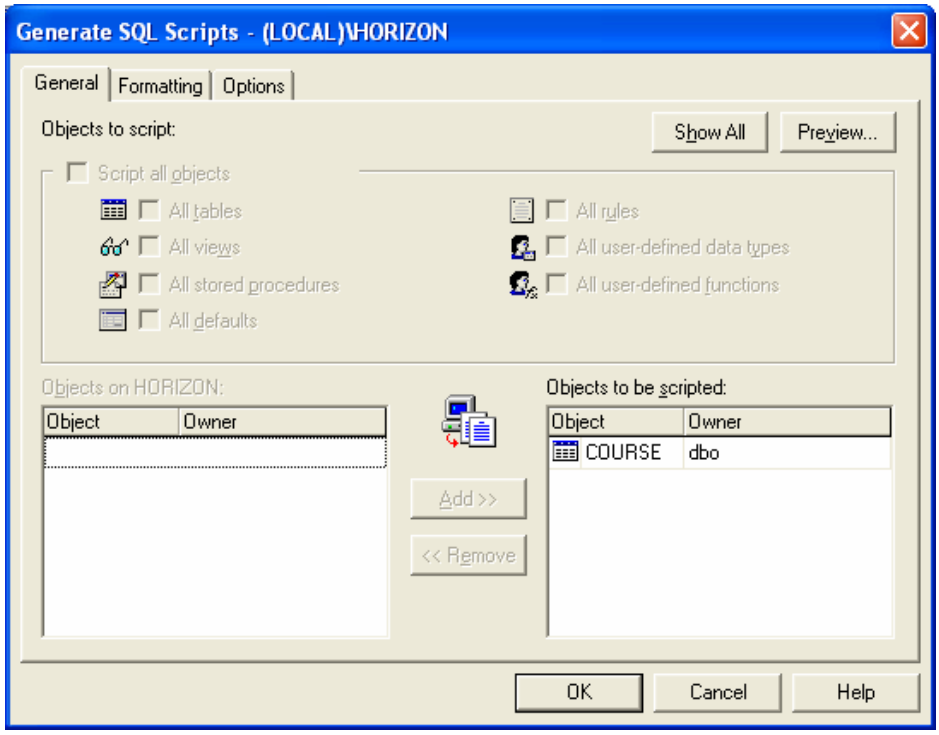
SQL Server Enterprise Manager

استخدام مولد الرماز

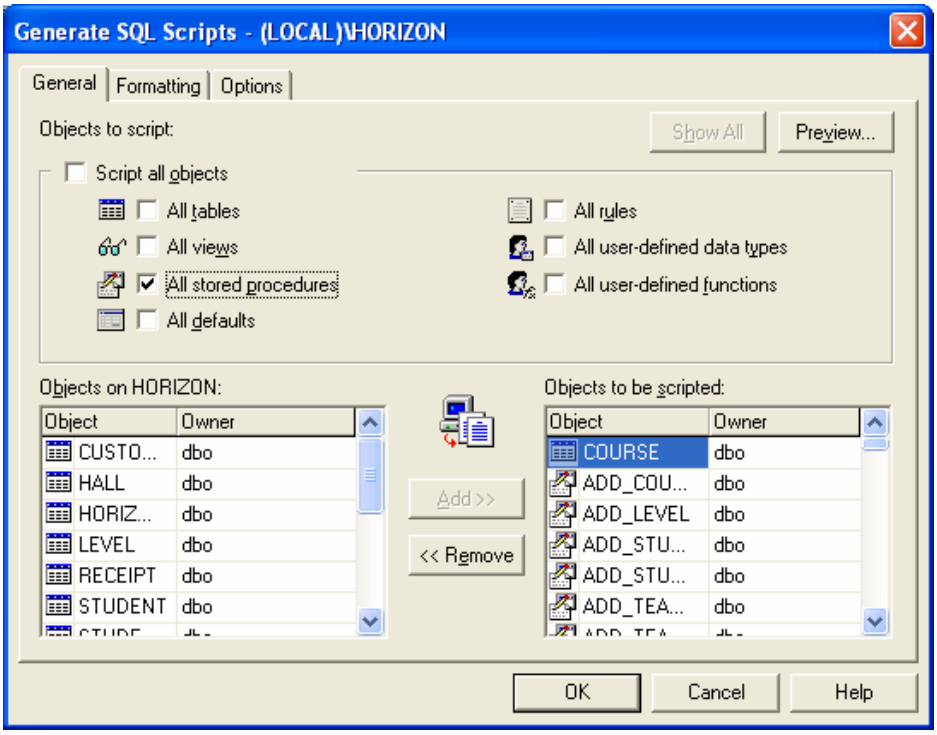
- عرض مولد الرماز
- توليد الرماز
- خصائص أخرى

يعتبر مولد الرماز المضمن في الأداة Enterprise Manager أحد أهم الخصائص التي تميز هذه الأداة وخاصة عندما ندرك سهولة استخدامه والخدمات التي يقدمها.

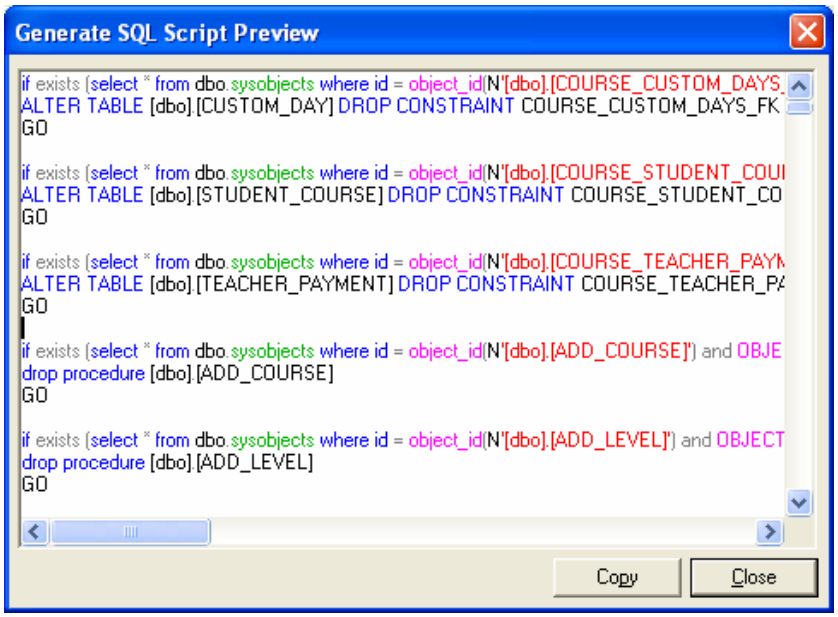
يمكننا الحصول على أي رماز لأي غرض من أغراض SQL Server بمجرد اختيار "All Tasks" بعد الضغط بالزر اليميني على ذلك الغرض، ثم اختيار "Generate SQL Script" لتظهر الواجهة الموضحة بالشكل التالي والتي تظهر الغرض الذي أردنا عرض الرماز الخاص به:



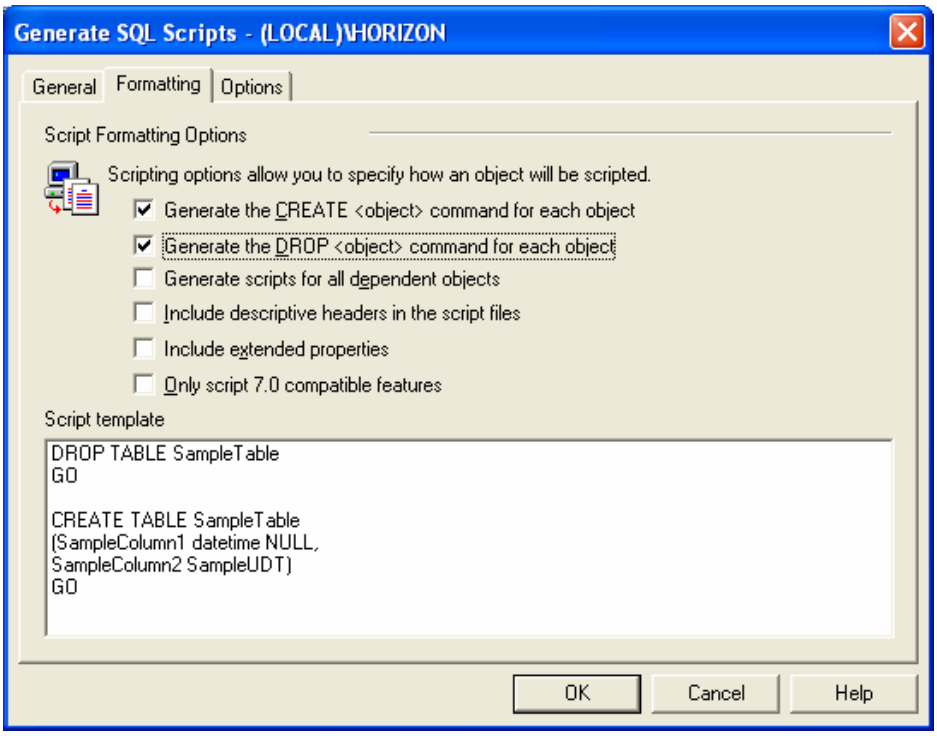
مع العلم أنه بإمكاننا إضافة أي غرض آخر إلى نفس ملف الرماز الذي سنقوم بتوليده، كما في الشكل التالي:



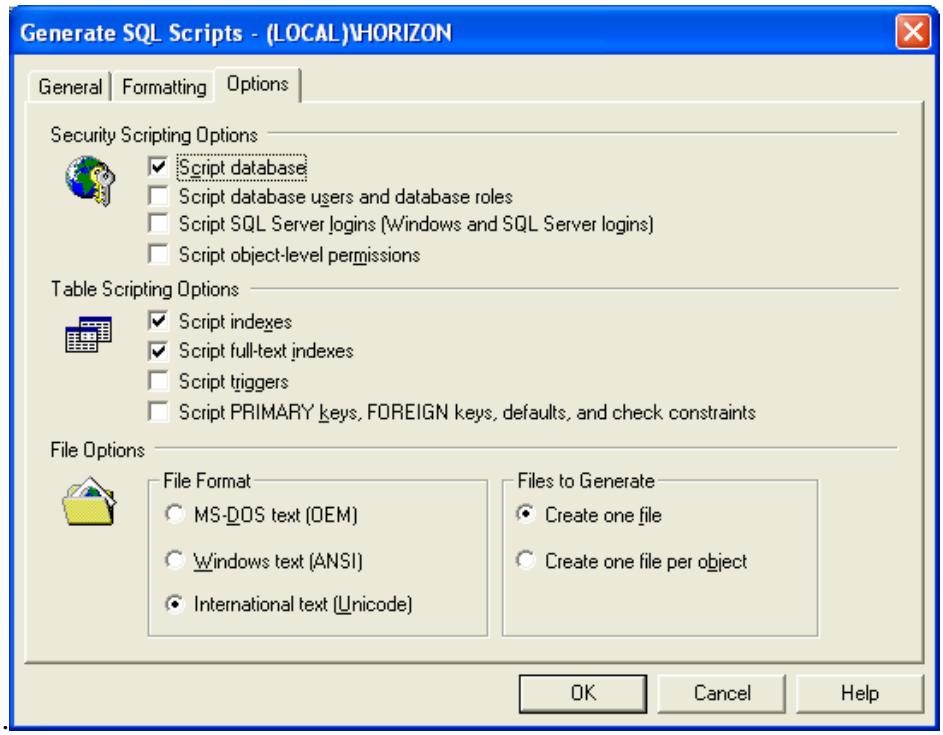
وأخيرا يمكننا باستخدام زر "Preview" أن نستعرض الرماز المتولد أو أن نخزنه على ملف معين:



يتميز مولد رماز الأداة Enterprise Manager بخصائص أخرى، فيمكننا –كما يوضح الشكل التالي– أن نتحكم بعرض ومكونات الرماز المتولد كأن ننفذ عمليات حذف للأغراض قبل إنشائها على سبيل المثال:



هذا بالإضافة إلى وجود خصائص متقدمة أخرى كتوليد رماز قاعدة المعطيات أو توليد رماز الفهارس وغيرها:


Generate SQL Scripts - (LOCAL)\HORIZON
General
Formatting
Options
Security Scripting Options
Script database
Script database users and database roles
Script SQL Server logins (Windows and SQL Server logins)
Script object-level permissions
Table Scripting Options
Script indexes
Script full-text indexes
Script triggers
Script PRIMARY keys, FOREIGN keys, defaults, and check constraints
File Options
File Format
MS-DOS text (OEM)
Windows text (ANSI)
International text (Unicode)
Files to Generate
Create one file
Create one file per object
OK
Cancel
Help

الفصل الثاني

عنوان الموضوع:

بيئة SQL server 2000

الكلمات المفتاحية:

محرك قاعدة المعطيات، مناقلة، فهرس، إجرائيات مخزنة، Enterprise Manager، Query Analyzer، Profiler، SQL Agent، Data Transformation Services.

ملخص:

يركز هذا الفصل على بيئة SQL Server 2000 ويُعرِّف، على نحو عمومي، مكوناته الأساسية بالإضافة إلى الميزات والتحسينات التي يتمتع بها مقارنةً بالنسخ السابقة منه.

أهداف تعليمية:

يهدف هذا الفصل إلى:

- التعريف ببيئة SQL server 2000
- التعريف بالمكونات الأساسية لـــ SQL server 2000
- إصدارات SQL server 2000

الميزات الجديدة في SQL server 2000

بيئة SQL server 2000

- SQL server 2000 عبارة عن منتج زبون/مخدّم لإدارة قواعد المعطيات

يتألف من عدد من برامج زبون/مخدّم نذكر منها: Query Analyzer، Enterprise Manager، Profiler، SQL Agent، Data Transformation Services.
يتألف 2000 ، وهو عبارة عن منتج زبون/مخدّم لإدارة قواعد المعطيات، من عدد من برامج زبون/مخدّم مختلفة تؤمن مجال واسع من الإمكانات والخدمات.

من تطبيقات الزبون في 2000 نذكر:
Query Analyzer، Enterprise Manager، Profiler، SQL Agent،
Data Transformation Services، حيث تتصل جميع التطبيقات السابقة مع محرك قاعدة المعطيات وتستخدم خدماته بطرق مختلفة.

تعتمد نماذج زبون/مخدّم التقليدية على وجود طبقتين (طبقة الزبون وطبقة المخدّم)، بينما تعتمد النماذج الحديثة على 3 طبقات أو n طبقة وتهدف إلى فصل منطق العمل عن طبقة التمثيل وعن طبقة الوصول للمعطيات، وبحيث يجري تقسيم النظام إلى مجموعة من الأغراض القابلة لإعادة الاستخدام.

مكونات SQL server 2000

- يُعتبر محرك قاعدة المعطيات المكون الأساسي والأهم لـــ SQL server 2000
- يُعتبر Enterprise Manager الواجهة الأساسية التي يجري من خلالها تنظيم وإدارة جميع مهمات قاعدة المعطيات
- يُعتبر SQL Query Analyzer المكان الأسهل لتنفيذ مخطوطات SQL

يُعتبر محرك قاعدة المعطيات المكون الأساسي والأهم لـــ SQL Server 2000، كما يمتلك النظام حزمة من التطبيقات المُساعدة

الإضافية كالأدوات التي تستخدم لإدارة بيئة العمل فيه نذكر منها:

- Enterprise Manager وهي الواجهة الأساسية التي يجري من خلالها تنظيم وإدارة جميع مهمات قاعدة المعطيات
- SQL Service Manager الذي يتيح إمكانية التحكم السهلة بالعديد من خدمات SQL، حيث يُستخدم لمراقبة هذه الخدمات
- SQL Query Analyzer الذي يُعتبر المكان الأسهل لتنفيذ مخطوطات SQL، حيث تُعبر كل نافذة من نوافذه عن اتصال مع قاعدة معطيات

- كما يمتلك النظام أدوات إضافية مثل: SQL Service Manager، DTS، SQL Profiler والتي سنأتي على ذكرها لاحقاً

محرك قاعدة المعطيات

يعمل محرك قاعدة المعطيات على:

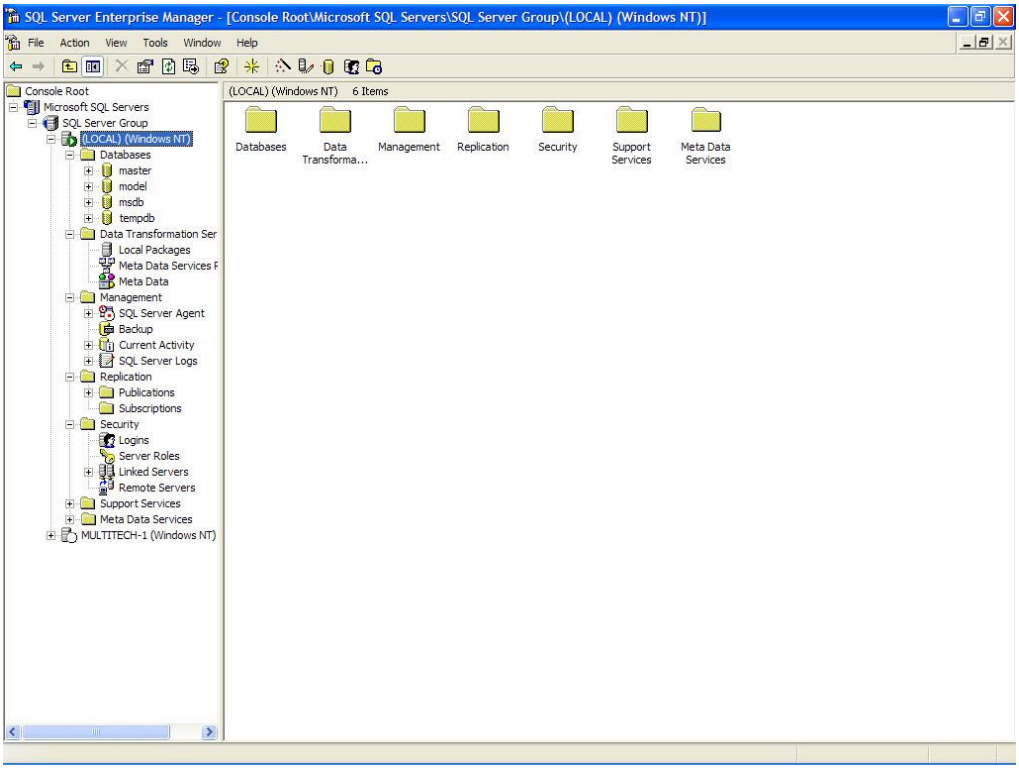
- تأمين مخزن موثوق للمعطيات
- تأمين البيئة الضرورية لتكامل المعطيات
- تأمين وصول سريع للمعطيات
- تأمين وصول متسق للمعطيات
- تأمين وصول آمن للمعطيات

يمكن تلخيص مهمات محرك قاعدة المعطيات بما يلي:

- تأمين مخزن موثوق للمعطيات المرسلة إليه، حيث تبدأ هذه العملية على مستوى العتاد (كاستخدام تقنية RAID)، وتستمر على مستوى محرك قاعدة المعطيات عن طريق تسجيل المناقلات (تسجيل كل التغييرات على قاعدة المعطيات من عمليات تراجع، أو عمليات معالجة أخطاء).
- تأمين الشروط الضرورية لتكامل المعطيات من أجل التأكيد على دقة واتساق المعطيات
- تأمين وصول سريع للمعطيات من خلال استخدام الفهارس
- تأمين وصول متسق للمعطيات من خلال استخدام مجموعة من القواعد التي تضمن الاتساق، مثل السماح لزبون واحد فقط بالتعديل على المعطيات في لحظة معينة، ومنع الزبائن الآخرين من الوصول إلى المعطيات أثناء تعديلها
- تأمين وصول آمن للمعطيات من خلال عدة مستويات من الأمان، على مستوى المخدم، وعلى مستوى قاعدة المعطيات، وعلى مستوى أغراض قاعدة المعطيات

Enterprise Manager

- هي الواجهة الأساسية التي يجري من خلالها تنظيم وإدارة جميع مهمات قاعدة المعطيات.
- تتضمن الإدارة:
  - إدارة عدة مخدّمات
  - وضع الإعدادات المناسبة للمخدّم
  - التحكم بالولوج وبمستخدمي وأدوار قاعدة المعطيات
  - نسخ واسترجاع قواعد المعطيات
  - بناء قواعد معطيات جديدة وإدارة مكوناتها



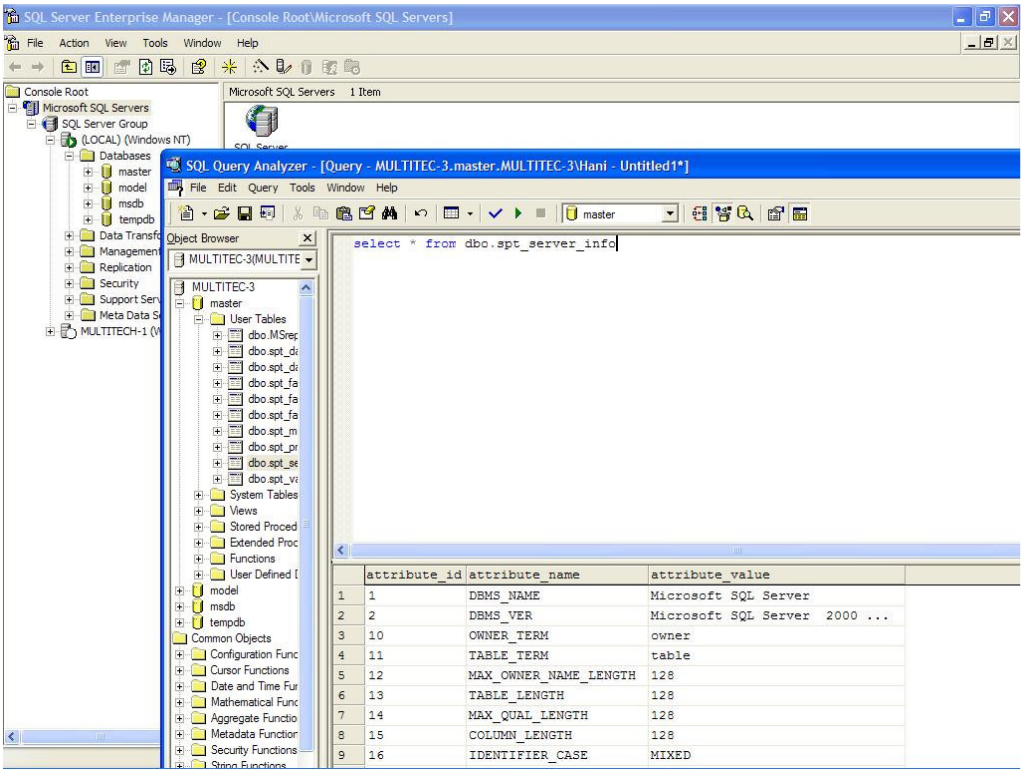
يُعتبر Enterprise Manager الواجهة الأساسية التي يجري من خلالها تنظيم وإدارة جميع مهمات قاعدة المعطيات.

فيما يلي استعراض لبعض مهمات Enterprise Manager:

- إدارة كاملة لعدة مخدمات من خلال واجهة واحدة تفاعلية ومريحة
- إدارة إعدادات المخدّم (حجم الذاكرة، عدد المعالجات التي يجب استخدامها، مسار تخزين المعطيات والملفات الأخرى)
- التحكم بالولوج وبمستخدمي قاعدة المعطيات وبأدوارهم
- تنظيم الأعمال الآلية من خلال SQL Agent
- نسخ واسترجاع قواعد المعطيات وتطوير خطط الصيانة
- بناء قواعد معطيات جديدة، وإدارة مكوناتها (جداول، فهارس، إجرائيات مخزنة)
- إعداد إجرائيات تكرار المعطيات والتحكم بها
- إدخال واستخراج المعطيات
- مراقبة فعاليات مخدّم SQL وتسجيل الأخطاء

SQL Query Analyzer

- SQL Query Analyzer هو المكان الأسهل لتنفيذ مخطوطات SQL، حيث تمثل كل نافذة من نوافذه اتصال مع قاعدة معطيات



يُعتبر SQL Query Analyzer المكان الأسهل لتنفيذ مخطوطات SQL. وتمثل كل نافذة من نوافذه اتصال مع قاعدة معطيات. كما يحوي إمكانية الاتصالات مع عدة مخدّمات مختلفة.

توجد في نسخة SQL server 2000 بعض التغييرات مقارنةً بالنسخ السابقة حيث تتضمن Object Viewer، بالإضافة إلى Stored Procedure Debugger ضمني.

SQL Server Agent

هي أداة جدولة تسمح بجدولة المخطوطات ومهمات الصيانة، والتعامل مع الإنذارات الآلية. كما تُعتبر أداة ضرورية في حال استعمال أداة التكرار (Replication).

- تعمل على نفس الجهاز التي يعمل عليها محرك قاعدة المعطيات
- يمكن إعدادها بحيث توزع المهام المشتركة على عدة مخدمات في حال استخدام إدارة متعددة المخدمات

تُعتبر SQL Server Agent أداة مهمة تساعد على جدولة المخطوطات وجدولة مهمات الصيانة، وتساعد في التعامل مع الإنذارات الآلية. كما تُعتبر أداة ضرورية في حال استعمال أداة التكرار (Replication).

يمكن النظر إلى SQL Server Agent على أنها خدمة تابعة لنظام Windows تعمل على نفس الآلة التي يعمل عليها محرك قاعدة المعطيات. ويمكن أن يتم إعدادها بحيث تقوم بتوزيع المهام المشتركة على عدة مخدمات، في حال استخدام إدارة متعددة المخدمات.

SQL Service Manager

- تطبيق صغير يتيح إمكانية تحكم سهلة بالعديد من خدمات SQL
- يُستخدم للمراقبة والتحكم بهذه الخدمات على أي جهاز ضمن الشبكة

يجري إقلاعه في معظم المخدمات آلياً مع إقلاع النظام

يُعتبر SQL Service Manager تطبيقاً صغيراً يتيح إمكانية تحكّم سهلة بالعديد من خدمات SQL مثل (محرك قاعدة المعطيات، SQL Agent، ومحرك بحث SQL، ومنظم المناقلات الموزعة.
كما يُستخدم SQL Service Manager للمراقبة والتحكم بهذه الخدمات على أي جهاز على الشبكة، حيث يستقصي عن حالة كل خدمة وذلك ضمن فترات زمنية معينة.
تقوم معظم المخدّمات بإقلاع هذا التطبيق آلياً مع إقلاع النظام.

SQL Profiler

- هي أداة من جهة الزبون تلتقط الاستعلامات والنتائج المنتقلة من وإلى المخدّم.
- من مهماتها:
  - التقاط جميع الاستعلامات المرسلة إلى المخدّم والاحتفاظ بها لتنفيذها لاحقاً عند الحاجة
  - الاستفادة منها في عملية إعادة البناء عند حدوث دخول غير مُرخّص

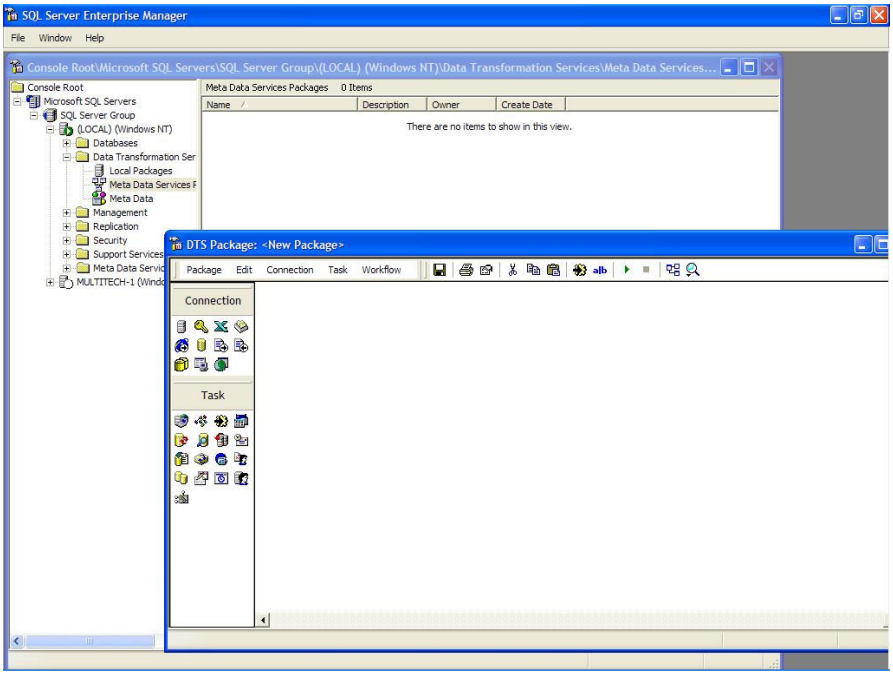
تُعتبر SQL Profiler أداة خاصة بالزبون تعمل على التقاط الاستعلامات والنتائج المنتقلة من وإلى المخدّم. تتلخص مهماتها فيما يلي:

- التقاط جميع الاستعلامات المرسلة إلى المخدّم والاحتفاظ بها لتنفيذها لاحقاً عند الحاجة
- الاستفادة من الاستعلامات المُخزنة في عملية إعادة البناء عند حدوث دخول غير مُرخّص

- يمكنها، من أجل الإجرائيات المخزّنة المعقدة، كشف الجزء المسبب للمشكلة من الإجرائية

Data Transformation Services

تُستخدم Data Transformation Services (DTS) أساساً كأداة لنقل المعطيات من مصدر إلى آخر.



تُعتبر Data Transformation Services (DTS) أداة قوية تُستخدم أساساً لنقل المعطيات من مصدر إلى آخر. وتتلخص بعض مهمات DTS:

- استيراد سريع لملف نصي إلى SQL Server
- الاتصال مع قواعد معطيات مختلفة، لنقل المعطيات منها أو إليها
- الاتصال مع موقع FTP (بروتوكول نقل الملفات) وتحميل ملفات منه

Replication Tool

- أداة من جهة المخدّم تستخدم لتحقيق التزامن بين عدة قواعد معطيات
- هناك ثلاثة أنواع من التكرار:

- تكرار لقطة
- تكرار مناقلة
- تكرار دمج

هي إحدى الأدوات العاملة من جهة المخدّم والتي تُستخدم لتحقيق التزامن بين عدة قواعد معطيات، وذلك عبر إرسال المعطيات من قاعدة معطيات إلى أخرى.

يدعم SQL Server ثلاثة أنواع من التكرار:

تكرار لقطة: يقوم المخدّم بأخذ لقطة أو صورة للمعطيات الموجودة في جدول وذلك في لحظة زمنية معينة، ويجري فيما بعد استبدال المعطيات الهدف (معطيات اللقطة) مع كل عملية تعديل.

تكرار مناقلة: ويتم فيها تكرار تنفيذ المناقلة على عدة قواعد معطيات (قواعد معطيات هدف)، وذلك بعد تنفيذها على قاعدة المعطيات الأساسية.

تكرار دمج: من الضروري في بعض الأحيان السماح لقواعد المعطيات الهدف بقبول التغييرات على الجداول المكررة ودمج هذه التغييرات مع بعضها لاحقاً.

تتمة مكونات SQL server 2000

- Microsoft Full Text Search Services الذي يقدم إمكانية البحث عن نص كامل، مما يفيد في عملية البحث عن حقول نصية طويلة في قاعدة المعطيات
- SQL Server Analysis Service الذي يقدّم خدمات أساسية تساعد على استخدام مخازن المعطيات
- Distributed Transaction Coordinator الذي يساعد في تنظيم المناقلات الموزعة والتأكد من تنفيذها بنفس طريقة المناقلات المحلية
-

يقدم Microsoft Full Text Search Services إمكانية البحث عن نص كامل ضمن قاعدة المعطيات مما يساعد في عملية البحث عن حقول نصية طويلة في القاعدة. لتنفيذ عمليات البحث تلك، يجري أولاً تحديد الجداول

أو قواعد المعطيات المراد البحث فيها، ومن ثم يجري بناء فهارس خاصة على هذه الجداول من أجل استخدامها في عملية البحث.

ويقدّم SQL Server Analysis Service خدمات أساسية من أجل استخدام مخازن المعطيات، عبر اتصاله مع عدة قواعد معطيات SQL Server تحوي مخازن معطيات.

أما Distributed Transaction Coordinator فيعمل على تنظيم المناقلات الموزعة والتأكد من تنفيذها بنفس طريقة المناقلات المحلية.

إصدارات SQL server 2000

- توجد إصدارات مختلفة من SQL server تعمل على بيئات مختلفة، لذا يعتمد اختيار الإصدار المناسب على قاعدة المعطيات المُراد بناؤها، وعلى المعالجة المطلوبة للمعطيات، وعلى بيئة العمل المطلوبة
- تُستعمل النسخة القياسية مع الأنظمة الصغيرة أو المتوسطة الحجم التي لا تحتاج إلى قابلية توسع ولا إلى متاحية عالية ولا إلى أداء عالي
- تُعتبر النسخة المؤسساتية أكثر النسخ شموليةً وكمالاً وهي النسخة المناسبة عند الحاجة إلى قابلية توسع وإلى متاحية عالية وإلى أداء عالي

توجد إصدارات مختلفة من SQL server تعمل على بيئات windows مختلفة، لذا يعتمد اختيار إصدار الـــSQL server 2000 على قاعدة المعطيات المُراد بناؤها، وعلى المعالجة المطلوبة للمعطيات، وعلى بيئة العمل المطلوبة.

تُستعمل النسخة القياسية لـــSQL Server 2000 مع الأنظمة الصغيرة أو متوسطة الحجم التي لا تحتاج إلى قابلية توسع ولا إلى متاحية، ولا إلى أداء عالي، بينما تُعتبر النسخة المؤسساتية أكثر النسخ شموليةً وكمالاً وهي النسخة المناسبة عند الحاجة إلى قابلية توسع وإلى متاحية عالية وإلى أداء عالي.

تكون النسخة القياسية مصممةً للتطبيقات من جهة المخدّم، في حين توجد نسخٌ أخرى من SQL Server مصممة لمستخدمين آخرين:

- نسخة المطور: نسخة خاصة بالتطوير واختبارات المستخدم النهائي
- النسخة الشخصية: نسخة خاصة بالمستخدمين عند حاجتهم إلى تشغيل تطبيقات تحتاج إلى قاعدة معطيات محلية
- نسخة محرك سطح المكتب: عبارة عن محرك قاعدة معطيات فقط من دون وجود أدوات إدارة

ما هو الجديد في SQL Server 2000 ؟

- استخدام الإجرائيات المُخزنة ضمن عبارات الاستعلامات SQL
- تعريف فهارس على المنظار
- إمكانية التقسيم الأفقي للجداول ضمن المنظار عبر عدة مخدّمات

يمكن للمُستخدم في النسخ السابقة من SQL Server تعريف إجرائيات SQL مُخزنة خاصة به، بالإضافة إلى استخدام مجموعة من التوابع مُسبقة التعريف، ولكن لا يمكن استخدام الإجرائيات المخزنة ضمن عبارات الاستعلامات SQL. أما في SQL Server 2000 فقد أصبحت هذه الإمكانية متاحة.

كذلك، يجري تنفيذ عمليات الدمج وتوابع التجميع على جداول قاعدة المعطيات، في النسخ السابقة من SQL server، في كل مرة يتم فيها طلب الاستعلام على منظار من المناظير المُعرّفة ضمن قاعدة المعطيات (حيث يُستخدم المنظار عادةً لتبسيط الاستعلامات المعقدة عبر تضمُنه لعمليات دمج ولتوابع تجميع). أما في SQL server 2000، فقد أصبح بالإمكان تعريف فهارس على المنظار بحيث تُخزَن نتيجة المنظار في قاعدة المعطيات وتجري عملية فهرستها باستخدام الفهارس، مما يساعد في تحسين الأداء وخصوصاً الأداء المتعلق بعمليات الدمج وبتنفيذ توابع التجميع.

كما يتيح SQL server 2000 إمكانية التقسيم الأفقي للجداول ضمن المنظار عبر عدة مخدّمات، بينما كانت هذه الإمكانية مقصورة على تقسيم الجداول ضمن مخدّم واحد مُعرّف المنظار ضمنه.

ما هو الجديد في SQL server 2000 (2)

- تخزين النصوص والصور الصغيرة مباشرةً في سطور المعطيات
- تحديد الفعل الذي يجب اتخاذه في حال جرى تجاوز شروط التكامل المرجعي
- تعريف أنماط جديدة ضمن SQL server 2000 يمكن استخدامها في أماكن مختلفة

يجري في النسخ السابقة من SQL server، تخزين الصور والنصوص ضمن صفحةٍ جديدةٍ مرتبطةٍ بسطر المعطيات، حيث يجري تخزين مؤشر يشير إلى الصفحة التي تحوي المعطيات، ضمن السطر. في حين يُقدّم

SQL server 2000 خياراً جديداً يسمح بتخزين النصوص والصور الصغيرة مباشرةً في سطر المعطيات مما يقلّل من حجم التخزين ومن عمليات الدخل/خرج.

كما يسمح SQL Server 2000 بتحديد الإجراء الذي يتوجب اتخاذه في حال جرى تجاوز شروط التكامل المرجعي للمعطيات، بدلاً من إظهار رسالة خطأ فقط كما في النسخ السابقة.

وقد جرى تعريف ثلاثة أنماط جديدة ضمن SQL server 2000 يمكن استخدامها في أماكن مختلفة وسيجري التعرّف عليها لاحقاً.

ما هو الجديد في SQL server 2000 (3)

- دعم لغة التأشير القابلة للتوسّع (XML)
- تحسينات في عمليات تعريف الفهارس
- تحسينات في آليات النسخ الاحتياطي وفي آليات استرجاع المعطيات
- تحسينات في آليات التحليل والتنقيب عن المعطيات

يدعم SQL server 2000 لغة التأشير القابلة للتوسّع (XML) حيث يتيح الإمكانيات التالية:

- إعادة نتيجة الاستعلام مباشرةً بصيغة XML؛
- استرجاع المعطيات من ملف XML كما لو أنها مخزنة في جدول SQL server؛
- الوصول إلى SQL server 2000 من خلال URL وباستخدام البروتوكول HTTP.

كما تضمَن SQL server 2000 تحسينات في تعريف الفهارس، شملت إمكانية تعريف فهارس على أعمدة الجداول، وإمكانية تحديد طريقة بناء الفهرس، بالإضافة إلى إمكانية ترتيب مفتاح الفهرس تصاعدياً أو تنازلياً. كما شملت التحسينات آليات النسخ الاحتياطية وآليات استرجاع المعطيات، بالإضافة إلى آليات التحليل والتنقيب عن المعطيات.

الفصل الثالث

عنوان الموضوع:

أدوات وبرامج SQL Server الخدمية (2)

.SQL Server Query Analyzer, SQL Debugger and SQL Profiler

الكلمات المفتاحية:

انظر ملف Glossary.doc المرفق.

ملخص:

نتناول في هذه الجلسة دراسة تفصيلية للأداة SQL Server Query Analyzer والأداة SQL Debugger المضمنة مع SQL server والمستخدمة لإدارة مهماته ووظائفه وتسهيل العمل في بيئته، بالإضافة إلى استعراض خصائص وميزات الأداة SQL Profiler.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- كيفية استخدام الأداة SQL Server Query Analyzer من أجل:
  - تحليل الاستعلامات
  - تنفيذ الاستعلامات وعرض النتائج
  - تصفح الأغراض
  - بناء المخطوطات

- كيفية استخدام الأداة SQL Debugger من أجل:
  - تنقيح المخطوطات
  - تنقيح الإجرائيات
  - تنقيح القوادح
  - تنقيح التوابع

- كيفية استخدام الأداة SQL Profiler من أجل:
  - تسجيل نشاطات المخدِّم
  - تحليل أداء الاستعلامات
  - تعقب الأحداث

- تتيح الأداة Enterprise Manager المُضَمنة مع SQL Server إمكانية إنشاء وإدارة قواعد المعطيات وأغراضها بالإضافة إلى إدارة المخدم ككل ومعالجة وظائفه. على كل حال، لاينفي وجود مثل هذه الأداة الحاجة من وقت إلى آخر، لكتابة استعلامات بلغة SQL أو تنفيذها بشكل مباشر

- كما تتيح الأداة Enterprise Manager إمكانية إنشاء أو تعديل الإجرائيات أو التوابع المعرّفة، إلا أن واجهة التحرير المستخدمة غير مريحة وغير سهلة الاستخدام كما أنها لا تقدم الخدمات المتوقعة، خاصة تلك التي تتيح عمليات بحث واستبدال أو فتح وتخزين الرماز المكتوب كما لاتوفر إمكانية تنفيذ ذلك الرماز

- تقدم الأداة SQL Query Analyzer خصائص متعددة كإمكانية تحليل الاستعلامات أو تحويلها إلى استعلامات تفاعلية وإمكانية تنقيحها، كما يمكن تطبيق الخصائص السابقة على الإجرائيات أو التوابع أو القوادح

SQL Server Query Analyzer

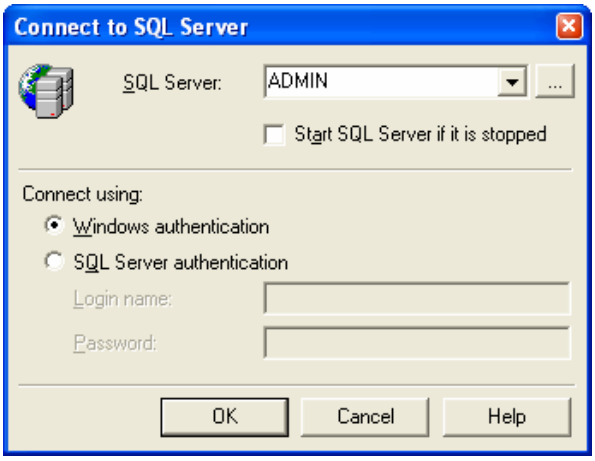
- تعريف
- تأسيس الاتصالات
- تحرير الاستعلامات
- تنفيذ الاستعلامات وعرض النتائج

- تعريف:

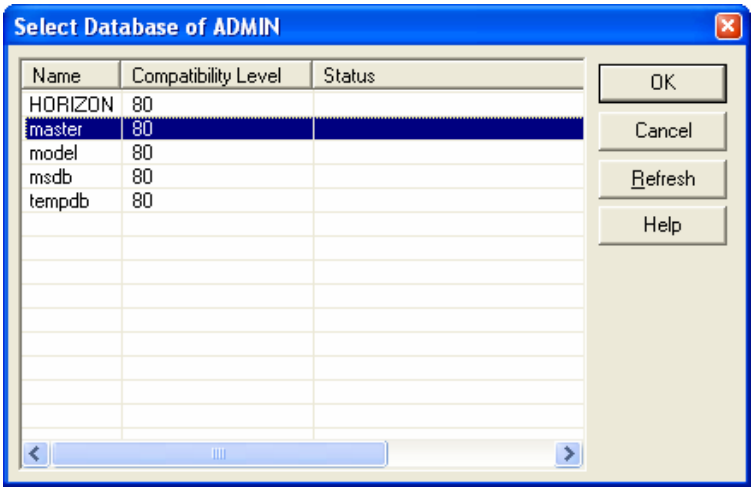
تتميز الأداة SQL Query Analyzer بسهولة الاستخدام وبساطته، كما تتعدى مهامها مجرد تحرير أو تنفيذ الاستعلامات لتصل إلى إمكانية تحليل الاستعلام بحد ذاته ومتابعته وتنقيحه، هذا بالإضافة إلى خدمات أخرى وخصائص تساهم في إدارة قواعد معطيات المخدم والأغراض التابعة لها.

- تأسيس الاتصالات:

ينبغي عند بدء تشغيل الأداة Query Analyzer أن يتم إنشاء اتصال مع المخدِّم الذي يجري التخاطب معه. يمثل الشكل التالي الواجهة التي يجري من خلالها تحديد معاملات الاتصال كإسم المخدم ونوع استراتيجية الاتصال، بالإضافة إلى خيارات أخرى كعرض قائمة بآخر الاتصالات التي تم إجراءها، أو خيار إجراء اتصال مباشر حالما يتم إغلاق الاتصال مع المخدم.



يجري –وبشكل افتراضي– فتح الاتصال مع قاعدة المعطيات التي سبق وجرى الدخول إليها من خلال SQL Server، مع العلم أنه يمكننا تغيير قاعدة المعطيات تلك من خلال اختيار "Change Database" من القائمة "Query" كما يوضح الشكل التالي:



- تحرير الاستعلامات:

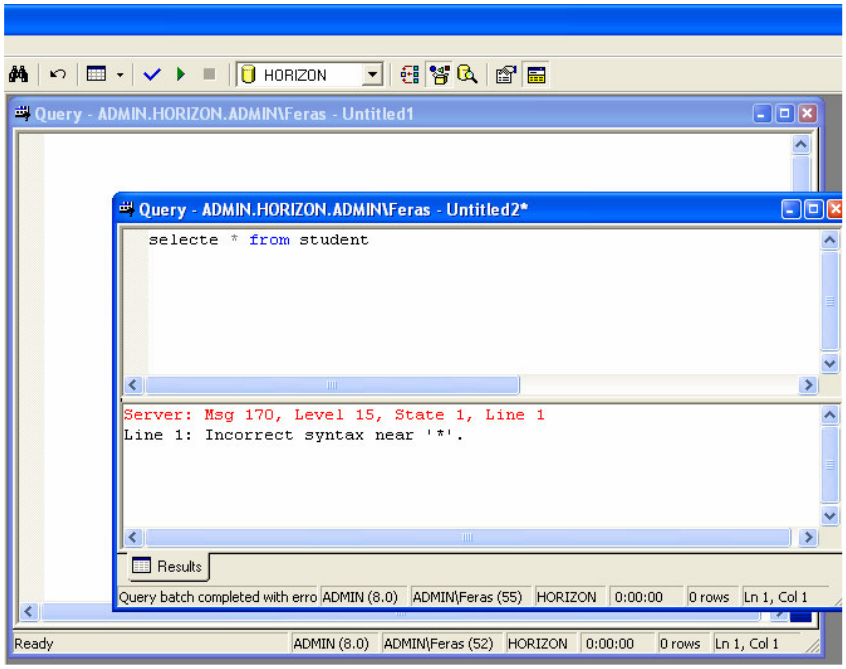
تُعتبر إمكانية تحرير الاستعلامات من أهم الميزات التي تتمتع بها الأداة SQL Query Analyzer لما تقدمه من تسهيلات في كتابة الاستعلام أو عرضه سواء كان ذلك من خلال شكل الرماز وعرض مفرداته القواعدية بأسلوب ملون يسهل التعامل، أو من خلال إتاحة إمكانيات التحرير المعروفة من قص ونسخ ولصق للنصوص بالإضافة إلى إمكانيات البحث والاستبدال والتراجع والانتقال إلى سطر معين ووضع إشارات مرجعية في النص؛

تقدم الأداة أيضا تسهيلات من نمط إمكانية حفظ مخطوطات SQL ضمن ملفات، أو فتح تلك الملفات المُخزنة ومعالجتها؛

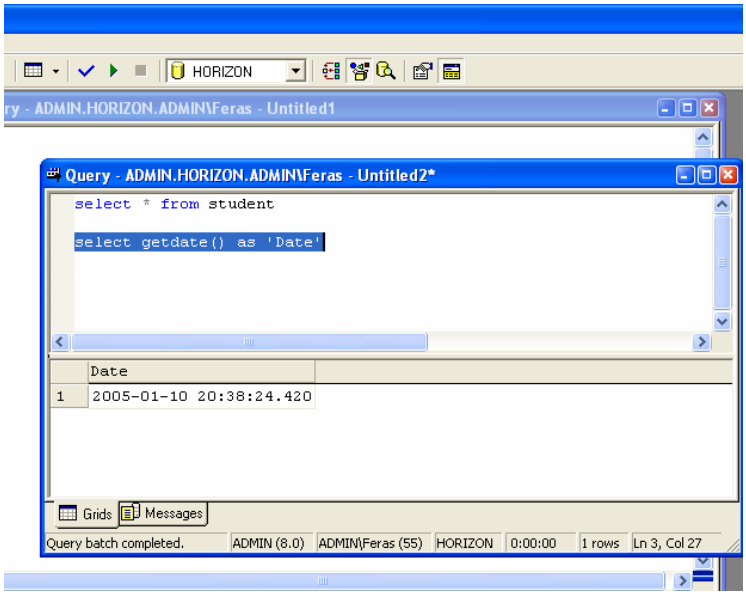
- تنفيذ الاستعلامات وعرض النتائج:

يمكننا بالإضافة إلى تحرير الاستعلامات أن نتأكد من صحتها وأن ننفذها باستخدام الأداة Query Analyzer؛

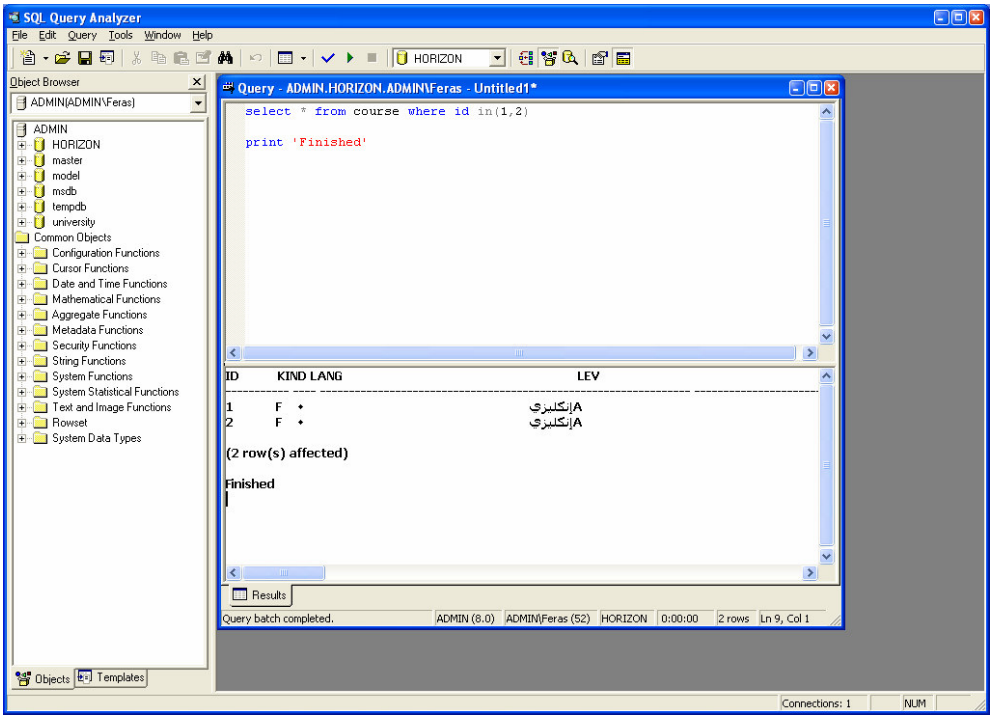
يجري التأكد من صحة الاستعلام قواعدياً إما من خلال اختيار "Parse" من قائمة "Query"، أو بالضغط على الزر المخصص لهذه العملية من شريط الأدوات، أو بتنفيذ الاختصار Ctrl+F5، يجري بعد ذلك تفسير مخطوطات SQL المكتوبة في كامل الواجهة الفعالة أو المخطوطات المحددة فقط، كما تظهر رسالة مناسبة تبين خلو الاستعلام من الأخطاء أو توضح نوع الخطأ المرتكب كما في الشكل التالي:



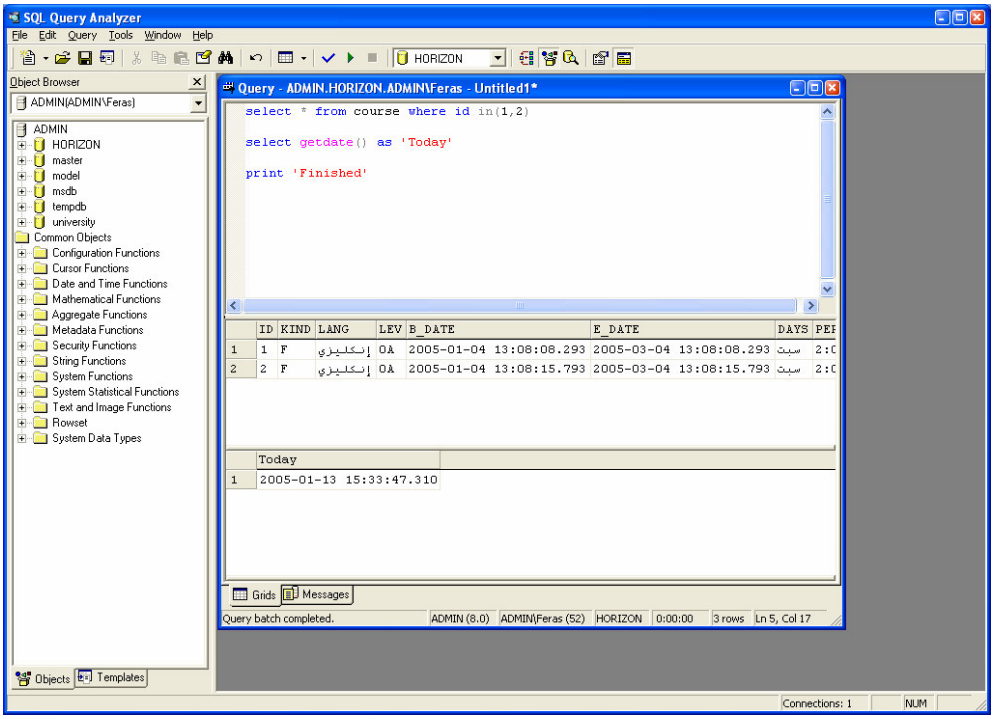
يمكننا تنفيذ الاستعلامات إما من خلال اختيار "Execute" من قائمة "Query" أو بالضغط على الزر المخصص لهذه العملية من شريط الأدوات أو بتنفيذ الاختصار F5 أو Ctrl+E، يجري بعد ذلك تنفيذ مخطوطات SQL المكتوبة في كامل الواجهة الفعالة أو المخطوطات المحددة فقط، كما تظهر رسالة تحتوي على نتائج الاستعلامات التي جرى تنفيذها؛



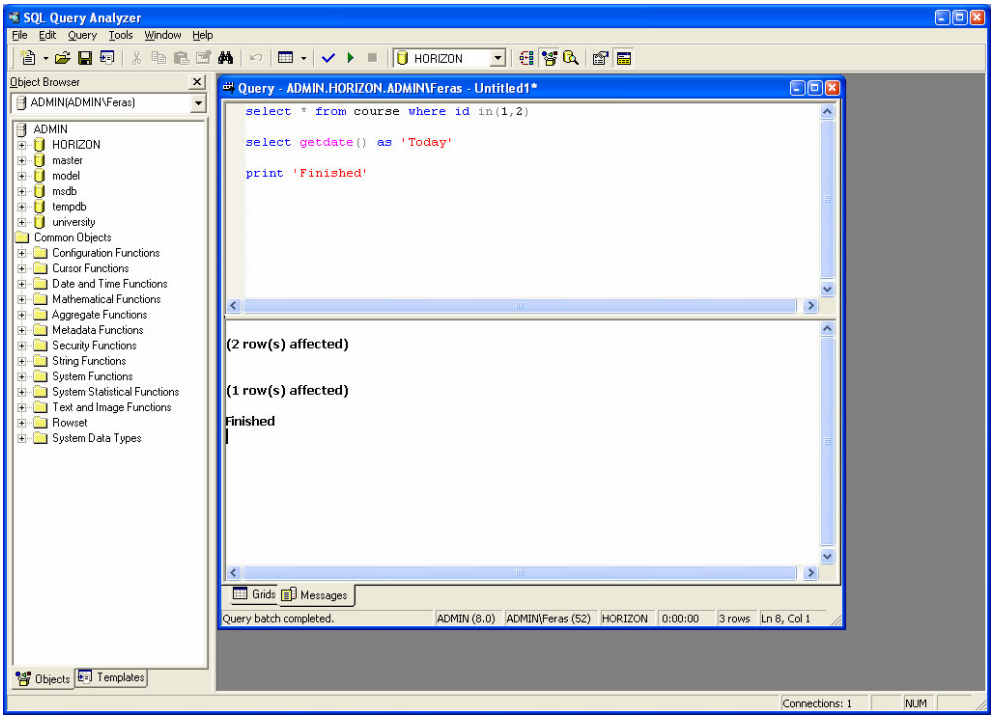
يمكن عرض النتائج بطريقتين، إما نصّية أو من خلال جدول خلايا. يعتبر العرض النصي للنتائج، الأسلوب المفضل عندما تحتوي الإجرائية على رسائل خاصة مكتوبة بالتعليمة "PRINT"، أو على رسائل خطأ، كما يمكننا اعتماد هذا الأسلوب في عرض النتائج عندما نرغب بنسخ الخرج إلى محرر نصوص معين، أو عندما يكون الخرج كبير جداً بحيث يصعب عرضه على شكل جدول؛ يوضح الشكل التالي خرج استعلام معين بأسلوب نصي:



تُستخدم طريقة العرض بأسلوب "جدول خلايا" عندما نرغب بعرض النتائج بشكل مرتب، بحيث تظهر كل نتيجة استعلام تحتوي على معطيات ضمن جدول خلايا خاص في واجهة النتائج ويمكننا تصفح النتائج في كل جدول من تلك الجداول، كما في الشكل التالي:



يعتبر هذا الأسلوب مفيداً عندما نرغب بحفظ نتيجة الاستعلام في ملفات منسقة أو عندما نرغب بنسخ ولصق النتائج إلى محرر خلايا معين، إلا أن السيئة الوحيدة في هذا الأسلوب تكمن في أننا لا نستطيع معاينة الرسائل التي يمكن عرضها من قبل المخطوطات المنفذة على نفس الواجهة إذ أنها تعرض في واجهة خاصة يطلق عليها اسم واجهة الرسائل، كما يبين الشكل التالي:



يمكننا التنقل بين أسلوب العرض النصي للنتائج وأسلوب العرض بشكل جدول خلايا من خلال الضغط على الاختصار Ctrl+T و Ctrl+D على الترتيب، أو من خلال اختيار الخيار المناسب من قائمة المهمات السريعة لواجهة تحرير الاستعلامات في الأداة Query Analyzer؛

يمكننا حفظ نتائج الاستعلامات في ملفات نصية أو ملفات تحرير خلايا –في حالة عرض النتائج على هيئة جدول خلايا–، بحيث ينبغي أولا أن نحدد الواجهة التي نريد حفظها وبالضغط على رمز الحفظ من شريط الأدوات يمكننا أن نحفظ النتائج. يجدر هنا بالذكر أن نوضح أن خرج الاستعلامات يمكن حفظه مع وضع فواصل بين الأعمدة.

- تصفح الأغراض
- قوالب مخطوطات SQL المضمنة
- البحث عن الأغراض
- اختصارات المستخدم المعرّفة
- أدوات المستخدم المعرّفة

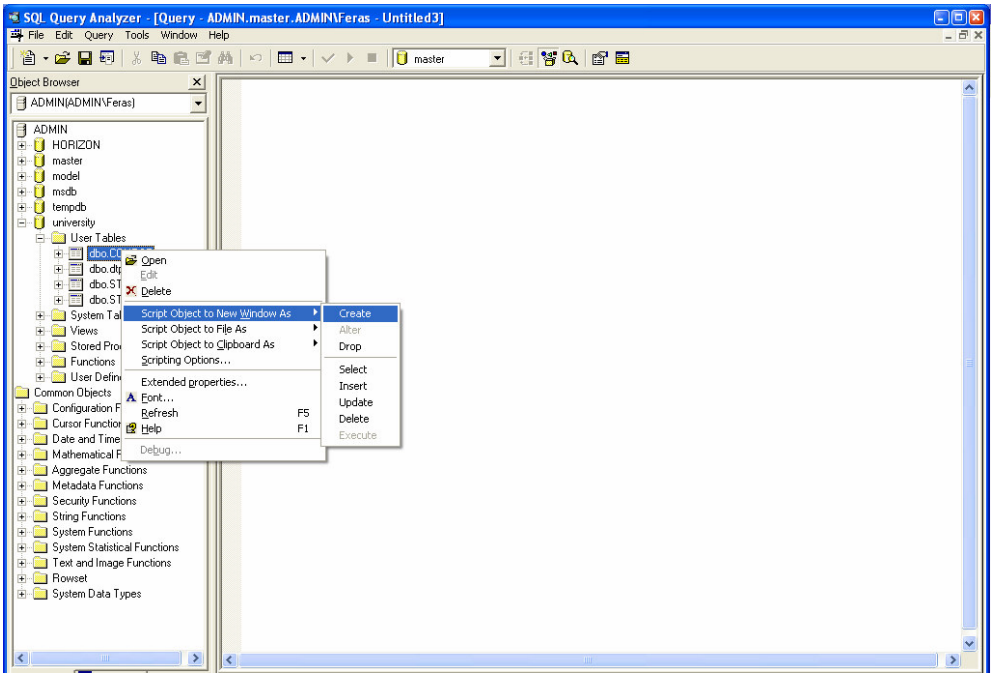
- تصفح الأغراض:

تعد ميزة تصفح الأغراض في الأداة SQL Query Analyzer من أهم الخصائص المميزة لها، بحيث يجري عرض قائمة بكافة الأغراض التي يحتويها SQL Server من مخدمات أو قواعد معطيات أو إجرائيات أو توابع معرّفة وغيرها؛

يجري تشغيل واجهة مستعرض الأغراض من خلال الزر المخصص لها في شريط أدوات SQL Query Analyzer أو بالضغط على المفتاح F8؛

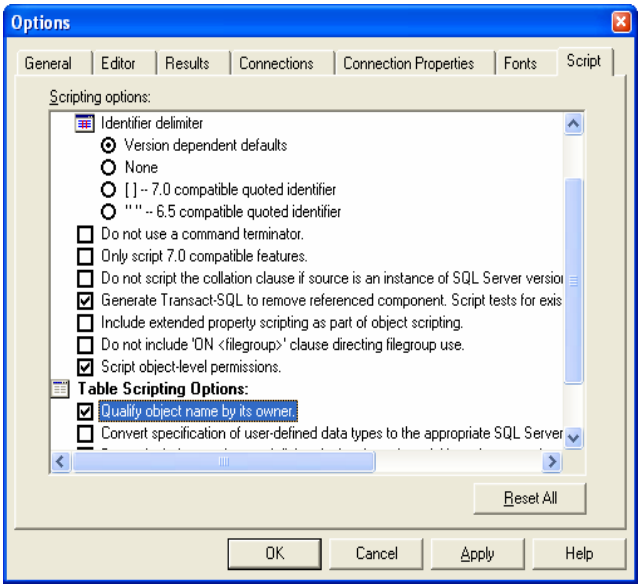
ما أن يتم فتح متصفح الأغراض حتى يجري إنشاء اتصال جديد مع SQL Server يختلف عن بقية الاتصالات المنشأة في واجهات الاستعلامات الأخرى، كما يجري إنشاء اتصال منفصل عند كل استعراض لغرض على مخدم مختلف، يمكننا أن نُبدّل ما بين الاتصالات المنشأة من خلال تغيير المخدم من القائمة التي تظهر في أعلى واجهة متصفح الأغراض؛

بالإضافة إلى إمكانيات عرض كافة الأغراض في SQL Server، فإن متصفح الأغراض يؤمن للمطور ميزات أخرى تتلخص في إمكانيات استخراج مخطوطات تلك الأغراض، بالإضافة إلى إمكانيات توليد الرماز لبناء أو تعديل أو حذف أياً منها كما يوضح الشكل التالي:



يمكننا أن نُعرِّف بعض الإعدادات التي تتعلق بكيفية توليد المخطوطات وبعض خصائصها من خلال اختيار "Script Options"، بحيث تظهر الواجهة الموضحة بالشكل التالي والتي نحدد من خلالها الخصائص التالية:

- ✓ تضمين ترويسات وصفية في المخطوطات
- ✓ تضمن اختبارات وجود الأغراض قبل إنشاءها
- ✓ تضمين المخطوطات اللازمة لإزالة الأغراض –في حال وجودها– قبل إنشائها من جديد
- ✓ تضمين المخطوطات التي تحدد نوع السماحيات ومستواها
- ✓ تضمين اسم مالك الغرض مع المخطوطات المنشأة

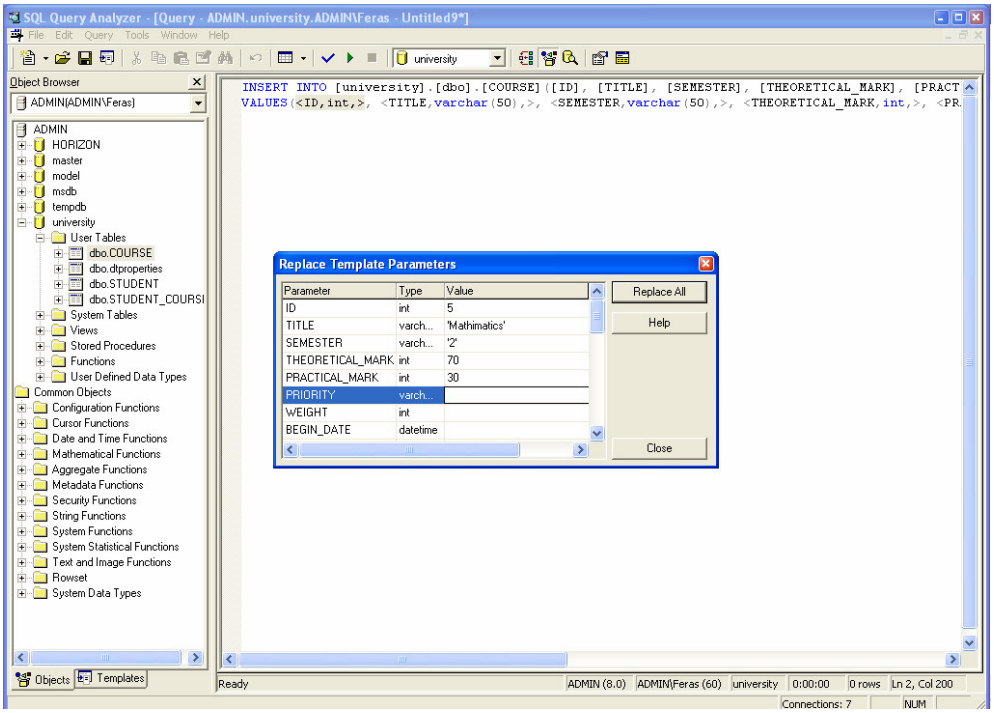


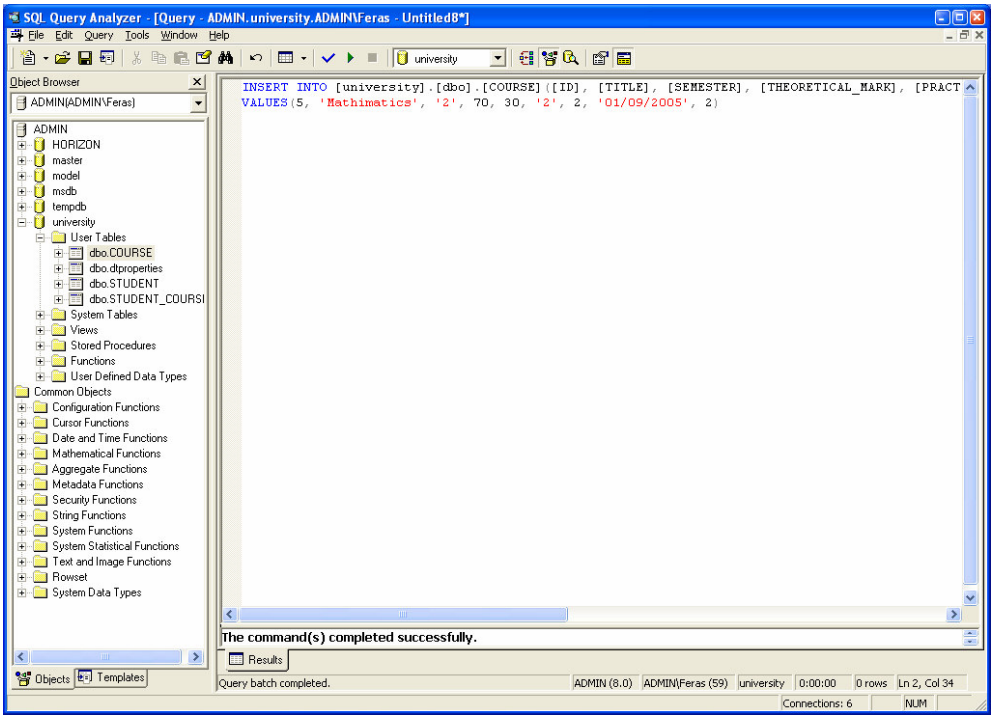
فيما يلي عرض للمخطوط المنشأ الناتج:

```
/****** Object:  Table [dbo].[COURSE]    Script Date: 13/01/2005 04:29:55 م ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[COURSE]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[COURSE]
GO
if not exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[COURSE]')
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
 BEGIN
CREATE TABLE [dbo].[COURSE] (
        [ID] [int] NOT NULL ,
        [TITLE] [varchar] (50) COLLATE Arabic_CI_AI NULL ,
        [SEMESTER] [varchar] (50) COLLATE Arabic_CI_AI NULL ,
        [THEORETICAL_MARK] [int] NULL ,
        [PRACTICAL_MARK] [int] NULL ,
        [PRIORITY] [varchar] (50) COLLATE Arabic_CI_AI NULL ,
        [WEIGHT] [int] NULL ,
        [BEGIN_DATE] [datetime] NULL ,
        [SESSION_PERIOD] [int] NULL ,
        CONSTRAINT [PK_COURSE] PRIMARY KEY  CLUSTERED
        (
                [ID]
        )  ON [PRIMARY]
) ON [PRIMARY]
END
GO
```

كما يمكننا إنشاء مخطوطات DDL من خلال مستعرض الأغراض كما لاحظنا، ويمكننا أيضاً إنشاء مخطوطات DML، كعرض لقوالب تعليمات Select أو Insert أو Update أو Delete أو حتى تنفيذ الإجرائيات والتوابع Exec.

تتميز هذه القوالب بخصائص تسهل عمليات تحرير الاستعلامات، فكما يوضح المثال التالي، يمكننا أن نولد مخطوط لإجراء عملية INSERT إلى أحد الجداول، ثم نختار من قائمة "Edit" "Replace Template Parameters" ونقوم بإدخال قيم العملية من خلال واجهة تخاطبية مريحة:

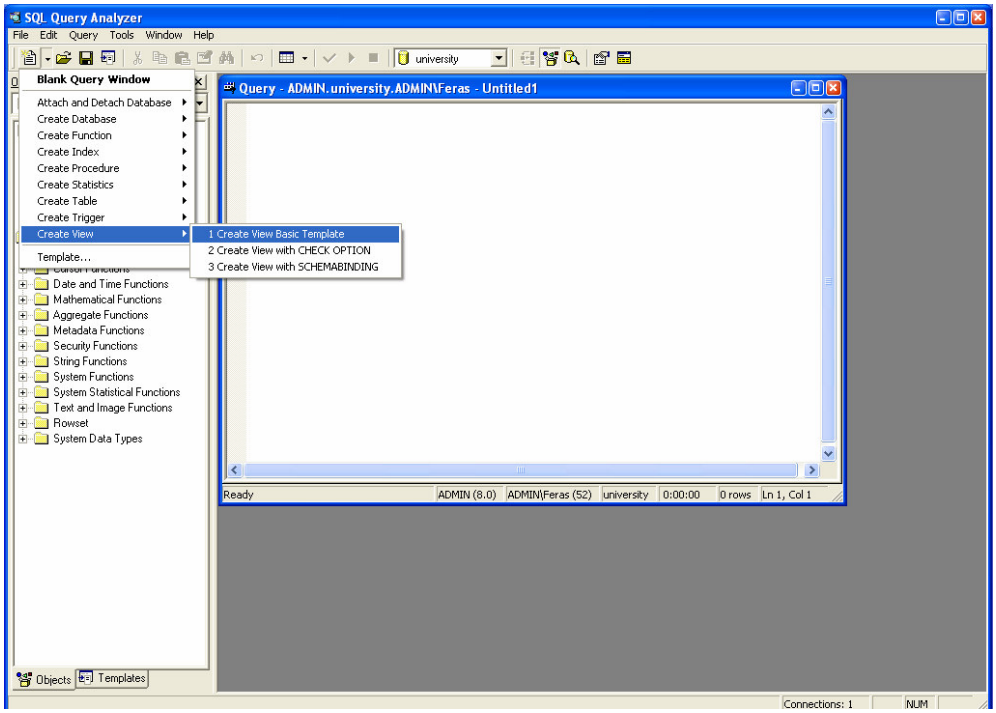




- قوالب مخطوطات SQL المضمنة:

تؤمن الأداة SQL Query Analyzer –بالإضافة إلى إمكانيات توليد مخطوطات DML– إمكانيات استخدام قوالب مخطوطات مسبقة التعريف لتنفيذ مهمات مختلفة كبناء جداول أو إجرائيات أو توابع أو قاعدة معطيات أو غيرها؛

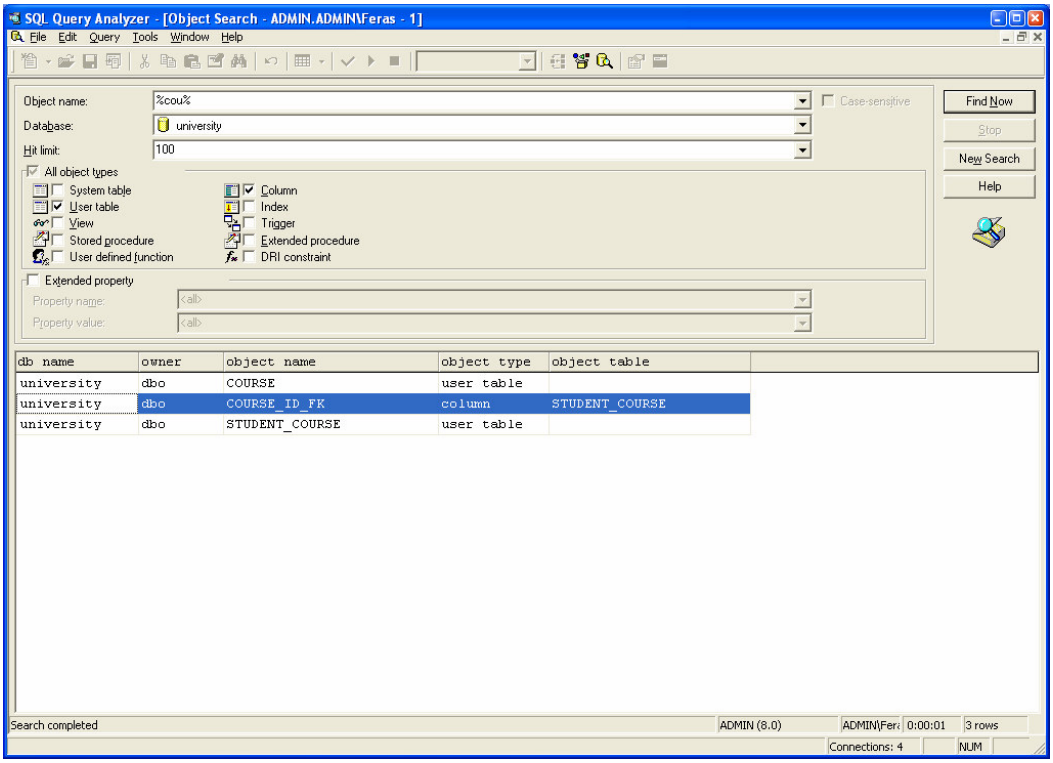
بإمكاننا فتح أو اختيار تلك القوالب من القائمة الملحقة برمز "ملف جديد" في شريط الأدوات كما يوضح الشكل التالي، أو من خلال واجهة "Templates" التي يمكن عرضها من أسفل واجهة مستعرض الأغراض:



كما يمكننا أن نحرر القالب الناتج يدوياً أو من خلال خصائص التحرير التي استعرضناها في الفقرة السابقة من حيث إضافة الاستعلامات أو استبدال المعاملات، بالإضافة إلى أننا نستطيع إنشاء قوالب جديدة وتخزينها لاستعمالها في وقت لاحق.

- البحث عن الأغراض:

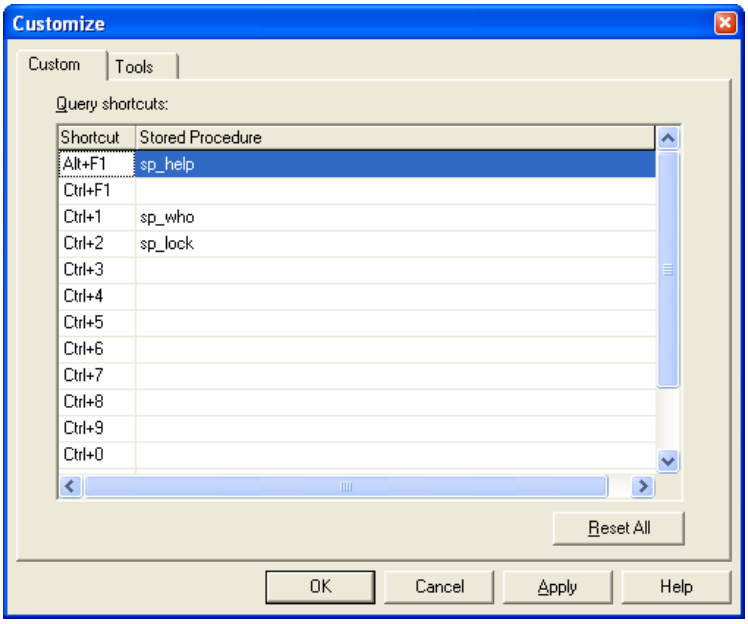
تزودنا أيضا الأداة SQL Query Analyzer بإمكانيات بحث فعّالة عن أمثال الأغراض المختلفة المتواجدة في SQL Server، يمكننا تشغيل واجهة البحث تلك من خلال قائمة "Tools" ثم "Object Search" ثم "New"، أو بالضغط مباشرةً على المفتاح F4، لتظهر بعد ذلك الواجهة الموضحة بالشكل التالي، والتي يمكننا من خلالها اختيار محددات ومعايير البحث التي نريدها:



تُسهّل علينا الواجهة السابقة عمليات تصفح الأمثال في SQL Server خاصةً عندما يزداد عددها بحيث يصعب البحث عنها من خلال مستعرض الأغراض.

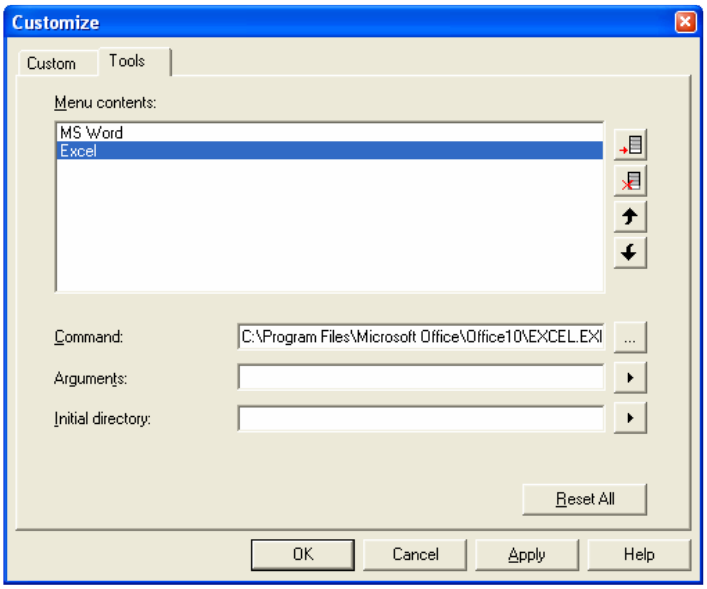
- اختصارات المُستخدم المعرّفة:

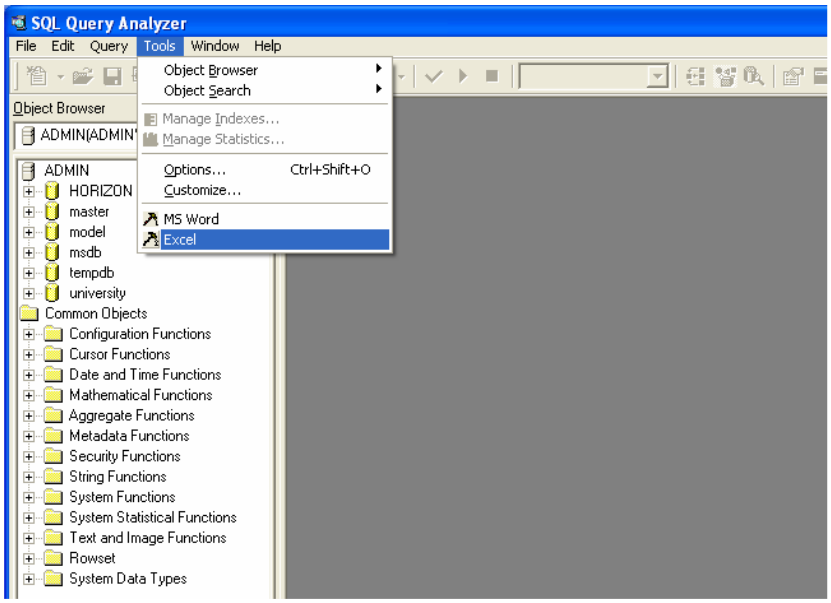
تزودنا الأداة SQL Query Analyzer بإمكانيات استعراض الاختصارات الموجودة أو تعريف اختصارات جديدة للعمليات التي نقوم بتنفيذها بشكل متواتر، فعلى سبيل المثال يؤي تنفيذ الاختصار Ctrl+1 إلى تنفيذ الإجرائية sp_who التي تعرض معلومات المستخدم الحالي، و يؤي تنفيذ الاختصار Alt+F1 إلى تنفيذ الإجرائية sp_help التي تعرض معلومات حول كافة الأغراض المتوافرة. يمكننا ببساطة تعديل تلك الاختصارات أو إضافة اختصارات جديدة من واجهة "Cutomize" الموضحة بالشكل التالي والتي يمكن الوصول إليها من قائمة "Tools":



- أدوات المستخدم المعرّفة:

تزودنا الأداة SQL Query Analyzer بإمكانيات إضافة أدوات وتعليمات جديدة إلى القائمة "Tools"، يتم ذلك من خلال الواجهة الفرعية "Tools" التي يمكن الوصول إليها من واجهة "Customize" السابقة، ويمكننا من خلالها تعريف تعليمات جديدة يمكــن من خلالها تشغيل تطبيقات جديدة.





SQL Debugger

- مقدمة
- تعريف
- الأساليب السابقة التي كانت تستخدم لتنقيح مخطوطات SQL.
- يعد منقِّح الاستعلامات SQL Debugger أحد أهم الأدوات المضمنة مع SQL Query Analyzer، وذلك لما يقدمه من إمكانيات متعددة في معالجة الاستعلامات المكتوبة بلغة SQL وخاصة الاستعلامات المعقدة منها؛
- يمكننا تشبيه الأداة SQL Debugger بأي منقِّح رماز في أية لغة برمجة، فكما نحتاج أثناء كتابة الرماز بلغة C++ –على سبيل المثال– لأن نقوم باختبار وتنقيح الرماز الذي نكتبه والتأكد من صحته أو مراقبة القيم المنطقية في بعض مواقعه أثناء التنفيذ، كذلك نحتاج لأداة مماثلة لتطبيق المهمات السابقة على المخطوطات المكتوبة بلغة SQL؛
- يمكننا الآن ومن خلال الأداة SQL Debugger أن نتابع أسلوب سير تنفيذ إجرائية معينة ببساطة وسهوله، في حين كانت هذه العملية – فيما مضى– تتطلب الكثير من الجهد والعناء، فقد كانت الطريقة المُتَّبعة لتنقيح المخطوطات المكتوبة في الإجرائيات أو التوابع تقتصر على إعادة كتابة بعض تعليمات SQL المكوِّنة لجزء من تلك الإجرائية وتنفيذها على نحو مستقل، وهو ما كان يسبب فقدان قيم المتحولات المحلية المعرَّفة والمستخدمة، وكان يدفعنا إلى تعريف المتحولات وإسناد قيم يدوية لها لاستخدامها في التنقيح؛ هذا بالإضافة إلى اقتصار إمكانية ملاحقة سير عمل إجرائية على استخدام الرسائل وطباعة العبارات التي تعرض الحالات المختلفة للإجرائية أو قيم بعض المعاملات أو المتحولات المحلية فيها.

SQL Debugger
تطور الأداة

- بدايات منقِّح استعلامات SQL
- المشاكل التي عانى منها
- الشكل الحالي لـــ SQL Debugger

• مرَّ منقِّح استعلامات SQL Server بالعديد من الأطوار حتى وصل إلى وضعه الحالي، فقد كانت البداية من خلال أداة خاصة يجري تنصيبها إلى جانب SQL Server وتنحصر مهماتها في تنقيح الاستعلامات، إلا أن استخدامها لم يكن سهلاً بما فيه الكفاية، فقد كانت تتطلب وجود العديد من التطبيقات الأخرى إلى جانبها لكي تعمل بالشكل المناسب، بالإضافة إلى حاجتها لبيئة خاصة تعمل عليها سواء كانت من نظام تشغيل أو إصدارات معينة من SQL Server أو حتى إصدارات معينة من Visual Studio، بالتالي ظهرت مشاكل مختلفة أثرت على استخداميّة هذه الأداة؛

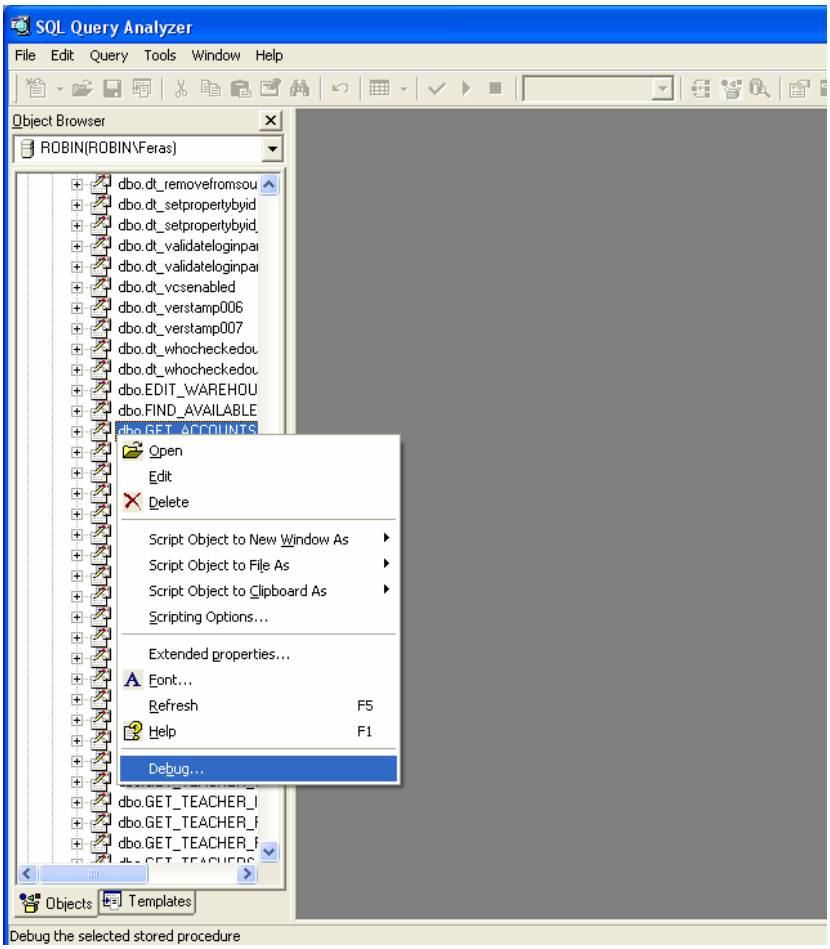
• واجهت أدوات تنقيح الاستعلامات أثناء تطورها العديد من المشاكل، بسبب نفور المستخدم منها خاصة مع العبء الإضافي المترتب على استخدامها، حيث رغب المطورون بوجود أداة سهلة الاستخدام والتنصيب مضمنة مع أدوات SQL Server ومتكاملة مع SQL Query Analyzer، وهذا ما تحقق من خلال الأداة SQL Debugger؛

• تقدم حالياً الأداة SQL Debugger المُضمَنة مع SQL Query Analyzer العديد من التسهيلات المتعلقة بتنقيح رماز الإجرائيات فيما يتعلق بمتابعة سير التعليمات أو إعداد نقاط التوقف أو عرض محتوى المتحولات المحلية أو العامة أو حتى الحصول على نتائج تنفيذ العبارات المكتوبة بلغة SQL والمُضمَنة في الإجرائية قيد المعالجة.

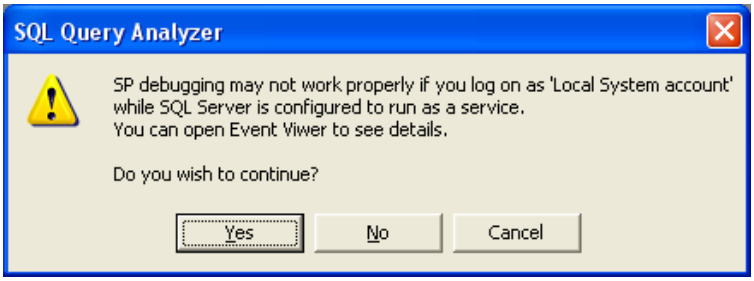
SQL Debugger
تشغيل الأداة

- تشغيل الأداة
- المشاكل التي يمكن أن تواجهنا أثناء التشغيل
- تغيير حساب الدخول إلى المخدِّم

- لا تعتبر طريقة تشغيل الأداة SQL Debugger بديهية بما فيه الكفاية، فلكي نقوم بتنقيح إجرائية معينة ينبغي أن نطبق الخطوات التالية:
  - نشغل أولاً الأداة SQL Query Analyzer
  - نقوم بعد ذلك بعرض شجرة متصفح الأغراض (يمكننا عرض تلك الشجرة من خلال الضغط على الزر F8)
  - نختار بعد ذلك الإجرائية التي نرغب بتنقيح رمازها، ثم نضغط عليها بالزر اليميني للفأرة ونختار "Debug" كما يوضح الشكل التالي:

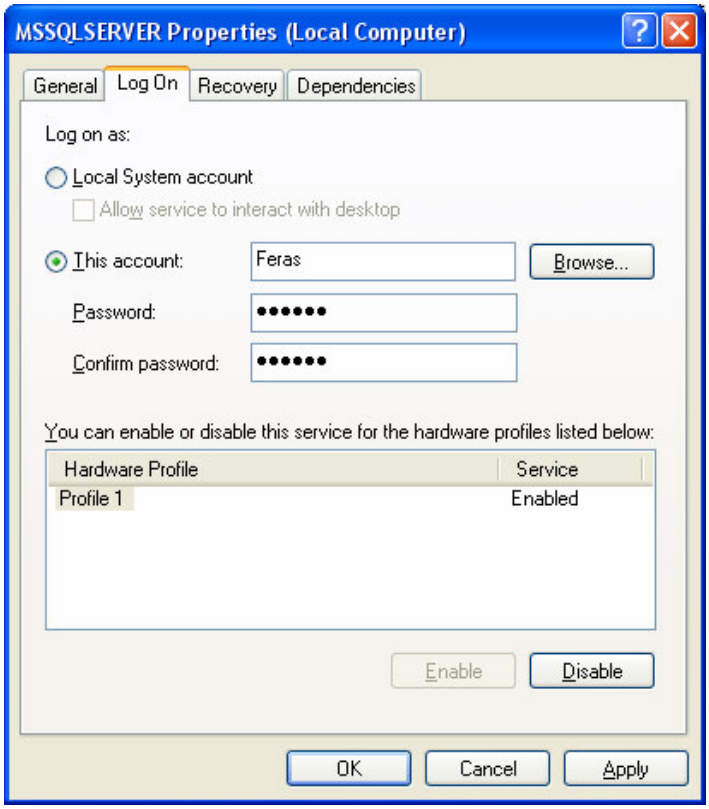


- تواجه المطور الذي يحاول تشغيل الأداة SQL Debugger والذي يعمل ضمن بيئة محلية –أي على نفس جهاز المخدم وليس على الشبكة–، أو بشكل عام، المطور الذي يستخدم حساب النظام المحلي، المشكلة الأكثر شيوعاً التي يمكن التعرض لها، فما أن يضغط على زر "Debug" حتى تظهر له الرسالة الموضحة بالشكل التالي:



والتي تحذر من أن الاستمرار في استخدام SQL Debugger ضمن حساب النظام المحلي قد يؤدي إلى نتائج غير متوقعة، وأنه من المفضل أن يتم تشغيل هذه الأداة باستخدام حساب آخر.

- يمكننا تجنب المشكلة السابقة من خلال استخدام حساب آخر للدخول إلى المخدم، سنقوم من خلال الخطوات التالية بتغيير حساب الدخول:
  - أولاً ومن سطح المكتب في نظام تشغيل Windows، نضغط بالزر اليميني للفأرة على "My Computer" ونختار "Manage"
  - نختار من شجرة "Computer Management" المجلد "Services and Applications" ثم "Services"؛
  - نختار "MSSQLSERVER" من قائمة الخدمات الموجودة وننقر عليها بالفأرة مرتين لفتح خصائصها؛
  - من قائمة "Log On" نختار "This account" ونملئ الحقول باسم دخول وكلمة سر المستخدم الذي يمثل المدير "Administrator" كما في الشكل التالي:



ينبغي بعد ذلك إعادة تشغيل هذه الخدمة "MSSQLSERVER" لكي يتم أخذ التغيرات –التي تم إعدادها– بعين الاعتبار.

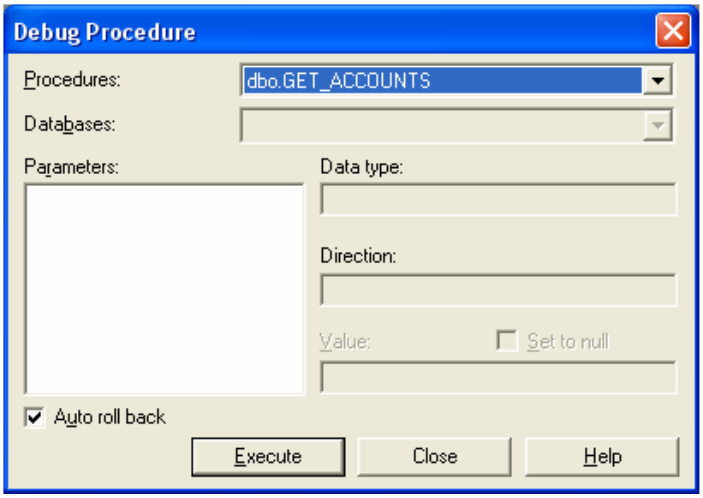
SQL Debugger

استخدام الأداة

- الإعدادات الافتراضية
- خصائص المنقِّح وأدواته

- أن يجري تشغيل الأداة SQL Debugger من أجل تنقيح إجرائية مختارة حتى يتم إظهار واجهة خاصة تطلب من المطور تحديد معاملات خاصة بالمنقح
- يوضح الشكل التالي مثالاً عن الواجهة التي نتحدث عنها، بحيث نلاحظ فيها اسم الإجرائية بالإضافة إلى مجموعة المعاملات التي يمكن استخدامها كمعاملات دخل، بالإضافة إلى مربع اختيار خاص وهو مربع "Auto roll back" يكون مختاراً بشكل افتراضي، ومهمته التراجع عن التعديلات التي تم إنجازها أثناء قيام SQL Debugger بعمله؛ تسمح لنا هذه الخاصة باختبار

وتنقيح إجرائيات تقوم بتعديلات في المعطيات دون تخزين تلك التعديلات فعلياً، وبالتالي يمكن تنقيح الإجرائية مراراً وتكراراً على نفس المعطيات في كل مرة، بحيث ينفذ SQL Debugger تعليمة "BEGIN TRANSACTION" قبل بدء تنفيذ الإجرائية، ثم ينفذ تعليمة "ROLLBACK TRANSACTION" بعد إيقافه. مع العلم أن عدم اختيار "Auto roll back" يدفع بالمنقِّح إلـــى تخزين التعديلات أثناء العملية، أي وكأننا قمنا بتنفيذ الإجرائية بشكل اعتيادي



تحذير

ينبغي توخّي الحذر عندما نقوم بتنقيح إجرائية ما تقوم بتعديل المعطيات بينما يقوم مستخدمون أو إجراءات في النظام بالولوج إلـــى تلك المعطيات أو تنفيذ استعلامات معينة، فالأقفال المفروضة على المعطيات من قبـــل الأداة SQL Debugger ســـتبقى إلـــى أن يجري إغلاق الأداة والتراجع عن التعديلات، مما قد يحرم بعض المستخدمين من الولوج إلى المعطيات أو يؤثر على أداء النظام. بالتالي ينبغي إجراء عملية التنقيح على معطيات تجريبية أو في بيئة تطوير ملائمة لا تؤخذ فيها مسألة التأثير على أداء النظام بعين الاعتبار.

- ما أن يتم الضغط على زر "Execute" في الواجهة السابقة حتى يجري تشغيل الواجهة الخاصة بالأداة SQL Debugger، بحيث تظهر مخطوطات SQL المكوِّنة للإجرائية مع سهم أصفر يشير إلى السطر الأول، واعتباراً من هذه النقطة يمكننا أن ننفـــذ الإجرائية خطوة بخطوة أو أن نُعد نقاط توقف خاصة لننتقل بالتنفيذ إليها

- يمكننا التحكم بتنفيذ كافة العمليات المتاحة من قبل SQL Debugger إما من خلال شريط الأدوات الخاص به أو من خـــلال قائمة المهمات السريعة المضمنة فيه أو من خلال اختصارات لوحة المفاتيح، وفيما يلي عرض لجدول يوضح كافة مهمـــات هـــذه الأداة:

التعليمة	الاختصار	الوصف
Go	F5	تنفيذ الإجرائية في وضع التنقيح إلى أول ورود لنقطة توقف أو إلى آخر الإجرائية.

Toggle breakpoi	F9	نقطة توقف على سطر محدد، مع العلم أنه لا يمكن وضع نقطة توقف على الأسطر الفارغة أو التي لا تتضمن رماز تنفيذي كأسطر التعليقات على سبيل المثال.
Remove All	Ctrl+Shift+F9	لإزالة جميع نقاط التوقف من الواجهة المختارة.
Step Into	F11	ل يتوضّع المنقِّح عندها، مع العلم أنه في حال صادف المنقِّح لاستدعاء لإجرائية أو تابع أو قادح أثناء التنفيذ، فإنه سينتقل لتنفيذ رماز المُستَدعِي.
Step Over	F10	ل يتوضّع المنقِّح عندها، مع العلم أنه في حال صادف المنقِّح لاستدعاء لإجرائية أو تابع أو قادح إنه سيقوم بتنفيذه ككتلة وحيدة دون الانتقال إليه وينتقل إلى التعليمة التالية من الإجرائية الحالية.
Step Out	Shift+F11	ل يتوضّع المنقِّح عندها، مع العلم أنه في حال صادف المنقِّح لاستدعاء لإجرائية أو تابع أو قادح أثناء التنفيذ، فإنه سينتقل إلى التعليمة التالية بعد تنفيذ رماز المُستَدعِي ولكن دون الانتقال إليه.
Run to Cursor	Ctrl + F10	تنفيذ الإجرائية في وضع التنقيح إلى أول ورود لخازن احتياطي أو إلى آخر الإجرائية.
Restart	Ctrl+shift+F5	لإعادة تنفيذ الإجرائية اعتباراً من البداية بعد إهمال التنفيذ الأخير.
Stop Debugging	Shift+F5	لإيقاف التنفيذ الحالي للمنقِّح.
Auto Rollback		أو إيقاف خاصة التراجع التلقائي عن التعديلات التي تجريها الإجرائية وهي في وضع التنقيح.

SQL Debugger

استخدام الأداة

- واجهات المنقِّح
- شريط حالة المنقِّح
- تتألف واجهات المنقِّح من خمسة أجزاء رئيسية، وهي:

o واجهة الرماز المصدري: وهي الواجهة العُلوية التي يظهر فيها الرماز المصدري للإجرائية أو التابع أو القادح قيد التنقيح، كما أنها الواجهة التي تعرض كيفية سير التنفيذ وكيفية التحكم بوضع نقاط التوقف. كما ويجدر بالذكر هنا أن إجراء أي تعديل على رماز الإجرائية بعد فتح SQL Debugger لن يؤدي إلى تعديل محتوى الرماز فيه، وبالتالي نحتاج لإعادة تشغيل المنقِّح في كل مرّة نرغب فيها بتحديث رماز الإجرائية أو التابع أو القادح قيد التنقيح؛

o واجهة المتحولات المحلية: وهي الواجهة التي تُظهر اسم وقيمة ونمط كل متحول محلي أو معامل دخل أو خرج مستخدم في المجال الحالي للإجرائية أو التابع أو القادح. لا يمكن إضافة أو حذف أي متحول من هذه الواجهة في حين يمكننا أن نغير في القيم التي يأخذها ذلك المتحول، مما يفيد في إجراء عدة اختبارات وتقييمات لقيم مختلفة لتلك المتغيرات أثناء التنفيذ؛

- واجهة التوابع العامة: وهي الواجهة التي تظهر فيها القيم الناتجة عن التوابع العامة والتي تعتمد على حالة التنفيذ الحالية للإجرائية. لا يمكن هنا أن نغير في القيم التي تأخذها تلك التوابع في حين يمكننا إضافة المزيد من التوابع ومراقبة تغيرات نتائجها، فعلى سبيل المثال، يمكننا إضافة التابع @@ROWCOUNT أو التابع @@IDENTITY أو التابع @@ERROR أو التابع @@NESTLEVEL؛
- واجهة استعراض المكدس: وهي الواجهة التي تعرض قائمة بالاستدعاءات المفتوحة، بحيث يتم عرض آخر استدعاء في قمة القائمة. يمكننا بالضغط على الاستدعاءات الموجودة في القائمة تلك أن ننتقل إلى تلك الإجرائية، بحيث تتغير قيم ومحتويات الواجهات السابقة بحسب الإجرائية أو التابع المفتوح؛
- واجهة النتائج النصية: وهي الواجهة التي تعرض النتائج النصية التي تنشأ عن الإجرائية قيد التنقيح، بالإضافة إلى أية نتيجة لتعليمة طباعة "PRINT" أو لأية رسالة خطأ.

- يعرض شريط الحالة في أسفل واجهة المنقِّح الأساسية لعدة معلومات تتعلق بـ:
  - حالة التنفيذ الحالية: "Running" أو "Completed" أو "Aborted"؛
  - اسم المخدم الذي يصل به SQL Debugger بالإضافة إلى معرِّف الدخول المستخدم؛
  - قاعدة المعطيات الحالية المستخدمة؛
  - رقم السطر والعمود الذي يتوضع عندهم مؤشر الكتابة في واجهة الرماز المصدري.

SQL Debugger
تنقيح القوادح والتوابع

- القوادح وتوابع المستخدم المعرّفة
- كيف يمكن تنقيح التوابع أو القوادح؟

تختلف بعض الأمور المتعلقة بتنقيح مخطوطات SQL عندما نتحدث عن القوادح أو التوابع، فلا يجري تنفيذ القوادح إلا عندما تحدث تعديلات معينة على المعطيات تؤدي بدورها إلى تشغيل القادح المعرّف على الجدول الذي حصلت فيه التغيرات، وبالتالي لا يمكن تنفيذ أو تنقيح القوادح مباشرةً في حين يمكن تنقيحها إذا ما تم استدعاءها أو تنفيذها من قبل إجرائية معينة؛

إذا لم تتوافر إجرائية تقوم باستدعاء أو تنفيذ التابع أو القادح الذي نرغب بتنقيحه، فلابد في هذه الحالة أن نقوم ببناء إجرائية خاصة لتقوم بعملية الاستدعاء المطلوبة.

SQL Server Profiler

- تعريف

- بنية الأداة SQL Server Profiler

تعتبر الأداة SQL Server Profiler إحدى أقوى أدوات التحليل والتدقيق في SQL Server، فهي تساعد المطور على فهم كيفية عمل التطبيق، وتكوِّن لديه صورة وافية عن كيفية أداء الاستعلامات وماذا يُتوقَّع منها؛ فمثلاً يمكننا من خلالها تحديد الاستعلامات التي تقوم بمسح جدول معين، أو تحديد الاستعلامات العشرة الأكثر سوءاً –من حيث الأداء– خلال الأسبوع الماضي، وما هي نسبة تحسن الأداء إذا ما استخدمنا فهرس معين على عمود معين في أحد الجداول، بالإضافة إلى تحديد احتمالات وجود استعلامات تؤدي إلى إقفال متبادل، وغير ذلك.

بإمكاننا استخدام الأداة SQL Server Profiler لتسجيل نشاطات المخدِّم وعرض تلك النشاطات أو تخزينها على ملف أو جدول خاص، كما يمكننا من خلالها تحديد أية أحداث نرغب بتعقبها وكيف نريد تجميعا وما هي المرشِّحات التي نريد تطبيقها.

يمتلك SQL Server 2000 مكوِّنان لمتابعة أنشطة المخدم وتعقبها أحدهما من جانب المخدم والآخر من جانب الزبون، يطلق اسم "SQL Trace Facility" على المكوِّن من جانب المخدم المسؤول عن إدارة صفوف الأحداث التي يتم إنشاءها على المخدم من قبل مولِّدي الأحداث، أما المكوِّن من جانب الزبون فهو SQL Profiler المزوَّد بواجهات تخاطبية متعددة المهام تسمح بتعقب الأحداث في الزمن الحقيقي.

SQL Server Profiler
بناء المتعقبات

- الخصائص عامة
- الأحداث
- أعمدة المعطيات
- المرشِّحات

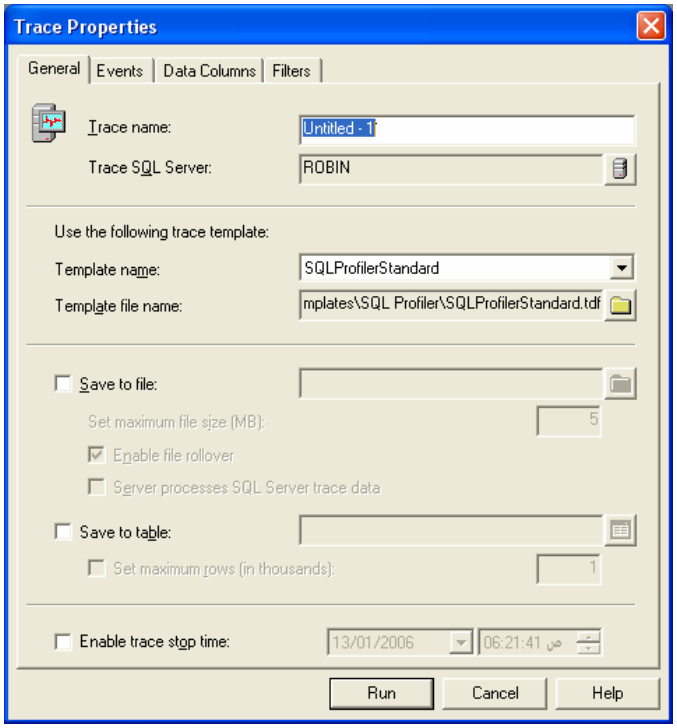
– بما أن الأداة SQL Server Profiler تساعد في تعقب عدد كبير جداً من الأحداث فإنه لابد من التشتت أثناء قراءة النتائج التي تعطيها الأداة، بالتالي نحتاج لأن نحدد كيفية عرض المعلومات وما هي الطريقة التي نرغب من خلالها أن نجمِّع النتائج، فعلى سبيل المثال، يمكننا تجميع كافة المعلومات الناتجة عن تعقُّب مستخدم معين؛

– قبل أن نحاول تعريف متعقِّب ما، لابد لنا من الإطلاع على قوالب المتعقِّبات مسبقة التعريف الموجودة والتي تحتوي على إعدادات خاصة بمتابعة وتعقُّب بعض الاحتياجات الشائعة، وفيما يلي عرض لبعض تلك القوالب مع وصف لخصائصها:
– القالب SQLProfilerSP_Counts: يتعقَّب أثر كافة الإجرائيات حالما تبدأ بالتنفيذ؛

- القالب SQL ProfilerStandard: يتعقَّب إتمام تنفيذ عبارات SQL وتنفيذ الإجراءات المنفذة عن بعد RPC وفترة تنفيذ كل منها؛
- القالب SQLProfilerTSQL: يتعقَّب بداية تنفيذ عبارات SQL والإجراءات المنفذة عن بعد؛
- القالب SQLProfilerTSQL_Duration: يتعقّب كامل فترة تنفيذ عبارات SQL والإجراءات المنفذة عن بعد؛
- القالب SQLProfilerTSQL_Grouped: يتعقَّب بداية تنفيذ عبارات SQL والإجراءات المنفذة عن بعد مجمَّعة مع التطبيقات والمستخدمين والإجراءات؛

- الخصائص العامة:

- يمكننا تشغيل الأداة SQL Server Profiler إما من قائمة "Tools" في الأداة Enterprise Manager أو من مجلد اختصارات SQL Server في قائمة ابدأ.
- لإنشاء متعقِّب جديد نختار من قائمة "File" ثم "New" ثم "Trace"، سيتم بعد ذلك طلب إنشاء اتصال مع المخدم ثم تظهر واجهة خصائص المتعقِّب الموضحة بالشكل التالي:



يظهر في هذه الواجهة عدة واجهات فرعية وهي "General" "Events" "Data Columns" و "Filters" على الترتيب، يمكننا في الواجهة "General" أن نحدد اسم المتعقِّب والطريقة التي نرغب من خلالها بتخزين خرج عملية التعقُّب، بحيث يمكننا تخزين النتيجة على ملف أو في جدول في قاعدة معطيات ما، بالإضافة إلى إمكانية تحديد نوع القالب الذي نرغب بالاعتماد عليه

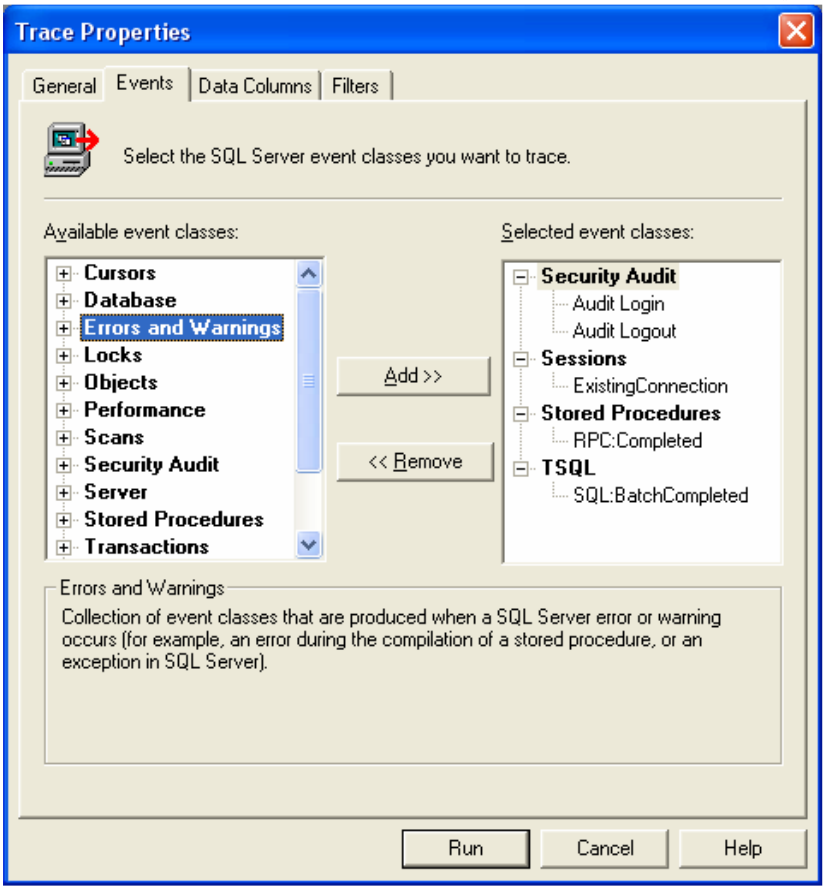
في بناء المتعقّب، فاختيارنا لأحد القوالب المعرَّفة يؤدي إلى تعديل قيم المعاملات الموجودة في واجهات الأحداث والمرشحات وأعمدة المعطيات بحيث يمكننا بعد ذلك تعديل بعض تلك القيم لتناسب احتياجاتنا؛

يمكننا أيضا من خلال هذه الواجهة تحديد وجدولة عملية التعقب، بحيث يمكن تشغيل تلك العملية ثم تعيير فترة زمنية مناسبة ليتم بعدها إيقاف المتعقِّب وتخزين النتائج آلياً؛

لا يعتبر اسم المتعقِّب الذي نقوم ببنائه ذو أهميه كبيرة، فهو لا يؤثر على بناء متعقبات أخرى، إنما تكمن الفائدة الوحيدة منه في متابعة عدة متعقبات يتم تنفيذها معاً؛

- الأحداث:

يمكننا من خلال الواجهة الفرعية "Events" أن نحدد الأحداث التي نرغب بالتقاطها خلال سير عملية التعقب، كما في الشكل التالي:



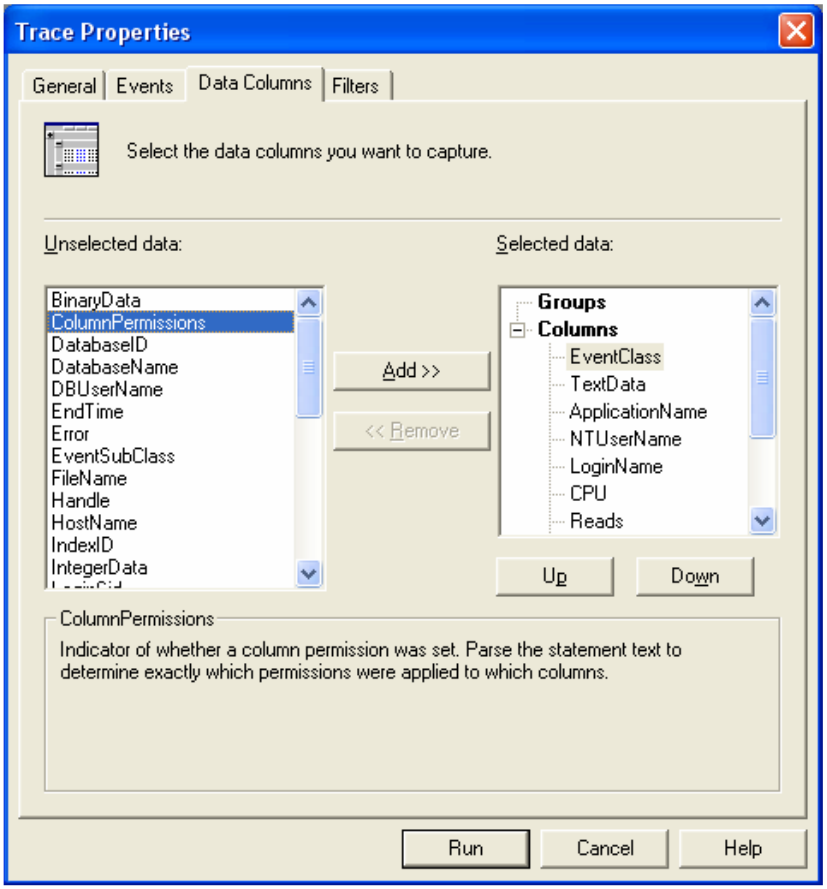
يمكننا أن نقوم بإضافة مجموعة كاملة من الأحداث كما يمكننا أن نحدد أحدها لإضافته، يمكننا كذلك إزالة أي حدث تمت إضـــافته مسبقاً؛

ينبغي هنا الإشارة إلى أن الإكثار من الأحداث التي نرغب بتعقبها قد يؤدي إلى تعقيد الخرج ويزيد نم صعوبة فهمه، بالتالي يُنصح عادة بإنشاء أكثر من متعقب يراقب كل منها سير مجموعة محددة من الأحداث؛

يفضل –وقبل البدء باستخدام الأداة SQL Server Profiler– أن نكوّن صورة واسعة عن كافة الأحداث التي يمكن تعقبها، يمكننا ذلك من خلال الإطّلاع على الوصف الخاص بكل حدث في أسفل واجهة "Event" الفرعية؛

- أعمدة المعطيات:

يمكننا من خلال الواجهة الفرعية "Data Column" أن نحدد ما هي المعلومات التي نرغب بعرضـــها وكيـــف ســـيتم ترتيبهـــا وتجميعها خلال سير عملية التعقب، كما في الشكل التالي:



إن ازدياد حجم المعلومات التي نرغب بعرضها قد يجعل من ملف أو جدول التعقُّب كبيراً جداً بحيث يؤثر ذلك سلباً علـــى عمليـــة التعقّب وعلى الأداء بشكل عام؛

يفضل –أثناء استخدام الأداة SQL Server Profiler– أن نكوِّن صورة واسعة عن كافة الطرائق التي يمكن من خلالها عرض نتيجة التعقب، يمكننا ذلك إما من خلال الإطّلاع على الوصف الخاص بكل طريقة في أسفل واجهة "Data Column" الفرعية، أو من خلال تجريب عرض تلك الأعمدة فعلياً ومراقبة النتائج؛

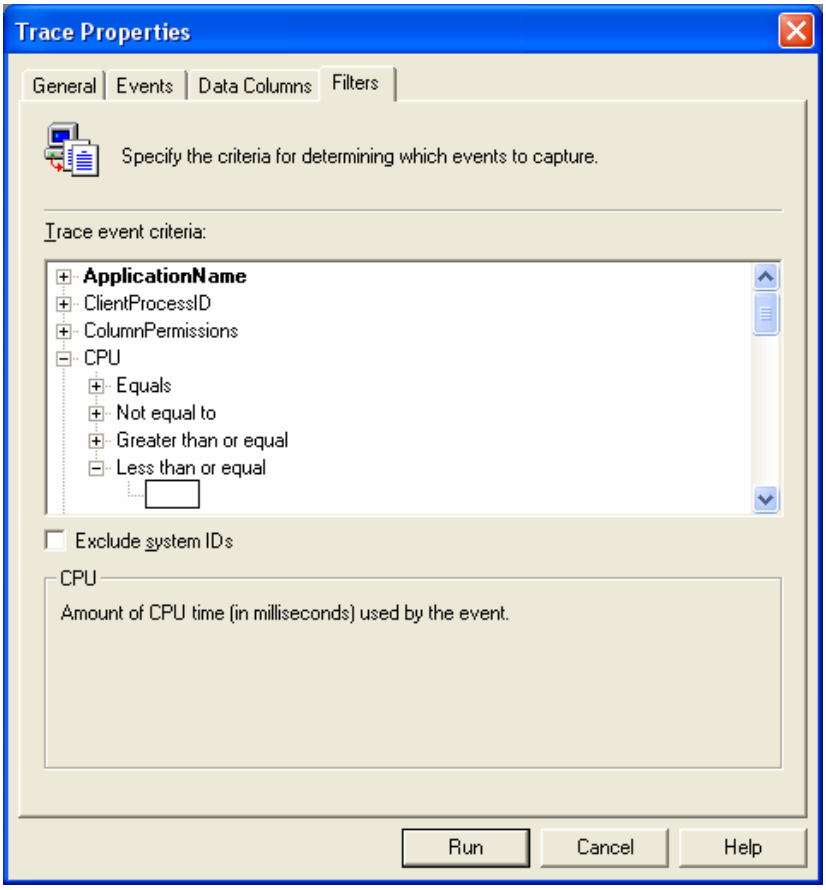
فيما يلي عرض لبعض خصائص أعمدة المعطيات:

- EventClass: وهو اسم الحدث الذي تم اختياره؛
- TextData: لعرض النص الفعلي (أي نص SQL على سبيل المثال)؛
- ApplicationName: لعرض اسم التطبيق الذي أدّى لإطلاق الحدث؛
- NTUserName: لعرض اسم المستخدم؛
- CPU: لعرض المدة المستهلكة من زمن المعالج لتنفيذ الحدث؛
- Reads: عدد عمليات القراءة المنطقية التي تم تنفيذها من قبل المخدِّم بسبب تنفيذ الحدث؛
- Writes: عدد عمليات الكتابة الفيزيائية التي تم تنفيذها من قبل المخدِّم بسبب تنفيذ الحدث؛
- Duration: الزمن الذي تم استهلاكه لتنفيذ الحدث؛
- ServerName: اسم المخدِّم الذي تمت عليه عملية التعقب.

يمكننا تجميع النتائج المعروضة في مجموعات مع بعضها البعض، ويتم ذلك من خلال تحريك أعمدة المعلومات التي نرغب بتجميعها ووضعها ضمن القائمة "Groups" بالترتيب الذي نفضله.

- المرشحات:

يمكننا من خلال الواجهة الفرعية "Filters" أن نفرض بعض القيود على بعض العناصر خلال سير عملية التعقب، ويمكننا تحرير تلك القيود كما في الشكل التالي:



تتوزع القيود في ثلاثة مجموعات رئيسية، وهي:

- يشبه / لا يشبه:

يمكننا من خلال هذا القيد أن نستثني بعض الأحداث التي لا تشبه القيمة المدخلة، يتم تطبيق هذا النوع من القيود على كافة أعمدة المعطيات ذات النتائج النصية مثل NTUserName أو ApplicationName، كما يمكننا أن نستخدم هنا الرمز % الذي يهمل ورود المحارف كما في المثال التالي %ADMIN%؛

- يساوي / لا يساوي / أكبر أو يساوي / أصغر أو يساوي:

يمكننا من خلال هذا القيد أن نستثني بعض الأحداث التي توافق الشرط المحدد، يتم تطبيق هذا النوع من القيود على كافة أعمدة المعطيات ذات النتائج الرقمية مثل DataBaseID أو CPU؛

- أكبر من / أصغر من:

يمكننا من خلال هذا القيد أن نستثني بعض الأحداث التي توافق الشرط أكبر من أو أصغر من قيمة محددة، يتم تطبيق هذا النوع من القيود على كافة أعمدة المعطيات ذات النتائج الرقمية بالإضافة إلى إمكانية تطبيقه على أعمدة التواريخ مثل StartTime؛

SQL Server Profiler
حفظ وتصدير خرج المتعقّبات

- كيفية حفظ خرج المتعقِّب
- كيفية استرجاع ملف أو جدول يحتوي خرج عملية تعقّب
- كيفية تحويل ملف إلى جدول

- تختلف الطريقة التي نرغب من خلالها بتخزين خرج عملية التعقُّب، بحيث يمكننا تخزين النتيجة على ملف أو في جدول في قاعدة معطيات، ويتم تحديد ذلك منذ البداية أي منذ أن قمنا بتحديد الخصائص العامة للمتعقِّب؛

- يتحدد عدد أعمدة الجدول –عندما نستخدم طريقة الحفظ في قاعدة معطيات– بعدد أعمدة المعطيات الذي تم تحديده أثناء تعريف محددات خرج المتعقّب، تعد هذه الطريقة في التخزين مناسبة في الحالات التي نرغب فيها بتحليل خرج عملية التعقُّب، بحيث يمكننا استخدام أدوات مثل SQL Query Analyzer من أجل إجراء استعلامات معقدة على المعطيات الناتجة، أو إعادة ترتيبها أو تجميعها وفق قيم معينة؛

- تتيح الأداة SQL Profiler إمكانية إعادة قراءة ملف أو جدول يحتوي خرج عملية تعقب سابقة، بهدف إجراء عمليات تحليل أوسع أو إعادة تنفيذ التعقُّب على نفس المخدم أو على نسخَةْ مَثَل منه، يتم ذلك من خلال الضغط على زر فتح ملف أو فتح جدول، بحيث أن فتح ملف يؤدي إلى عرض واجهة خاصة باستعراض الملفات على قرص الحاسب لنحدد من خلالها الملف الذي يحتوي المتعقب الذي نريد تنفيذه، أما زر فتح جدول فسيؤدي إلى فتح واجهة خاصة لبناء اتصال مع قاعدة المعطيات التي تحتوي الجدول الذي سبق وخزّننا عليه خرج عملية تعقُّب سابقة نريد فتحها؛

- قد يكون الملف أو جدول المتعقب الذي نريد فتحة ضخماً بحيث قد يؤدي إلى بعض المشاكل في الأداء خاصة عندما نحاول تحميله ككل دفعة واحدة، لذلك يؤمن SQL Profiler إمكانية تعريف مرشحات خاصة يمكن أن تطبّق على الملف الذي نقوم بفتحه؛

- يزود SQL Server تابع خاص مسبق التعريف وهو التابع fn_trace_gettable يمكن من خلاله أن نقوم بتحويل ملف يحتوي خرج عملية تعقُّب سابقة إلى جدول؛ يقدم هذا التابع ميزات خاصة يمكن استخدامها كمرشحات للملف الذي يتم تحويله إلى

جدول بحيث يتم إهمال بعض القيم وتحويل قيم أخرى بغرض الاختصار وتصغير حجم الجدول، ما يؤدي إلى تسريع وتحسين أداء عمليات التحليل التي يمكن أن نطبقها على الجدول الناتج؛

الفصل الرابع

عنوان الموضوع:

تنصيب وتحديث SQL Server.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

تتناول هذه الجلسة خصائص عملية تنصيب SQL Server 2000 بمعظم جوانبها، سواء كان ذلك من خلال تحديد المتطلبات الأساسية لعملية التنصيب، أو من خلال اختيار كافة الخصائص والإعدادات الأولية لـ SQL Server 2000 أثناء سير تلك العملية.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- إصدارات SQL Server 2000 المختلفة، وأي إصدار هو الأنسب للتنصيب
- المتطلبات الأولية العتادية والبرمجية لتنصيب SQL Server 2000
- خيارات إعداد عملية التنصيب خطوة بخطوة
- كيف يمكن التأكد واختبار نجاح عملية التنصيب، وكيف يمكن معالجة أخطاء التنصيب في حال وقوعها؟
- كيف يمكن تغيير الإعدادات والخيارات المحددة خلال عملية التنصيب بعد انتهائها؟
- تنصيب SQL Server 2000 عن بعد
- تحديث الإصدارات السابقة من SQL Server

- تبدأ عملية إدارة قاعدة معطيات SQL Server منذ الخطوة الأولى من استخدام هذا النظام، بحيث تؤمن لنا أداة التنصيب اختبارات أولية للمتطلبات التي نحتاجها لعمل SQL Server وذلك من خلال واجهات تخاطبية تطرح على المستخدم عدّة أسئلة قبل المباشرة بعملية التنصيب؛
- تستغرق عملية التنصيب ككل من 10 إلى 30 دقيقة بحسب الجهاز الذي تتم عليه العملية وبحسب الإعدادات والخصائص المختارة مسبقاً، وتعتبر بساطة إعداد SQL Server وتجهيزه للعمل نقطة هامة في صالح مايكروسوفت مقارنة مع تعقيد هذه المهمة في Oracle أو DB2؛
- سنناقش في الشرائح التالية بالتفصيل الغرض من كافة الخصائص والخيارات التي يعمل SQL Server اعتماداً عليها والتي قمنا بتحديدها أثناء عملية التنصيب –علماً أن تلك الخصائص تمتلك مسبقاً قيماً تلقائية– آخذين بعين الاعتبار إمكانية تعديل معظم تلك الخصائص بعد الانتهاء من عملية التنصيب.

اختيار الإصدار المناسب من SQL Server

- SQL Server Enterprise Edition
- SQL Server Standard Edition
- SQL Server Developer Edition
- SQL Server Personal Edition
- SQL Server Windows CE Edition

- يدعم SQL Server خمسة إصدارات وهي:
  - إصدار المؤسسة Enterprise
  - الإصدار القياسي Standard
  - إصدار المطوّر Developer
  - الإصدار الشخصي Personal
  - إصدار الأنظمة المحمولة Windows CE

- بقدر ما يستخدم SQL Server Enterprise Edition كمدير معطيات لأي تطبيق مكتبي، فإنه من الممكن أيضاً استخدام هذا الإصدار لأغراض تقييم مختلفة، فقد جرى تصميم كلٍ من الإصدارين
SQL Server Enterprise Edition و SQL Server Standard Edition ليعملا ضمن بيئة زبون/مخدِّم، إلا أنهما يختلفان

عن بعضهما البعض في بعض الخصائص المتعلقة بقابلية التوسع، فقد صُـمِّم الأول ليدعم 32 معالجاً و 64 جيغابايت من الذاكرة الحية (RAM)، في حين يكون الثاني محدوداً بأربعة معالجاتٍ كحد أقصى و 2 جيغابايت من الذاكرة الحية.

- يتمتع SQL Server Developer Edition بنفس خـصائص SQL Server Enterprise Edition إلا أنـه مُـرخص لأعمال التطوير والاختبارات فقط
- صُمِّم SQL Server Personal Edition بشكل رئيسي كمخزن معطيات للتطبيقات موجه بشكل أساسي للحواسب النقالة
- صُمِّم SQL Server Windows CE ليعمل كمخزن معطيات للتطبيقات التي تعمل في بيئة CE

المتطلبات الأولية

العتاديات

تفضل مايكروسوفت أن تتوافر عدة متطلبات عتادية على الجهاز الذي يجري تنصيب SQL Server عليه، وعلى الرغم مـن أن عملية التنصيب تختلف من جهازٍ إلى آخر، يبقى من الضروري توفير متطلباتٍ أوليةٍ لازمةٍ لعمل تلك الأداة، مع العلـم أن الأداء المتوقع يتناسب طرداً مع نوعية وحداثة التجهيزات العتادية المُستخدمة. تخص هذه المتطلبات الموارد التالية،:

- المعالج
- الذاكرة
- القرص الصلب

تفضل مايكروسوفت أن تتوافر عدة متطلبات عتادية على الجهاز الذي يجري تنصيب SQL Server عليه، وعلى الرغم مـن أن عملية التنصيب تختلف من جهازٍ إلى آخر، يبقى من الضروري توفير متطلباتٍ أوليةٍ لازمةٍ لعمل تلك الأداة، مع العلـم أن الأداء المتوقع يتناسب طرداً مع نوعية وحداثة التجهيزات العتادية المُستخدمة. تخص هذه المتطلبات الموارد التالية،:

- **المعالج:**

- يعتبر معالج Intel Pentium ذو السرعة 166 MHz الحد الأدنى الذي يمكن استخدامه لتشغيل أي إصدار مـن إصـدارات SQL Server 2000 على نظام تشغيل Windows NT أو Windows 2000، في حين يرتبط التشغيل على Windows 95 أو Windows 98 بخصائص نظام تشغيل؛
- يمكن ترقية أداة SQL Server 2000 لتدعم –كحد أقصى– 32 معالجاً على الحاسبات التي تشغِّل نظام Windows 2000 Datacenter Server، مع العلم أنه يمكننا الحصول على أداء جيد جداً وكافٍ لمعظم متطلبات المعالجة الكبيرة من خلال استخدام حاسب بمعالج مزدوج.

- **الذاكرة:**

- يُعتبر 64 ميغابايت، الحد الأدنى اللازم من الذاكرة إذا كنا نقوم بتنصيب إصدار SQL Server Standard، في حين يتطلب SQL Server Enterprise Edition، 128 ميغابايت من الذاكرة كحد أدنى؛

- يتطلب SQL Server Personal Edition 64 ميغابايت من الذاكرة كحد أدنى عندما يتم تنصيبه على نظام تشغيل Windows 2000 في حين يمكن تشغيله على بقية نظم التشغيل الأخرى مع 32 ميغابايت من الذاكرة فقط؛
- بشكلٍ عام، كلما ازداد حجم الذاكرة الرئيسية المستخدمة، كلما ازداد احتمال قرائة المعطيات المطلوبة من الذاكرة الرئيسية، على احتمال قراءتها من القرص الصلب، لذلك يُفضل أن يعمل SQL Server على جهاز يمتلك من الذاكرة الرئيسية 256 MB أو 512 MB على الأقل.
- **القرص:**
- يعرض الجدول التالي، حجم القرص المطلوب لنظام SQL Server 2000 وخصائصه الأخرى التي يمكن تنصيبها معه بغض النظر عن الحجوم المطلوبة لتخزين قواعد المعطيات أو السجلات أو النسخ الاحتياطية أو قواعد المعطيات المؤقتة:

نمط التنصيب	حجم القرص المطلوب (ميغابايت)
تنصيب كامل	270
تنصيب تقليدي	250
أدوات الزبون	100
خدمات التحليل	من 50 إلى 130
English-Query	

المتطلبات البرمجية

تختلف البرمجيات –التي يتطلب تنصيب SQL Server وجودها– من نسخةٍ إلى أخرى، فهناك الكثير من النقاط التي ينبغي أخذها بعين الاعتبار فيما يتعلق بنظام التشغيل ونظام الملفات المُستَخدم أو حتى إصدار نظام التشغيل أو التحديثات التي يمكن تنفيذها عليه:

نظام التشغيل:

يعرض الجدول التالي نظم التشغيل المتوافقة مع بعض إصدارات SQL Server:

إصدار SQL Server	نظم التشغيل المتوافقة معه
Enterprise Editio	Windows NT 4.0 Server, NT4.0 Server Enterprise Edition, ows 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacenter Server
Standard Editio	Windows NT 4.0 Server, NT4.0 Server Enterprise Edition, Windows 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacenter Server
Personal Editio	Windows 98, Windows Me, Windows XP, NT4.0 Workstation, Windows 2000 professional, NT 4.0 and Windows 2000 Server (all)

نظام الملفات:

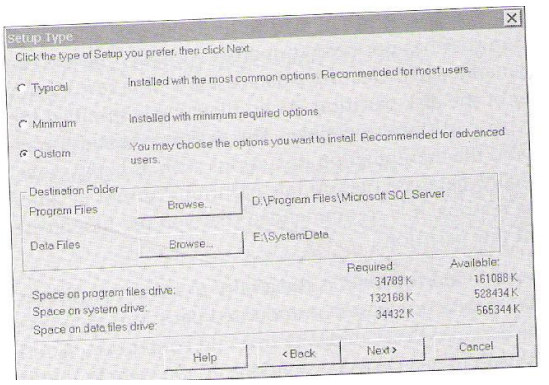
يمكن تنصيب SQL Server على أقراص مهيّئة بنظام ملفات FAT أو FAT32 أو NTFS مع حجم عنقود يبلغ 64 كيلوبايت، في حين لا يُنصح أبداً باستخدام خاصة ضغط الأقراص الصلبة –المتاحة فقط في نظام NTFS– لأنها تؤثر سلباً على أداء النظام.

خيارات إعداد عملية التنصيب

يمكن تصنيف خيارات إعداد عملية التنصيب من خلال النقاط التالية:

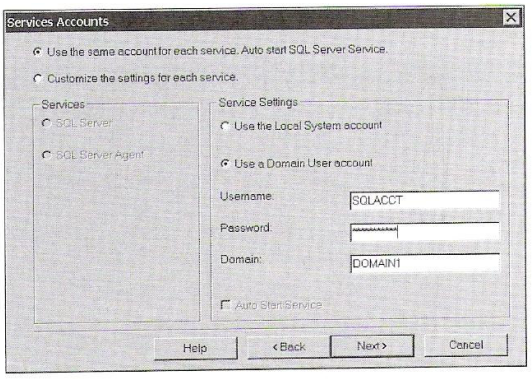
- تحديد مكونات SQL Server ومواقع ملفاته:

الخاصّية	التنصيب الشائع	التنصيب المختصر	التنصيب المخصص
مخدِّم قاعدة المعطيات	نعم	نعم	اختياري
أدوات التحديث	نعم	لا	اختياري
تكرار المعطيات	نعم	نعم	اختياري
تقنية البحث عن النص الكامل	نعم	لا	اختياري
أدوات إدارة الزبون	كافة الأدوات	لا	اختياري
الاتصالية مع الزبون	نعم	نعم	نعم
الدليل المرجعي لـــSQL Server	نعم	لا	اختياري
أدوات التطوير	المنقِّح فقط	لا	اختياري
عيِّنات الرماز	لا	لا	اختياري
الإعدادات الإقليمية	نعم	نعم	اختياري

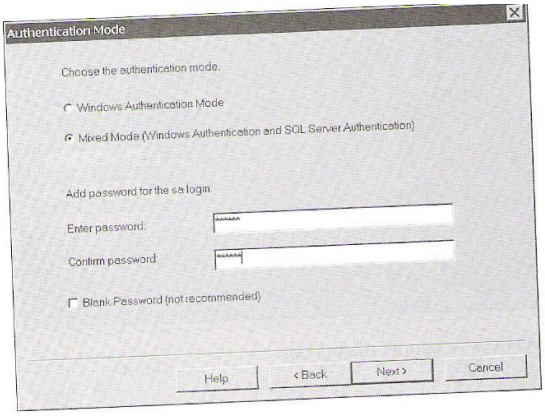


- حساب SQL Server وحساب وكيل SQL Server

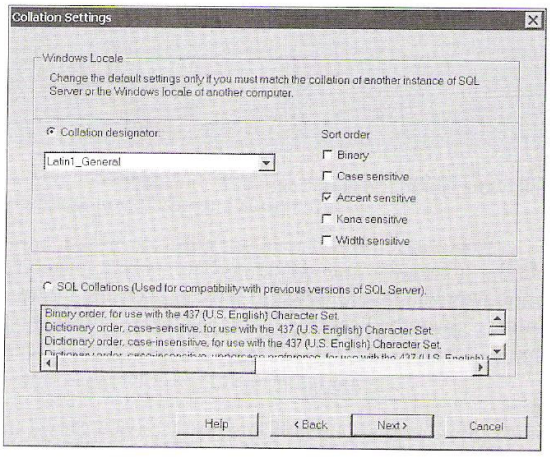
يوضح الشكل التالي الواجهة التي يتم من خلالها تحديد خصائص تلك الحسابات أثناء عملية التنصيب:



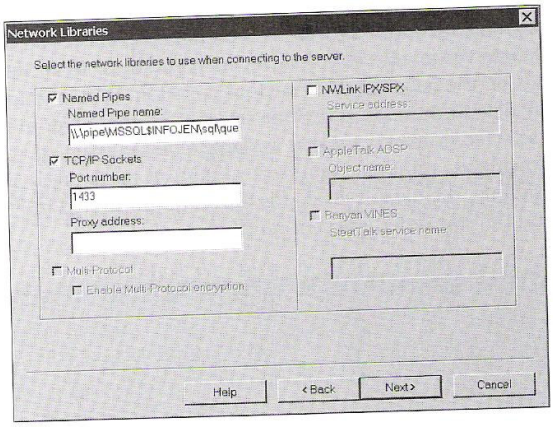
- اختيار نمط التعرُّف على الهوية المستخدم في SQL Server:



- تحديد الإعدادات الإقليمية التلقائية:

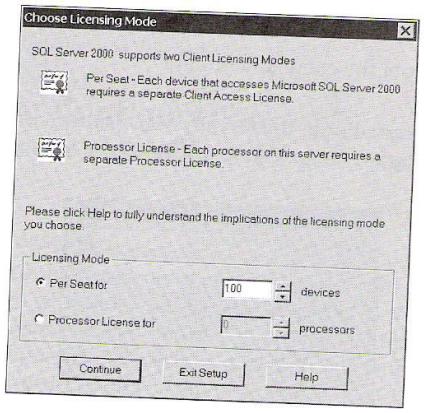


- اختيار مكتبات الشبكة:



- اعتماد أحد خيارات ترخيص SQL Server:

يوضح الشكل التالي الواجهة التي يمكننا من خلالها اختيار نمط الترخيص الذي نرغب باستخدامه:



ما أن يتم اختيار وتحديد المتطلبات العتادية والبرمجية الأولية يمكننا المباشرة فوراً بعملية تنصيب نظام SQL Server؛ يتكوّن برنامج التنصيب المستخدَم من واجهات تخاطبية خاصة تسمح للمستخدِم أن يقوم بتحديد خيارات التنصيب بأسلوب سهل وبسيط، يمكن تصنيف تلك الخيارات من خلال النقاط التالية:

- تحديد مكونات SQL Server ومواقع ملفاته:
  - ينبغي أولاً أن نحدد اسم نسخة SQL Server التي نقوم بتنصيبها مع العلم أنه بإمكاننا إنشاء عدة نسخ؛
  - بعد ذلك ينبغي تحديد أحد أنواع عملية التنصيب الثلاثة المتاحة، وهي إما التنصيب الشائع الذي يقوم بتنزيل معظم ملفات SQL Server وخياراته الشائعة، أو التنصيب بأقل قدر ممكن من الخيارات، أو التنصيب المُخصص والذي يتيح للمستخدم إمكانية اختيار الخصائص التي يرغب بتنصيبها.
  - يعرض الجدول المُرفق خصائص SQL Server المتاحة والتي يمكن اختيارها في كل نوع من أنواع التنصيب الثلاثة السابقة.
- يتم في هذه المرحلة أيضاً تحديد الموقع الذي سنقوم بتخزين ملفات SQL Server فيه، ويقصد هنا بملفات SQL Server، أي ملفات التطبيق بحد ذاته بالإضافة إلى ملفات المعطيات؛
- يعتبر المسار التالي هو المسار التلقائي الذي يجري تنصيب تلك الملفات فيه:

systemRoot\Program Files\Microsoft SQL Server

- يعتبر اعتماد المسار التلقائي لملفات التطبيق الخيار الأفضل من وجهة نظر الأداء والاتِّساق مع التطبيقات الأخرى، إلا أن البعض يفضل فصل عملية تخزين ملفات SQL Server في جزء مخصص، وذلك من وجهة نظر إدارية، مع العلم أنه لابد لبعض الملفات الخاصة من أن يتم نسخها إلى جزء القرص الحاوي على نظام التشغيل؛
- إلاّ أنه –ومن جهة نظرٍ أخرى– ينبغي أن يجري تنصيب ملفات المعطيات على جزء خاص ومنفصل من النظام وذلك بهدف تحسين الأداء وتسريع الاستعلامات التي سيجري تطبيقها مستقبلاً على قواعد المعطيات المبنية على ذلك الجزء؛
- كذلك من المستحسن أن يتم نقل قاعدة المعطيات المؤقتة Tempdb –بعد الانتهاء من عملية التنصيب– إلى قرص خاص بها.

- حساب SQL Server وحساب وكيل SQL Server:
- يعمل SQL Server ضمن Windows NT وَ Windows 2000 كخدمة من خدمات النظام، بالتالي، فهو يعمل إما ضمن سياق حسابات النظام المحلية أو ضمن سياق حسابات مجال معين أو ضمن سياق حسابات المستخدم المحلية؛
- في حال جرى تعريف SQL Server ليعمل كحساب نظام محلي فلن نستطيع أن نستخدم بعض خدماته الشبكية، خاصة تلك التي تتعلق بتكرار المعطيات أو استخدام بريد SQL على سبيل المثال؛
- يجري –وبشكل تلقائي– تنصيب كلاً من خدمة SQL Server وخدمة وكيل SQL Server للعمل ضمن بيئة المستخدم الحالي المسجَّل في النظام، أي الذي سجل دخوله ويقوم بتنفيذ عملية التنصيب، بالتالي ينبغي أن يكون هذا المستخدم عضواً من أعضاء مجموعة مديري النظام ليقدِّم كافة السماحيات الممكنة لخدمة SQL Server التي يجري تنصيبها؛

- اختيار نمط التعرُّف على الهوية المستخدم في SQL Server:
- يتوفر في SQL Server نمطين من أنماط التعرُّف على الهوية، نمط Windows في التعرُّف على الهوية Windows Authentication Mode والنمط المختلط Mixed Mode؛
- يعتمد النمط الأول على مفهوم التعرُّف على هوية المستخدم الذي يتصل مع SQL Server، من خلال حساب الـ Windows الذي يستخدمه؛
- يجري في النمط الثاني التعرُّف على هوية المستخدم إما من حساب الـ Windows الخاص به أو من مُعرِّف المستخدم في SQL Server، بحيث تتطلب هنا عملية الولوج أن يزود المستخدم النظام بكلمة مرور لحساب sa (SQL Server System Administration)، كما يوضح الشكل المُرافق.

- تحديد الإعدادات الإقليمية التلقائية:
- ينبغي –أثناء عملية التنصيب– أن نقوم بتحديد الخصائص التلقائية للإعدادات الإقليمية المُستخدمة في SQL Server، ويقصد بذلك، طريقة ترميز الصفحات أو محارف المعطيات غير القياسية NON Unicode؛
- يجري –وبشكل تلقائي– اعتماد نظام الترميز المختار ليطبَّق على كافة قواعد المعطيات المنشأة في SQL Server، مع العلم أنه بإمكاننا تخصيص تلك العملية من خلال السماح بتحديد إعدادات إقليمية مختلفة لكل قاعدة معطيات أو كل جدول أو حتى كل عمود؛
- يتوافر نمطان مختلفان من الإعدادات الإقليمية يمكننا اختيار أحدهما أثناء سير عملية التنصيب: إعدادات SQL الإقليمية، والإعدادات الإقليمية المحلية:
  - يجري اعتماد إعدادات SQL الإقليمية عندما نقوم بتنصيب SQL Server على جهاز يحتوي مسبقاً على أحد إصدارات SQL Server، إصدار 6.5 أو إصدار 7.5 على سبيل المثال، كما يحتوي على قواعد معطيات نرغب أن نحافظ على توافق معطياتها مع عملية التنصيب الجديدة؛
  - يمكننا أن نقوم بتخصيص أكبر للعملية من خلال تحديد خصائص الإعدادات الإقليمية التي نرغب باستخدامها كما يوضح الشكل المُرافق.

- اختيار مكتبات الشبكة:
- تعبِّر مكتبات الشبكة عن مجموعة من ملفات DLL التي تسمح لـــ SQL Server أن يستخدم بروتوكولات مختلفة للاتّـــصال عبر الشبكة؛
- يمكن تشغيل عدّة مكتبات بآنٍ واحد مما يسمح لـــ SQL Server أن يُخدِّم طلبات الزبائن المختلفة ضمن بيئة شـــبكيّة غيـــر متجانسة؛
- يوضح الشكل المُرافق الواجهة التي يمكننا من خلالها أن نحدد أنواع البروتوكولات التي نرغب باستخدامها، مع الخـــصائص المتعلّقة بكل منها. كما يجدر هنا أن نذكر ضرورة توافق خصائص المكتبة –التي سيستخدمها الزبون للاتصال بالمخدِّم– من جهة الزبون، مع المكتبة من جهة المخدِّم.

- خيارات ترخيص SQL Server:
- ينبغي –أثناء سير عملية التنصيب– أن نختار نمط الترخيص الذي نرغب باستخدامه، وينبغي أن يكون هذا الخَيار نهائياً، إذ لا يمكننا تغييره بعد الانتهاء من عملية التنصيب.
- يوجد نمطان أساسيان لعملية الترخيص، هما:
  - الترخيص حسب الجلسة:
- ينبغي في هذا النمط بناء جلسة لكل مخدِّم مع كل زبون يتّصل به؛
- لا تحدُّ هذه الطريقة من إمكانيات الاتصالات المتاحة، فكل جهاز يمكنه التواصل مع المخدِّم من خلال عدّة اتصالات؛
- يعتبر هذا النمط من الترخيص، الأنسب للمؤسسات الصغيرة أو تلك التي تتكون من عدد محدود من المستخدمين.
  - الترخيص حسب المعالج:
- ينبغي في هذا النمط تحديد رخصة لكل معالج يشغِّل SQL Server؛
- يمكن من خلال هذا النمط تخديم عدد غير محدد من المستخدمين، وهو يصلح في المؤسسات الضخمة أو في التطبيقات التـــي تعمل على أساس وُلوج عبر الوِب.

تأكيد عملية التنصيب

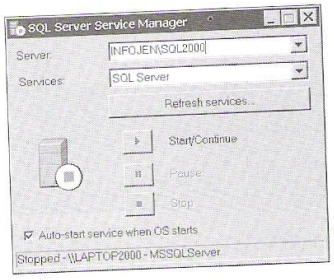
ينبغي –وبعد الانتهاء من عملية التنصيب– أن نتأكد من صحتها؛
ينبغي هنا التأكد من عدّة نقاط:
التأكد من تنصيب SQL Server بشكل صحيح، ومن أنه قد تم إنشاء قواعد معطيات النظام في المكان المخصص لها، وأن أدوات إدارة SQL Server موجودة، وأنه بإمكاننا تشغيل أو إيقاف تلك الخدمات، كما ينبغي أن نقوم بإنشاء اتصال مع إحـــدى قواعـــد المعطيات وإجراء استعلام بسيط عليها؛

بشكل عام، ليس من الضروري إجراء تلك الاختبارات إذا لم تظهر أية رسالة خطأ أثناء سير عملية التنصيب، مع العلم أن معظـــم المشاكل التي يمكن مواجهتها –وفي معظم الأحيان– تكون نتيجةً لإعداد غير صحيح أثناء عملية التنصيب.

تشغيل وإيقاف SQL Server

- على الرغم من أن SQL Server معَّد ليعمل تلقائياً عند إقلاع نظام التشغيل بشكل تلقائي، إلا أنه لابد من وجود حالات خاصة تستدعي توفر طريقة يدوية لتشغيل SQL Server أو إيقافه مؤقتاً أو بشكل دائم؛
- تمنع عملية الإيقاف المؤقت لخدمة SQL Server إمكانية إنشاء اتصال جديد معه، في حين تسمح للاتصالات الحالية بالبقاء في حالة عمل. تفيد هذه العملية في منع الاتصالات الجديدة من الولوج إلى المخدِّم خاصةً خلال عمليات الاختبار أو الصيانة؛
- يمكن تشغيل أو إيقاف خدمات SQL Server بشكل مؤقت أو دائم من خلال إحدى الأدوات التالية:
  - SQL Server Service Manager؛
  - SQL Server Enterprise Manager؛
  - لوحة التحكم؛
  - لائحة التعليمات Command Prompt.

- يمكننا الولوج إلى Service Manager من خلال أيقونة Services Manager الموجودة في شريط مهمات Windows، أو من خلال مجموعة برامج SQL Server في قائمة البرامج، يوضح الشكل التالي الواجهة التي يمكننا من خلالها القيام بتشغيل أو إيقاف SQL Server:



- يمكننا أيضاً التحكم بكيفية تشغيل SQL Server من الأداة Enterprise Manager من خلال اختيار المخدِّم المناسب ثم تطبيق المهمة المناسبة بعد الضغط بالزر اليميني عليه؛
- يمكننا أيضاً اختيار الخدمة التي نريد تشغيلها أو إيقافها من قائمة الخدمات الموجودة في لوحة التحكم ثم استعراض قائمة المهمات السريعة؛
- أخيراً يمكننا التحكم بالعمليات السابقة من خلال لائحة التعليمات، وذلك بتنفيذ التعليمة net start، أو net pause، أو net continue، أو net stop متبوعة بإسم الخدمة أو نسخة SQL Server التي نرغب بتطبيق تلك التعليمة عليها وذلك من أجل التشغيل أو إيقاف التشغيل بشكل مؤقت أو متابعة التشغيل أو إيقاف التشغيل بشكل دائم، على الترتيب، فمثلا: يمكننا من خلال التعليمة التالية تشغيل الخدمة mssqlserver كما يلي:

```
net start mssqlserver
```

كما يمكننا تشغيل إحدى نسخ SQL Server كما يلي:

net start mssql$***instanceName***

إعدادات ما بعد التنصيب

- تغيير كلمة مرور sa
- إعداد حسابات مديري النظام
- إعداد سجل الأخطاء
- تطبيقات المخدِّم الشبكيّة

بعد الانتهاء من عملية تنصيب SQL Server، والتأكد من نجاحها، لابد لنا من تنفيذ عدة مهمات أخرى يطلق عليها اسم مهمات ما بعد التنصيب:

- تغيير كلمة مرور sa:
- لابد لمدير قاعدة المعطيات من أن يغير كلمة مرور حساب sa مباشرةً بعد الانتهاء من عملية تنصيب SQL Server، وذلك إذا ما لم يتم إسناد كلمة مرور لهذا الحساب أثناء عملية التنصيب، أو إذا كان المسؤول عن إسناد كلمة المرور هذه قد اشترك بهذه الكلمة مع أشخاص آخرين أثناء عملية التنصيب؛
- يمكننا تغيير كلمة مرور حساب sa إما من صفحة خصائص ذلك الحساب الذي يمكن الوصول إليه من خلال مجلد Security في الأداة Enterprise Manager، أو من خلال إجرائية sp_password.
- إعداد حسابات مديري النظام:
- لا يعتبر حساب sa هو الحساب الوحيد الذي يتمتع بسماحيات ولوج مطلقة على SQL Server، ذلك لأنه عند تنصيب SQL Server يتم اعتبار كافة أعضاء مجموعة المدراء المحليين الموجودة على المخدِّم –وبشكل تلقائي– مستخدمين ذوي سماحيات مطلقة أيضاً؛
- يفضل هنا –من وجهة نظر إدارية– التمييز بين مدراء نظام التشغيل ومدراء نظام إدارة قواعد المعطيات، فليس من الضروري أن يتمتع مستخدم خبير لنظام التشغيل Windows بالخبرات الكافية لإدارة SQL Server؛
- يمكننا إنشاء مجموعة خاصة للمدراء وحسابات أخرى غير حساب sa، نستطيع من خلالها تحديد مستويات مختلفة من سماحيات الولوج إلى SQL Server.
- إعداد سجل الأخطاء:
- يمتلك SQL Server سجل أخطاء يُخزِّن فيه كافة المعلومات المتعلقة بالأخطاء الحاصلة ورسائلها؛
- يُخزِّن SQL Server عند كل عملية تشغيل جديدة، السجل القديم ويُعِدّ سجل جديد بلاحقة من الشكل "1." ليخزِّن فيه المعلومات؛
- يُخزِّن SQL Server وبشكل تلقائي لغاية ستة سجلات أخطاء ذوات لواحق من الشكل "1." "2." "3." الخ... ويمكننا تحديد ذلك العدد من السجلات من خلال مجلد Management في أداة Enterprise Manager. يجري تخزين تلك السجلات بشكل تلقائي في المسار التالي:
Program Files\Microsoft SQL Server\Mssql\Log\Errorlog

كما يمكننا بناء سجل جديد من خلال الإجرائية sp_cycle_errorlog.

- تطبيقات المخدِّم الشبكيّة:
- تستخدم تطبيقات المخدِّم الشبكيّة "Server Network Utility" –المتواجدة ضمن مجلد برامج SQL Server في قائمة **إبدأ**– في تغيير إعدادات مكتبات الشبكة بعد الانتهاء من عملية التنصيب؛
- يمكننا من خلال هذه التطبيقات، تفعيل أو عدم تفعيل البروتوكولات بالإضافة إلى خصائص متعددة أخرى، كاستخدام البوابات على سبيل المثال.

المشاكل التي يمكن التعرض لها خلال عملية التنصيب

- يعتبر سجل الأخطاء، المكان الأول الذي ينبغي تفحصّه في حال وقوع أية مشكلة أثناء سير عملية التنصيب؛
- يُصدِر برنامج التنصيب سجلاً خاصاً بالأخطاء المرتكبة أثناء سير تلك العملية، وهو الملف sqlstp.log الموجود ضمن مجلد System في نظام التشغيل Windows؛
- يحتوى هذا الملف على معلومات وافية تساعد المستخدم على تتبع المشكلة الحاصلة ومعرفة أسبابها، مع العلم أن أغلب المشاكل المتوقع حدوثها أثناء سير عملية التنصيب تكون بسبب مشاكل شبكيّة أو بسبب فشل في إقلاع الخدمات؛

التنصيب عن بعد

- يسمح لنا برنامج تنصيب الأداة SQL Server باختيار حاسب بعيد على الشبكة لكي يكون هدفاً لعملية التنصيب؛
- عند إجراء عملية تنصيب عن بعد، فإن سيناريو الأحداث يتخذ المسار التالي:
  - يجمع برنامج التنصيب كافة معلومات الدخل المطلوبة لتنصيب SQL Server؛
  - يتم تخزين كافة معلومات الدخل السابقة ضمن الملف setup.iss؛
  - يتم نسخ الملف السابق عبر الشبكة إلى الجهاز الهدف لتبدأ عليه عملية تنصيب SQL Server اعتماداً على الإعدادات المُخزنة ضمن الملف المُرسَل؛
- ينبغي إعادة تشغيل الحاسب الهدف بعد الانتهاء من عملية التنصيب.

تحديث إصدارات سابقة من SQL Server

- يدعم SQL Server 2000 عملية تحديث كلٍ من نسختي SQL Server 6.5 و SQL Server 7.0، مع العلم أنه إذا ما رغبنا بتحديث SQL Server 6.0 إلى SQL Server 2000، فينبغي أولاً تحديث الأول إلى SQL Server 6.5 أو SQL Server 7.0 ومنه إلى الثاني؛
- عند إجراء عملية تنصيب نسخة SQL Server 2000 على حاسب يمتلك نسخة SQL Server 7.0، فإنه بإمكاننا اختيار إعادة كتابة الملفات الجديدة فوق الملفات القديمة، مما يؤدي إلى تحديث النسخة القديمة وتحويل ملفات قاعدة المعطيات وتهيئتها من جديد لتعمل على النسخة الأحدث؛

- عند إجراء عملية تنصيب نسخة SQL Server 2000 على حاسب يمتلك نسخة SQL Server 7.0، فإنه يمكننا أن نحـــافظ على النسخة القديمة وعلى قواعد المعطيات المخزنة عليها، لنقوم بإجراء عملية التنصيب الجديدة بمعزل عنها، كما يمكننا بعد ذلك أن نستخدم أداة خاصة مع SQL Server 2000 أو استخدام الإجرائية sp_attach_db لكي نقوم باستيراد قواعد المعطيات مـــن النسخة القديمة إلى الحديثة؛
- عند إجراء عملية تنصيب لنسخة SQL Server 2000 على حاسب يمتلك نسخة SQL Server 6.5، فإنه لا يمكننا تحديث قواعد المعطيات العاملة على النسخة القديمة مباشرةً أثناء سير عملية التنصيب، في حين نستطيع إجراء تلك العملية بعد تنـــصيب النسخة الحديثة؛

ملاحظة:

ينبغي –وقبل البدء بأي عملية تحديث من أية نسخة إلى أية نسخة أخرى– أن نقوم أولاً بإجراء **تخزين احتياطي** لتجنب أية مشكلة قد تؤدي إلى خسارة المعطيات. فبإجراء هذه العملية يمكننا أن نطمئن أنفسنا بأنه مهما كانت النتيجة يمكننا –وفي أسوء الحالات– العودة إلى النقطة التي كنّا قد بدأنا منها.

تنصيب وإعداد SQL Server Client

- بعد الانتهاء من تنصيب SQL Server على المخدِّم، ينبغي الآن أن نعمل على توصيل الزبائن المرتبطة به على الشبكة
- غالباً ما تكون حاسبات الزبائن عبارة عن طرفيات تتطلب تطبيقات خاصة للتواصل مع نسخة من SQL Server عبر الشبكة
- سنناقش فيما يلي، برمجيات SQL Server Client التي ينبغي تنصيبها والتي ستسهل من عمليات الاتصال الواجب إنشاءها بين الحاسبات الزبائن من جهة وبين المخدِّم من جهة أخرى

بنية SQL Server Client

- أنماط تطبيقات الزبائن التي تتصل مع المخدِّم:
  - OLE DB Consumers
  - تطبيقات ODBC
  - تطبيقات DB-Library

- عملية التواصل ما بين الزبون والمخدِّم.
- يدعم SQL Server 2000 عدّة طرائق لوصل حاسبات الزبائن مع نسخة من المخدِّم، بحيث ترتبط عمليات الوصـــل تلـــك، والتطبيقات الواجب تنصيبها، بشكل كبير، مع نوع الزبون وموقعه، بالإضافة إلى البنية التحتية للشبكة والتوصيلات المستخدَمة؛
- تتوافر عدّة أنماط شائعة من الزبائن التي يمكن أن تتّصل مع قاعدة معطيات مخـــدِّم SQL Server 2000، بحيـــث تُعتبـــر الزبائن هنا تطبيقات برمجية يمكن أن يجري فرزها إلى إحدى التصنيفات التالية:
  - OLE DB Consumers (أو مستهلكي OLE DB):

تستخدِم هذه التطبيقات أحد نوعين من OLE DB Providers، المقدَّمة من مايكروسوفت للاتصال مع SQL Server، وهما: OLE DB Provider الخاص بـ SQL Server وَ OLE DB Provider الخاص بـ ODBC، بحيث يُعتبر الأول هو المفضل للاستخدام على اعتبار أنه مخصص ليتوافق مع SQL Server؛

- تطبيقات ODBC:

تعتمد هذه التطبيقات على بنية ODBC في إنشاء الاتصالات مع المخدِّم، وتُعتبر هذه الطريقة، المعيار المستخدم من قبل بعض أدوات SQL Server Client كالأداة SQL Query Analyzer والأداة Enterprise Manager على سبيل المثال؛

- تطبيقات DB-Library:

وهي عبارة عن تطبيقات الواجهة البرمجية الأصلية (API) المُستخدمة للاتصال مع SQL Server؛

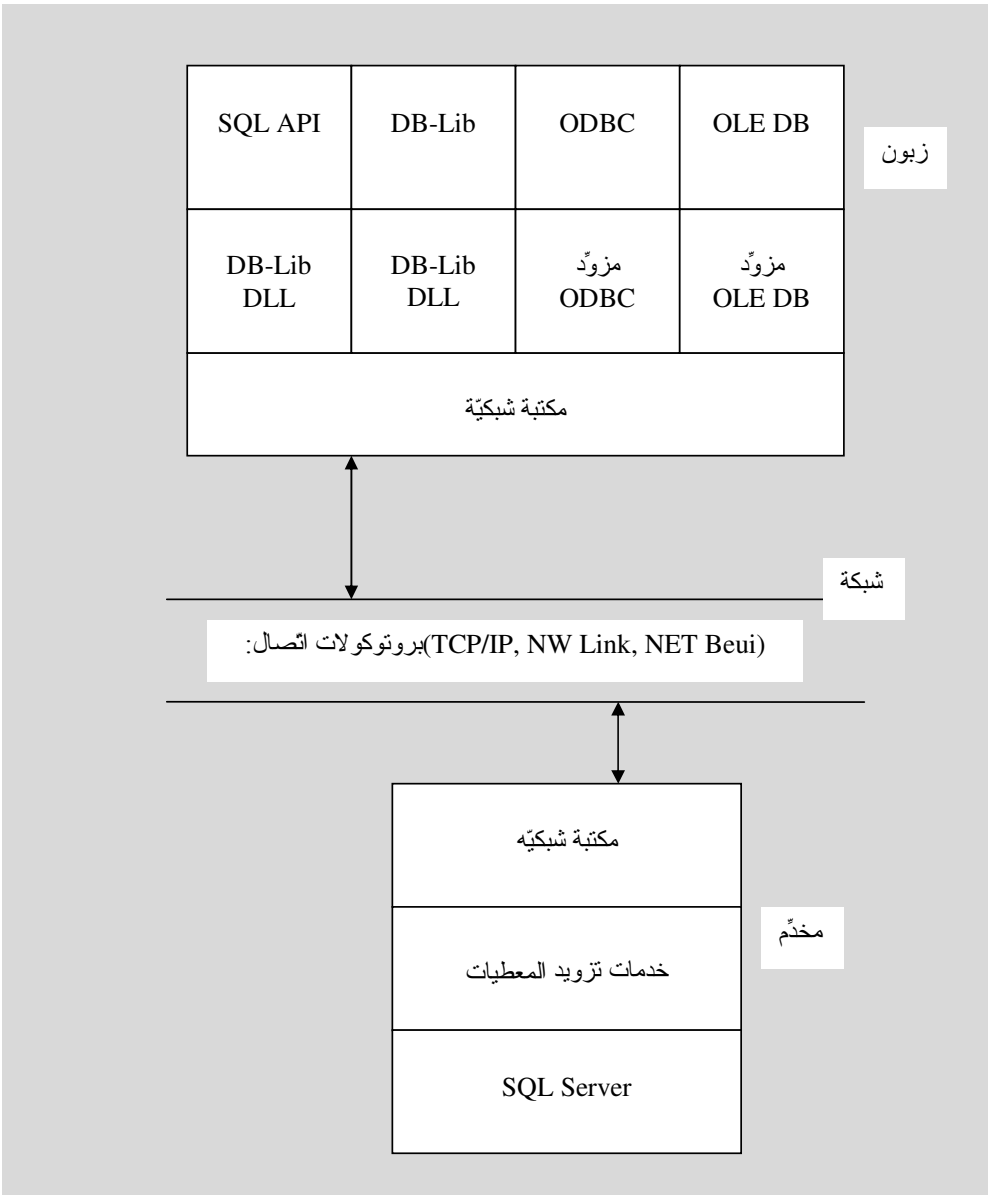
تعتبر التطبيقات التي تستخدم DB-Library للاتصال مع SQL Server، هي التطبيقات الأقدم –والتي تعتمد على واجهة التعليمات ISQL Command-Line المضمَّنة مع SQL Server 2000– في إجراء الاتّصالات.

- ينبغي على كافة الزبائن التي تتصل مع نسخة من SQL Server، أن تستخدم مكتبة شبكيّة مماثلة للمكتبة المُستخدَمة من جانب المخدِّم، والتي تُستخدم للتعامل مع الاتصال القائم ما بين تطبيق الزبون وبروتوكول الشبكة؛
- يعرض المخطط المُرافق كيفية سير عملية التواصل ما بين الزبون والمخدِّم، باستخدام المكوِّنات سابقة الذِّكر:
  - يطلب تطبيق الزبون أحد التطبيقات البرمجية اللازمة للتواصل مع المخدِّم: OLE DB أو ODBC أو DB-Library أو أحد أنواع واجهات التطبيقات البرمجية المضمَّنة SQL API؛
  - يجري بعد ذلك التواصل مع المكتبة الشبكيّة لإعداد عملية الاتّصال؛
  - تؤمن المكتبة الشبكيّة أسلوباً مناسباً لعملية نقل المعطيات أو طلبات الزبون عبر الشبكة، وذلك باستخدام بروتوكول النقل المناسب؛
  - تستقبل مكتبة المخدِّم الشبكيّة طلبات الزبون وتنقلها إلى النسخة المناسبة من SQL Server المتواجد على المخدِّم.

تنصيب أدوات الزبائن

- تُوصَف عملية تنصيب SQL Server Client وإعداده للتواصل مع قاعدة معطيات المخدِّم بالعملية البسيطة نوعاً ما، بحيث يمكننا من خلال برنامج تنصيب SQL Server نفسه أن نقوم بعملية التنصيب تلك على أجهزة الزبائن، وذلك بعد اختيار خصائص وإعدادات خاصة بهذه العملية، كما سنلاحظ في الشرائح القادمة؛

ولكن، وقبل البدء بعملية تنصيب الأدوات الإدارية لـ SQL Server Client ، وملفات المكتبات الأخرى المختلفة، لابدّ لنا أن نتأكد من إمكانية إجراء عملية التنصيب على الحاسب الذي نتعامل معه.


SQL API
DB-Lib
ODBC
OLE DB
زبون
DB-Lib DLL
DB-Lib DLL
مزوِّد ODBC
مزوِّد OLE DB
مكتبة شبكيّة
شبكة
بروتوكولات اتّصال: (TCP/IP, NW Link, NET Beui)
مكتبة شبكيّه
خدمات تزويد المعطيات
SQL Server
مخدِّم

1.4.15 المتطلبات الأولية لأجهزة الزبائن

- المعالج
- الذاكرة
- القرص
- نظام التشغيل
- التطبيقات الشبكيّة

ينبغي أن يحقق جهاز الزبون، المتطلبات الأساسية التالية والتي تعتبر الحدّ الأدنى المطلوب لكي نستطيع القيام بعملية تنصيب SQL Server Client على ذلك الجهاز. نحتاج إلى:

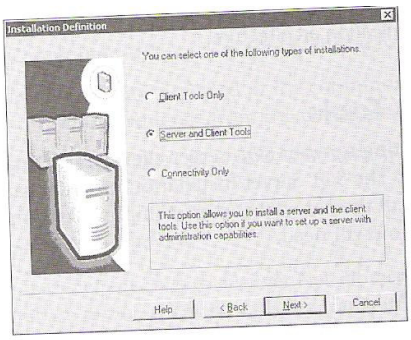
- معالج Intel، أو أي معالج آخر متوافق معه، بسرعة تبلغ 166 ميغاهرتز كحدّ أدنى؛
- ذاكرة رئيسية RAM لا تقل عن 32 ميغابايت، مع العلم أن بعض التطبيقات تتطلب 64 ميغابايت على الأقل؛
- مساحة تخزين على القرص الصلب لا تقل عن 95 ميغابايت، فقط لتطبيقات إدارة الزبون؛
- نظام تشغيل مناسب. يمكن استخدام Windows NT 4.0 أو أي إصدار من Windows 2000 أو Windows Me أو Windows 98، مع العلم أن Windows XP يدعم بعض إصدارات SQL Server 2000، وأه يمكننا استخدام Windows 95 لأغراض الاتصال فقط؛
- تطبيقات شبكيّة:

تمتلك أنظمة تشغيل Windows NT وَ Windows 2000 وَ Windows Me وّ Windows XP وَ Windows 98 وَ Windows 95 تطبيقات شبكية مُضمّنة فيها، في حين تتطلب Banyan Vinos أو Apple Talk ADSP تطبيقات شبكية إضافية، مع العلم أن بروتوكول NW Link يقدّم دعماً لزبائن Novel NetWare IPX/SPX.

خصائص عملية التنصيب

يمكننا إجراء عملية تنصيب أدوات وبرمجيات الزبون بطريقتين، إما من خلال برنامج تنصيب SQL Server 2000 نفسه، أو من خلال تنفيذ ملف موجود في القرص المدمج الخاص بـ SQL Server 2000؛

تعتبر الطريقة الأولى هي الأبسط والأوضح لعملية التنصيب تلك، بحيث يمكننا من خلالها تحديد عدّة خيارات، كما يوضح الشكل التالي:



- تنصيب أدوات الزبون فقط
- تنصيب أدوات المخدِّم والزبون
- تنصيب مكوّنات الاتصال مع المخدِّم فقط

يمكننا إجراء عملية تنصيب أدوات وبرمجيات الزبون بطريقتين، إما من خلال برنامج تنصيب SQL Server 2000 نفسه، أو من خلال تنفيذ ملف موجود في القرص المدمج الخاص بـــ SQL Server 2000؛

تعتبر الطريقة الأولى هي الأبسط والأوضح لعملية التنصيب تلك، بحيث يمكننا من خلالها تحديد عدّة خيارات، كما يوضح الـــشكل المُرافق.

- يمكننا من خلال هذه الواجهة، اختيار تنصيب أدوات الزبون فقط، أو أدوات المخدِّم والزبون معاً، أو الملفات المسؤولة عـــن عملية الاتصال بين المخدِّم والزبون، مع العلم أن اختيار تنصيب أدوات الزبون فقط يتضمن أيضاً ملفات الاتصال؛

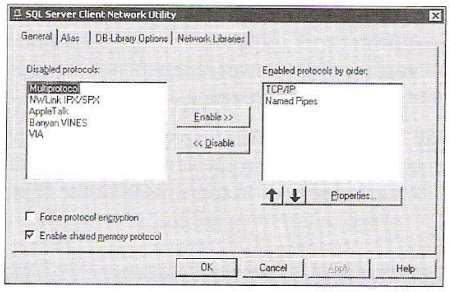
- يقوم برنامج التنصيب الخاص بالزبون بنسخ ملفات SQL Server Client –وبشكل تلقائي– إلى نفس الجـــزء المخـــصص لملفات نظام التشغيل من القرص الصلب، ولا يمكننا القيام بتغيير ذلك المسار، مع العلم أن عملية التنصيب تلك ستؤدي إلى استبدال أدوات الزبائن الخاصة بـــ SQL Server 7.0 –في حال وجودها– بأدوات الزبائن الخاصة بـــ SQL Server 2000، ولا يمكننا الحفاظ على الأدوات القديمة حتى ولو كان المخدِّم يحتوي على كل من SQL Server 2000 وَ SQL Server 7.0.

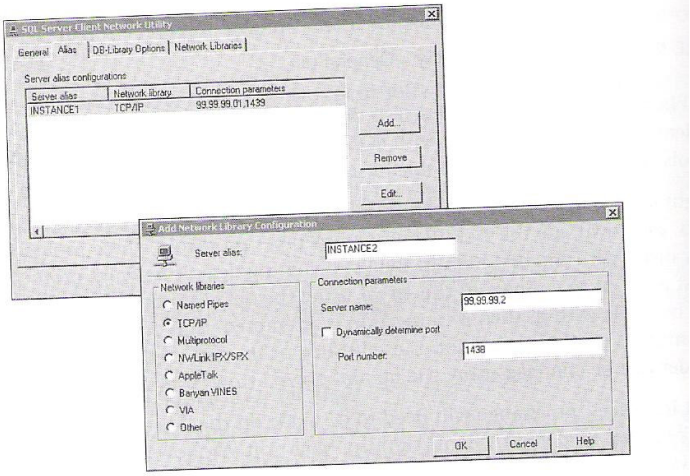
أما بالنسبة للطريقة الأخرى المتّبعة في عملية التنصيب، فيمكن تنصيب أدوات SQL Server Client من خـــلال تنفيـــذ الملـــف sqlredis.exe الموجود ضمن القرص الخاص بـــ SQL Server 2000، مع العلم أن هذه العملية تـــسمح بتنـــصيب المكوّنـــات المسؤولة عن الاتّصال مع المخدِّم فقط، ولا يمكننا تنصيب أدوات الزبون من خلالها؛

تتميز هذه الطريقة في التنصيب، في كونها بسيطة وهادئة، بحيث تؤدي إلى نسخ مباشر لكل من المكونات الأساسية لـــ OLE DB و ODBC، بالإضافة إلى مزوِّد OLE DB الخاص بـــ SQL Server ، ومتحكِّم SQL Server ODBC، والمكتبات الـــشبكيّة التلقائية الخاصة بـــ SQL Server Client.

إعداد الزبون

- تعتبر عملية إعداد الزبون، الخطوة التالية الواجب تنفيذها من أجل إتمام عملية التنصيب،وهي عبارة عن إعداد اتصالات حاسب الزبون ليصبح قادراً على التواصل مع المخدِّم؛
- يمكننا استخدام الإعدادات الشبكيّة التلقائية –التي يتم وضعها من قبل أدوات SQL Server Client– دون أي تعديلات إضافية باستثناء تحديد اسم جهاز المخدِّم الذي ينبغي على الزبائن أن تتصل به.
- تختلف الإعدادات في الحالات التي تتطلب أن يتّصل الزبون بعدّة مخدِّمات، يمكن أن تستَخدم عدّة بروتوكولات اتصال عبر الشبكة؛
- يمكننا من خلال أدوات SQL Server Client أن نحدد كافة الخصائص اللازمة لعملية التواصل الأخيرة تلك، ولكن، قبل البدء بتلك العملية ينبغي أن نضمن توافر مكتبات شبكيّة متوافقة على كل من حاسبات المخدِّم والزبائن، بالإضافة إلى ضرورة استخدام بروتوكولات اتصال مناسبة لتلك المكتبات على كل من الطرفين؛
- يمكننا أن نضمن تنصيب المكتبات الشبكيّة من خلال برنامج تنصيب SQL Server Client، في حين ينبغي التأكد من توافر بروتوكولات الاتصال المستخدَمة، والتي يتم تنصيبها عادةً كجزء من عملية تنصيب نظام التشغيل؛
- يتم –بعد الانتهاء من عملية تحديد المتطلبات الأولية السابقة– إعداد برمجيات الزبون للاتصال مع عدّة نسخ مختلفة من المخدِّمات من خلال إحدى الطريقتين التاليتين:
  - اختيار البروتوكولات الشبكيّة المناسبة للتواصل مع المخدِّم وذلك بتحديد أسماء المخدِّمات المتوافرة على الشبكة، من خلال أداة SQL Server Client وكما يوضح الشكل التالي:





- إنشاء مصادر معطيات ODBC مناسبة تسمح بالتواصل مع المخدِّمات المطلوبة باستخدام مجال عنونة معروف DNS، مع العلم أنه يمكننا استخدام JDBC عوضا عن ODBC عندما تلعب تطبيقات Java دور الزبون الذي يطلب الاتصال.

الاتصال مع SQL Server من خلال الانترنت

- يمكننا ومن خلال إتاحة إمكانية الولوج إلى SQL Server من خلال شبكة الانترنت، أن نجعل المعطيات متاحة لكافة الزبائن حول العالم؛
- تتيح هذه العملية إمكانية رفع مستوى التشارك بالمعطيات إلى درجة عالية جداً، إلا أنها –وفي الوقت نفسه– تضيف عبئاً إضافياً فيما يتعلق بضرورة حماية المعطيات؛
- ينبغي لكي يتم إنشاء الاتصال مع SQL Server من خلال الانترنت، أن يكون كلا الطرفين من المخدِّم والزبون متصلان بالشبكة العالمية، كما يتوجب على المخدِّم أن يدعم تشغيل بروتوكول TCP/IP. وبتوافر هذين المُتطلَّبين يمكن للزبون التواصل مع المخدم من خلال استخدام عنوان IP معين أو اسم مجال مناسب (DNS)؛
- تختلف طرائق حماية معطيات المخدم التي يمكن اتّباعها عند استخدام SQL Server على شبكة الانترنت، بحيث يمكن –من جهة– أن نقوم بإخفاء المخدِّم خلف جدار ناري لعزل الشبكة عن المستخدمين الذين يقومون بالولوج إلى الأجهزة المخصصة للانترنت، بحيث يتم استقبال الطلبات من عناوين TCP/IP محددة دون أخرى. كما يمكننا من جهة أخرى أن نقوم بحماية المعطيات من خلال فرض أسلوب اتصال مشفر ما بين المخدِّم والزبائن، يضمن أن يطلب كل زبون جديد الولوج إلى المخدِّم قبل الحصول على المعطيات.

الفصل الخامس

عنوان الموضوع:

إدارة نظام SQL Server وقواعد معطياته.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

تتناول هذه الجلسة كيفية إدارة نظام SQL Server 2000، وإدارة قواعد معطيات النظام المعرّفة فيه، كما تستعرض الطرائـــق المختلفة التي تسمح بالولوج إلى جداول النظام والاستعلام عن معطياتها.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- قواعد معطيات النظام
  - قاعدة المعطيات master
  - قاعدة المعطيات msdb
  - قاعدة المعطيات model
  - قاعدة المعطيات tempdb
  - قاعدة المعطيات Distribution
- جداول النظام
  - جداول دليل النظام وخصائصها
  - جداول دليل قاعدة المعطيات وخصائصها
- إجرائيات النظام المخزّنة وأشهر أنواعها
- مناظير مخططات المعلومات
- توابع النظام

- سنركز من خلال الشرائح التالية على كيفية إدارة نظام SQL Server، كما سنعمل على دراسة قواعد معطيات النظام وكيفية إدارة جداولها، والطرائق الأخرى المختلفة التي تسمح لمدير قواعد المعطيات تلك بإنشاء الاستعلامات واستعراض النتائج؛

- تعتبر المفاهيم التي سنقوم بدراستها، هامة جداً في عملية الإدارة تلك، فكما يقوم الميكانيكي بإصلاح السيارة عندما يصيبها عطل ما، كذلك ينبغي على مدير قاعدة المعطيات أن يتحمل كافة المسؤوليات المتعلقة بصيانة وإصلاح وإدارة قاعدة معطيات المؤسسة التي يعمل بها.

مسؤوليات مدير قاعدة المعطيات

- يمكن حصر مسؤوليات مدير قاعدة المعطيات في مهمتين أساسيتين:
  - ضمان تكامل المعطيات؛
  - ضمان إتاحة المعطيات لكافة المستخدمين.

- أهمية المعطيات في الشركات؛
- تتراوح مهمات مدير قاعدة المعطيات من إعداد التصميم الفيزيائي، إلى تنصيب SQL Server، أو حتى تنفيذ التصميم المنطقي، وتحسين الأداء، بالإضافة إلى مهمات أخرى تتعلق بالتخزين الاحتياطي واسترجاع المعطيات؛
- يُصنّف عمل مدير قاعدة المعطيات على أنه الأكثر توتّراً وتطلُّباً ومتعةً واحتراماً؛
- يُتوقَّع من مدير قاعدة المعطيات أن **يعرف** كل شيء وأن **يرى** كل شيء وأن **يتنبَّأ** بكل شيء.
- يمكن حصر مسؤوليات مدير قاعدة المعطيات في مهمتين أساسيتين هما: ضمان تكامل المعطيات وضمان إتاحتها للمستخدمين؛
- يمكننا اعتبار تلك المهمات، بسيطة من حيث المفهوم، ولكن في الحقيقة، يترتب عنها مسؤوليات كبير جداً، فعلى سبيل المثال: تقدِّر بعض الشركات قيمة معطياتها بمليون دولار لكل 100 MB، كما أن شركات أخرى قد تفشل أو تُفَلَّس في حال فقدان معطياتها؛
- يختلف التوصيف الحقيقي لمهمات مدير قاعدة المعطيات بشكل كبير، ففي بعض الحالات –حالات الشركات الصغيرة– يمكن أن تتراوح مهمات مدير قاعدة المعطيات من إعداد التصميم الفيزيائي، إلى تنصيب SQL Server، أو حتى تنفيذ التصميم المنطقي، وتحسين الأداء، بالإضافة إلى مهمات أخرى تتعلق بالتخزين الاحتياطي واسترجاع المعطيات. في حين يمكن تخصيص المهمات السابقة بشكل أدقّ في الشركات الأكبر حجماً؛

- يمكننا تصنيف المهمات الأساسية في إدارة قاعدة المعطيات من خلال النقاط التالية:

- تنصيب وإعداد SQL Server؛
- تخطيط وتصميم قواعد المعطيات؛
- إدارة تخزين المعطيات؛
- ضمان الأمن والتحكم بطرائق الولوج؛
- توليف قاعدة المعطيات؛
- تنفيذ عمليات التخزين الاحتياطي واسترجاع المعطيات، بالإضافة إلى عمليات التعافي.

- تُنسَب عادةً مهمات إدارة الإجرائيات المُخزَّنة إلى مدير قاعدة المعطيات أيضاً، وذلك لأنها تُخزَّن كأغراض في قاعدة المعطيات، على الرغم من أنه يمكن تصنيفها في الإطار العام من ضمن مهمات المطوّرين لما تحتويه من رماز Transact-SQL معقّد، كما ينبغي الإشارة إلى ضرورة توخي الحذر عند تعامل مدير قاعدة المعطيات مع الإجرائيات المخزّنة من دون التعاون مع المطوّرين؛

- يُصنّف عمل مدير قاعدة المعطيات على أنه الأكثر توتّراً وتطلُّباً، في حين أنه في الوقت نفسه الأكثر متعةً واحتراماً؛

- يُتوقَّع من مدير قاعدة المعطيات أن **يعرف** كل شيء وأن **يرى** كل شيء وأن **يتنبَّأ** بكل شيء، بحيث يُكافئ المدير بمقدار الجهود التي يقدّمها.

قواعد معطيات النظام

- يقوم برنامج تنصيب SQL Server بإنشاء أربعة قواعد معطيات –بشكلٍ تلقائي– أثناء سير تلك العمليّة
- قاعدة المعطيات master
- قاعدة المعطيات msdb
- قاعدة المعطيات model
- قاعدة المعطيات tempdb
- قاعدة المعطيات Distribution

- يقوم برنامج تنصيب SQL Server بإنشاء أربعة قواعد معطيات –بشكلٍ تلقائي– أثناء سير تلك العمليّة؛
- تعتبر قواعد المعطيات تلك هامة جداً في دعم عمليات SQL Server، كما أنها توصف بأنها تحتوي على معطيات مترفِّعة (سامية) (Meta Data)، أي معطيات حول المعطيات، مع العلم أنه لا يمكن حذف قواعد المعطيات تلك؛

سنعرض فيما يلي وصفاً أولياً لقواعد المعطيات التي يُنشئها SQL Server:

- قاعدة معطيات master:

تعتبر قاعدة المعطيات master كما يظهر من اسمها، المخزن الرئيسي لكافة معلومات SQL Server، فكافة الأغراض المُعرَّفة على مستوى المخدِّم من حسابات ولوج، وإعدادات النظام، وإجرائيات النظام المخزَّنة، وغيرها، تُخزَّن في قاعدة المعطيات master، مع العلم أنه لا يمكن أن يجري تشغيل SQL Server إذا ما تعرّضت هذه القاعدة لمشاكل أو أضرار، بالتالي، يُفضل القيام بتخزين نسخة احتياطية منها لضمان استرجاعها في حال حدوث مشاكل؛

- قاعدة معطيات msdb: تُخزَّن في قاعدة معطيات msdb معلومات متعددة تتعلق بوكيل SQL Server، كما تُخزَّن فيها معلومات مختلفة أخرى تتعلق بالعمليات أو الأعمال المعرَّفة أو رسائل التنبيه بالإضافة إلى المعلومات التي تتعلق بعمليات التخزين الاحتياطي، مما يزيد من أهمية تخزين نسخة احتياطية من قاعدة المعطيات msdb حتى ولو لم يجر استخدام وكيل SQL Server؛

- قاعدة معطيات model: تُعتبر قاعدة معطيات model، القالب الذي تعتمد عليه كافة قواعد المعطيات الأخرى التي يجري إنشاؤها من قبل المستخدمين، فكل قاعدة معطيات ينبغي أن تتضمن مجموعة أساسية من الأغراض التي تُعرَف باسم دليل قاعدة المعطيات، فمع كل إنشاء لقاعدة معطيات جديدة، يجري نسخ عدّة أغراض من قاعدة المعطيات model إلى قاعدة المعطيات الجديدة، مع العلم أنه يمكننا تعريف أغراض جديدة وإضافتها إلى قاعدة المعطيات model.

مثال:

يمكننا أن نُنشئ جدول معين في قاعدة المعطيات model، سنلاحظ بعد ذلك إنشاء ذلك الجدول في كل قاعدة معطيات جديدة نقوم ببنائها.

إن الاحتفاظ بنسخة احتياطية من قاعدة المعطيات model يسمح لنا بالحفاظ على التعديلات التي قمنا بإجرائها على قاعدة المعطيات تلك، بالتالي يمكننا استخدامها في كل مرة نقوم بتنصيب SQL Server، أو في الحالات التي يمكن أن تتضرر فيها قاعدة المعطيات model.

- قاعدة معطيات tempdb: تُخزَّن في قاعدة المعطيات tempdb كافة الأغراض المؤقتة في SQL Server، من الجداول والإجرائيات المُخزَّنة المؤقتة التي يجري إنشاؤها بشكل صريح، بالإضافة كافة الأغراض المؤقّتة الأخرى التي يجري إنشاؤها واستخدامُها من قبل النظام؛

تعتبر قاعدة المعطيات tempdb بمثابة مسوَّدة لكافة الأعمال التي يقوم بها SQL Server، فعلى سبيل المثال، ينفذ SQL Server العديد من عمليات ترتيب المعطيات باستخدام قاعدة المعطيات tempdb قبل إعادة النتائج إلى إجراء المستخدم، كما يعتمد SQL Server كثيراً على قاعدة المعطيات هذه في إنشاء الفهارس؛

إن تخزين قاعدة المعطيات tempdb على حيِّز خاص من القرص مهم جداً ويمتلك ميّزات كبيرة خاصةً في سبيل تحسين الأداء؛

لا داعٍ للحفاظ على نسخة احتياطية من tempdb، خاصةً وأن SQL Server يقوم بإنشائها في كل مرّة يُقلِع فيها.

- قاعدة معطيات Distribution: لا تعتبر قاعدة المعطيات Distribution من ضمن أنواع قواعد المعطيات التي يجري إنشاؤها بشكل تلقائي على الرغم من أنها تعتبر تقنياً من ضمن أنواع قواعد معطيات النظام؛

يجري تنصيب قاعدة المعطيات Distribution بشكل تلقائي عندما يجري اختيار تنصيب خاصة التكرارية في SQL Server.

جداول النظام

- يستخدم SQL Server جداولاً خاصة يطلق عليها اسم "جداول النظام" وذلك لتخزين الإعدادات الخاصة به، بالإضافة إلى المعلومات الخاصة بأغراضه؛

يتم تصنيف جداول النظام في نوعين أساسيين، هما:

- جداول system catalog تحتوي على معلومات عن عملية التنصيب ككل؛
- جداول database catalog تحتوي على معلومات خاصة عن كل قاعدة معطيات.

يستخدم SQL Server جداولاً خاصة يُطلق عليها اسم "جداول النظام" وذلك لتخزين الإعدادات الخاصة به، بالإضافة إلى المعلومات الخاصة بأغراضه؛
يُشار عادة إلى جداول النظام بالرمز sys، بحيث يسبق ذلك الرمز اسم الجدول ليعبِّر عنه، مثلا: sysdatabases؛
يجري إنشاء بعض أنواع جداول النظام على شكل مناظير، مثل جدول syslogins؛

يجري تصنيف جداول النظام في نوعين أساسيين، هما:

- جداول system catalog:

تحتوي جداول دليل النظام على معلومات عن عملية التنصيب ككل، ويتم تخزين هذه الجداول في قاعدة المعطيات master؛

- جداول database catalog:

تحتوي جداول دليل قاعدة المعطيات على معلومات خاصة عن كل قاعدة معطيات، ويجري تخزين هذه الجداول في كل قاعدة معطيات بالإضافة إلى قاعدة المعطيات master. تمتلك قاعدة المعطيات msdb –بالإضافة إلى دليل قاعدة المعطيات الخاص بها– جداول نظام إضافية ترتبط بوكيل SQL Server وبعمليات الخزن الاحتياطي والاسترجاع بالإضافة إلى جداول خاصة بتخزين السّجلّ ونقله (مع العلم أن الخاصة الأخيرة تتوافر فقط في الإصدار Enterprise Edition)؛
من الجدير بالذكر، أنه يجري إنشاء جداول نظام إضافية في كل من قاعدة المعطيات master وقاعدة المعطيات Distribution، عندما نقوم بإعداد خاصة التكرارية في SQL Server.

سنعرض في الشرائح التالية وصفاً سريعاً لمعظم جداول دليل النظام ودليل قاعدة المعطيات.

الاسم		الوصف
Sysaltfiles		مات حول ملف قاعدة المعطيات tempdb، وبما أن قاعدة المعطيات تلك يتم إنشاؤها عند للأداة SQL Server، بالتالي لابد من تخزين هذا الجدول في قاعدة المعطيات master.
Syscharsets		يتضمن مجموعات المحارف المختارة أثناء عملية التنصيب بالإضافة إلى طلبات الترتيب.
Sysconfigures		من قيم إعدادات النظام التي سيجري استخدامها عند تشغيل الأداة SQL Server.

Syscurconfigs		ضمن قيم إعدادات النظام الحالية، وهو جدول يتم إنشاؤه في كل مرة يُستعلم عنه.
Sysdatabases		يتضمن معلومات عن كافة قواعد المعطيات.
Sysdevices		ضمن مسار كل ملف قاعدة معطيات بالإضافة إلى مسار ملفات النَسْخ الاحتياطي.
Syslanguages		لكل لغة منصَّبة على المخدِّم، مع العلم أنه لا يحتوي على مسار اللغة الإنكليزية على اعتبار أنها اللغة التلقائية المنصّبة تلقائياً.
Sysloginns		يتضمن معلومات عن كل مستخدم للنظام، اعتماداً على الجدول sysxlogins.
Sysmessages		يتضمن كافة رسائل الأخطاء ورسائل التحذير.
sysoledbusers		عن أسماء المستخدمين وكلمات المرور المستخدمة للاتصال مع المخدمات، ويعتمد على الجدول
sysperfinfo		حول العدَّادات المستخدمة لقياس الأداء والتي يمكن عرضها من خلال Performance
Sysprocesses		حول كافة الإجراءات التي تعمل على SQL Server والتي تتعلق بالمخدِّم أو بالزبون.
Sysremotelogins		مستخدم بعيد يستطيع أن يستدعي إجرائيات مُخزَّنة.
Sysxlogins		عن كافة المستخدمين.

جداول دليل النظام

فيما يلي عرضٌ لأهم الجداول والمناظير التي تتوافر في دليل النظام:

جداول دليل قاعدة المعطيات

فيما يلي عرضٌ لأهم الجداول والمناظير التي تتوافر في دليل قاعدة المعطيات:

الاسم		الوصف
Syscolumns		ضمن مسار كل عمود في كافة الجداول والمناظير، بالإضافة إلى كل المعاملات في كافة الإجرائيات المخزّنة المستخدمة.
Syscomments		يتضمن مسار كل منظار أو قاعدة أو قادح أو شرط اختبار أو شرط تلقائي أو إجرائية مُخزَّنة، كما يتضمن نص استعلام SQL المستخدم لبناء ذلك الغرض.
Sysconstraints		يربط الشروط بالأغراض التي تمتلكها، اعتماداً على الجدول sysobject.
Sysdepends		يخزن العلاقات ما بين الأغراض التي تعتمد على بعضها البعض، كالعلاقة ما بين الجداول والمناظير.
Sysfiles		يتضمن معلومات عن كافة ملفات قاعدة المعطيات،

		بالإضافة إلى اسم وحجم الملفات الفيزيائية والمنطقية.
Sysforeignkeys		يتضمن علاقات الـــ foreign keys الموجودة بين الجداول في قاعدة المعطيات.
Sysindexes		يتضمن معلمات حول الفهارس المستخدمة في قاعدة المعطيات.
Sysindexkeys		يتضمن معلومات عن كافة مفاتيح الفهارس المستخدمة.
Sysmembers		يتضمن معلومات حول كافة الأعضاء في قاعدة المعطيات، وذلك من أجل توزيع الأدوار.
Syspermitions		يتضمن معلومات حول كافة السماحيات الممنوحة أو المنتزعة من المستخدمين أو المجموعات أو الأدوار.
Sysusers		يتضمن معلومات حول مختلف أنواع المستخدمين الذين يملكون سماحيات ولوج إلى قاعدة المعطيات، سواء كان مستخدم Windows مجموعة Windows أو مستخدم SQL Server أو دور SQL Server.

جداول النظام في قاعدة معطيات msdb

بالإضافة إلى جداول دليل قاعدة المعطيات القياسية، فإن قاعدة المعطيات msdb تحتوي على جداول خاصة بوكيل SQL Server أو بعمليات الخزن الاحتياطي أو بعمليات نقل السجّل بالإضافة إلى الجداول الخاصة بخطط صيانة قاعدة المعطيات؛

فيما يلي عرضٌ لأهم جداول وكيل SQL Server:

الاسم	الوصف
Sysalerts	يتضمن اسم ورقم وكافة المعلومات الأخرى المتعلّقة بكافة رسائل الخطأ المعرّفة.
sysdownloadlist	صفّ من الأعمال التي ينبغي تنزيلها من مخدِّمات أخرى.
sysjobhistory	يتضمن سجل محفوظات لكافة الأعمال التي جرى إنجازها.
sysjobs	يتضمن معلومات حول كافة الأعمال الخاصة بوكيل SQL Server.
sysjobschedules	يتضمن معلومات عن كافة الأعمال المجدولة لوكيل SQL Server.
sysjobsteps	يتضمن معلومات حول الخطوات المرسومة لكل عمل من أعمال وكيل SQL Server.

فيما يلي عرضٌ لأهم جداول الخزن الاحتياطي المستخدمة:

الاسم	الوصف
Backupfile	يتضمن معلومات حول كل ملف سجلّ أو ملف معطيات سبق أن جرى تخزينه احتياطياً.
Retorefile	يتضمن معلومات حول كل ملف جرى استرجاعه.
Restorehistory	يتضمن معلومات حول كافة عمليات الاسترجاع السابقة.

فيما يلي عرضٌ لجداول خطط الصيانة المستخدمة:

الاسم	الوصف
sysdbmaintplan_databases	يمثل سطراً لكل قاعدة معطيات مضمَّنة في خطة الصيانة.
sysdbmaintplan_history	عبارة عن سجلّ محفوظات يمثل خطط الصيانة المُنفَّذة.
sysdbmaintplan_jobs	يمثل سطراً لكل عمل في خطة الصيانة.
Sysdbmaintplans	يمثل سطراً لكل خطة صيانة، ويتضمن الاسم والمعرِّف وتاريخ الإنشاء.

فيما يلي عرضٌ لبعض الجداول المستخدمة في نقل السجلّ:

الاسم	الوصف
log_shipping_databases	يمثل اسم ومعرِّف قاعدة المعطيات التي يجري نقلها.
log_shipping_plan_history	يتضمن معلومات حول كافة خطط نقل السجلات التي جرى استخدامها.
log_shipping_plans	يتضمن معلومات حول كل خطة نقل سجلّ يجري استخدامها.

بالإضافة إلى جداول دليل قاعدة المعطيات القياسية، فإن قاعدة المعطيات msdb تحتوي على جداول خاصة بوكيل SQL Server أو بعمليات الخزن الاحتياطي أو بعمليات نقل السجّل بالإضافة إلى الجداول الخاصة بخطط صيانة قاعدة المعطيات؛

إجرائيات النظام المخزّنة

- الهدف من إنشاء إجرائيات النظام المخزَّنة:

تعتبر إجرائيات النظام المُخزّنة –التي يجري إنشاؤها أثناء سير عملية تنصيب الأداة SQL Server– مكمِّلةً لعمليـــة إدارة SQL Server، فالهدف الأساسي من إنشائها هو تأمين وسيلة تساعد مدير قاعدة المعطيات في استعلام أو تعديل جداول دليل النظـــام أو جداول دليل قاعدة المعطيات من دون الولوج بشكل مباشر إلى تلك الجداول؛

- الفوائد المتوقّعة من إجرائيات النظام المخزّنة
- خصائص إجرائيات النظام المخزّنة:
  - مخزّنة في قاعدة المعطيات master
  - تتمتع بالبادئة _sp
  - يملكها dbo
  - ذات مدى عام
- محاذير إجراء التعديلات على إجرائيات النظام المخزّنة
- تمرين

– مكمّلةً لعملية إدارة SQL Serverتعتبر إجرائيات النظام المخزّنة –التي يتم إنشاؤها أثناء سير عملية تنصيب الأداة ، فالهدف الأساسي من إنشائها هو وسيلة لمدير قاعدة المعطيات من أجل استعلام أو تعديل جداول دليل النظام أو SQL Server جداول دليل قاعدة المعطيات من دون الولوج بشكل مباشر إلى تلك الجداول؛

يمكننا من خلال إجرائيات النظام المخزّنة أن نقوم بكافة عمليات الإدارة التي كنا نقوم بها من خلال الأداة Enterprise Manager، فعلى سبيل المثال، يمكننا إضافة مستخدم SQL Server جديد من خلال الضغط بالزر اليميني على Logins في الأداة Enterprise Manager ثم اختيار New Login وتعبئة الحقول المطلوبة، ولكن ماذا نفعل إذا ما كنا نرغب بإضافة 200 مستخدم دفعة واحدة؟؟؟
إن استخدام الطريقة السابقة في تعريف مستخدمي SQL Server لا تفيد في هذه الحالة، ويكون الحل البديل ببناء مخطوطة نقوم من خلالها باستدعاء إجرائية النظام المخزّنة sp_addlogin مع المعاملات المناسبة من اسم المستخدم وكلمة المرور وغيرها، ثم ننفذ هذا المخطوط في كل مرة نرغب بإضافة المستخدمين الـ 200.

لن نستطيع من خلال هذه الجلسة أن نقوم باستعراض كافة إجرائيات النظام المخزّنة المتوافرة، وذلك لعددها الكبير، فبنظرة سريعة لإجرائيات قاعدة المعطيات master المخزّنة يمكننا أن نجد حوالي 1000 إجرائية مخزّنة، بالتالي تعتبر MSDN وَ Books Online –وهو ملف المساعدة المضمّن مع SQL Server– من أهم المراجع التي يمكن الاعتماد عليها للحصول على المعلومات، ذلك بالإضافة إلى الانترنت، كما يعتبر أسلوب عرض الإجرائيات المخزّنة في شجرة الأداة SQL Query Analyzer مفيداً بحيث يمكننا استعراض تلك الإجرائيات أو تنفيذها، مع العلم أن سحب إحداها إلى واجهة Query Analyzer والضغط على Shift + F1 يؤدي مباشرة عرض المعلومات التوثيقية المتعلّقة بتلك الإجرائية.

تتميز إجرائيات النظام المخزّنة بالخصائص المشتركة التالية:
- كلها مخزّنة في قاعدة المعطيات master؛
- تمتلك كافة الإجرائيات المخزّنة البادئة _sp؛
- يعتبر المستخدم dbo أو Database Owner مالك الإجرائية؛

- تمتلك كافة إجرائيات النظام المخزّنة مدى عام، أي أنه يمكن تنفيذها من أية قاعدة المعطيات، كما أن تنفيذها يجري ضمن سياق قاعدة المعطيات تلك.

- لا ينبغي إجراء تعديلات مباشرة على جداول النظام، ولا ينبغي إجراء أية تعديلات على إجرائيات النظام المخزّنة، فإذا كنا نرغب بتعديل إحداها، فيفضّل في هذه الحالة أن يجري نسخ تعريف تلك الإجرائية من عمود text في جدول syscomments أو من خلال خصائص تلك الإجرائية في الأداة Enterprise Manager، ثم إنشاء إجرائية جديدة وإجراء كافة التعديلات عليها. مع العلم أنه ينبغي توخّي الحذر الشديد أثناء محاولة تعديل محتوى إجرائيات النظام المخزّنة التي تقوم بدورها بإجراء تعديلات على جداول النظام، فهذه الإجرائيات معقدة وذات اعتماديات متعددة المستويات، وحصول أي خطأ أو ضياع في محتويات تلك الإجرائيات قد يودي بـ SQL Server ككل كما تُستبعَد المساعدة في هذه الحالات حتى من عملاء مايكروسوفت أنفسهم.

أشهر إجرائيات النظام المخزّنة المستخدمة

على الرغم من توافر المئات من إجرائيات النظام المخزّنة، إلا أننا في الحقيقة لن نستخدم أكثر من مجموعة من 10إلى 20 إجرائية مخزّنة في النظام الذي نقوم بإدارته؛

فيما يلي عرضٌ لأهم أنواع إجرائيات النظام المخزّنة وأكثرها شيوعاً:

اسم الإجرائية	الوصف
sp_help	تقوم بعرض قائمة بكافة أغراض قاعدة المعطيات، استخدامها لعرض معلومات عن غرض محدد، مع العلم أنه يمكننا استخدامها للحصول على معلومات عن أي غرض. **تمرين1**
Sp_helpdb	تقوم بعرض معلومات عن قواعد المعطيات ككل، أو عن قاعدة معطيات محددة.
Sp_helprotect	لعرض معلومات غرض معين أو عرض قائمة بالسماحيات المتاحة.
Sp_Who	تعيد معلومات حول اتصالات SQL Server الحالية. **تمرين4**
Sp_lock	لاستعراض معلومات القفل المطبقة على إجراء أو إجراءين أو كافة الإجراءات. وهي إجرائيات نظام مخزّنة مفيدة في حالات تشخيص الأقفال أو حالات الإقفال المتبادل.
Sp_configure	لعرض أو تعديل إعدادات التعريف العامة.

Sp_dboption	لعرض أو تعديل إعدادات قاعدة معطيات معينة.

على الرغم من توافر المئات من إجرائيات النظام المخزّنة، إلا أننا في الحقيقة لن نستخدم أكثر من مجموعة من 10إلى 20 إجرائية مخزّنة في النظام الذي نقوم بإدارته؛

طرائق أخرى في استعلام جداول النظام

اعتمدنا حتى الآن في كافة الشرائح السابقة، على إجرائيات النظام المخزّنة لاسترجاع المعلومات حول دليل النظام ودليل قواعد المعطيات، إلا أنه يوجد طريقتان أخريتان لاسترجاع تلك المعطيات المترفِّعة، هما:

- Information Schema Views
- System Functions

اعتمدنا حتى الآن في كافة الشرائح السابقة، على إجرائيات النظام المخزّنة لاسترجاع المعلومات حول دليل النظام ودليل قواعد المعطيات، إلا أنه يوجد طريقتان أخريتان لاسترجاع تلك المعطيات السامية، هما:

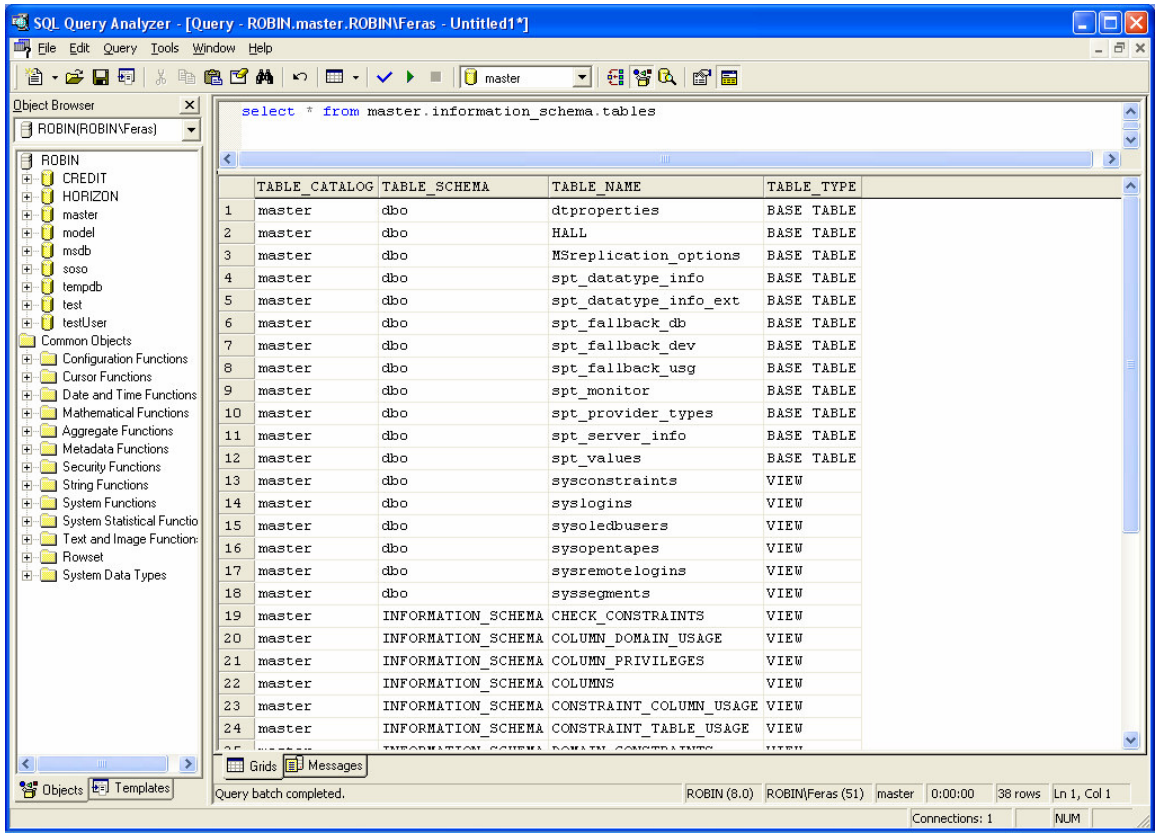
- Information Schema Views:

تعتبر مناظير مخططات المعلومات المزوَّدة من قبل ANSI-92، مناظيراً خاصة بتزويد معطيات النظام؛
يدعم SQL Server و ANSI-92 مخطط تسمية بثلاثة أجزاء وذلك عند الإشارة إلى غرض معين على مخدِّم محلِّي، تشير إليه ANSI-92 من خلال المصطلح *catalog.schema.object* في حين تشير إليه مايكروسوفت من خلال المصطلح *database.owner.object*، بحيث، تعتبر قاعدة المعطيات –وبشكل تلقائي– هي القاعدة الحالية إذا لم يتم الإشارة إليها، كما يعتبر المالك –وبشكل تلقائي– هو المستخدم الحالي إذا لم تتم الإشارة إليه.

مثال1:

تؤدي التعليمة التالية إلى استعراض معلومات حول جداول قاعدة المعطيات master:

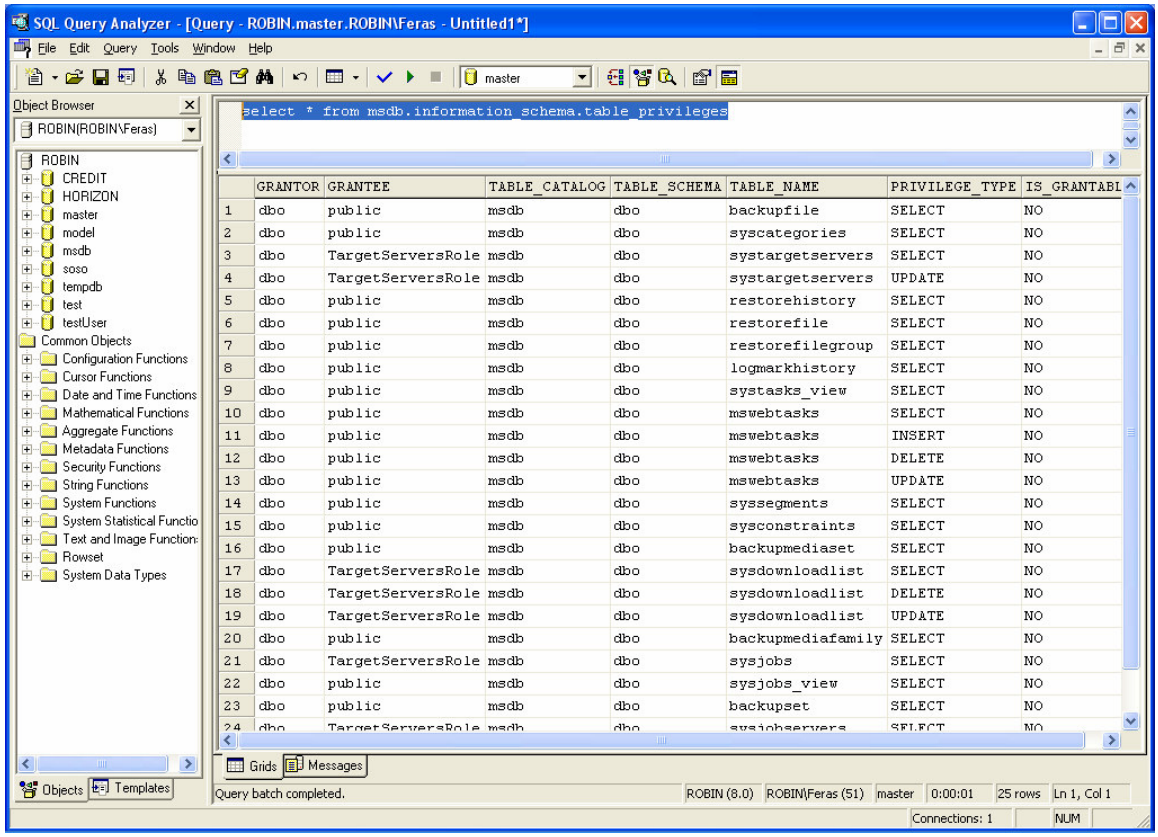
*select * from master.information_schema.tables*



مثال2:

تؤدي التعليمة التالية إلى استعراض معلومات حول السماحيات المتاحة على أغراض قاعدة المعطيات msdb:

*select * from msdb.information_schema.table_privileges*



لحسن الحظ، تبقى أسماء مناظير مخططات المعلومات معبِّرة نوعاً ما، ومنها على سبيل المثال:

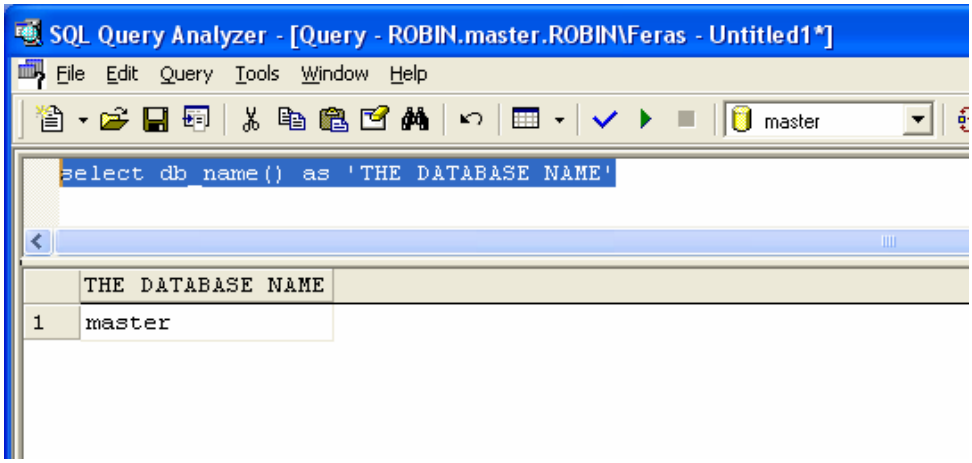
TABLES, COLUMNS, TABLE_PRIVILEGES, VIEWS.

- System Functions:

بإمكاننا أيضاً من خلال توابع النظام، أن نقوم باستعراض المعلومات حول المعطيات المترفِّعة المتعلّقة بدليل النظام أو دليل قواعد المعطيات، وذلك باستخدام Transact-SQL بشكل مباشر، بحيث يمكننا تنفيذ توابع خاصة تُعيد قيم محددة؛

فعلى سبيل المثال يمكننا استخدام التابع التالي للحصول على اسم قاعدة المعطيات الحالية:

select db_name() as 'THE DATABASE NAME'



ومن التوابع الأخرى الشهيرة:

SUSER_NAME()	لاسترجاع اسم المستخدم الحالي
DB_ID()	لاسترجاع معرف قاعدة المعطيات
DB_NAME()	لاسترجاع اسم قاعدة المعطيات
OBJECT_NAME()	لاسترجاع اسم غرض ما
	لاسترجاع التاريخ الحالي

الفصل السادس

عنوان الموضوع:

إنشاء وإدارة قواعد المعطيات.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

سنناقش في هذه الجلسة اسلوب إنشاء قواعد المعطيات وإدارتها، وسنتناول بالتفصيل مكوناتها، كما سنركز على التعليمات والأدوات المُستخدمة لتحقيق تلك المهمات.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- ما هي قاعدة المعطيات؟
- ملفات قاعدة المعطيات:
  - ملف المعطيات
  - ملف سجلّ المناقلات
- تخزين المعطيات في SQL Server:
  - الصفحة
  - الامتداد وأنواعه
- صفحات التحكم بملفات المعطيات
- كيف يمكننا إنشاء قاعدة المعطيات؟
- إدارة قاعدة المعطيات:
  - إدارة تضخم الملفات
  - توسيع وتقليص ملفات قاعدة المعطيات
  - التحكم بخصائص قاعدة المعطيات

- تعتبر قاعدة المعطيات، بالتأكيد، السبب الرئيسي لوجود SQL Server؛

- سنناقش في هذه الجلسة كيف يتم إنشاء قواعد المعطيات وإدارتها, وسنتناول بالتفصيل مكوناتها، كما سنركز على التعليمات والأدوات المستخدمة لتحقيق تلك المهمات؛

- ينبغي أن ندرك قبل البدء ببناء قاعدة المعطيات، أهمية القرارات التي يجري اتّخاذها أثناء عملية البناء والإدارة وكيف يمكن أن تؤثر على أداء النظام ككل، فعلى الرغم من سهولة التعامل مع SQL Server في عملية التصميم تلك، إلا أنه لابد لمدير قاعدة المعطيات من أن يأخذ بعين الاعتبار أن تصميم وإدارة قاعدة المعطيات لن ينتهي هنا، وأنه يترتب عليه القيام بالعديد من الإجراءات الأخرى مستقبلاً، خاصة فيما يتعلق بحفظ قاعدة المعطيات واسترجاعها أو القرارات التي ينبغي اتخاذها عندما تتضخم قاعدة المعطيات بشكل كبير، أو عند الحاجة لإدارة سجلات المناقلات وغيرها، وهو ما سنستعرضه في الجلسات القادمة.

قاعدة المعطيات

- تعريف:

تعبِّر قاعدة المعطيات عن بنية تخزينيّة خاصة لأغراض معينة من المعطيات؛

- ملفات قاعدة المعطيات:
  - يحتوي "**ملف المعطيات**" على كافة أغراض قاعدة المعطيات من جداول أو فهارس أو غيرها؛

يخزِّن "**ملف سِجلّ المناقلات**" كافة التغيرات التي تحدث على المعطيات

- تعبِّر قاعدة المعطيات عن بنية تخزينيّة خاصة لأغراض معينة من المعطيات.

- تتكون قاعدة المعطيات على الأقل من ملفين أساسيين، يُطلق على الأول اسم "ملف المعطيات" ويحتوي على كافة أغراض قاعدة المعطيات من جداول أو فهارس أو غيرها، ويُطْلق على الثاني اسم "ملف سِجلّ المناقلات" والذي يخزِّن كافة التغيرات التي تحدث على المعطيات.

- يشير كل ملف معطيات أو ملف سجلّ مناقلات، حصراً إلى قاعدة معطيات وحيدة –في الحالة العامة– ، أما في الحالات التي نقوم فيها بتعريف "مجموعات ملفات" ضمن قاعدة المعطيات، فيمكننا أن نربط كل ملف معطيات مع "مجموعة ملفات" قاعدة معطيات وحيدة.

تخزين المعطيات في SQL Server

- الصفحة:

كتل متتالية من 8KB ضمن "ملفات المعطيات"؛

- الامتداد:

عبارة عن سلسلة من ثماني صفحات، وله نوعان:

- الامتداد المختلط:

تُخزّن الامتدادات المختلطة صفحات من أكثر من غرض وحيد.

- الامتدادات المنتظمة:

لا يمكن أن تخزّن هذه الامتدادات إلا صفحات من غرض وحيد سواء كان جدول أو فهرس.

الصفحة:

- يخزن SQL Server المعطيات في كتل متتالية من 8KB ضمن "ملفات المعطيات"، تُعرف تلك الكتل باسم الصفحات؛
- تمثل الصفحة أصغر وحدة دخل/خرج يستخدمها SQL Server أثناء تبادل المعطيات من وإلى القرص، بحيث تقدّر سعتها بحوالي 8060 بايت من المعطيات في الحالات الطبيعية، مع العلم أنها يمكن أن تتحمل لغاية 8192 بايت.
- لا يمكن أن تتسع الصفحة لأكثر من سطر وحيد، وتعتبر هذه الميزة عاملاً هاما في قياس المساحة المستهلكة من القرص، فعلى سبيل المثال: إذا قُدِّر حجم سطر جدول معين بحوالي 4050 بايت، فإن مساحة القرص التي ينبغي حجزها لهذا الجدول تساوي ضعف حجم المعطيات المخزّنة على اعتبار أن الصفحة تتسع لسطر وحيد.

الامتداد:

- يستخدم SQL Server –أثناء تحصيص المساحات لجدول ما أو فهرس معين– بغية تخفيض تكلفة العمليات الداخلية وزيادة فعالية عمليات الدخل / خرج، ما يسمّى بالامتداد؛
- يُعرَف الامتداد على أنه عبارة عن تتالي ثماني صفحات، أو بعبارة أخرى، هو 64 KB من مساحة التخزين.
- هناك نوعين أساسيين من الامتدادات، الامتداد المختلط والامتدادات المنتظمة:
  - الامتداد المختلط:

يتم تحصيص المساحات الخاصة بكل جدول أو فهرس –في البداية– باستخدام الامتداد المختلط، تخزّن الامتدادات المختلطة صفحات من أكثر من غرض وحيد.

  - الامتدادات المنتظمة:

كما يبدو من اسم هذا النوع من الامتدادات، لا يمكن أن تخزّن هذه الامتدادات إلا صفحات من غرض وحيد سواء كان جدول أو فهرس، مما يسمح لـ SQL Server أن يحسّن من أداء عمليات القراءة والكتابة وأن يقلل من التجزئة المفروضة على القرص خاصةً وأن ملف المعطيات يتكوّن في هذه الحالة من مجموعات ثمانيّة من الصفحات.

ملفات قاعدة المعطيات

-صفحات التحكم-

- صفحات التحكم بملفات المعطيات:
  - صفحة الترويسة: تتضمن هذه الصفحة معلومات مختلفة عن الملف، كقاعدة المعطيات التي ينتمي إليها، ومجموعة الملفات المُحتوى فيها؛
  - صفحة مساحات التخزين الحرَّة: تتضمن هذه الصفحة معلومات مختلفة حول المساحات الحرَّة المتاحة على الصفحات التي يتكوَّن منها ملف المعطيات؛
  - صفحة خارطة التحصيص الشاملة: تتضمن هذه الصفحة معلومات مختلفة تتعلّق بتعقّب الامتدادات المحصّصة؛
  - صفحة خارطة التحصيص الشاملة الثانوية: تتضمن هذه الصفحة معلومات مختلفة تتعلّق بتعقّب الامتدادات المختلطة المحصّصة.

قبل أن نتكلم عن ملفات قواعد المعطيات وأنواعها، لابد لنا أولاً من الإشارة إلى كيفية تحكُّم SQL Server بالمساحات التي يقوم بتحصيصها لكل ملف من ملفات المعطيات تلك، **حيث يجري التحكم من خلال تحصيص صفحات خاصّة، في كل أول امتداد من كل ملف معطيات؛**

غالبا ما يتم تحميل صفحات التحكم تلك مباشرة إلى الذاكرة عند تشغيل SQL Server، خاصةً وأن المعطيات المخزّنة في تلك الصفحات كثيفة نسبياً بالإضافة إلى كثرة عمليات الولوج عليها؛

- تمثل الصفحة الأولى (page 0) في كل ملف معطيات، صفحة الترويسة في ذلك الملف، تتضمن هذه الصفحة معلوماتٍ مختلفةً حول الملف، كقاعدة المعطيات التي ينتمي إليها، ومجموعة الملفات المُحتوى فيها، بالإضافة على الحجم الأدنى للملف ومقدار تزايد النمو؛

- تمثل الصفحة الثانية (page 1) في كل ملف معطيات، صفحة مساحات التخزين الحرَّة في ذلك الملف، تتضمن هذه الصفحة معلوماتٍ مختلفةً حول المساحات الحرَّة المتاحة على الصفحات التي يتكوَّن منها ملف المعطيات، بحيث يمكن من خلالها أن يتم تتبُّع فيما إذا كانت الصفحات قد حصِّصت للأغراض المختلفة، وفيما إذا كانت الامتدادت المعرَّفة مختلطة أو منتظمة، بالإضافة إلى تقدير تقريبي للمساحة الحرّة المتبقية، مع العلم أن كل صفحة من هذا النوع يمكنها أن تتعقب المساحات الحرّة في أكثر من 8000 صفحة متتالية، كما يمكن تحصيص صفحات تعقُّب أخرى إذا ما اقتضت الحاجة لذلك؛

- تمثل الصفحة الثالثة (page 2) في كل ملف معطيات، صفحة خارطة التحصيص الشاملة في ذلك الملف، تتضمن هذه الصفحة معلومات مختلفة تتعلّق بتعقّب الامتدادات المحصّصة، بحيث يمكن لكل صفحة من هذا النوع أن تتعقّب ما يصل إلى حوالي 63,904 امتداد، بالإضافة إلى إمكانية تحصيص صفحات تعقُّب أخرى إذا ما اقتضت الحاجة إلى ذلك؛

- تمثل الصفحة الرابعة (page 3) في كل ملف معطيات، صفحة خارطة التحصيص الشاملة الثانوية في ذلك الملف، تتضمن هذه الصفحة معلومات مختلفة تتعلّق بتعقّب الامتدادات المختلطة المحصّصة، بحيث يمكن لكل صفحة من هذا النوع أن تتعقّب ما يصل إلى حوالي 63,904 امتداد مختلط، بالإضافة إلى إمكانية تحصيص صفحات تعقُّب أخرى إذا ما اقتضت الحاجة إلى ذلك.

ملفات قاعدة المعطيات

- الملفات الأولية
- الملفات الثانوية
- مجموعات الملفات
- ملف سجلّ المناقلات
- الملفات الأولية:

يُعتبر "الملف الأولي" أول ملف يتم التصريح عنه في عبارة بناء قاعدة المعطيات، ولا يمكن إنشاء أي قاعدة معطيات من دون هذا الملف. يتضمن الملف الأولي كافة أغراض النظام التي يجري نسخها من قاعدة المعطيات *model* والتي يشار إليها عادة باسم دليل قاعدة المعطيات؛

عادة ما يشار إلى الملف الأولي باللاحقة mdf.؛

- الملفات الثانوية:

يجري إنشاء ملفات معطيات إضافية لتخزين أغراض المستخدم بصورة مستقلة عن دليل قاعدة المعطيات، وذلك في سبيل تحقيق أهداف مختلفة تتعلق بالأداء أو التخزين الاحتياطي للمعطيات أو التعافي، يُطلق عادةً على ملفات المعطيات الإضافية تلك اسم الملفات الثانوية؛

عادة ما يشار إلى ملفات المعطيات الثانوية باللاحقة ndf. ؛

- مجموعات الملفات:

نستطيع من خلال مجموعات الملفات file groups أن نقرر فيما إذا كنّا نرغب بتخزين غرض محدد من قاعدة المعطيات في مكان معين من القرص، ويجري ذلك، أولاً بتعريف مجموعة ملفات محددة ضمن قاعدة المعطيات التي نعمل عليها، ثم بتوسيع قاعدة المعطيات لتشمل سوّاقة أخرى أو عدّة سواقات، وبعد ذلك يمكننا وَضع الأغراض التي نريد تخزينها في مجموعة الملفات الجديدة المُنشأة.

- ملف سجلّ المناقلات:

تتطلب كل قاعدة معطيات أيضاً ملف خاص يطلق عليه اسم ملف سجلّ المناقلات، ويمكننا تعريف أكثر من سجلّ مناقلات وحيد مع العلم أن استخدام مثل هذه العملية ليس بالأمر الشائع لأن ولوج تلك الملفات سيكون بشكل تسلسلي بحيث كلّما امتلأ أحد الملفات يجري الانتقال إلى الملف الثاني وبالتالي لا يُتوقع تغير مفيد في الأداء؛

عادة ما يشار إلى ملفات سجل المناقلات باللاحقة ldf.؛

بخلاف ملفات المعطيات، لا تنفذ ملفات سجلّ المناقلات عمليات الدخل/خرج في صفحات من KB 8، بحيث يقوم SQL Server بكتابة المناقلات التي يتم تنفيذها والتغيرات التي تحدث على المعطيات بأسرع طريقة ممكنة على ملف سجلّ المناقلات؛

يمكن أن يزداد حجم ملف سجلّ المناقلات مع الزمن، وكما لاحظنا من تعريف هذا النوع من الملفات، فلا داع لوجوده على نفس القرص مع ملفات قاعدة المعطيات الأخرى، بحيث يفضّل –تقنياً– أن يتم تخزينه على نظام جزئي منفصل.

إنشاء قواعد المعطيات

- يمكننا إنشاء قاعدة المعطيات عادةً بأستخدام إحدى الطرق التالية: إما من خلال تنفيذ مخطوط مكتوب بلغة T-SQL، أو من خلال استخدام الأداة Enterprise Manager؛
- يتوقف الزمن اللازم لإنشاء قاعدة المعطيات على عدد وحجم ملفات قاعدة المعطيات المعرّفة عند إنشائها، خاصةً وأنَّ إنشاء تلك الملفات يتم مع إنشاء قاعدة المعطيات نفسها؛
- يعيد SQL Server رسالة خطأ مناسبة وتتوقف عملية إنشاء قاعدة المعطيات وملفاتها المرافقة إذا ما لم تتوافر الحجوم المطلوبة لتخزين تلك الملفات على القرص.

إنشاء قاعدة المعطيات باستخدام Create Database Wizard

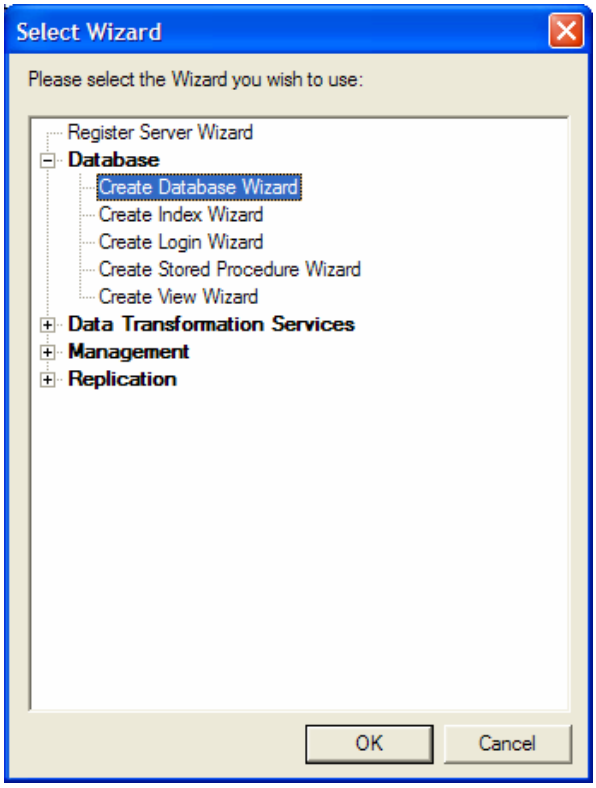
- تشغيل واجهات Create Database Wizard؛

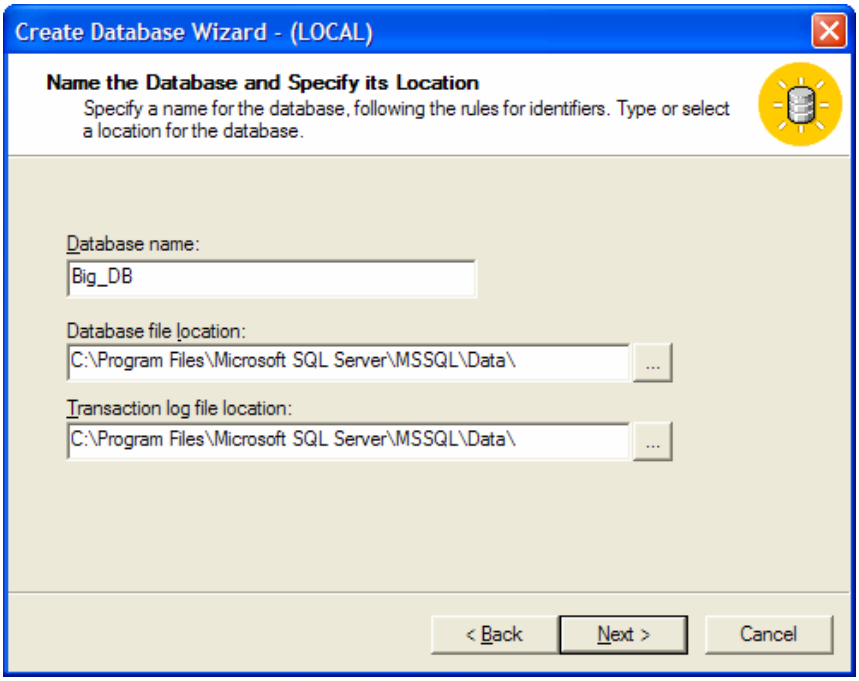
- يمكننا من خلال واجهة إنشاء قاعدة المعطيات باستخدام Wizard أن نقوم بإدخال خَيارات مختلفة لقاعدة المعطيات، كإسمها أو موقع ملف قاعدة المعطيات أو اسم ذلك الملف أو حجمه الأولي أو مقدار تزايده وغيرها من المعلومات؛

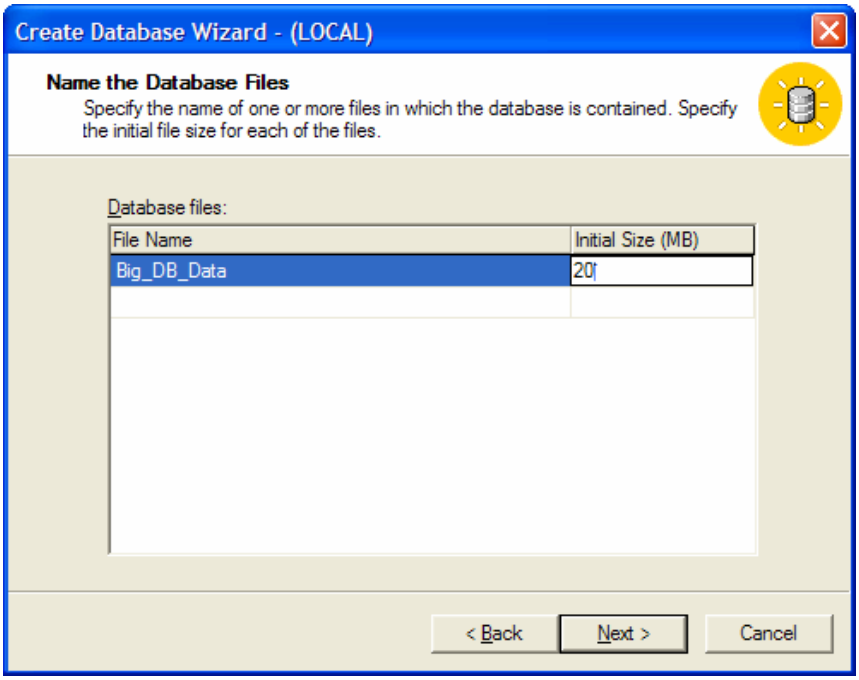
- التحكم بكيفية توزّع مواقع تخزين ملفات المعطيات غير متاح من خلال واجهات الـ Wizard.

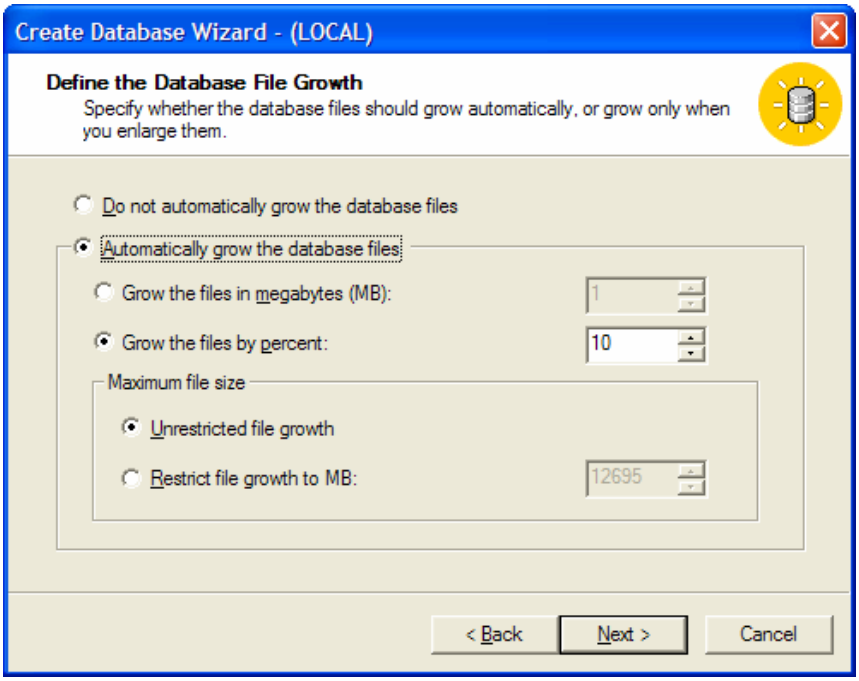
نستطيع من خلال واجهات Create Database Wizard أن نقوم بإنشاء قاعدة المعطيات خطوة بخطوة، يمكننا الوصول إلى هذه الواجهة إما من خلال أيقونة "العصا السحرية" في شريط أدوات الأداة Enterprise Manager أو من خلال الخَيار Wizard في قائمة الأدوات Tools، أو من خلال لوحة المهمات الخاصة بقاعدة المعطيات؛

يمكننا من خلال واجهة إنشاء قاعدة المعطيات باستخدام Wizard أن نقوم بإدخال خَيارات مختلفة لقاعدة المعطيات، كاسمها أو موقع ملف قاعدة المعطيات أو اسم ذلك الملف أو حجمه الأولي أو مقدار تزايده وغيرها من المعلومات؛


Select Wizard
Please select the Wizard you wish to use:
Register Server Wizard
Database
Create Database Wizard
Create Index Wizard
Create Login Wizard
Create Stored Procedure Wizard
Create View Wizard
Data Transformation Services
Management
Replication
OK
Cancel


Create Database Wizard - (LOCAL)
Name the Database and Specify its Location
Specify a name for the database, following the rules for identifiers. Type or select a location for the database.
Database name:
Big_DB
Database file location:
C:\Program Files\Microsoft SQL Server\MSSQL\Data\
...
Transaction log file location:
C:\Program Files\Microsoft SQL Server\MSSQL\Data\
...
< Back
Next >
Cancel


Create Database Wizard - (LOCAL)
Name the Database Files
Specify the name of one or more files in which the database is contained. Specify the initial file size for each of the files.
Database files:
File Name
Initial Size (MB)
Big_DB_Data
20
< Back
Next >
Cancel



تعتبر عملية إنشاء قاعدة المعطيات بالعملية البسيطة، وبشكل عام، لا داعٍ لاستخدام Wizard بعدّة صفحات لإنشائها، وخصوصاً أن أهم الخيارات التي ينبغي توصيفها أثناء إنشاء قاعدة المعطيات، وهي التحكم بكيفية توزّع مواقع تخزين ملفات المعطيات، غير متاحة لسوء الحظ من خلال واجهات الـ Wizard.

1.6.08 إنشاء قاعدة المعطيات باستخدام الأداة Enterprise Manager

- على الرغم من أن واجهات خصائص قواعد المعطيات في الأداة Enterprise Manager لا تقدّم المزيد من الخدمات في تحديد توزع ملفات قاعدة المعطيات، عمّا تقدمه واجهات Create Database Wizard، إلا أنها تقدم خدمات أخرى بأسلوب أكثر مرونة؛
- نلاحظ وجود ثلاث واجهات رئيسية، وهي:
  - الواجهة General: اسم قاعدة المعطيات وتحديد الاعدادات الإقليمية الخاصة بمعطياتها؛
  - الواجهة Data Files: الأسماء المنطقية لملفات قاعدة المعطيات والمسار الفيزيائي لتلك الملفات، بالإضافة مجموعة الملفات ومعاملات الامتداد الأخرى؛

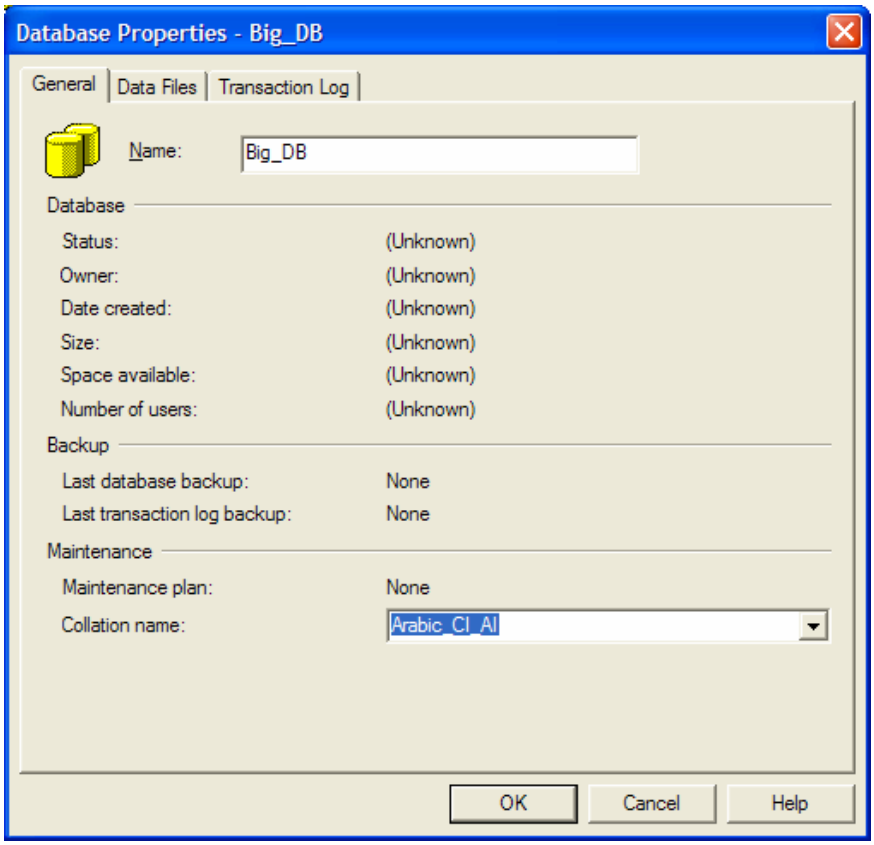
الواجهة Transaction Log: المعلومات المتعلّقة بملفات سجل أو سجلات المناقلا

- على الرغم من أن واجهات خصائص قواعد المعطيات في الأداة Enterprise Manager لا تقدّم المزيد من الخدمات فـــي تحديد توزع ملفات قاعدة المعطيات، عمّا تقدمه واجهات Create Database Wizard، إلا أنها تقدم خدمات أخرى بأسلوب أكثر مرونة؛

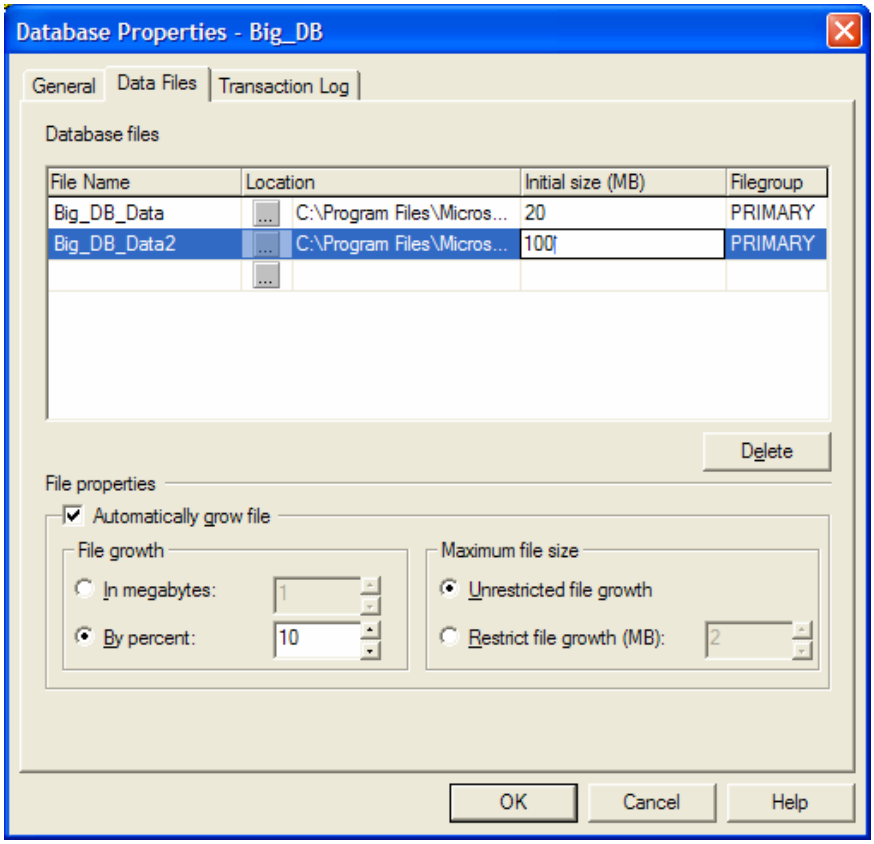
- يمكننا الوصول إلى هذه الواجهات من خلال اختيار New Database من قائمة المهمات السريعة الخاصة بقاعدة المعطيات المحددة في شجرة الأداة Enterprise Manager؛

- نلاحظ وجود ثلاثة واجهات رئيسية، وهي: General، Data Files، Transaction Log.

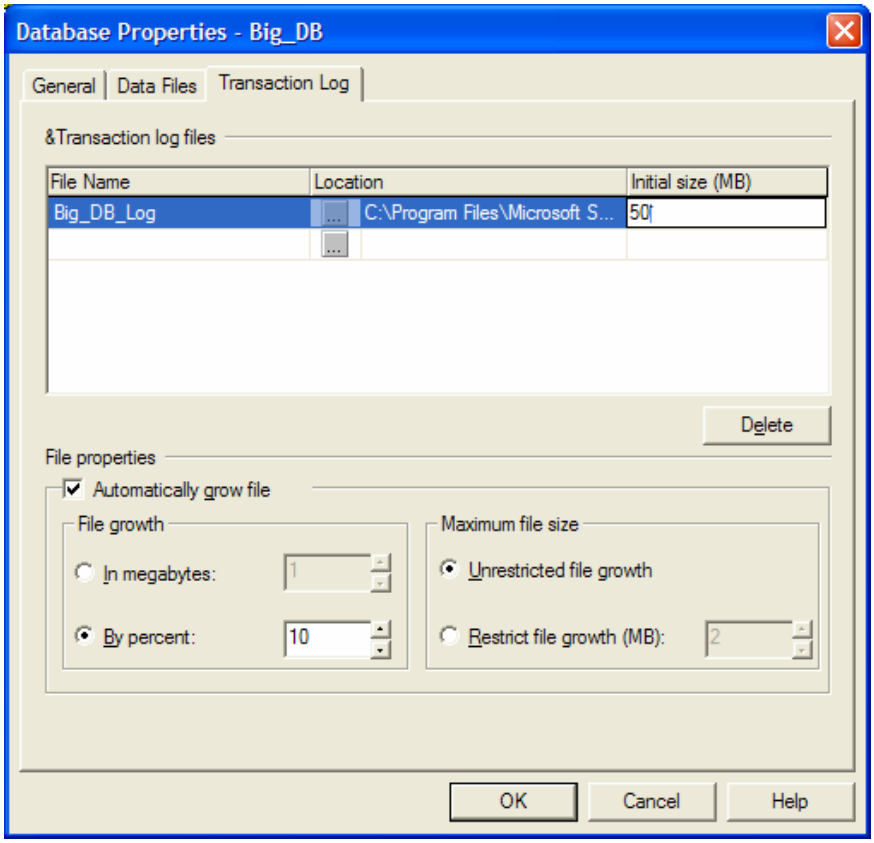
- تسمح لنا الواجهة General بإدخال اسم قاعدة المعطيات وتحديد الإعدادات الإقليمية الخاصة بمعطياتها؛



- تسمح لنا الواجهة Data Files بإدخال الأسماء المنطقية لملفات قاعدة المعطيات والمسار الفيزيائي لتلك الملفات، بالإضافة إلى مجموعة الملفات ومعاملات الامتداد الأخرى، مع العلم أن كافة تلك المدخلات لها قيم تلقائية، فعلي سبيل المثال: اسم ملف المعطيات له الشكل التالي databasename_data ومساره التلقائي هو نفسه مسار ملفات المخدِّم؛



- تسمح لنا الواجهة Transaction Log بإدخال المعلومات المتعلّقة بملفات سجل أو سجلات المناقلات مع العلم أنها تماثل واجهة Data Files في كل شيء ماعدا "مجموعة الملفات" بحيث لا يمكن تخزين ملفات سجلّ المناقلات ضمن مجموعة ملفات؛


Database Properties - Big_DB
General
Data Files
Transaction Log
&Transaction log files
File Name
Location
Initial size (MB)
Big_DB_Log
C:\Program Files\Microsoft S...
50
Delete
File properties
Automatically grow file
File growth
In megabytes:
1
By percent:
10
Maximum file size
Unrestricted file growth
Restrict file growth (MB):
2
OK
Cancel
Help

إنشاء قاعدة المعطيات باستخدام الأداة T-SQL

- سواء كنا نقوم بإنشاء قاعدة المعطيات بالاعتماد على الأداة Enterprise Manager أو عمّا تقدمه واجهات Create Database Wizard من خدمات، فبكلتا الحالتين، يقوم SQL Server بتوليد مخطوطات T-SQL مناسبة وينفذها من دون أن نلحظ ذلك؛

- يمكننا استخدام إمكانيات T-SQL بشكل مباشر لبناء قاعدة المعطيات أو لتوليد مخطوطات تساعدنا على أن نقوم بإنشاء قاعدة المعطيات في كل مرّة ننفذها فيها، ونستطيع من خلال استخدام مخطوطات Transact SQL أن ننوّع في الرماز بحيث نشمل على كافة المعاملات والملفات وغيرها من خصائص بناء قاعدة المعطيات؛

- مثال.

- سواء كنا نقوم بإنشاء قاعدة المعطيات بالاعتماد على الأداة Enterprise Manager أو عمّا تقدمه واجهات Create Database Wizard من خدمات، فبكلتا الحالتين، يقوم SQL Server بتوليد مخطوطات T-SQL مناسبة وينفذها من دون أن نلحظ ذلك؛

- يمكننا استخدام إمكانيات T-SQL بشكل مباشر لبناء قاعدة المعطيات أو لتوليد مخطوطات تساعدنا على أن نقوم بإنشاء قاعدة المعطيات في كل مرّة ننفذها فيها، ونستطيع من خلال استخدام مخطوطات Transact SQL أن ننوّع في الرماز بحيث نشمل على كافة المعاملات والملفات وغيرها من خصائص بناء قاعدة المعطيات؛

- فيما يلي عرض لمثال يوضّح المخطوطة اللازمة لبناء قاعدة معطيات:

```
Create Database Big_db
ON PRIMARY
(
      NAME = Big_DB_Dat ,
      FILENAME = 'C:\data\Big_DB.mdf' ,
      SIZE = 10 MB ,
      MAXSIZE = 50 MB ,
      FILEGROWTH = 15%
) ,

FILEGROUP Big_DB_Data
(
      NAME = Big_DB_Data_dat ,
      FILENAME = 'd:\ data\Big_DB_Data.ndf'
      SIZE = 50GB ,
      MAXSIZE = 100 GB ,
      FILEGROWTH = 10GB
)
```

```
LOG ON
(
      NAME = 'Big_DB_log'  ,
      FILENAME= 'f:\ log\Big_DB_log.ldf' ,
      SIZE = 50 MB ,
      MAXSIZE = 100 MB ,
      FILEGROWTH = 10 MB
)
COLLATE Arabic_CI_AI
GO
```

شرح المثال السابق

Create Database Big_db	اسم قاعدة المعطيات
ON PRIMARY	مجموعة الملفات الأولية
(	
NAME = Big_DB_Dat ,	الاسم المنطقي للملف الأولي
FILENAME = 'C:\data\Big_DB.mdf' ,	المسار الفيزيائي للملف الأولي
SIZE = 10 MB ,	حجم الملف
MAXSIZE = 50 MB ,	الحجم الأعظمي الذي يمكن أن يصل إليه الملف الأولي
FILEGROWTH = 15%	نسبة تضخم الملف الأولي عندما يمتلئ بالمعطيات، مع اعتبار الحجم الأعظمي ومساحة القرص.
) ,	
FILEGROUP Big_DB_Data	مجموعة الملفات الثانوية
(	
NAME = Big_DB_Data_dat ,	الاسم المنطقي للملف الثانوي
FILENAME = 'd:\ data\Big_DB_Data.ndf'	المسار الفيزيائي للملف الثانوي
SIZE = 50GB ,	حجم الملف
MAXSIZE = 100 GB ,	الحجم الأعظمي الذي يمكن أن يصل إليه الملف الثانوي
FILEGROWTH = 10GB	نسبة تضخم الملف الثانوي عندما يمتلئ بالمعطيات، مع اعتبار الحجم الأعظمي ومساحة القرص.
)	

LOG ON	ملف سجلّ المناقلات
NAME = 'Big_DB_log' ,	الاسم المنطقي لملف سجلّ المناقلات
FILENAME= 'f:\ log\Big_DB_log.ldf' ,	المسار الفيزيائي للملف
SIZE = 50 MB ,	حجم الملف
MAXSIZE = 100 MB ,	الحجم الأعظمي الذي يمكن أن يصل إليه الملف
FILEGROWTH = 10 MB	نسبة تضخم الملف عندما يمتلئ بالمعطيات، مع اعتبار الحجم الأعظمي ومساحة القرص.
)	
COLLATE Arabic_CI_AI	الإعدادات الإقليمية لمعطيات قاعدة المعطيات
GO	تنفيذ

إدارة قاعدة المعطيات

بعد الانتهاء من المرحلة الأولى، أي إنشاء قاعدة المعطيات، تبدأ المرحلة الثانية المستمرة وهي مرحلة إدارة قاعدة المعطيات تلك؛

يمكننا تصنيف عمليات إدارة قاعدة المعطيات إلى نوعين أساسيين

- معالجة بنية الملفات
- إعداد الخصائص الملائمة للاستخدام في قاعدة المعطيات

بعد الانتهاء من المرحلة الأولى، أي إنشاء قاعدة المعطيات، تبدأ المرحلة الثانية المستمرة وهي مرحلة إدارة قاعدة المعطيات تلك؛

يمكننا تصنيف عمليات إدارة قاعدة المعطيات وفق نوعين أساسيين، معالجة بنية الملفات، وإعداد الخصائص الملائمة للاستخدام في قاعدة المعطيات؛

سنناقش في الشرائح التالية كلا النوعين السابقين بالتفصيل.

إدارة قاعدة المعطيات - معالجة بنية الملفات
إدارة تضخم الملفات

لاحظنا سابقاً أن SQL Server يعمل على إدارة نمو الملفات وذلك بتوسيع تلك الملفات آلياً من خلال استخدام نسب النمو التي نحددها عند بناء قاعدة المعطيات بحيث يتم اعتمادها في زيادة حجم الملفات آليا كلما اقتضت الحاجة إلى ذلك؛

لا يأخذ مفهوم إدارة تضخُّم ملفات قاعدة المعطيات، والمُلقى على عاتق مدير قاعدة المعطيات، منحاه الإداري بالشكل الأمثل إلا عندما يقوم المدير دورياً بتفحُّص تزايد حجوم الملفات ويعمل على زيادة تلك الحجوم عند الحاجة دون الاعتماد على التحصيص الآلي للمساحات باستخدام امتدادات جديدة.

يعمل على إدارة نمو الملفات وذلك بتوسيع تلك الملفات آلياً من خلال استخدام نسب النمو SQL Serverلاحظنا –كما مرّ معنا مسبقاً– أن المحددة أثناء بناء قاعدة المعطيات بحيث يتم اعتمادها في زيادة حجم الملفات آليا كلما اقتضت الحاجة إلى ذلك؛

–في الحقيقة– يعتبر اعتماد التعريف السابق كجزء من مفهوم إدارة قواعد المعطيات، غير دقيق بما فيه الكفاية، فما يحدث عادةً أنه عندما يزداد حجم ملفات قاعدة المعطيات بحيث لا تتسع للمزيد من المعطيات، يجري إيقاف كافة الأنشطة المُطبَّقة على قاعدة المعطيات تلك، ثم يجري اختبار توافر مساحات إضافية على القرص بما يعادل مقدار التزايد المعرَّف بحسب المعامل FILEGROWTH، ومن ثم يجري توسيع الملفات؛
تتكرر العملية السابقة بالكامل في كل مرّة تمتلئ فيها الملفات بالمعطيات، وينطبق الأمر نفسه عندما نتعامل مع "مجموعات الملفات"؛

عموماً، ونتيجة لما سبق، لا بد لنا من الإشارة إلى أن مفهوم إدارة تضخُّم ملفات قاعدة المعطيات، والمُلقى على عاتق مدير قاعدة المعطيات، لا يأخذ منحاه الإداري بالشكل الأمثل عندما يجري الاعتماد على استخدام الامتدادات في توسيع حجم ملفات قاعدة المعطيات، في حين يتحقق ذلك عندما يقوم المدير دورياً بتفحُّص تزايد حجوم الملفات ويعمل على زيادة تلك الحجوم عند الحاجة دون الاعتماد على التحصيص الآلي للمساحات باستخدام امتدادات جديدة؛

إدارة قاعدة المعطيات - معالجة بنية الملفات

توسيع قواعد المعطيات

- يمكننا توسيع قواعد المعطيات بأسلوبين، إما آليا أو يدوياً؛
- يمكننا توسيع قاعدة المعطيات يدوياً إما بزيادة حجم ملفات قاعدة المعطيات أو بإضافة ملفات معطيات جديدة؛
- نستطيع إجراء التعديلات اليدوية السابقة بطريقتين مختلفتين أيضاً:
  - كتابة مخطوطات T-SQL خاصة بتعديل قاعدة المعطيات؛
  - استخدام الأداة Enterprise Manager؛

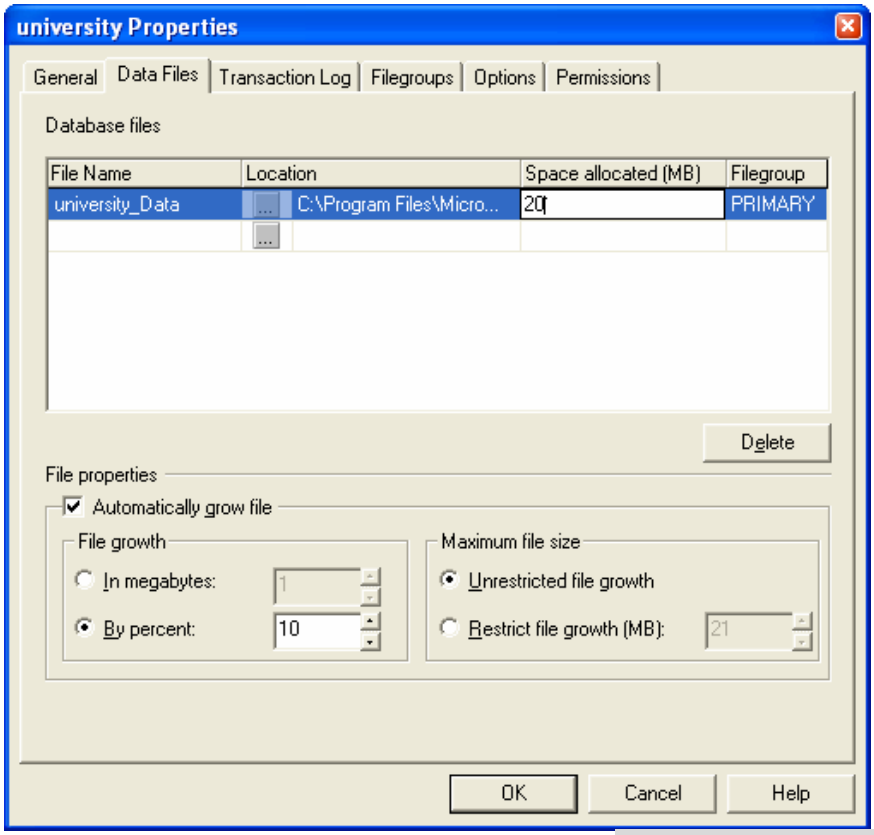
- كما مر معنا سابقاً، يمكننا توسيع قواعد المعطيات بأسلوبين، إما آليا أو يدوياً؛
- تناولنا –بالتفصيل– في الشرائح السابقة كيف تجري عملية توسيع قاعدة المعطيات آلياً، أما بالنسبة للتوسيع اليـدوي لقاعـدة المعطيات، فيمكننا إما أن نقوم بزيادة حجم ملفات قاعدة المعطيات أو أن نقوم بإضافة ملفات جديدة؛
- نستطيع إجراء التعديلات اليدوية السابقة بطريقتين مختلفتين أيضاً، إما من خلال كتابة مخطوطات T-SQL خاصة بتعـديل قاعدة المعطيات، أو من خلال استخدام الأداة Enterprise Manager؛
- سنقوم في المثال التالي بعرض مخطوط مكتوب بلغة T-SQL يقوم أولاً بزيادة حجم ملف قاعدة المعطيات ثم يقوم بإضـافة ملف معطيات جديد:

```
ALTER DATABASE Big_DB
MODIFY FILE
(
  NAME= Big_DB_Dat ,
  SIZE = 20MB
)
GO

ALTER DATABASE Big_DB
ADD FILE
(
  NAME= Big_DB_Dat2  ,
  FILENAME = 'e:\data\Big_DB_Data2.ndf'   ,
SIZE = 20MB
MAXSIZE = 100 MB   ,
FILEGROWTH = 5MB
)
GO
```

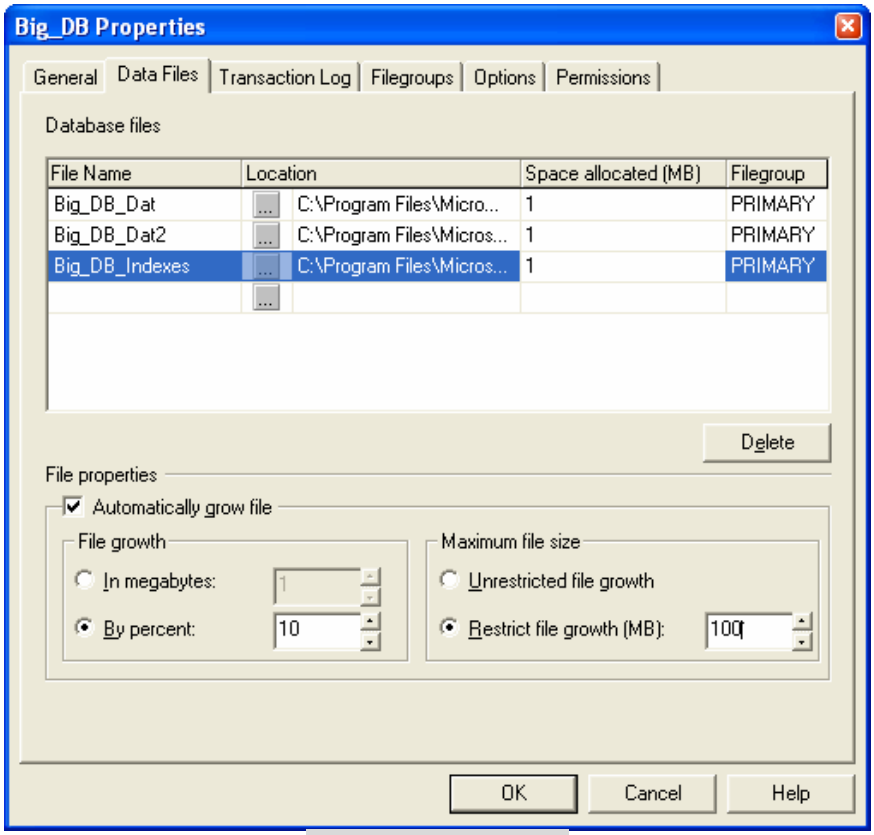
ALTER DATABASE Big_DB	اسم قاعدة المعطيات وتعليمة التعديل	**زيادة حجم ملف قاعدة المعطيات**
MODIFY FILE	تعليمة تعديل ملف المعطيات	
(		
NAME = Big_DB_Dat ,	الاسم المنطقي لملف المعطيات	
SIZE = 20MB	الحجم الجديد للملف	
)		
GO	تنفيذ	

إضافة ملف معطيات جديد	اسم قاعدة المعطيات وتعليمة التعديل	ALTER DATABASE Big_DB
	تعليمة إضافة ملف معطيات جديد	ADD FILE
		(
	الاسم المنطقي لملف المعطيات الجديد	NAME= Big_DB_Dat2 ,
	المسار الفيزيائي لملف المعطيات الجديد	AME = 'e:\data\Big_DB_Data2.ndf',
	حجم الملف	SIZE = 20MB ,
	ل الذي يمكن أن يصل إليه ملف المعطيات الجديد	MAXSIZE = 100MB ,
	الملف عندما يمتلئ بالمعطيات، مع اعتبار الحجم الأعظمي ومساحة القرص.	FILEGROWTH = 5MB
		)
	تنفيذ	GO

يمكننا أيضاً استخدام الأداة Enterprise Manager لزيادة حجم ملف ما أو إضافة ملف جديد، وذلك من خـــلال واجهـــة " Data Files" ضمن واجهة خصائص قاعدة المعطيات التي نرغب بتعديلها كما يوضح الشكل التالي:



تعديل المساحة المخزّنة لملف معطيات



إضافة ملف معطيات جديد

إدارة قاعدة المعطيات - معالجة بنية الملفات

تقليص قواعد المعطيات

- يزوّدنا SQL Server بخيار خاص يطلق عليه اسم AUTOSHRINK Option نستطيع من خلاله السماح لـــ SQL Server بأن يقوم بعمليات بحث تلقائي عن المساحات الفارغة في ملفات المعطيات وتقليصها؛

- يعتبر ملف سجلّ المناقلات من بين ملفات قاعدة المعطيات الأكثر تعرّضاً للنمو بشكل غير مألوف، وهو بالتالي الملف الأكثر تطلُّباً لإجراء عمليات تقليص.

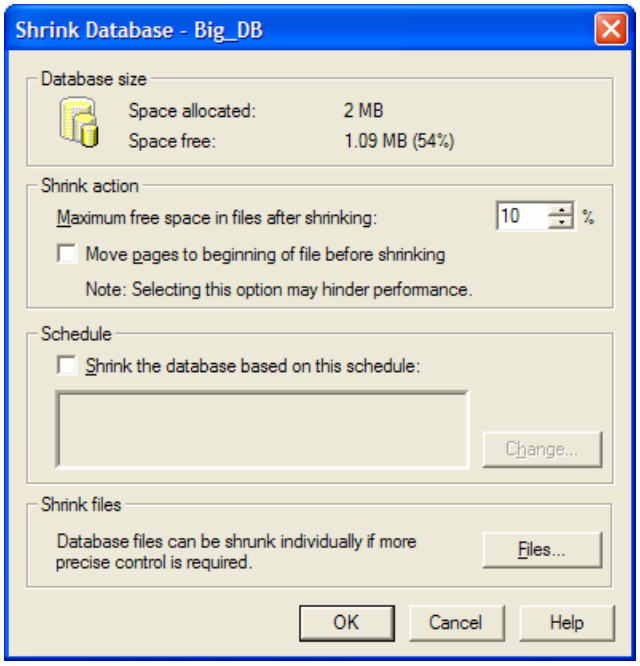
مرَّ معنا حتى الآن كيف يمكن أن يتم توسيع قواعد المعطيات، ولكن هل يمكننا أن نعمل على تقليص الحجم أيضاً؟

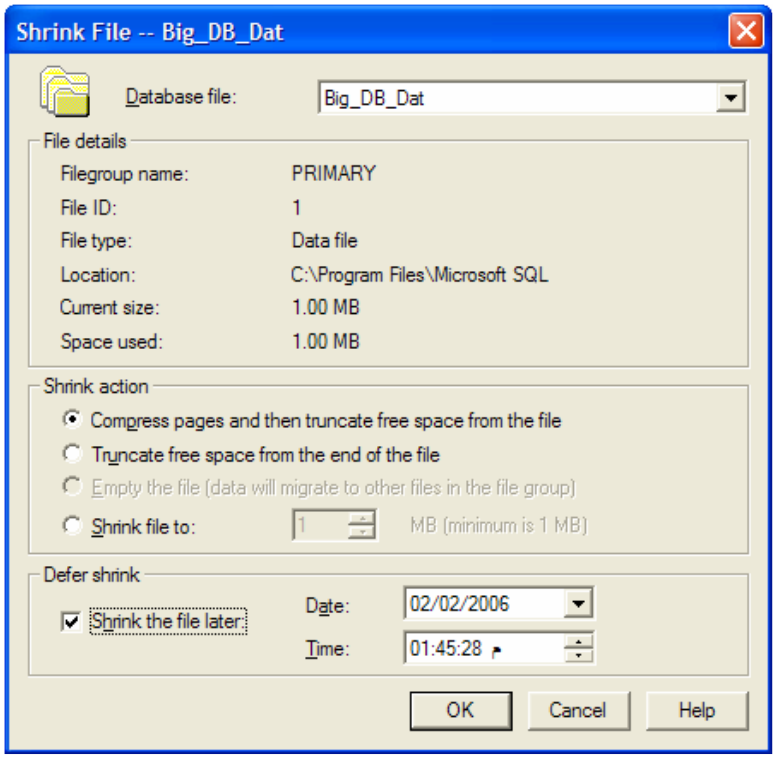
- يزوّدنا SQL Server بخيار خاص يطلق عليه اسم AUTOSHRINK Option نستطيع من خلاله السماح لـــ SQL Server بأن يقوم بعمليات بحث تلقائي عن المساحات الفارغة في ملفات المعطيات وتقليصها؛

- مع العلم أنه توجد حاجة حتميَّة للمساحات التي يمكننا الحصول عليها نتيجةً لإجراء عمليات التقليص، إلا أنه لا يُفضَّل القيام بتلك العمليات إذاما كنا على علم مسبقاً بأن قاعدة المعطيات تلك سوف تتوسع ويزداد حجمها في المستقبل القريب، فبالمحافظة على مساحات فارغة في ملفات قاعدة المطيات، يمكننا أن نتجنّب العبء الإضافي المترتّب والذي يمكن أن نعاني منه عندما نحتاج لمساحات تخزين إضافية، آخذين بعين الاعتبار أن كثرة عمليات التوسيع والتقليص تؤدي إلى تجزئة ملفات قاعدة المعطيات وتؤخر من أداء عمليات الدخل خرج؛

- يعتبر ملف سجلّ المناقلات من بين ملفات قاعدة المعطيات الأكثر تعرّضاً للنمو بشكل غير مألوف، وهو بالتالي الملف الأكثر تطلُّباً لإجراء عمليات تقليص عليه؛
- يمكننا أن نتخيل كيف يزداد حجم هذا الملف إذا ما افترضنا أن يقوم المستخدم بتنفيذ كمية كبيرة جدا من عمليات التحديث على العديد من سجلات قاعدة المعطيات، عندئذٍ، لابد لملف سجلّ المعطيات من أن يزداد بالحجم ليشمل على كافة السجلات المتغيرة تلك؛
- يمكننا تقليص حجم ملف سجل المعطيات من خلال السماح باقتصاص كافة السجلات غير الفعالة ما بين محتوياته، ما يؤدي بالضرورة إلى اختزال كبير في حجم ذلك الملف، مع العلم أنه ينصح دائما بالاحتفاظ بنسخة احتياطية من الملف الذي نقوم بتقليصه.

- يمكننا استخدام الأداة Enterprise Manager من أجل القيام بعملية التقليص تلك، وذلك من خلال اختيار "All Tasks" من قامة المهمات السريعة لقاعدة المعطيات التي نرغب بتقليصها، ثم نختار "Shrink Database" لتظهر الواجهة التاليه التي يمكننا من خلالها أن نحدد مقدار التقليص الذي نرغب بتنفيذه على قاعدة المعطيات المختارة، بالإضافة إلى إمكانية جدولة العملية تلك إلى وقت لاحق من خلال الضغط على الزر Files:


Shrink Database - Big_DB
Database size
Space allocated: 2 MB
Space free: 1.09 MB (54%)
Shrink action
Maximum free space in files after shrinking: 10 %
Move pages to beginning of file before shrinking
Note: Selecting this option may hinder performance.
Schedule
Shrink the database based on this schedule:
Change...
Shrink files
Database files can be shrunk individually if more precise control is required.
Files...
OK
Cancel
Help


Shrink File -- Big_DB_Dat
Database file: Big_DB_Dat
File details
Filegroup name: PRIMARY
File ID: 1
File type: Data file
Location: C:\Program Files\Microsoft SQL
Current size: 1.00 MB
Space used: 1.00 MB
Shrink action
Compress pages and then truncate free space from the file
Truncate free space from the end of the file
Empty the file (data will migrate to other files in the file group)
Shrink file to: 1 MB (minimum is 1 MB)
Defer shrink
Shrink the file later:
Date: 02/02/2006
Time: 01:45:28
OK
Cancel
Help

إدارة قاعدة المعطيات

خيارات قاعدة المعطيات

- أنواع الخيارات
- استعراض خيارات قاعدة معطيات معينة

يمتلك SQL Server خمسة تصنيفات مختلفة من الخيارات التي يمكن أن تستخدم للتحكم بسلوك قاعدة المعطيات، وفيما يلي عرض لتلك التصنيفات مع قيم الخيارات الموافقة لها:

الخَيار	التصنيف
AUTO_CLOSE {ON|OFF}	Auto Options
AUTO_CREATE_STATISTICS {ON|OFF}	
AUTO_UPDATE_STATISTICS {ON|OFF}	
AUTO_SHRINK {ON|OFF}	
CURSOE_CLOSE_ON_COMMIT {ON|OFF}	Cursor Options
CURSOR_DEFAULT {LOCAL|GLOBAL}	
RECOVERY {FULL|BULK_LOGGED|SIMPLE}	Recovery Options
TORN_PAGE_DETECTION {ON|OFF}	
SINGLE_USER|RESTRICTED_USER|MULTI_USER	State Options
OFFLINE|ONLINE	
READ_ONLY | READ_WRITE	
ANSI_NULL_DEFAULT {ON|OFF}	SQL Options
ANSI_NULLS {ON|OFF}	
ANSI_PADDING {ON|OFF}	
ANSI_WARNING {ON|OFF}	
ARITHABORT {ON|OFF}	
CONCAT_NULL_YIELDS_NULL {ON|OFF}	
NUMERIC_ROUNDABORT {ON|OFF}	
QUOTED_IDENTIFIER {ON|OFF}	
RECURCIVE_TRIGGERS {ON|OFF}	

يمكن إعداد معظم خيارات قاعدة المعطيات من واجهة Options ضمن واجهة خصائص قاعدة المعطيات المحددة، وذلك باستخدام الأداة Enterprise Manager مع العلم أنه يمكننا استخدام تعليمات T-SQL مباشرةً لتعديل قيم تلك الخصائص، مثال:

```
ALTER DATABASE Big_DB
  SET AUTO_SHRINK OFF
```

يمكننا الحصول على معلومات أشمل حول إعدادات خصائص قاعدة معطيات معينة من خلال استخدام إجرائية النظام sp_helpdb، أو من خلال استخدام التابع DATABASEPROPERTYX، مثال:

بتنفيذ التعليمة

```
sp_helpdb Big_DB
```

يمكننا الحصول على كافة الخيارات لقاعدة المعطيات المحددة، مرتبة بشكل متلاحق في عمود STATUS ضمن نتيجة الاستعلام:

```
Status=ONLINE, Updateability=READ_WRITE, UserAccess=MULTI_USER,
Recovery=SIMPLE, Version=539, Collation=Arabic_CI_AI, SQLSortOrder=0, IsAutoClose,
IsAutoShrink, IsTornPageDetectionEnabled, IsAutoCreateStatistics, IsAutoUpdateStatistics
```

الفصل السابع و الثامن

القسم الأول

عنوان الموضوع:

إنشاء وإدارة الجداول والفهارس، وضمان تكامل المعطيات.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

سنناقش في هذه الجلسة كيف يتم إنشاء جداول وفهارس قواعد المعطيات وإدارتها, وسنتناول بالتفصيل مكوناتها، كما سنركز على التعليمات والأدوات المستخدمة لتحقيق تلك المهمات، بالإضافة إلى أننا سنعمل على دراسة مفهوم تكامل المعطيات وكيف يتم تحقيقه.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- جدول قاعدة المعطيات
  - ما هو الجدول؟
  - كيف يتم إنشاء الجداول؟
- أنماط المعطيات وأنواعها
- بناء الجداول باستخدام مخطوطات T-SQL
- بناء الجداول باستخدام الأداة Enterprise Manager
- استعراض خصائص الجداول
- استعراض مخطوطات T-SQL التي يتم توليدها من خلال الأداة Enterprise Manager
- استعراض معطيات جدول معين
- حذف الجداول
- الجداول المؤقتة
- الفهارس
  - الفهارس العنقودية
  - الفهارس غير العنقودية
- إنشاء الفهارس باستخدام T-SQL

- إنشاء الفهارس باستخدام الأداة Enterprise Manager
- إدارة الفهارس
- حذف الفهارس
- مفهوم تكامل المعطيات
- أنماط تكامل المعطيات
  - تكامل المجال
  - تكامل الكيان
  - التكامل المرجعي
- طرائق تحقيق تكامل المعطيات
  - تكامل المعطيات التصريحي
  - تكامل المعطيات الإجرائي
- القيود:
  - المفتاح الأولي
  - شرط التفرد
  - المفتاح الخارجي
- خاصة التكامل المرجعي الشلالي
  - قيد الاختبار
- إنشاء القيود باستخدام الأداة Enterprise Manager
- القواعد:
  - إنشاء وإدارة القواعد باستخدام تعليمات T-SQL
  - إنشاء وإدارة القواعد باستخدام الأداة Enterprise Manager
- القيم التلقائية:
  - إنشاء وإدارة القيم التلقائية باستخدام تعليمات T-SQL
  - إنشاء وإدارة القيم التلقائية باستخدام الأداة Enterprise Manager

- يعتبر الجدول البنية الأساسية التي تعتمد عليها قاعدة المعطيات العلاقاتية، وهو يعتبر بشكل أو بآخر بنيةً لتخزين المعطيات

- سنناقش في هذه الجلسة كيف يتم إنشاء جداول المعطيات وكيف تتم إدارتها، اعتباراً من عمليات البناء أو استعراض محتوايها وانتهاءً بحذف تلك الجداول, وسنتناول بالتفصيل مكوناتها، كما سنركز على التعليمات والأدوات المستخدمة لتحقيق تلك المهمات

- سنتناول بعد ذلك مفهوم الفهارس وأنواعها، وكيف تتم عملية إنشاءها وإدارتها وحذفها، وما هي الفائدة منها، بالإضاقة إلى دراسة التعليمات والأدوات المستخدمة لتحقيق تلك العمليات

- وبعد الانتهاء من عمليات بناء مكونات قاعدة المعطيات من الجداول والفهارس، لابد لنا من التطرق إلى كيفية ضمان تكامل المعطيات في قاعدة المعطيات التي تمّ بناؤُها، بحيث سنناقش الأنماط المقترحة والتي يمكن أن تستخدم في تلك العملية، كما سنستعرض أنواع الشروط والقواعد والخصائص التلقائية المتاحة لضمان تكامل المعطيات، أي ما يُعرف بالـــ Constrains والـــ Rules والـــ Defaults

جدول قاعدة المعطيات

- تعريف الجدول: يعتبر الجدول البنية الأساسية التي تعتمد عليها قاعدة المعطيات العلاقاتية، وهو عبارة عن مجموعة من الأعمدة ذات خصائص معينة، تُستخدم لتخزين المعطيات

- إنشاء الجداول
- تعريف الجدول: يعتبر الجدول البنية الأساسية التي تعتمد عليها قاعدة المعطيات العلاقاتية، وهو عبارة عن مجموعة من الأعمدة ذات خصائص معينة تُستخدم لتخزين المعطيات، بحيث يتم تمثيل تلك المعطيات على هيئة أسطر مخزّنة في ذلك الجدول؛ تعبّر أسطر الجدول عادةً عن كيانات ترتبط بها مجموعة من الواصفات، مثلا: الكيان: موظّف، مع واصفات من نمط الاسم الأول أو الكنية...، بحيث يتم تمثيل تلك الواصفات من خلال أعمدة الجدول، كما في المثال التالي:

EMPLOYEE_NUMBE	F_NAME	L_NAME	PHONE	DEPT	SALARY
1	عماد	السيد	1234567	20	10000
2	سامر	حسن	2222222	20	10000
3	أمين	حداد	3333333	30	20000
4	منى	الغانم	4444444	40	12000
5	سناء	نجم	5555555	40	14000
6	فراس	دياب	6666666	20	21000

- إنشاء الجداول: تستخدم كل من تعليمتي CREATE TABLE و UPDATE TABLE لإنشاء وتعديل الجداول على الترتيب؛

ينبغي عند إنشاء جدول جديد، أن نحدد اسم ذلك الجدول، واسم كل عمود من أعمدته، بالإضافة إلى نمط المعطيات الخاص بكلٍ من تلك الأعمدة؛

نعرض فيما يلي لمثال يوضح المخطوط اللازم لبناء جدول جديد:

```
CREATE TABLE EMPLOYEE (
        EMPLOYEE_NUMBER int NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE varchar (14) NULL,
        DEPT   smallint NULL,
        SALARY float NULL
)
```

يفضّل عادة التصريح عن امكانية أن يحتوي العمود على قيم فارغة NULL، مع العلم أن العملية تلك اختيارية؛

نلاحظ في المثال السابق وجود عدّة أنماط معطيات مختلفة (int, varchar, smallint, float)، تعبّر عن طريقة تخزين المعطيات؛

سنناقش في الشرائح التالية الأنواع المختلفة لأنماط المعطيات التي يمكن استخدامها في الأداة SQL Server، وذلك نظراً لضرورة إدراك أهمية أنماط المعطيات المتاحة قبل المباشرة ببناء قاعدة المعطيات وجداولها.

أنماط المعطيات

- تعريف:

تعبِّر "أنماط المعطيات" عن النمط المستخدم لتخزين المعطيات في الجداول، وتستخدم لفرض بعض القيود على نمط وحجم المعطيات المخزّنة في الجداول؛

- تعتبر عملية انتقاء نمط المعطيات المناسب لكل عمود من أعمدة الجدول، عملية هامة جداً، خاصةً فيما يتعلق بضمان فعاليّة تخزين المعطيات أو نمط الأداء أو توافقية النظام ككل؛
- تزوّد الأداة SQL Server عدّة أنواع من أنماط المعطيات مصنّفة ضمن عدّة مجموعات، وهي:

التصنيف	اسم النمط	سعة التخزين (بايت)
المحارف		0-8000
	VARCHAR [(n)]	0-2GB
	TEXT	
محارف UNICODE	NCHAR [(n)]	0-8000
	NVARCHAR [(n)]	4000 محرف
	NTEXT	0-2 GB
الثنائيات	BINARY [(n)]	0-8000
	VARBINARY [(n)]	
التاريخ والوقت	DATETIME	8
	SMALLDATETIME	4
أنماط الأعداد الصحيحة	INT	4
	BIGINT	8
	SMALLINT	2
	TINYINT	1
أنماط الأرقام العشرية (الأنماط الرقمية)	DECIMAL [(p[,s])]	2-17
	NUMERIC [(p[,s])]	2-17
أرقام حقيقية (الأنماط الرقمية التقريبية)	FLOAT [(n)]	8
	REAL [(n)]	4
أنماط الترميز المالي	MONEY	8
	SMALLMONEY	4
صور	IMAGE	0-2 GB
معرّفات عامة	UNIQUEIDENTIFIER	16
أنماط خاصّة	BIT	1
	CURSOR	0-8
	TIMESTAMP	8
	SYSNAME	256

	TABLE	
	SQL_VARIANT	0-8016

• تعبِّر "أنماط المعطيات" عن النمط المستخدم لتخزين المعطيات في الجداول، بحيث ينبغي إسناد نمط معطيات لكل عمود جديد تتم إضافته إلى أي جدول، ما يضمن تحقيق مكاملة – ولو كانت أولية –للمعطيات التي يتم إدخالها، وذلك من خلال فرض بعض القيود على نمط وحجم تلك المعطيات

• تعتبر عملية انتقاء نمط المعطيات المناسب لكل عمود من أعمدة الجدول، عملية هامة جداً، خاصةً فيما يتعلق بضمان فعاليّة تخزين المعطيات أو نمط الأداء أو توافقية النظام ككل

• تزوّد الأداة SQL Server عدّة أنواع من أنماط المعطيات مصنّفة ضمن عدّة مجموعات بحسب نوع المعطيات التي تُستخدَم لتخزينها، يعرض الجدول التالي قائمة بتلك التصنيفات مع الأنماط التي تتكون منها بالإضافة إلى سعات تخزين كل منها:

التصنيف	اسم النمط	سعة التخزين (بايت)
محارف		0-8000
	VARCHAR [(n)]	0-2GB
	TEXT	
محارف UNICODE	NCHAR [(n)]	0-8000
	NVARCHAR [(n)]	4000 محرف
	NTEXT	0-2 GB
ثنائيات	BINARY [(n)]	0-8000
	VARBINARY [(n)]	
تاريخ ووقت	DATETIME	8
	SMALLDATETIME	4
أعداد صحيحة	INT	4
	BIGINT	8
	SMALLINT	2
	TINYINT	1
أرقام عشرية	DECIMAL [(p[,s])]	2-17
	NUMERIC [(p[,s])]	
أرقام حقيقية	FLOAT [(n)]	8

	REAL [(n)]	4
ترميز مالي	MONEY	8
	SMALLMONEY	4
صور	IMAGE	0-2 GB
معرّفات عامة	UNIQUEIDENTIFIER	16
أنماط خاصّة	BIT	1
	CURSOR	0-8
	TIMESTAMP	8
	SYSNAME	256
	TABLE	
	SQL_VARIANT	0-8016

سنتناول في الشرائح التالية خصائص كلاً من تلك الأنماط والتصنيفات بالتفصيل.

أنماط المعطيات
-المحارف والأنماط الثنائية-

- يتضمن التصنيف المحرفي على ستة أنواع مختلفة من أنماط المعطيات، وهي CHAR و VARCHAR و NCHAR و NVARCHAR و NTEXT، بحيث يتميز كل نوع منها بمميزات خاصة ويستخدم لأغراض مختلفة؛\
- تُخزن الأنماط BINARY و VARBINARY ما يصل إلى 8000 بايت أيضاً، في حين أنها تستخدم لتخزين معطيات ثنائية، مع العلم أنه يمكن تخزين قيم ست عشرية من خلال أنماط التخزين الثنائية تلك؛\
- يمكننا استخدام الأنماط TEXT و NTEXT و IMAGE لتخزين سلاسل المحارف والمعطيات الثنائية التي يمكن أن يتجاوز حجمها 8000 بايت.\
- يُستخدم نمط المعطيات المحرفي –بشكل عام– لتخزين سلاسل الأحرف، ويتمتع هذا النمط بخصائص متعددة بحيث يمكن أن يكون ذو طول ثابت أو متغير، بايت واحد أو بايتاتUnicode
- يتضمن هذا التصنيف على ستة أنواع مختلفة من أنماط المعطيات، وهي CHAR و VARCHAR و NCHAR و NVARCHAR و NTEXT، بحيث يتميز كل نوع منها بمميزات خاصة ويستخدم لأغراض مختلفة؛
- يعتبر كل من النمطين CHAR و VARCHAR الأكثر شيوعاً ما بين أنماط المعطيات المحرفيّة، بحيث يخزن كل منهما، المعطيات ثابتة ومتغيرة الطول على الترتيب

- يستخدم النمط CHAR عادةً للتعبير عن الأعمدة ذات الأطوال الثابتة، كالعمود الذي يعبر عن الجنس في جدول البيانات الشخصية لموظف، على سبيل المثال، بحيث يمكن أن يتكون من محرف وحيد "ذ" للذكر أو "أ" للأنثى
- عندما نعبر عن أحد أعمدة المحارف في جدول ما بالشكل "CHAR (30)"، فسيتم حجز ثلاثون بايتاً لذلك العمود حتى ولواستخدمنا ستة محارف منها فقط
- يعتبر النمط VARCHAR هو المفضّل للتعبير عن الأعمدة التي يمكن أن تحتوي على معطيات متغيرة الطول، كالاسم أو الوصف أو غيرها، فالمساحة المحجوزة باستخدام هذا النمط تساوي تماما طول الحقل المُدخل، مع العلم أن الطول الأعظمي لذلك الحقل محدود بالقيمة المدخلة أثناء اختيار هذا النمط
- يمكننا تخزين ما يصل إلى 8000 محرف فقط، باستخدام كلا النمطين السابقين

• يُستخدم كل من النمطين NCHAR و NVARCHAR لنفس أغراض النمطين السابقينCHAR و VARCHAR على الترتيب، في حين يختلفان عنهما بالمساحة التي يتم حجزها، بحيث يمكننا تخزين ما يصل إلى 4000 محرف فقط، وذلك لأن هذين النمطين يستخدمان لتخزين محارف الـ UNICODE التي يحتاج كل منها إلى بايتين عوضاً عن البايت الواحد

• تُخزن الأنماط BINARY و VARBINARY ما يصل إلى 8000 بايت أيضاً، في حين أنها تستخدم لتخزين معطيات ثنائية، مع العلم أنه يمكن تخزين قيم ست عشرية من خلال أنماط التخزين الثنائية تلك؛

• يمكننا استخدام الأنماط TEXT و NTEXT و IMAGE لتخزين سلاسل المحارف والمعطيات الثنائية التي يمكن أن يتجاوز حجمها 8000 بايت، فعلى سبيل المثال، يمكننا اسناد النمط TEXT للعمود الذي نرغب باستخدامه لتخزين محتويات ملفات نصية، ويمكننا اعتماد NTEXT عوضاً عن TEXT في الحالات التي تتطلب تخزين محارف UNICODE، كما يعتبر اختيار النمط IMAGE الخَيار المفضل عندما نرغب بتخزين الصور

1.7-8.05 أنماط المعطيات
- الأنماط الرقمية -

• يُستخدم نمط المعطيات الرقمي –بشكل عام– لتخزين الأرقام، ويتمتع هذا النمط بخصائص متعددة وله أنواع مختلفة لكل منها استخداماتها

• الأعداد الصحيحة:

يتضمن هذا التصنيف على أربعة أنواع مختلفة من أنماط المعطيات، وهي INT و BIGINT و SMALLINT و TINYINT، بحيث يتميز كل نوع منها بمميزات خاصة ويستخدم لأغراض مختلفة خاصة فيما يتعلق بمجال الترقيم وحجمه لكل منها.

• الأنماط الرقمية التقريبية:

يُستخدم كل من النمطين FLOAT و REAL عادةً، لتخزين معطيات علمية أو إحصائية ذات مجال قيم واسع، إلا أنها تتميز بعدم الدقة المطلقة فيما يتعلق بالأرقام العشرية.

- الأنماط الرقمية:

يُستخدم كل من النمطين DECIMAL و NUMERIC عادةً، لتخزين معطيات ذات مجال قيم واسع وتتطلب دقة حسابية عالية، ويعتبر هذان النمطان متماثلان تماماًp

- أنماط الترميز المالي:

يزوِّد SQL Server كلاً من نمطي المعطيات MONEY و SMALLMONEY وذلك لتخزين المعطيات الماليّة، يمكن تشبيه هذان النمطان بأنماط المعطيات الرقمية مع دقّة تصل إلى أربعة خانات عشرية.

- يُستخدم نمط المعطيات الرقمي –بشكل عام– لتخزين الأرقام، ويقسم هذا النمط إلى أربعة مجموعات رئيسية، وهي:
  - أنماط الأعداد الصحيحة؛
  - الأنماط الرقمية التقريبية؛
  - الأنماط الرقمية؛
  - أنماط الترميز المالي.

- الأعداد الصحيحة:

يتضمن هذا التصنيف على أربعة أنواع مختلفة من أنماط المعطيات، وهي INT و BIGINT و SMALLINT و TINYINT، بحيث يتميز كل نوع منها بمميزات خاصة ويستخدم لأغراض مختلفة خاصة فيما يتعلق بمجال الترقيم وحجمه لكل منها؛
تعتبر الأرقام الصحيحة ملائمة تماماً لتخزين ومعالجة مجال واسع من القيم، بالتالي تعتبر الأمثل للاستخدام في الحالات التي تتطلب قيماً ثابتة كالمفاتيح على سبيل المثال؛
يوضح الجدول التالي حجوم التخزين ومجال القيم التي يمكن حجزها باستخدام كل من أنماط تخزين الأعداد الصحيحة:

النمط	حجم التخزين(بايت)	القيمة الدنيا	القيمة العليا
TINYINT	1	0	255
SMALLINT	2	-32,768	32,767
INT	4	-2,147,483,64	2,147,483,647
BIGINT	8	-92233720	-92233720

تجدر الإشارة هنا إلى أن الأداة SQL Server تسمح بالتبديل من نمط معطيات رقمي صحيح ذو حجم تخزين أو مجال ترقيم أقل إلى آخر ذو مجال أكبر، دون أي مشكلة.

- الأنماط الرقمية التقريبية:

يُستخدم كل من النمطين FLOAT و REAL عادةً، لتخزين معطيات علمية أو إحصائية ذات مجال قيم واسع، إلا أنها تتميز بعدم الدقة المطلقة فيما يتعلق بالأرقام العشرية؛

- الأنماط الرقمية:

يُستخدم كل من النمطين DECIMAL و NUMERIC عادةً، لتخزين معطيات ذات مجال قيم واسع وتتطلب دقة حسابية عالية، ويعتبر هذان النمطان متماثلان تماماً، فالاسمان المستخدمان هما في الحقيقة مترادفات للتعبير عن نفس النمط، يمكننا عند استخدام هذا النوع من الأنماط أن نحدد معاملين أساسيين يعبر أولهما عن عدد الأرقام التي ينبغي حجزها ككل، ويعبر الثاني عن عدد الأرقام التي ينبغي حجزها بعد الفاصلة. مثال:

يعبر النمط DECIMAL(5,2) عم امكانية تخزين القيم التي تتراوح ما بين 999.99- إلى 999.99.

يوضح الجدول التالي حجوم التخزين التي يتم حجزها اعتماداً على الرقم الأول المعرّف لهذا النمط والذي يطلق عليه اسم "دقّة النمط العشري":

الدقة	حجم التخزين(بايت)
1-9	5
10-19	9
20-28	13
29-38	17

- أنماط الترميز المالي:

يزوِّد SQL Server كلاً من نمطي المعطيات MONEY و SMALLMONEY وذلك لتخزين المعطيات الماليّة، يمكن تشبيه هذان النمطان بأنماط المعطيات الرقمية مع دقّة تصل إلى أربعة خانات عشرية؛

يوضح الجدول التالي حجوم التخزين التي يتم حجزها باستخدام أنماط التخزين المالي:

النمط	حجم التخزين(بايت)	القيمة الدنيا	القيمة العليا
SMALLMONEY	4	-214,748.3648	214,748.3647
MONEY	8	-922,337,203,685,477.5808	-922,337,203,685,477.5807

أنماط المعطيات
- التاريخ والوقت-

- يُستخدم نمط معطيات التاريخ والوقت لتخزين القيم الزمنية، ويتمتع هذا النمط بخصائص متعددة وله أنواع مختلفة لكل منهـا استخداماتها.

يتضمن هذا التصنيف على نوعين مختلفين من أنماط المعطيات، هما DATETIME و SMALLDATETIME، بحيث يتميز كل نوع منها بمميزات خاصة ويستخدم لأغراض مختلفة خاصة فيما يتعلق بالمجال الزمني ودقته؛

- يُستخدم نمط معطيات التاريخ والوقت لتخزين القيم الزمنية، ويتمتع هذا النمط بخصائص متعددة وله أنواع مختلفة لكل منهـا استخداماتها، وهي:

- أنماط DATETIME:

يتضمن هذا التصنيف على نوعين مختلفين من أنماط المعطيات، هما DATETIME و SMALLDATETIME، بحيث يتميز كل نوع منها بمميزات خاصة ويستخدم لأغراض مختلفة خاصة فيما يتعلق بالمجال الزمني ودقته؛

يوضح الجدول التالي حجوم التخزين ومجال القيم التي يمكن حجزها باستخدام نمطي تخزين التاريخ والوقت:

النمط	حجم التخزين(بايت)	الدقة	القيمة الدنيا	القيمة العليا
DATETIME	8	3/100 ثانية	JAN. 1, 1753	DEC. 31, 9999
SMALLDATETIME	4	1دقيقة	JAN. 1, 1990	DEC. 31, 2079

لا يوجد في SQL Server نمط معطيات لتخزين التاريخ فقط أو الوقت فقط، فعندما نقوم بإدخال تاريخ معين بدون وقت إلى حقل ما من نمط DATETIME أو SMALLDATETIME، فإن قيمة الوقت ستأخذ بشكل تلقائي ساعة منتصف الليل مـن ذلـك اليوم؛

وبالمثل، فإن إدخال الوقت دون التاريخ، فسيأخذ التاريخ القيمة التلقائية التالية: JAN. 1, 1900

مثال:

لنقم بإنشاء جدول ما بعمود وحيد من نمط DATETIME:

```
CREAT TABLE datetest
(
        dateCol DATETIME not null
)
GO
```

الآن، لنقم بإدخال القيمة التالية كما يلي:

```
INSERT INTO datetest VALUES
('2/2/2006')
GO
```

لنقم بعد ذلك باستعلام القيمة المُدخلة كما يلي، ولنراقب النتيجة:

```
SELECT * FROM datetest
```

عندما نقوم بإدخال قيم زمنية إلى جدول معين، ينبغي التعبير عن القيمة تلك على هيئة سلسلة محارف؛
يزوّدنا SQL Server بعدّة أشكال مختلفة للتعبير عن التاريخ منها:

الهيئة	مثال
monthname dd[,] yy[yy]	February 4, 2006
dd monthname yy[yy]	4 February 2006
yyyy monthname dd	2006 February 04
mm/dd/yy[yy]	02/04/06
mm-dd-yy[yy]	02-04-06
mm.dd.yy[yy]	02.04.2006
[yy]yymmdd	060204

مع العلم أنه يدعم أشكال أخرى كالتاريخ الهجري على سبيل المثال؛
عندما تكون اللغة الانكليزية هي اللغة المعتمدة على الحاسب، فستكون هيئة التاريخ المستخدمة من قبل SQL Server من الــشكل mdy ، مع العلم أنه يمكننا أن نغيّر ذلك الشكل التلقائي من خلال تعليمة dateformate؛

أما بالنسبة إلى الزمن، فيمكننا التعبير عنه بطريقتين، إما AM PM أو 24 ساعة.

أنماط المعطيات
-أنماط أخرى-

- نمط المعرّفات العامة:

يستخدم هذا النوع من أنماط المعطيات لتوليد معرّفات عامة لعمود معين؛

- نمط الختم الزمني timestamp:

يولد هذا النمط رقماً ثنائياً فريداً على مستوى قاعدة المعطيات ككل، بطول ثماني بايتات؛

- نمط المعطيات المنطقي bit:

يستخدم هذا النمط ببساطة للتعبير عن القيم المنطقية on\off أو true\false، بحيث يؤمن هذا النمط إمكانيات تخزين للقيم 1 أو 0 أوNULL ويستخدم لذلك الغرض سعة تخزين تبلغ بايتاً واحداً فقط.

- نمط المعطيات المتغيّر SQL_VARIANT:

يستخدم هذا النمط للتعبير عن الحالات التي يمكن فيها تخزين قيم مختلفة في نفس العمود، كأن يستخدم مثلاً أحد الأعمدة ليخـــزن محارف في أحد الأسطر وقيماً رقمية في أسطر أخرى.

- النمط CURSOR والنمط TABLE:

لا يمكن استخدام هذا النمط للتعبير عن نمط معطيات عمود ما في جدول معين، إنما يعتبر بنية تخزينية لها استخداماتها الخاصة.

- نمط المعرّفات العامة:

يستخدم هذا النوع من أنماط المعطيات لتوليد معرّفات عامة لعمود معين؛
يتم توليد هذه القيمة من خلال الاعتماد على عملية ربط تتم ما بين الخاصة ROWGUIDCOL وبين التابع ( )NEWID، بحيث يضمن SQL Server أن القيمة الناتجة عن هذه العملية فريدة من نوعها عبر كافة الحواسيب حول العالم؛
تظهر الفائدة من نمط المعطيات هذا في حالات خاصة، كالتي تتطلب الحفاظ على معرّف فريد لسطر قاعدة المعطيات، كمـــا فـــي حالات تنفيذ عمليات تكرار المعطيات.

- نمط الختم الزمني timestamp:

يولد هذا النمط رقماً ثنائياً فريداً على مستوى قاعدة المعطيات ككل بطول ثماني بايتات؛
على الرغم من أن الختم فريد على مستوى قاعدة المعطيات، إلا أنه ليس من المفضل استخدامه كمعرّف لـــسطر المعطيـــات فـــي الجدول، خاصةً وأنه يتغير عند كل عملية إضافة أو تعديل على السطر.

- نمط المعطيات المنطقي bit:

يستخدم هذا النمط ببساطة للتعبير عن القيم المنطقية on\off أو true\false، بحيث يؤمن هذا النمط إمكانيات تخزين للقيم 1 أو 0 أوNULL ويستخدم لذلك الغرض سعة تخزين تبلغ بايتاً واحداً فقط.

- نمط المعطيات المتغيّر SQL_VARIANT:

يستخدم هذا النمط للتعبير عن الحالات التي يمكن فيها تخزين قيم مختلفة في نفس العمود، كأن يستخدم مثلاً أحد الأعمدة ليخـــزن محارف في أحد الأسطر وقيماً رقمية في أسطر أخرى.

- النمط CURSOR:

لا يمكن استخدام هذا النمط للتعبير عن نمط معطيات عمود ما في جدول معين، إنما يعتبر بنية تخزينية لها استخداماتها الخاصـــة، وهو يستخدم عادةً للتصريح عن المعاملات والمتغيرات كما في الإجرائيات المخزّنة.

- النمط TABLE:

يشبه هذا النمط تماماً نمط المعطيات CURSOR في خصائصه، بحيث يمكن استخدامه في الإجرائيات المخزّنة وفي التوابع المعرّفة.

8 أنماط المعطيات المعرّفة من قبل المستخدم

- يمكننا من خلال الأداة SQL Server أن نقوم بتعريف أنماط معطيات خاصة اعتماداً على أنماط معطيات النظام المتاحة؛

- تعتبر هذه العملية مفيدة جداً، وخاصةً عندما نرغب ببناء قاعدة معطيات على أسس متّسقة، مع العلم أنه يمكننا زيادة الأمور تعقيداً من خلال استخدام بعض القواعد أو القيم التلقائية أو إضافة المزيد من العمليات على نمط المعطيات الجديد المُنشأ؛

- نستطيع أن نعرّف نمط معطيات جديد بإحدى الطريقتين التاليتين:

من خلال استخدام الإجرائية المخزّنة sp_addtype؛ أو من خلال الأداة Enterprise Manager، وذلك باختيار إضافة نمط معطيات جديد من أيقونة User Defined Data Types ضمن قاعدة المعطيات التي نرغب بإضافة النمط الجديد إليها.

- يمكننا من خلال الأداة SQL Server أن نقوم بتعريف أنماط معطيات خاصة اعتماداً على أنماط معطيات النظام المتاحة؛
- فعلى سبيل المثال، يمكننا إنشاء نمط معطيات جديد باسم معين ذو نمط varchar(64)، بالتالي، فأي عمود من النمط الجديد سيكون varchar(64)؛
- تعتبر هذه العملية مفيدة جداً، وخاصةً عندما نرغب ببناء قاعدة معطيات على أسس متّسقة، مع العلم أنه يمكننا زيادة الأمور تعقيداً من خلال استخدام بعض القواعد أو القيم التلقائية أو إضافة المزيد من العمليات على نمط المعطيات الجديد المُنشأ؛
- كيف يتم بناء نمط معطيات جديد؟

نستطيع أن نعرّف نمط معطيات جديد بإحدى الطريقتين التاليتين:

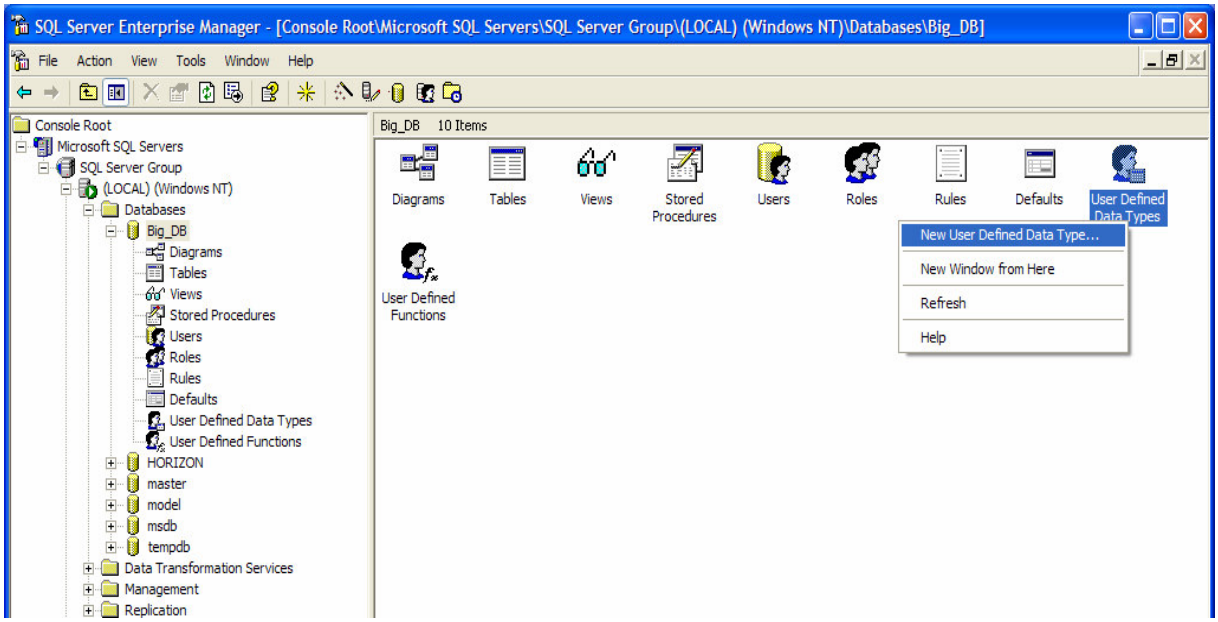
- من خلال استخدام الإجرائية المخزّنة sp_addtype، والتي تأخذ معاملان هما اسم النمط الجديد ونوعه؛

مثال:

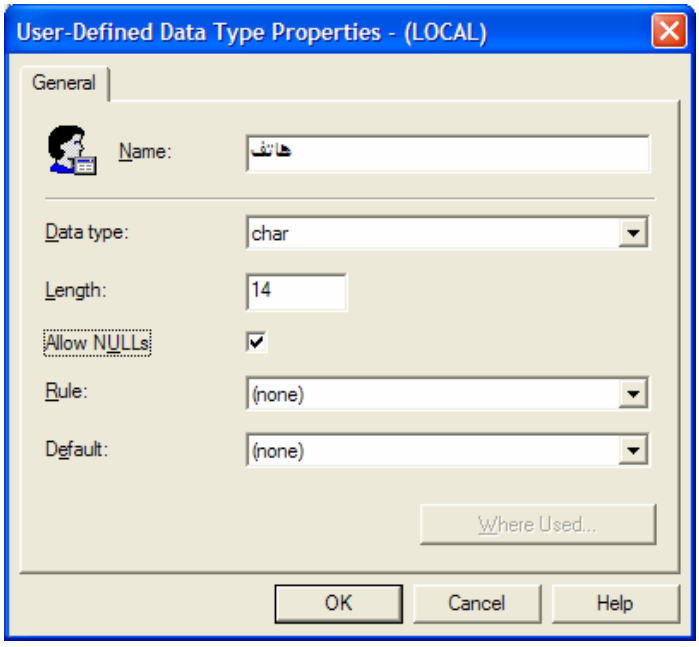
```
sp_addtype phone, 'char (13)'
```

بالمقابل، فأنه يمكننا استخدام الإجرائية sp_droptype لحذف نمط المعطيات المُعرَّف من قبل المستخدم؛

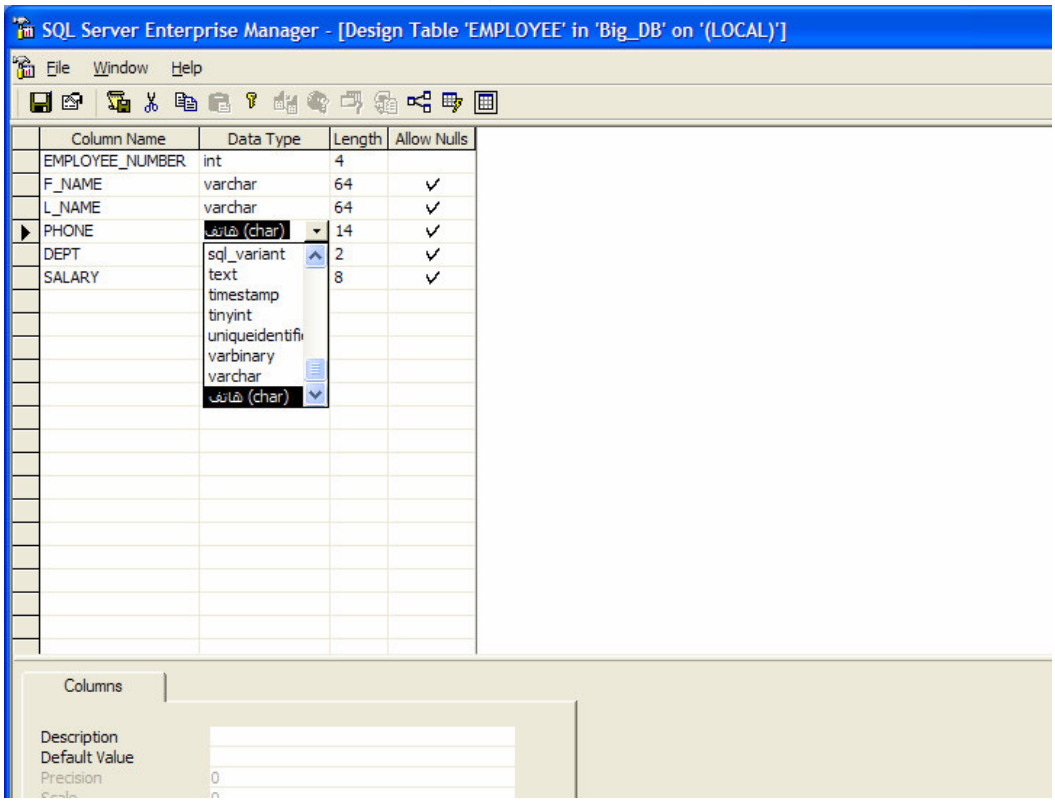
- يمكننا أيضاً تعريف نمط معطيات جديد من خلال الأداة Enterprise Manager، وذلك باختيار إضافة نمط معطيات جديد من أيقونة User Defined Data Types ضمن قاعدة المعطيات التي نرغب بإضافة النمط الجديد إليها،



لتظهر بعد ذلك الواجهة الموضحة بالشكل التالي، وفيها يتم تحديد كل من اسم نمط المعطيات الجديد ونوعه وطوله أو حجمه، كما نلاحظ إمكانية السماح لهذا النمط بأن يقبل القيم الفارغة NULLS، بالإضافة إلى إمكانية إدراج قواعد أو قيم افتراضية مسبقة التعريف، مع العلم أننا سنتناول مفهوم القواعد والقيم الافتراضية بالتفصيل لاحقاً.



نلاحظ أنه يمكننا بعد الانتهاء من تعريف النمط الجديد أن نقوم باستخدامه بشكل طبيعي كأي نمط معطيات آخر، مع العلم أنه لا يمكننا حذف أي نمط معطيات مُعرَّف في حال كونه قيد الاستخدام في أحد جداول قاعدة المعطيات.



إنشاء جداول قاعدة المعطيات

- الآن وبعد الانتهاء من التعرف على مختلف أنواع أنماط المعطيات التي يمكن استخدامها لتخزين المعطيات، يمكننا أن ننتقل إلى الخطوة التالية وهي عملية إنشاء الجداول؛
- تؤمن الأداة SQL Server طريقتين مختلفتين لبناء وإنشاء الجداول وذلك إما من خلال استخدام الأداة Enterprise Manager وما تقدمه من واجهات خاصة بتصميم الجداول، أومن خلال كتابة مخطوطات T-SQL مناسبة وتنفيذها؛
- ينبغي –وفي كلتا الطريقتين السابقتين– لكي يتم إنشاء جدول جديد أن نراعي ثلاثة نقاط رئيسية، وهي تسمية الجدول، وتحديد الأعمدة، وإسناد الخصائص المناسبة للأعمدة التي تمَّ إنشاؤها.

تسمية الجداول

يتم عادةً إنشاء الجدول ضمن قاعدة معطيات، وفي معظم الأحيان، يعتبر مالك الجدول هو نفسة المستخدم الذي قام بإنشائه؛

تعتبر عملية إنشاء الأغراض تحت ملكية المستخدم dbo هي المفضلة بشكل عام، وذلك لما لها من فوائد تصنيفية من جهة، وفيما يتعلق بالأداء والأمن من جهة أخرى؛

- بعض معايير تسمية الجداول:

ينبغي عند تسمية جدول ما، أن يكون اسمه معبِّراً عن محتوياته، عما ينبغي تجنب استخدام الجمل أثناء التعبير عم اسم الجدول، ينبغي أن يكون اسم الجدول مفرداً، يمكن أن يتسع اسم الجدول ليصل إلى 128 محرفاً، يفضل عادةً تجنب استخدام المحارف الخاصة في تسمية الجدول –مع العلم أنه يمكن استخدامها–، كما يمكن الاستعاضة عن الفراغ بالمحرف "_"، وهذا أمر شائع؛

- إعادة تسمية الجدول:

كثيراً ما نرغب بتغيير اسم الجدول بعد بنائه، تعتبر هذه العملية متاحة من خلال الأداة SQL Server بأسلوبين مختلفين، باستخدام الأداة Enterprise Manager أو من خلال استخدام إجرائيات نظام مخزّنة.

يتم عادةً إنشاء الجدول ضمن قاعدة معطيات، وفي معظم الأحيان، يعتبر مالك الجدول هو نفسة المستخدم الذي قام بإنشائه؛

يمكننا تعريف اسم الجدول من خلال الثلاثية التالية:

databaseName.owner.tableName

مع العلم أنه يمكن لكل من أعضاء sysadmin و dbowner و ddladmin أن يقوموا بتحديد اسم مالك الجدول صراحةً في تعليمة إنشاء الجدول، مثال:

CREAT TABLE myTable.feras.employee

- تعتبر عملية إنشاء الأغراض تحت ملكية المستخدم dbo هي المفضلة بشكل عام، وذلك لما لها من فوائد تصنيفية من جهة، وفيما يتعلق بالأداء والأمن من جهة أخرى؛
- يمتلك كافة أعضاء sysadmin و dbowner و ddladmin سماحيات القيام ببناء الجداول، فعندما يقوم أي مستخدم أو عضو من تلك المجموعات بإنشاء جدول جديد، فإن المالك الافتراضي لذلك الجدول هو dbo؛

- بعض معايير تسمية الجداول:

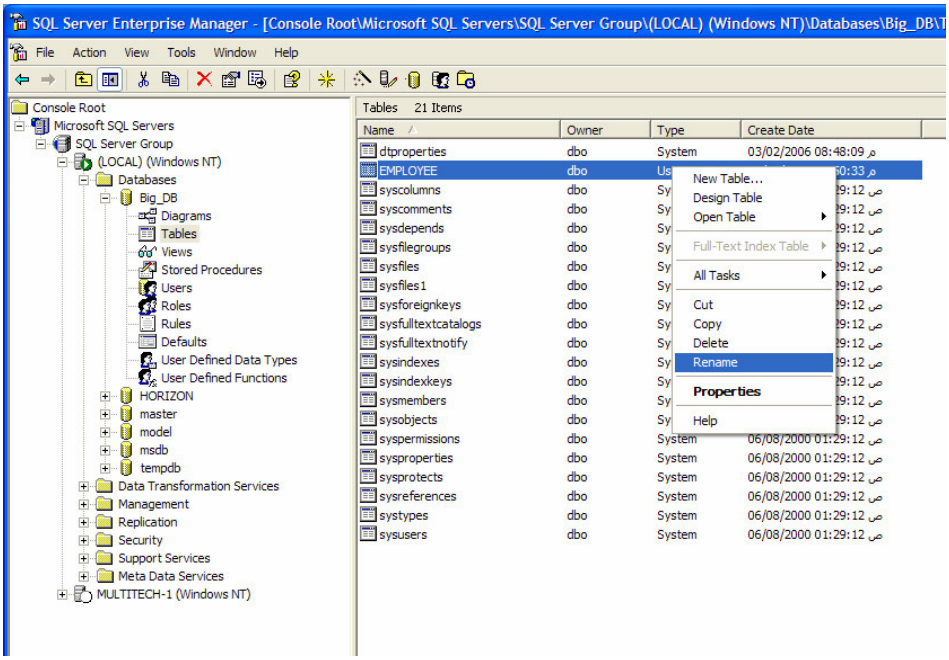
ينبغي عند تسمية جدول ما، أن يكون اسمه معبِّراً عن محتوياته، عما ينبغي تجنب استخدام الجمل أثناء التعبير عم اسم الجدول؛
ينبغي أن يكون اسم الجدول مفرداً، مثلاً Employee؛
يمكن أن يتسع اسم الجدول ليصل إلى 128 محرفاً تتضمن أحرفا و أرقاما بالإضافة إلى محارف خاصة أو فراغات؛
يفضل أن يكون المحرف الأول من اسم الجدول حرفاً أبجدياً؛

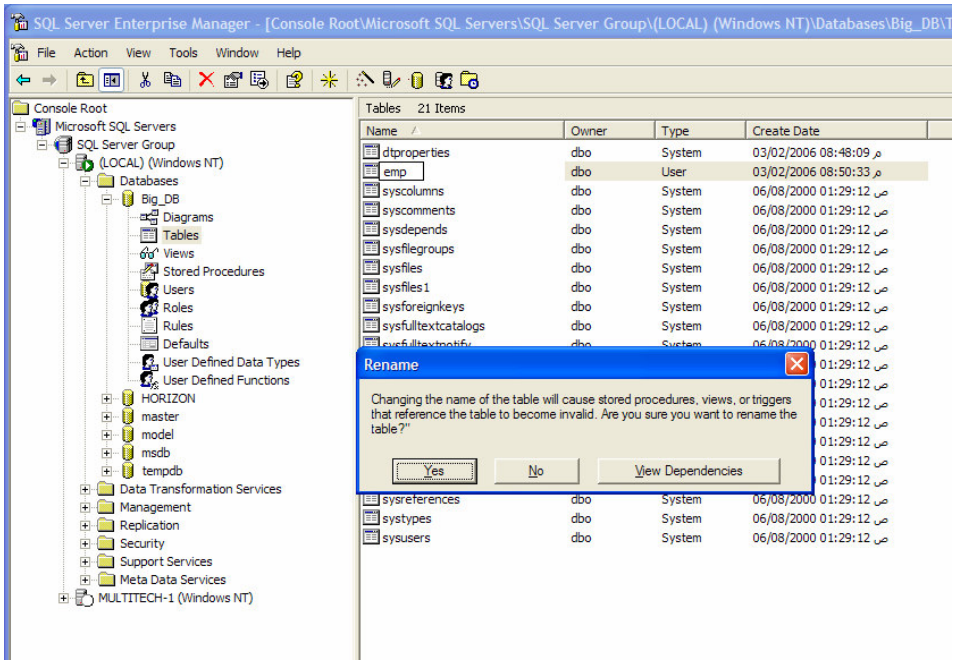
يفضل عادةً تجنب استخدام المحارف الخاصة في تسمية الجدول –مع العلم أنه يمكن استخدامها–، كما يمكن الاستعاضة عن الفراغ بالمحرف "_"، وهذا أمر شائع؛

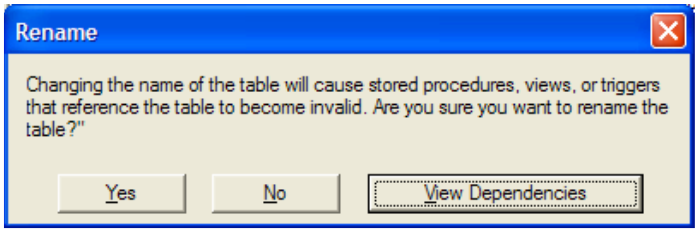
- إعادة تسمية الجدول:

كثيراً ما نرغب بتغيير اسم الجدول بعد بنائه، تعتبر هذه العملية متاحة من خلال الأداة SQL Server بأسلوبين مختلفين:

يمكننا تغيير اسم الجدول من خلال اختيار "rename" من قائمة المهمات السريعة لذلك الجدول، بحيث تظهر لنا واجهة خاصة لتحذيرنا من أن عملية التغيير تلك يمكن أن تؤثر على بعض الأغراض التي ترتبط بالجدول، كإجرائيات مخزّنة مثلاً أو توابع معرّفة أو مناظير أو غيرها، بحيث يمكننا من خلال هذه الواجهة استعراض تلك الأغراض واتخاذ القرار بتغيير اسم الجدول بعد ذلك، بالإضافة إلى أنه بإمكاننا التراجع عن تلك العملية؛







يمكننا أيضاً أن نقوم بتغيير اسم الجدول من خلال استخدام إجرائية النظام sp_rename والتي تأخذ معاملين أولهما اسم الغـــرض الذي نرغب بإعادة تسميته والثاني الاسم الجديد، مع العلم أنه يمكننا استعراض الأغراض الأخرى التي يمكن أن ترتبط بالجـــدول الذي نقوم بتغيير اسمه من خلال استخدام إجرائية النظام sp_depends.

بناء الجداول باستخدام مخطوطات T-SQL
تحديد الأعمدة وخصائصها

سنناقش في الشرائح التالية كيف يمكن استخدام مخطوطات T-SQL من أجل بناء الجداول وتحديد أعمدتها وخصائص تلك الأعمدة، بالإضافة إلى إمكانية ربط الجدول بخصائص أخرى كتحديد موقع الجدول أو إضافة شروط واختبارات ستناولها بالتفصيل لاحقاً؛

- تحديد الأعمدة:

ينبغي عند إنشاء جدول جديد أن نقوم بتحديد اسم كل عمود من أعمدته، بالإضافة إلى نمط المعطيات الخاص بكلٍ من تلك الأعمدة؛

- خصائص الأعمداة:

تمتلك الأعمدة أيضاً خصائص ينبغي أن يتم إسنادها إليها، ومنها –على سبيل المثال– إمكانية أن يحتوي العمود على قيم فارغة أو لا، أو أن يمتلك العمود لقيمة معينة أو أن يتمتع بخاصة التزايد المستمر identity property؛

  - القيم الفارغة NULLS:

يفضّل عادة التصريح عن امكانية أن يحتوي العمود على قيم فارغة، مع العلم أن العملية تلك اختيارية؛

  - خاصة التزايد المستمر identity property:

تعتبر هذه الخاصة من أهم خصائص الأعمدة وأكثرها شيوعاً واستخداماً؛

  - تحديد موقع الجدول:

يدعم SQL Server إمكانية تحديد مكان تخزين الجدول أو الفهرس، مع العلم أنه لا بد من الإشارة إلى أنه يمكن أن يكون لهذه العملية دور كبير في تحسين الأداء والمساهمة في تسهيل عملية التخزين الإحتياطي، بحيث يكفي إجراء عملية تخزين احتياطي لمجموعة ملفات من أجل حفظ كافة الجداول التي تتكون منها؛
يمكننا أيضاً أن نفصل الملفات النصية أو الصور ليتم تخزينها في أماكن منفصلة من القرص.

من أجل بناء الجداول وتحديد أعمدتها وخصائص تلك الأعمدة، بالإضافة إلى T-SQLسنناقش في الشرائح التالية كيف يمكن استخدام مخطوطات إمكانية ربط الجدول بخصائص أخرى كتحديد موقع الجدول أو إضافة شروط واختبارات ستناولها بالتفصيل لاحقاً؛

تحديد الأعمدة:

ينبغي عند إنشاء جدول جديد أن نقوم بتحديد اسم كل عمود من أعمدته، بالإضافة إلى نمط المعطيات الخاص بكلٍ من تلك الأعمدة؛
نعرض فيما يلي لمثال يوضح المخطوط اللازم لبناء جدول جديد:

```
CREATE TABLE  BigDB.dbo.EMPLOYEE (
        EMPLOYEE_NUMBER int NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE char (14) NULL,
        DEPT   smallint NULL,
        SALARY float NULL
)
```

- خصائص الأعمداة:

تمتلك الأعمدة أيضاً خصائص ينبغي أن يتم إسنادها إليها، ومنها –على سبيل المثال– إمكانية أن يحتوي العمود على قيم فارغة أو لا، أو أن يمتلك العمود لقيمة معينة أو أن يتمتع بخاصة التزايد المستمر identity property؛

- القيم الفارغة NULLS:

يفضّل عادة التصريح عن امكانية أن يحتوي العمود على قيم فارغة، مع العلم أن العملية تلك اختيارية؛
يسمح SQL Server للعمود بأن يأخذ القيمة NULL بشكل تلقائي أثناء إنشاءه؛

- خاصة التزايد المستمر identity property:

تعتبر هذه الخاصة من أهم خصائص الأعمدة وأكثرها شيوعاً واستخداماً؛
يمكن تشبيه نمط معطيات عمود يتمتع بهذه الخاصة بالنمط الرقمي العشري من دون توصيف للأرقام التي ترد بعد الفاصلة؛
تعتبر الفائدة الرئيسية من هذه الخاصة، هي توليد المفاتيح، خاصة وأن الأرقام التسلسلية المتزايدة تتميز بكونها فريدة على مستوى أسطر الجدول، مما يضمن عدم التكرار وعدم حدوث الأخطاء أثناء تعريف المفاتيح؛
ينبغي عند إعداد هذه الخاصة، أن نقوم بتحديد معاملين أساسيين، هما، البداية والتزايد، بحيث يعبّر البداية عن الرقم الذي سيستخدم لبدء عملية الترقيم، أما المعامل الثاني فيعبر عن مقدار التزايد الذي سيتم تطبيقه لإنشاء الرقم التالي؛
مثال:
سنعدل على تعليمة إنشاء الجدول في المثال السابق وسنقوم بتطبيق خاصة IDENTITY على العمود EMPLOYEE_NUMBER، بحيث يبدأ الترقيم من 100 ويزداد بمعدل 10 أرقام

```
CREATE TABLE  BigDB.dbo.EMPLOYEE (
        EMPLOYEE_NUMBER int IDENTITY (100, 10) NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE char (14) NULL,
        DEPT   smallint NULL,
        SALARY float NULL
)
```

-
-

- تحديد موقع الجدول:

مع ازدياد حجم قاعدة المعطيات، وخاصة الجداول والفهارس، ينبغي اتخاذ قرارات خاصة فيما يتعلق بتحسين الأداء وتسريع الاستعلامات؛

فعلى سبيل المثال يمكننا اعتبار وجود استعلامات كثيرة تنفَّذ على كل من الجدولين Employee و Dept في آن واحد، فلا بد هنا من الوقوف عد مكان تخزين كلا الجدولين، بحيث لا يعتبر التخزين المتفرّق لكل منهما، على جزءين مختلفين من القرص الصلب، مفيداً في مسألة تسريع الأداء؛

يدعم SQL Server إمكانية تحديد مكان تخزين الجدول أو الفهرس، مع العلم أنه لا بد من الإشارة إلى أنه يمكن أن يكون لهذه العملية دور كبير في تحسين الأداء والمساهمة في تسهيل عملية التخزين الإحتياطي، بحيث يكفي إجراء عملية تخزين احتياطي لمجموعة ملفات من أجل حفظ كافة الجداول التي تتكون منها؛

يمكننا أيضاً أن نفصل الملفات النصية أو الصور ليتم تخزينها في أماكن منفصلة من القرص؛

مثال:

سنعدل على تعليمة إنشاء الجدول في المثال السابق وسنقوم بإضافة التعليمات التي نقوم من خلالها بتحديد موقع التخزين المعتمد للجداول أو الصور:

```
CREATE TABLE   BigDB.dbo.EMPLOYEE (
        EMPLOYEE_NUMBER int IDENTITY (100, 10) NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE char (14) NULL,
        DEPT   smallint NULL,
        SALARY float NULL,
        PHOTO image NULL
)
ON DISK1

TEXTIMAGE_ON DISK3


GO


CREATE TABLE   BigDB.dbo.DEPT(
        DEPT_NUMBER int IDENTITY (10, 10) NOT NULL,
        NAME varchar (64)  NOT NULL,
        DESCRIPTION varchar (80) NOT NULL,
        CODE char (2) NULL
)
ON DISK2

GO
```

بناء الجداول باستخدام مخطوطات T-SQL

تحديد قيود الجداول

- تزوّدنا قيود الجداول بإمكانيات تضمن –إلى حدّ ما– تكامل المعطيات في تلك الجداول؛

- بالإضافة إلى قيد القيم الفراغة NOT NULL / NULL، يزوّدنا SQL Server بخمسة قيود من أنواع أخـــرى، وهـــي: PRIMARY KEY و FOREIGN KEY و UNIQUE و CHECK و DEFAULT؛

- سنتفرّد لاحقاً بشرائح خاصة لتفصيل كل من تلك القيود، ولكن سنقوم الآن بعرض المخطوطات التي تمكّننا من تعريف بعض القيود تلك أثناء وبعد إنشاء الجداول.

- تزوّدنا قيود الجداول بإمكانيات تضمن –إلى حدّ ما– تكامل المعطيات في تلك الجداول؛

- بالإضافة إلى قيد القيم الفراغة NOT NULL / NULL، يزوّدنا SQL Server بخمسة قيود من أنواع أخـــرى، وهـــي: PRIMARY KEY و FOREIGN KEY و UNIQUE و CHECK و DEFAULT؛

- سنتفرّد لاحقاً بشرائح خاصة لتفصيل كل من تلك القيود، ولكن سنقوم الآن بعرض المخطوطات التي تمكّننا من تعريف بعض القيود تلك أثناء وبعد إنشاء الجداول؛

- مثال1:

سنعدل على تعليمة إنشاء الجدول في المثال السابق وسنقوم بإضافة التعليمات التي نقوم من خلالها بإضافة المفاتيح الرئيسية لكـــلا الجدولين Employee و Dept بالإضافة إلى المفتاح الخارجي FOREIGN KEY إلى جدول الموظف:

```
CREATE TABLE   BigDB.dbo.EMPLOYEE (
        EMPLOYEE_NUMBER int IDENTITY (100, 10) CONSTRAINT EMP_PK
        PRIMARY KEY NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE char (14) NULL,
        DEPT   smallint CNSTRAINT EMP_DEPT_FK REFERENCES
        dept(DEPT_NUMBER) NOT NULL,
        SALARY float NULL,
        PHOTO image NULL
)
ON DISK1

TEXTIMAGE_ON DISK3


GO


CREATE TABLE   BigDB.dbo.DEPT(
        DEPT_NUMBER int IDENTITY (10, 10) CONSTRAINT DEPT_PK
        PRIMARY KEY NOT NULL,
        NAME varchar (64)  NOT NULL,
        DESCRIPTION varchar (80) NOT NULL,
        CODE char (2) NULL
)
```

- مثال2:

سنعدل على تعليمة إنشاء الجدول في المثال السابق وسنقوم بإضافة التعليمات التي نقوم من خلالها بإضافة المفاتيح الرئيسية لكـــلا الجدولين Employee و Dept بالإضافة إلى المفتاح الخارجي FOREIGN KEY إلى جدول الموظف وذلك بعد إنشاء الجداول:

```
CREATE TABLE   BigDB.dbo.EMPLOYEE (
EMPLOYEE_NUMBER int IDENTITY (100, 10) NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE char (14) NULL,
DEPT   smallint NOT NULL,
        SALARY float NULL,
        PHOTO image NULL
)
ON DISK1

TEXTIMAGE_ON DISK3


GO


CREATE TABLE   BigDB.dbo.DEPT(
DEPT_NUMBER int IDENTITY (10, 10) NOT NULL,
        NAME varchar (64)  NOT NULL,
        DESCRIPTION varchar (80) NOT NULL,
        CODE char (2) NULL
)
ON DISK2

GO

ALTER TABLE BigDB.dbo.DEPT ADD
CONSTRAINT DEPT_PK PRIMARY KEY(DEPT_NUMBER)
GO

ALTER TABLE  BigDB.dbo.EMPLOYEE ADD
CONSTRAINT EMP_PK PRIMARY KEY
(EMPLOYEE_NUMBER)
CNSTRAINT EMP_DEPT_FK FOREIGN KEY(DEPT)
REFERENCES BigDB.dbo.dept(DEPT_NUMBER)
```

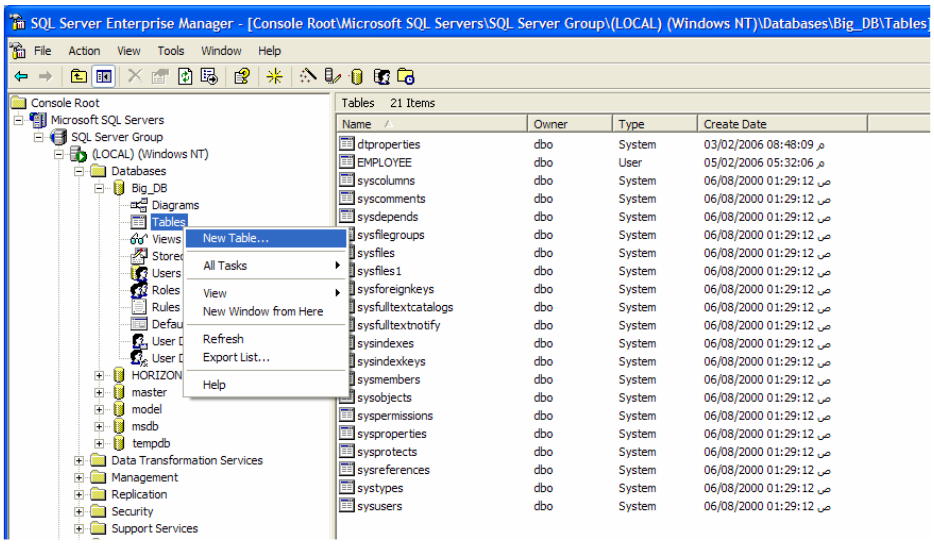
بناء الجداول باستخدام الأداة Enterprise Manager

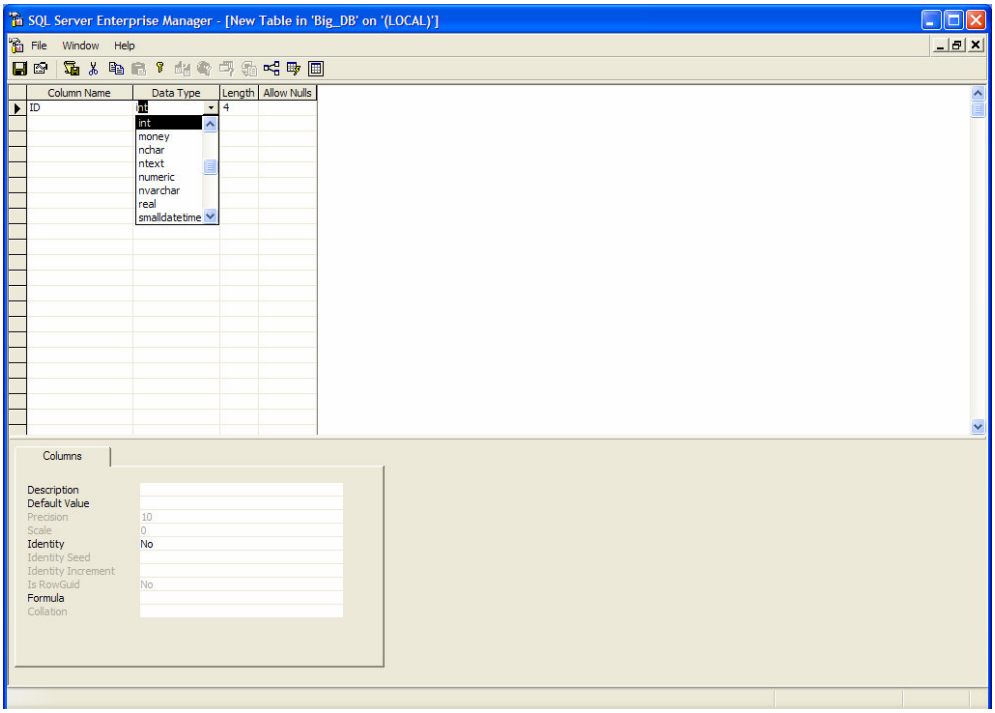
يؤمن SQL Server ومن خلال الأداة Enterprise Manager إمكانيات فعالة جداً في بناء جداول قاعدة المعطيات وذلــك مــن خلال استخدام واجهات تخاطبية سهلة الاستخدام؛

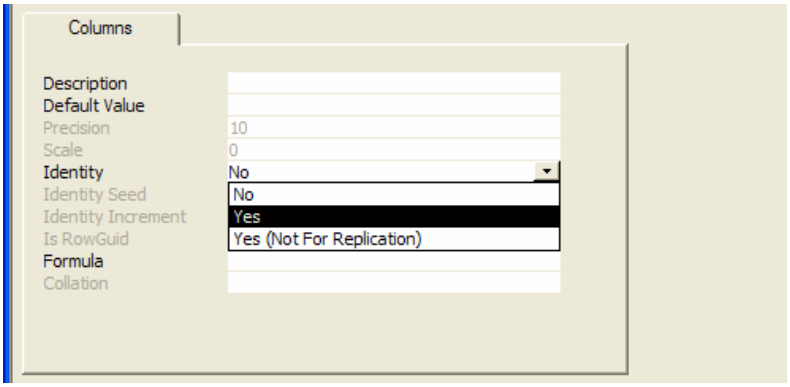
يمكننا إنشاء جدول جديد باستخدام واجهات المصمِّم التخاطبية من خلال اختيار "New Table" من قائمة المهام السريعة لمجلد "Tables" ضمن قاعدة المعطيات المختارة، لننتقل بعد ذلك إلى واجهة خاصة ببناء الجدول، بحيث نحدد فيها أسماء الأعمدة وأنماط معطياتها وحجومها، بالإضافة إلى إمكانية تحديد خصائص تلك الأعمدة، كأن نحدد فيما إذا يُسمح للعمود بأخذ قيم فارغة أو فيما إذا كان العمود يتمتع بخاصة التزايد المستمر أو خاصة قيمة تلقائية على سبيل المثال؛

يؤمن SQL Server ومن خلال الأداة Enterprise Manager إمكانيات فعالة جداً في بناء جداول قاعدة المعطيات وذلك من خلال استخدام واجهات تخاطبية سهلة الاستخدام؛

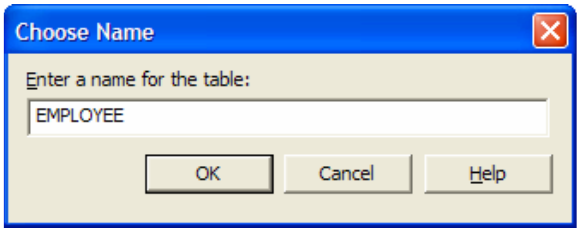
يمكننا إنشاء جدول جديد باستخدام واجهات المصمِّم التخاطبية من خلال اختيار "New Table" من قائمة المهام السريعة لمجلد "Tables" ضمن قاعدة المعطيات المختارة، لننتقل بعد ذلك إلى واجهة خاصة ببناء الجدول، بحيث نحدد فيها أسماء الأعمدة وأنماط معطياتها وحجومها، بالإضافة إلى إمكانية تحديد خصائص تلك الأعمدة، كأن نحدد فيما إذا يُسمح للعمود بأخذ قيم فارغة أو فيما إذا كان العمود يتمتع بخاصة التزايد المستمر أو خاصة قيمة تلقائية على سبيل المثال؛




SQL Server Enterprise Manager - [New Table in 'Big_DB' on '(LOCAL)']
File
Window
Help
Column Name
Data Type
Length
Allow Nulls
ID
int
4
int
money
nchar
ntext
numeric
nvarchar
real
smalldatetime
Columns
Description
Default Value
Precision
10
Scale
0
Identity
No
Identity Seed
Identity Increment
Is RowGuid
No
Formula
Collation


Columns
Description
Default Value
Precision
10
Scale
0
Identity
No
Identity Seed
Identity Increment
Is RowGuid
Formula
Collation
No
Yes
Yes (Not For Replication)

- بعد الانتهاء من تعريف وتحديد الأعمدة وخصائصها، يمكننا أن نقوم بتخزين التغيرات في الواجهة الأخيرة تلك، لتظهر مباشرة رسالة خاصة تطلب منّا إدخال اسم الجدول:

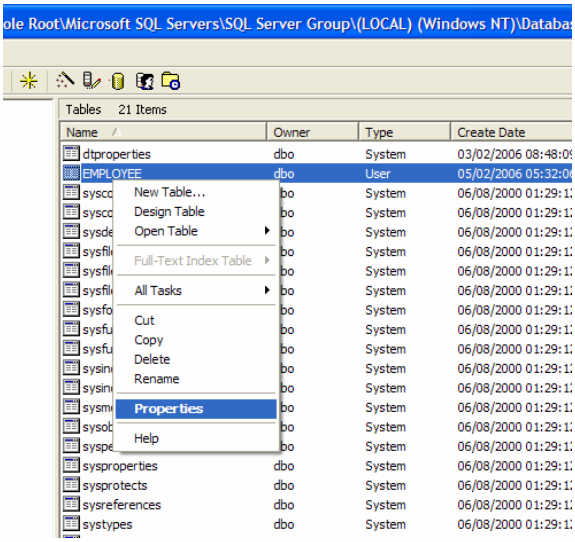


- يسمح لنا مصمِّم الجداول الذي تؤمنه الأداة Enterprise Manager بإمكانية بناء الجداول والأعمدة بأسلوب مرن جداً، بحيث لا يوجد أي داعٍ لكتابة مخطوطات sql للقيام بتلك العمليات، مع العلم أنه في الحقيقة يتم توليد تلك المخطوطات وتنفيذها آلياً من خلال هذه الأداة ولكن دون أن يشعر المستخدم بذلك.

استعراض خصائص الجداول باستخدام الأداة Enterprise Manager

- يمكننا وبسهولة بعد الانتهاء من عملية بناء الجداول باستخدام الأداة Enterprise Manager أن نقوم باستعراض خصائص تلك الجداول أو حتي القيام بالتعديلات على الأعمدة وخصائصها؛
- تسمح لنا واجهة خصائص الجدول باستعراض أعمدة الجدول وخصائصها، مع امكانيات إجراء تعديلات على بعضها، كما يمكننا من خلالها استعراض سماحيات الولوج المسموحه لمستخدمي SQL Server على هذا الجدول؛
- تسمح لنا واجهة خصائص الجدول والفهرس –التي يمكن الوصول إليها من داخل واجهة إدارة تصميم أعمدة الجدول– ببناء وإدارة القيود المختلفة أو الفهارس بالإضافة إلى التحكم بمالك الجدول أو بأماكن توضُّع مجموعات الملفات لذلك الجدول.

- يمكننا وبسهولة بعد الانتهاء من عملية بناء الجداول باستخدام الأداة Enterprise Manager أن نقوم باستعراض خصائص تلك الجداول أو حتي القيام بالتعديلات على الأعمدة وخصائصها؛
- تسمح لنا واجهة خصائص الجدول باستعراض أعمدة الجدول وخصائصها، مع امكانيات إجراء تعديلات على بعضها، كما يمكننا من خلالها استعراض سماحيات الولوج المسموحه لمستخدمي SQL Server على هذا الجدول؛



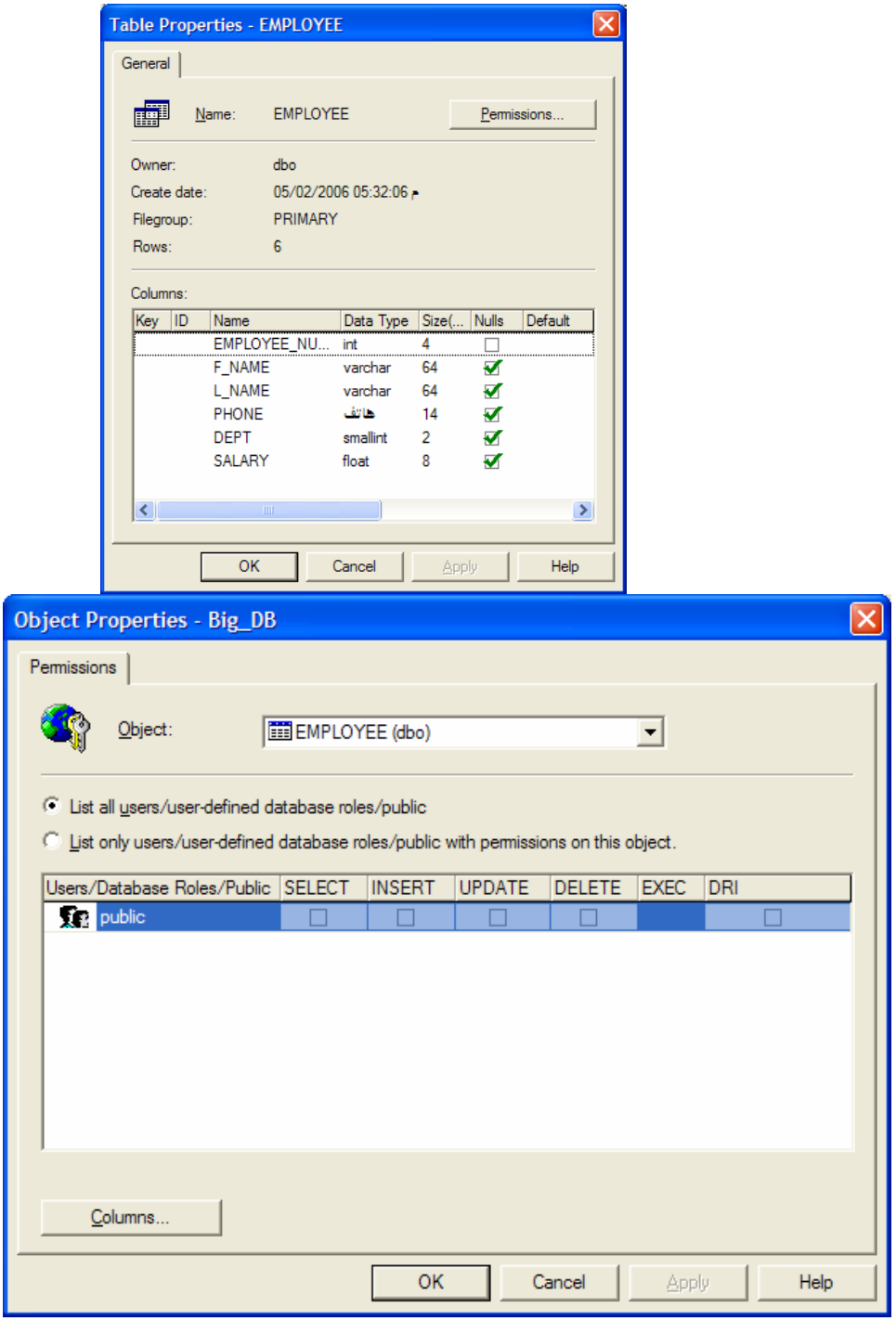
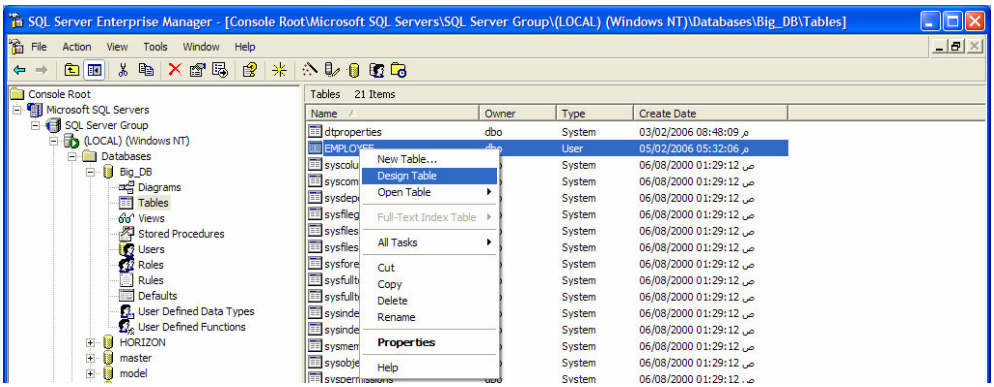
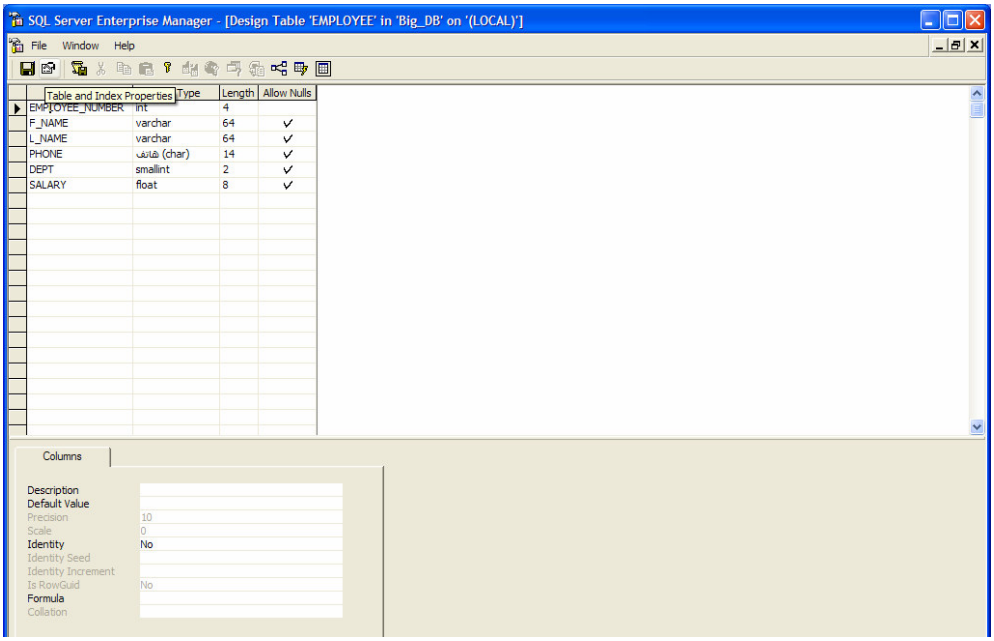
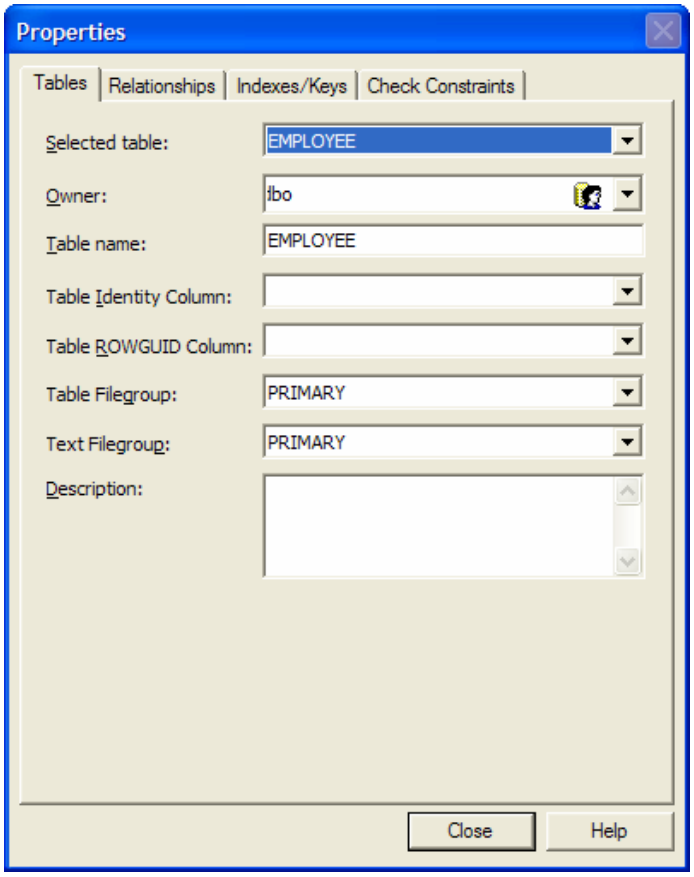


Table Properties - EMPLOYEE
General
Name: EMPLOYEE
Permissions...
Owner: dbo
Create date: 05/02/2006 05:32:06 م
Filegroup: PRIMARY
Rows: 6
Columns:
Key ID Name Data Type Size(... Nulls Default
EMPLOYEE_NU... int 4
F_NAME varchar 64
L_NAME varchar 64
PHONE هاتف 14
DEPT smallint 2
SALARY float 8
OK Cancel Apply Help
Object Properties - Big_DB
Permissions
Object: EMPLOYEE (dbo)
List all users/user-defined database roles/public
List only users/user-defined database roles/public with permissions on this object.
Users/Database Roles/Public SELECT INSERT UPDATE DELETE EXEC DRI
public
Columns...
OK Cancel Apply Help

- تسمح لنا واجهة خصائص الجدول والفهرس –التي يمكن الوصول إليها من داخل واجهة إدارة تصميم أعمدة الجدول– ببناء وإدارة القيود المختلفة أو الفهارس بالإضافة إلى التحكم بمالك الجدول أو بأماكن توضُّع مجموعات الملفات لذلك الجدول؛







استعراض مخطوطات T-SQL التي يتم توليدها من خلال الأداة Enterprise Manager

- قد يتساءل الكثير من المستخدمين الذين يسعون لكي يصبحوا مدراء قاعدة معطيات، لماذا ينبغي على مدير قاعدة المعطيات أن يتعامل مع مخطوطاتSQL –التي قد تكون معقدة للبعض– من أجل بناء قاعدة المعطيات أو الأغراض الأخرى، مع وجود واجهات تخاطبية وأدوات تصميم سهلة التعامل؛

- من جهة أخرى، يتجنّب العديد من مدراء قاعدة المعطيات المحترفين استخدام الواجهات التخاطبية لإنشاء أغراض قاعدة المعطيات، ويلتزمون باستخدام مخطوطات SQL، وذلك لأن المخطوطات تلك يمكن حفظها وتنفيذها في أي وقت، في حين ينبغي إعادة بناء تلك الأغراض التي استُخدمَت الواجهات في بنائها؛

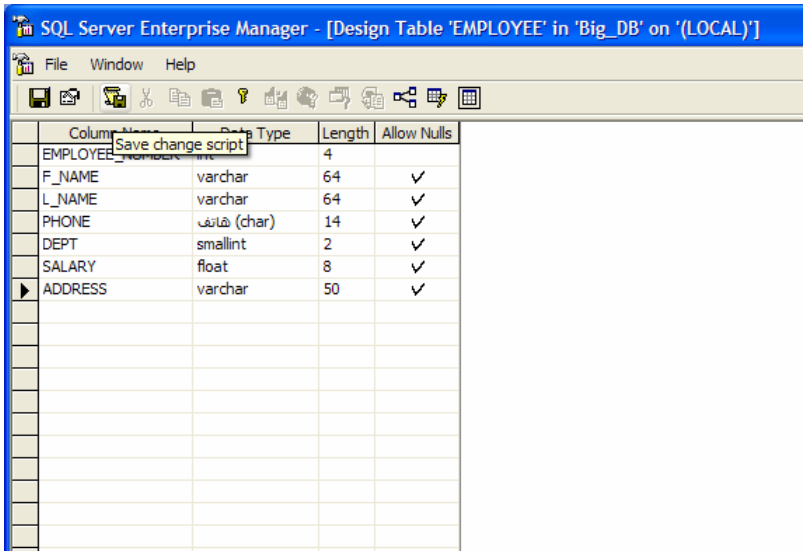
استطاعت مايكروسوفت من خلال الأداة SQL Server أن توجد حلاً نهائيا للمشكلة السابقة، وذلك من خلال توفير أساليب وأدوات عالية الفعالية لتوليد مخطوطات SQL تعبّر عن الأغراض التي يتم إنشاءها أو تعديلها.

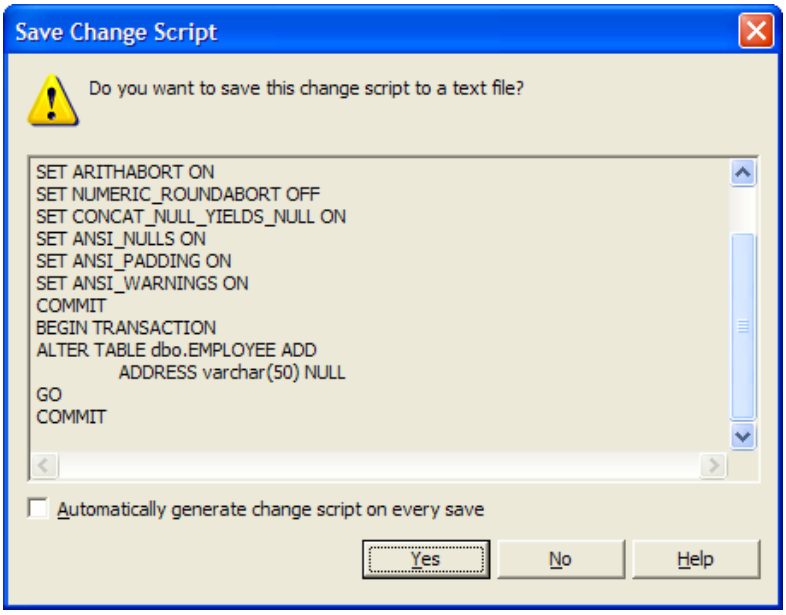
- قد يتساءل الكثير من المستخدمين الذين يسعون لكي يصبحوا مدراء قاعدة معطيات، لماذا ينبغي على مدير قاعدة المعطيات أن يتعامل مع مخطوطاتSQL –التي قد تكون معقدة للبعض– من أجل بناء قاعدة المعطيات أو الأغراض الأخرى، مع وجود واجهات تخاطبية وأدوات تصميم سهلة التعامل؛

- من جهة أخرى، يتجنّب العديد من مدراء قاعدة المعطيات المحترفين استخدام الواجهات التخاطبية لإنشاء أغراض قاعدة المعطيات، ويلتزمون باستخدام مخطوطات SQL، وذلك لأن المخطوطات تلك يمكن حفظها وتنفيذها في أي وقت، في حين ينبغي إعادة بناء تلك الأغراض التي استُخدمَت الواجهات في بنائها؛

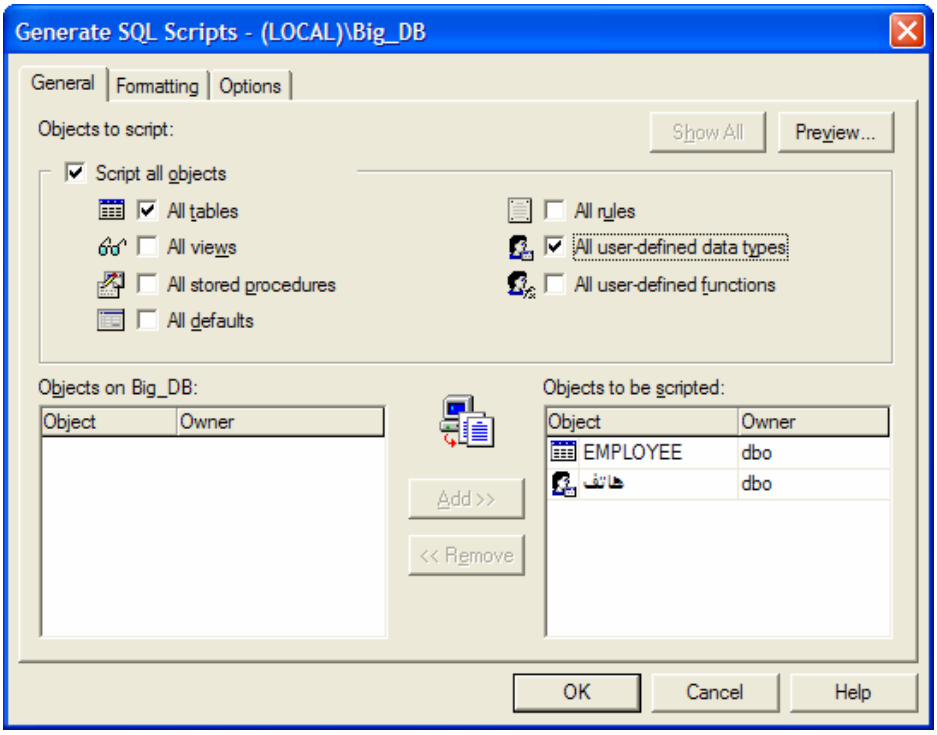
- استطاعت مايكروسوفت من خلال الأداة SQL Server أن توجد حلاً نهائيا للمشكلة السابقة، وذلك من خلال توفير أساليب وأدوات عالية الفعالية لتوليد مخطوطات SQL تعبّر عن الأغراض التي يتم إنشاءها أو تعديلها؛

- يمكننا من خلال أيقونة "Save change script" أن نحتفظ بالمخطوطات المتولدة أثناء إنشاء أو تعديل الجدول المحدد أو أيه خاصة من خصائصة:





يمكننا أيضاً –وفي أي وقت– توليد مخطوطات كاملة للجداول أو أية أغراض أخرى ضمن قاعدة المعطيات وذلك من خلال اختيار "Generate SQL Script" من قائمة "أدوات" في الأداة SQL Enterprise Manager.



استعراض معطيات الجداول من خلال الأداة Enterprise Manager

يمكننا من خلال الأداة Enterprise Manager أن نقوم باستعراض معطيات جدول معين أو حتى تعديل تلك المعطيات، وذلك من خلال اختيار "Open Table" من قائمة المهمات السريعة للجدول المختار، نلاحظ بعد ذلك وجود ثلاثة خيارات، وهي:

- Return All Rows:

يعيد هذا الخيار كافة أسطر الجدول؛

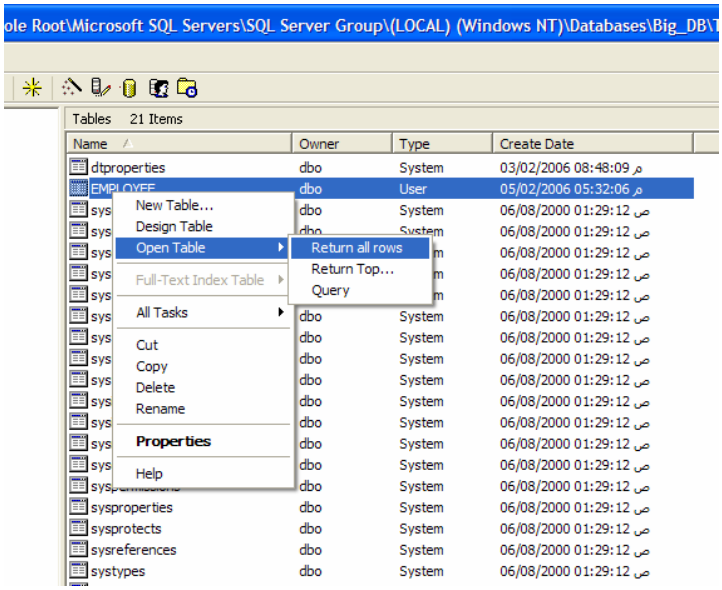
- Return Top:

يسمح هذا الخيار للمستخدم بتحديد رقم ليعيد من الجدول كافة الأسطر بعدد أعظمي يكافئ الرقم المُدخل؛

- Query:

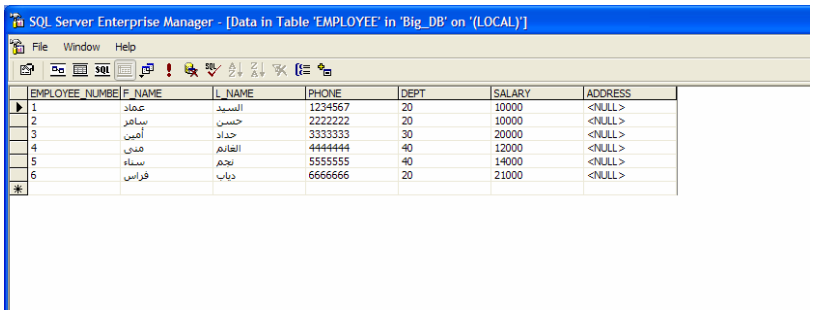
يمكننا من خلال هذا الخَيار أن نحدد استعلاماً مناسبا للأسطر التي نرغب بإعادتها، بحيث تظهر لنا واجهات خاصة تسمح ببناء الاستعلام الذي نريده.

- يمكننا من خلال الأداة Enterprise Manager أن نقوم باستعراض معطيات جدول معين أو حتى تعديل تلـك المعطيـات، وذلك من خلال اختيار "Open Table" من قائمة المهمات السريعة للجدول المختار، نلاحظ بعد ذلك وجود ثلاثة خيارات، وهي:



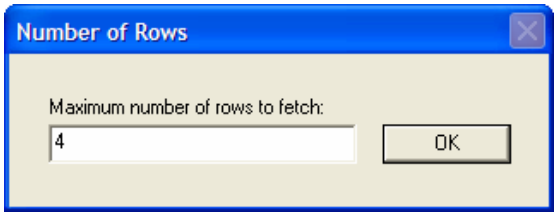
- Return All Rows:

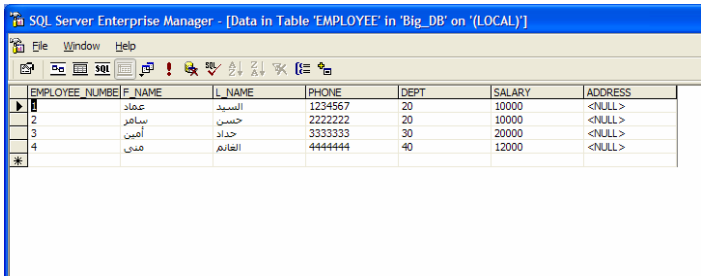
يعيد هذا الخيار كافة أسطر الجدول؛



- Return Top:

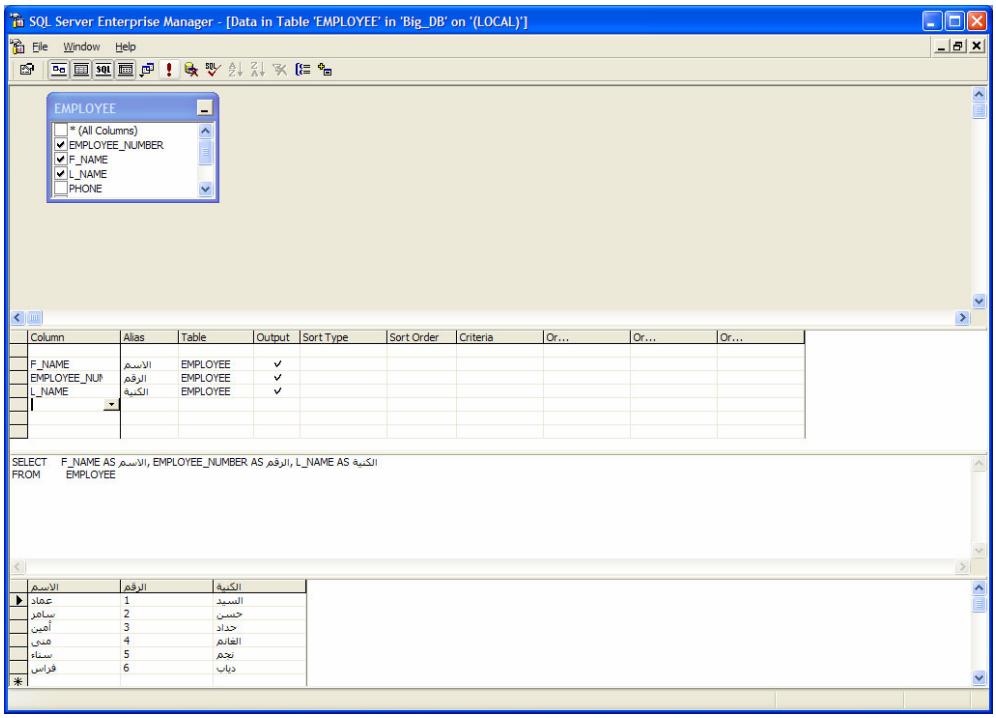
يسمح هذا الخيار للمستخدم بتحديد رقم ليعيد من الجدول كافة الأسطر بعدد أعظمي يكافئ الرقم المُدخل؛





- Query:

يمكننا من خلال هذا الخَيار أن نحدد استعلاماً مناسبا للأسطر التي نرغب بإعادتها، بحيث تظهر لنا واجهات خاصة تسمح ببناء الاستعلام الذي نريده.



حذف الجداول

نستطيع حذف الجداول من قاعدة المعطيات بطريقتين مختلفتين، إما من خلال استخدام تعليمة DROP TABLE كما في المثال التالي:

DROP TABLE *EMPLOYEE*

مع العلم أنه يمكن حذف عدة جداول بآن واحد من خلال تعليمة DROP TABLE وحيدة، وذلك بسرد أسماء تلك الجداول مفصولة عن بعضها البعض باستخدام فاصلة ","؛
أو من خلال الأداة Enterprise Manager وذلك باختيار "Delete" من قائمة المهمات السريعة للجدول الذي نرغب بحذفه؛

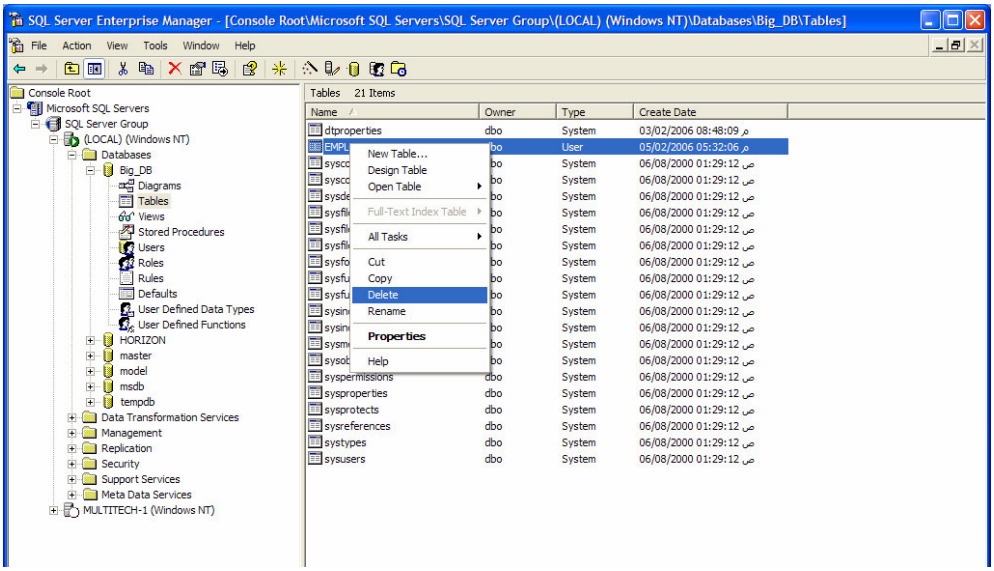
عندما نقوم بحذف جدول ما يحتوي في أعمدته على مفتاح خارجي من جدول آخر، فإنه لا بد لنا من حذف الجدول المانح للمفتاح قبل الجدول الحاوي على المفتاح الخارجي، إذا ما أردنا حذف هذا الأخير.

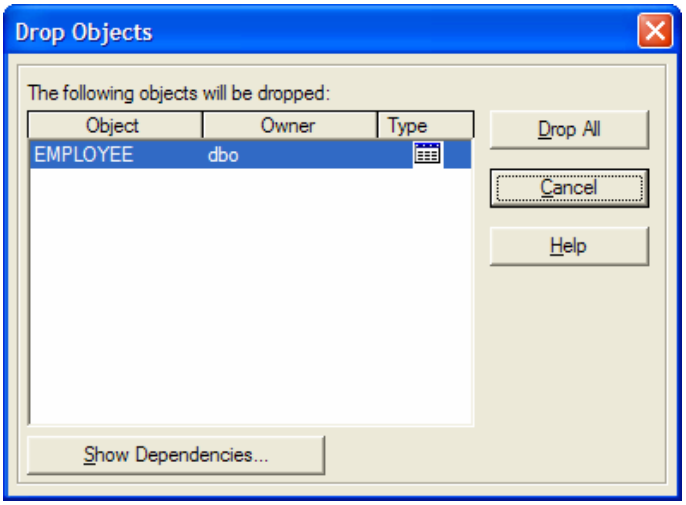
إن القيام بحذف جدول ما، يؤدي بالضرورة إلى حذف كافة القيود أو القوادح المرتبطة بذلك الجدول؛
نستطيع حذف الجداول من قاعدة المعطيات بطريقتين مختلفتين، إما من خلال استخدام تعليمة DROP TABLE كما في المثال التالي:

DROP TABLE *EMPLOYEE*

مع العلم أنه يمكن حذف عدة جداول بآن واحد من خلال تعليمة DROP TABLE وحيدة، وذلك بسرد أسماء تلك الجداول مفصولة عن بعضها البعض باستخدام فاصلة ","؛

أو من خلال الأداة Enterprise Manager وذلك باختيار "Delete" من قائمة المهمات السريعة للجدول الذي نرغب بحذفه.





عندما نقوم بحذف جدول ما يحتوي في أعمدته على مفتاح خارجي من جدول آخر، فإنه لا بد لنا من حذف الجدول المانح للمفتـــاح قبل الجدول الحاوي على المفتاح الخارجي، إذا ما أردنا حذف هذا الأخير.

إن القيام بحذف جدول ما، يؤدي بالضرورة إلى حذف كافة القيود أو القوادح المرتبطة بذلك الجدول؛

الجداول المؤقتة

- تعتبر الجداول المؤقتة مفيدة، بحيث تستخدم لتخزين المعطيات مؤقتاً أثناء القيام بعمليات معالجة، كبناء جدول ما يستخدم معطيات مشتقّة ناتجة عن مجموعة حسابات على سبيل المثال؛
- تعتبر الجداول في SQL Server مؤقتة، إذا ما سبق اسمها الرمز "#"؛
- يتم إنشاء الجداول المؤقتة بشكل تلقائي في قاعدة المعطيات tempdb، ويتم حذف تلك الجداول–ان لم يتم ذلك بشكل صريح– عندما تنتهي الجلسة التي تم إنشاؤهم فيها؛
- يمكننا أن نميز بين نوعين من الجداول المؤقتة، الجداول المؤقتة الخاصة والجداول المؤقتة العامة:
  - تتم الإشارة إلى الجداول المؤقّتة الخاصة من خلال استخدام الرمز # قبل اسم الجدول، وهي تتبع فقط للجلسة التي تم إنشاؤها فيها؛
  - تتم الإشارة إلى الجداول المؤقّتة العامّة من خلال استخدام الرمز ## قبل اسم الجدول، وهي تتميز بإمكانية الولوج إليها من قبل أية اتصال بأية قاعدة معطيات؛
- تعتبر الجداول الأخرى التي يتم إنشاؤها في قاعدة المعطيات tempdb والتي لا تُسبق بالرمز #، جداول معطيات مؤقتة أيضاً، إلا أنها –في الحقيقة– أكثر استدامةً من الجداول الأخرى، بحيث لا يتم حذفها تلقائياً، إنما يتم ذلك مع إعادة تشغيل الأداة SQL Server بحيث يُعاد بناء قاعدة المعطيات tempdb ككل من جديد.

الفهارس

مقدمة وتعريف

- مفهوم كومة المعطيات؛

أسلوب لتخزين المعطيات في الجداول. تراكمياً؛

- المسح الكامل للجدول:

عبارة عن عملية مسح لكامل لكومة المعطيات حتى العثور على النتيجة؛

يمكن تشبيه العملية السابقة بعملية بحث عن معلومه معينة في كتاب لا يحتوي على أي فهرس، بالتالي، وفي كل مرّة نرغب فيها بالبحث عن ذلك الموضوع ينبغي العودة إلى الصفحة الأولى من الكتاب، والتنقل صفحة تلو الأخرى حتى الوصول إلى الهدف المراد؛

- يمكننا تشبيه فهرس المعطيات تماماً بفهرس الكتاب؛

- تضمن الفهارس عادةً تحسين الأداء وتسريع الاستعلامات.

- مرّ معنا مسبقاً أن جداول قاعدة المعطيات عبارة عن بنية لتخزين المعطيات، –وفي الحقيقة– يتم تخزين المعطيات تلك في الجدول بدون أي أساس من الترتيب، بحيث تتم عملية إضافة المعطيات بشكل تراكمي، وهذا ما يُعرف باسم الكومة؛

- يتم –في كل مرة نرغب فيها بالبحث عن جزء من المعطيات– إجراء عملية مسح لكامل كومة المعطيات حتى العثور على النتيجة، وهذا ما يُعرف باسم "المسح الكامل للجدول"؛

- يمكن تشبيه العملية السابقة بعملية بحث عن معلومه معينة في كتاب لا يحتوي على أي فهرس، بالتالي، وفي كل مرّة نرغب فيها بالبحث عن ذلك الموضوع ينبغي العودة إلى الصفحة الأولى من الكتاب، والتنقل صفحة تلو الأخرى حتى الوصول إلى الهدف المراد؛

- كما يبدو مما سبق، تعتبر عملية البحث ضمن كومة معطيات غير مفهرسة بالعملية المضنية جداً؛

- يمكننا تشبيه فهرس المعطيات تماماً بفهرس الكتاب، فهو عبارة عن دليل معنون بحسب مفتاح معين، فكما أن فهرس الكتاب يتكوّن من محددين أساسيين هما العنوان ورقم الصفحة، كذلك فإن فهرس المعطيات يتكون من محددين أساسيين هما المفتاح، كرقم الموظف على سبيل المثال، ورقم معرّف لسطر المعطيات الذي يحتوي على المعلومة المطلوبة؛

- تضمن الفهارس عادةً تحسين الأداء وتسريع الاستعلامات، كما أنها تستخدم في كثير من الأحيان لضمان عدم تكرار المعطيات كما في حالالت المفاتيح الأولية أو شروط التفرُّد.

القسم الثاني

ما هو مدى استخدام الفهارس؟

- **لماذا لا يتم استخدام الفهارس واعتمادها في كل شيء؟**

- مساوئ الفهارس؛

• نعلم إذاً أن الفهارس عادةً ما تؤدي إلى تحسين الأداء وتسريع الاستعلامات،بالإضافة إلى أنها تستخدم في كثير من الأحيان لضمان عدم تكرار المعطيات، ولكن، وفي هذه الحالة قد يخطر ببالنا سؤال هام!!! **لماذا لا يتم استخدام الفهارس واعتمادها في كل شيء؟**

لا تعتبر الفهارس الخيار الأمثل دائماً، فعلى الرغم من أهميتها الكبيرة جداً وضرورة استخدامها الملحّه في جداول قاعدة المعطيات، إلا أن هناك بعض القيود التي لابد من أخذها بعين الاعتبار والتي تؤثر على مسألة اعتماد الفهارس في كل شيء، منها:

- الحجم الإضافي:

تتطلب الفهارس حجوم تخزين إضافية على القرص، تماماً كما يتطلب فهرس الكتاب صفحاتٍ إضافية؛

- خير الأمور الوسط:

إذا أردنا فهرسة كل كلمة من الكتاب فسنحصل على فهرس أكبر من الكتاب نفسه؛

- عبء إضافي:

ينبغي، ومع تزايد عمليات الإضافة والحذف والتعديل على معطيات الجدول، أن يتم إجراء تحديثات مماثلة عل فهارس تلك المعطيات، بالتالي، يمكننا هنا أن نتوقع العبء الناتج عن ازياد وتيرة تلك العمليات وما قد يؤثر على الأداء بشكل عام؛

على الرغم من النقاط السلبية السابقة التي يمكن أن تحدث أثناء استخدام الفهارس، إلا أن استخدام هذه البنية له دور أساسي في عملية إدارة قاعدة المعطيات، ولا يمكن الاستغناء عنه.

أنواع الفهارس

سنقوم بدراسة نوعين أساسيين من أنواع الفهارس المستخدمة المستخدمة من قبل الأداة SQL Server، وسنراعي من خلال الشرائح التالية كيفية إنشاء وإدارة كلا من هذين النوعين بالتفصيل:

- الفهارس العنقودية:

يتم عادةً في الفهارس العنقودية تخزين المعطيات في الجداول بشكل مرتب على أساس قيم مفتاح الفهرس، يقول البعض، أن كافة معطيات الجدول مخزّنة في الفهرس بشكل مرتّب، بينما يعتبر البعض الآخر أن أوراق الفهرس، أو المستوى الأخير منه، هو الجدول بحد ذاته؛

- الفهارس غير العنقودية:

تعتمد الفهارس غير العنقودية على مفتاح الفهرس العنقودي المنشأ على الجدول من أجل البحث عن المعطيات؛
لا يتم عادةً في الفهارس غير العنقودية إعادة ترتيب معطيات الجدول، بالتالي يمكن استخدام أكثر من فهرس غير عنقودي في الجدول؛

- سنقوم بدراسة نوعين أساسيين من أنواع الفهارس المستخدمة المستخدمة من قبل الأداة SQL Server، وسنراعي من خلال الشرائح التالية كيفية إنشاء وإدارة كلا من هذين النوعين بالتفصيل:

- يدعم SQL Server استخدام الفهارس العنقودية وغير العنقودية، حيث يعتبر الاختلاف الرئيسي بينهما هو في طريقة تخزين المعطيات في الجداول، بحيث تكون مرتبة في الأول وعشوائية في الثاني؛
- الفهارس العنقودية:

يتم عادةً في الفهارس العنقودية تخزين المعطيات في الجداول بشكل مرتب على أساس قيم مفتاح الفهرس، يقول البعض، أن كافة معطيات الجدول مخزّنة في الفهرس بشكل مرتّب، بينما يعتبر البعض الآخر أن أوراق الفهرس، أو المستوى الأخير منه، هو الجدول بحد ذاته؛
باعتماد مفهوم الفهارس العنقودية، يختفي مفهوم الكومة نهائياً، كما أنه لا يمكن استخدام أكثر من فهرس عنقودي وحيد ضمن الجدول، وذلك لأن المعطيات أصبحت تُخزَّن الآن مرتّبة فيزيائياً ضمن الجدول؛
تقدِّم الفهارس العنقودية إمكانية ولوج سريع للقيم التي يتم البحث عنها –بشكل متكرر– اعتماداً على مجال معين أو تلك التي يتم الوصول إليها علىأساس مرتَّب؛

مثال:

يعتبر اعتماد فهرس على الاسم والكنية –في جدول دليل الهاتف مثلاً– أكثر فعاليّة بشكل ملحوظ من اعتماد فهرس على رقم الهاتف في ذلك الجدول، وذلك على اعتبار أن البحث عن "باسم يوسف" على سبيل المثال سيعيد كافة النتائج المطلوبة بشكل أكثر فعالية بحيث أن الجدول مرتب على أساس الاسم والكنية وكافة النتائج المطلوبة يفترض أن تكون مخزنة بالقرب من بعضها البعض اعتمادا علىالفهرس.

- الفهارس غير العنقودية:

لا يتم عادةً في الفهارس غير العنقودية، وكما يبدو من اسمها، إعادة ترتيب معطيات الجدول، بالتالي يمكن استخدام أكثر من فهرس غير عنقودي في الجدول؛

يمكننا إنشاء ما يصل إلى 249 فهرس غير عنقودي لكل جدول، مع العلم أنه لا يوجد داع لاستخدام عدد كبير جدا من الفهارس على الجدول، فكما مرّ معنا سابقاً، يمكن أن تؤثر زيادة عدد الفهارس سلباً؛

تعتمد الفهارس غير العنقودية على مفتاح الفهرس العنقودي المنشأ على الجدول من أجل البحث عن المعطيات؛

بما أن المعطيات غير مرتّبة ضمن الفهارس غير العنقودية، بالتالي لا يعتبر استخدام هذا النوع من الفهارس مفيداً في عمليات البحث عن مجال معطيات، في حين يعتبر استخدام الفهارس غير العنقودية فعّالاً بشكل واضح عندما نرغب بالبحث عن قيم ثابتة؛

مثال:

بالعودة إلى جدول دليل الهاتف في المثال السابق، يعتبر البحث عن شخص معين في الدليل اعتماداً على رقم هاتفه أكثر فعالية باستخدام فهرس غير عنقودي مطبّق على رقم الهاتف.

إنشاء الفهارس باستخدام تعليمات T-SQL

- سنناقش فيما يلي القواعد التي يمكن استخدامها لإنشاء الفهارس باستخدام T-SQL؛

- يفضّل عادةً أن يفهم مدير قاعدة المعطيات تماماً للقواعد المستخدمة في بناء وإنشاء الفهارس، خاصة وأن هذه العملية من العمليات التي تتكرر بشكل كبير، بجيث يتم حذف وإعادة بناء اللفهارس بشكل مستمر من أجل عمليات تحسين الأداء؛

سنناقش فيما يلي القواعد التي يمكن استخدامها لإنشاء الفهارس باستخدام T-SQL؛

يفضّل عادةً أن يفهم مدير قاعدة المعطيات تماماً للقواعد المستخدمة في بناء وإنشاء الفهارس، خاصة وأن هذه العملية من العمليات التي تتكرر بشكل كبير، بجيث يتم حذف وإعادة بناء اللفهارس بشكل مستمر من أجل عمليات تحسين الأداء؛

يتم إنشاء باستخدام تعليمة CREAT INDEX، وفيما يلي عرض لكامل قواعد تلك العملية:

CREATE [UNIQUE] [CLUSTERED] [NONCLUSTERED] INDEX *indexName*
ON { *table* | *view* }
(*column* [ASC | DESC] [, … n])
[WITH < index_option> [, … n]]

[ON *fileGroup*]
<indexOption> :: =
{ PAD_INDEX |
FILLFACTOR = *fillFactor* |
IGNORE_DUP_KEY |
DROP_EXISTING |
STATISTICS_NORECOMPUTE |
SORT_IN_TEMPDB

الشرح المرافق للجدول السابق:

UNIQUE	للتأكيد بعدم السماح بتكرار الأسطر، فإذا وُجِد تكرار أسطر في المعطيات يفشل إنشاء الفهرس ككل.
CLUSTERED | NONCLUSTERED	للتعبير أن الفهرس المنشأ عنقودي أو غير عنقودي، مع العلم أن NONCLU هي القيمة التلقائية، وأنه لا يمكن إنشاء أكثر من فهرس عنقودي وحيد لكل جدول.
indexName	عبارة عن اسم الفهرس المراد إنشاؤه.
table | *view*	ِل أو المنظار الذي نرغب بإنشاء الفهرس عليه، بحيث يدعم SQL Server امكانية إنشاء فهارس على المناظير أيضاً.
column	يعبر عن العمود أو الأعمدة التي نرغب بفهرستها.
ASC | DESC	للتعبير عما إذا كان ينبغي أن يتم ترتيب الفهرس تصاعدياً أم تنازلياً مع العلم أن القيمة ASC هي القيمة التلقائية.
ON *fileGroup*	لتحديد أية مجموعة ملفات ينبغي أن يتم تخزين الفهرس عليها.
PAD_INDEX	للتعبير عن ضرورة ترك نسبة فراغ في المستويات غير الورقيّة من الفهرس، مع العلم أن تلك النسبة يتم تحديدها من خلال المعامل FILLFACTOR.
FILLFACTOR = *fillFactor*	تعبر عن النسبة المستخدمة لملئ الصفحات الورقية من الفهرس أثناء إنشاءه، تفيد هذه الخاصة بتجنُّب إجراء تقسيم صفحات الفهرس عند امتلائها بالمعطيات بعد إجراء عمليات إضافة أو تعديل.
IGNORE_DUP_KEY	ستخدام هذه الخاصة عندما يتم إنشاء فهرس من نو ع UNIQUE، والهدف منها عمد التراجع عن المناقلة التي تحتوي عملية إضافة لمفتاح مكرر، بحيث يتم رفض التكرار ولكن لا يتم التراجع عن كامل المناقلة.
DROP_EXISTING	تستخدم هذه العملية لحذف الفهرس في حال وجودة عندما نقوم بإنشائه

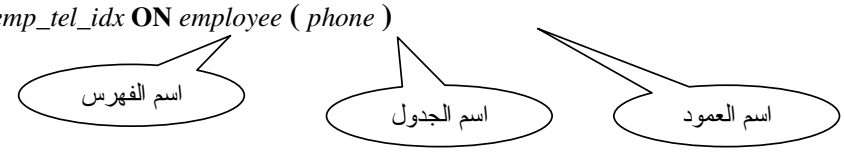
لاسم، مع العلم أنه يمكن تحسين الأداء بشكل ملحوظ عندما نقوم بإعادة بناء الفهارس العنقودية.	
للتعبير عن أنه لن يتم إعادة حساب إحصائيات الجدول تلقائياً.	STATISTICS_NORECOMPUTE
عن إمكانية استخدام قاعدة المعطيات TEMPDB لتخزين عمليات الترتيب المرحلية التي يتم استخدامها أثناء بناء الجدول.	SORT_IN_TEMPDB

مثال عن إنشاء الفهارس باستخدام تعليمات T-SQL

- سنقوم فيما يلي بتطبيق بعض القواعد التي مرّت معنا في الشريحة السابقة وذلك لإنشاء الفهارس باستخدام مخطوطات T-SQL؛

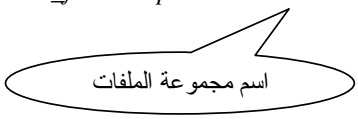
- يعبر المثال التالي عن مخطوط SQL الأكثر اختصاراً والذي يمكننا من خلاله إنشاء فهرس غير عنقودي جديد في **قاعدة المعطيات** الحالية و**مجموعة الملفات** الحالية أيضاً:

CREAT INDEX *emp_tel_idx* **ON** *employee* **(** *phone* **)**



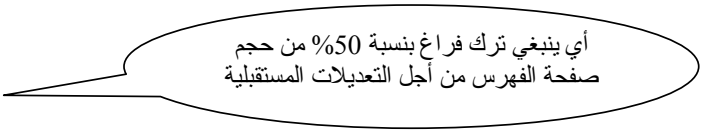
- يُنصح في معظم الحالات أن يتم فصل مكان تخزين الفهرس عن الجدول، بالتالي يمكننا إضافة الفهرس إلى مجموعة ملفات خاصة كما في المثال التالي:

CREAT INDEX *emp_tel_idx* **ON** *employee* **(** *phone* **)**
ON *index_fileGroup1*



- يعبر المثال التالي عن مخطوط SQL أكثر تعقيداً والذي يمكننا من خلاله إنشاء فهرس عنقودي فريد ذو خصائص مختلفة:

CREAT UNIQUE CLUSTERED INDEX *emp_tel_idx*
ON *employee* **(** *phone* **)**



WITH
PAD_INDEX,
FILLFACTOR = 50,
IGNORE_DUP_KEY,
STATISTICS_NORECOMPUTE
ON *index_fileGroup1*

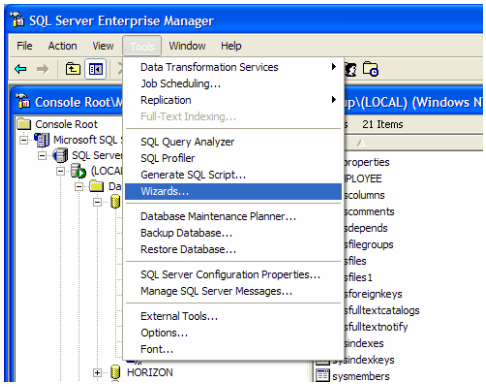
إنشاء الفهارس باستخدام الأداة Enterprise Manager

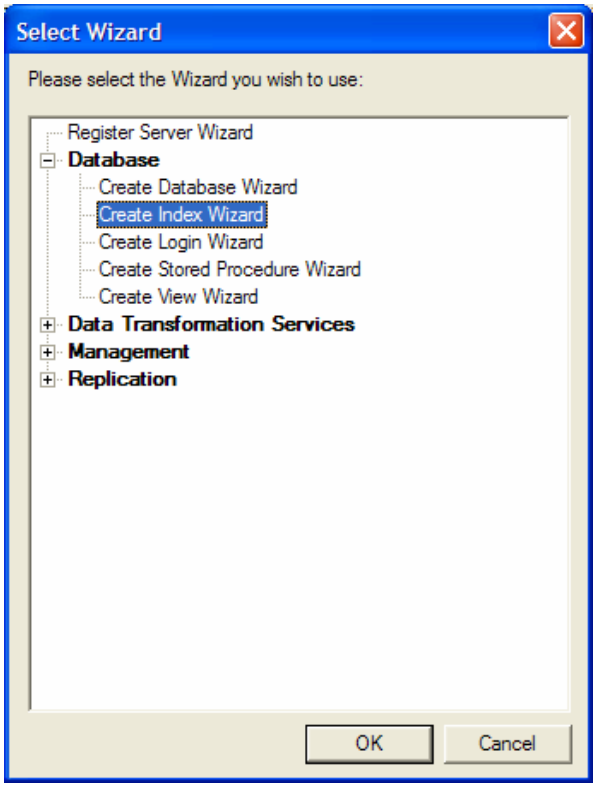
- تقدم الأداة SQL Server Enterprise Manager العديد من التسهيلات فيما يتعلّق ببناء الفهارس، بحيث يمكن إنشاء الفهارس من خلال واجهات مستخدم تخاطبية سهلة الاستخدام وبطريقتين مختلفتين:
  - من خلال استخدام Create Index Wizard؛
  - من خلال خصائص الجدول نفسه في قاعدة المعطيات.

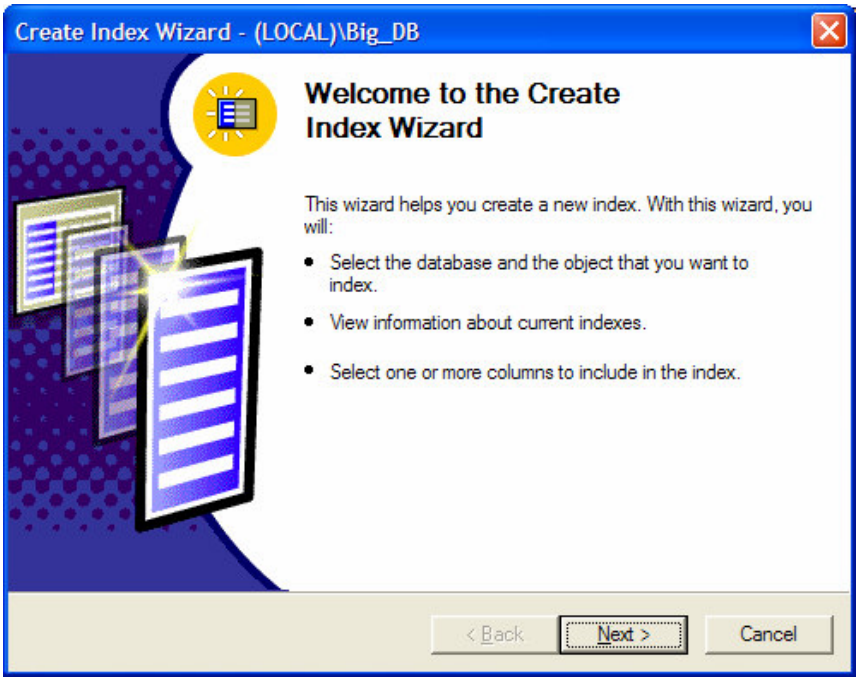
- تقدم الأداة SQL Server Enterprise Manager العديد من التسهيلات فيما يتعلّق ببناء الفهارس، بحيث يمكن إنشاء الفهارس من خلال واجهات مستخدم تخاطبية سهلة الاستخدام وبطريقتين مختلفتين:

- من خلال استخدام Create Index Wizard:

يمكننا الوصول إلى الواجهات الخاصة بـ Create Index Wizard من خلال القائمة "Tools" ثم اختيار "Wizards..."، بعد ذلك يمكننا اختيار Create Index Wizard من ضمن القائمة "Database".

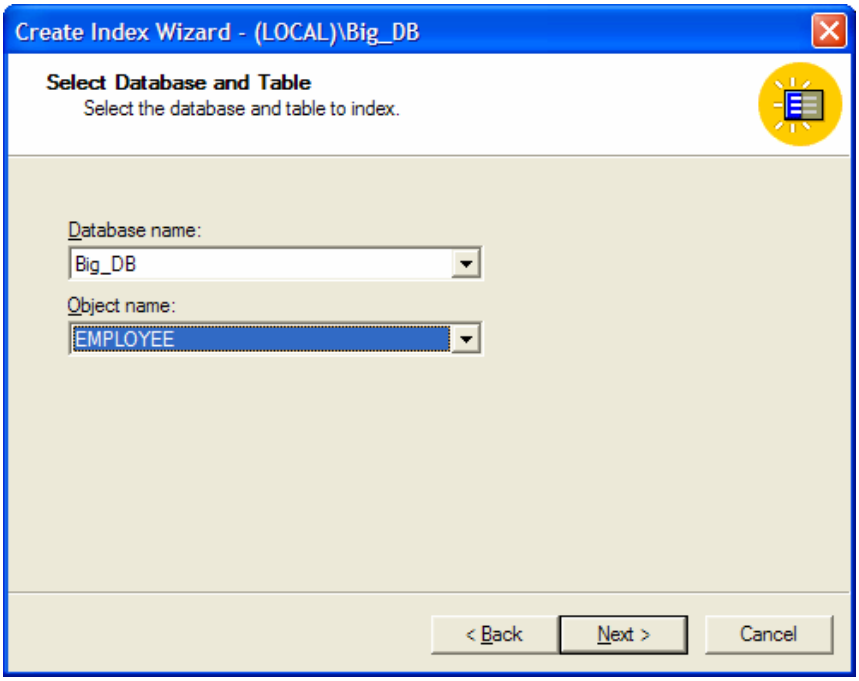



Select Wizard
Please select the Wizard you wish to use:
Register Server Wizard
Database
Create Database Wizard
Create Index Wizard
Create Login Wizard
Create Stored Procedure Wizard
Create View Wizard
Data Transformation Services
Management
Replication
OK
Cancel

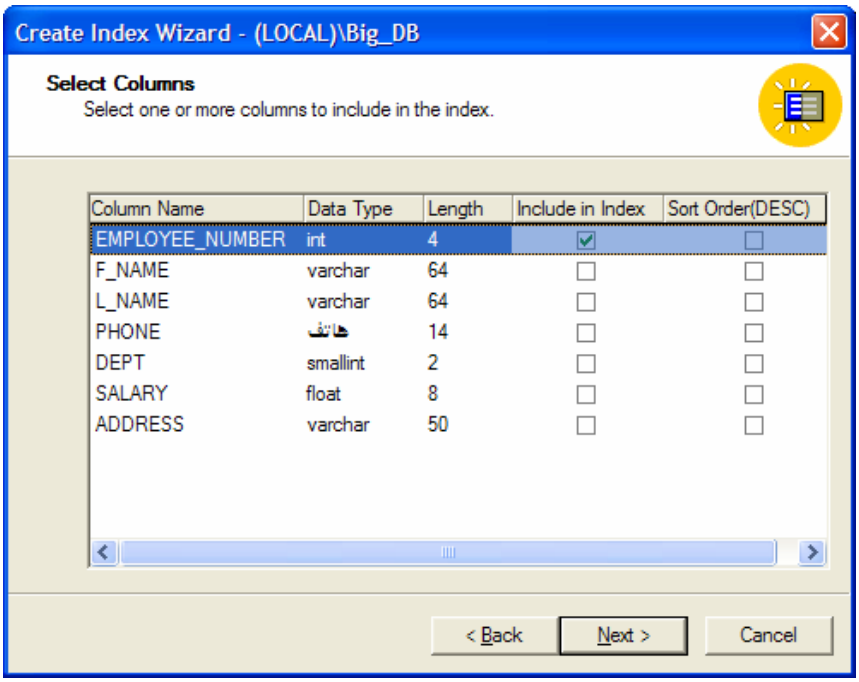


تظهر لنا بعد ذلك واجهة خاصة لتعبر لنا عند بداية عملية إنشاء الفهرس؛

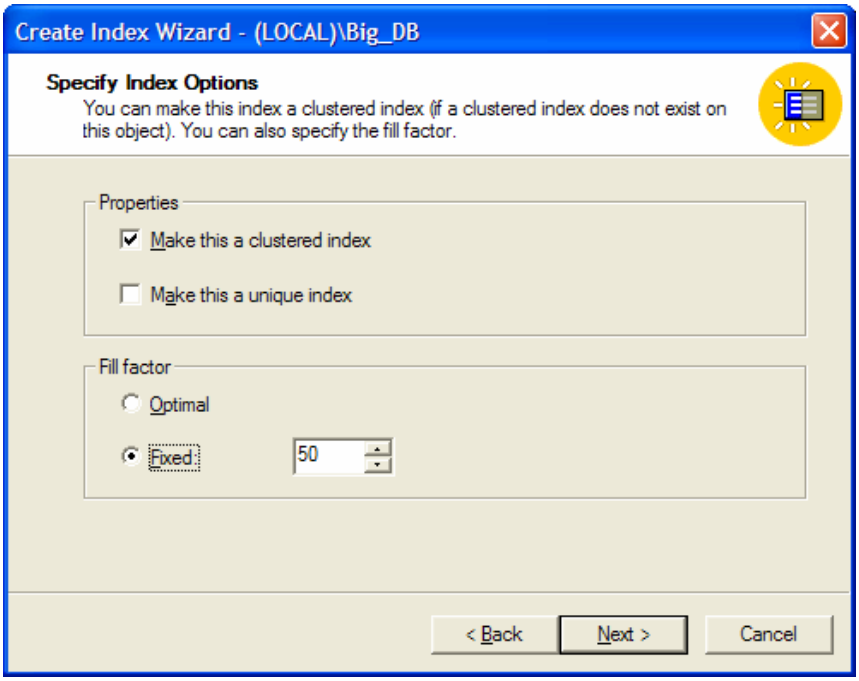
يمكننا من خلال التنقل بين الواجهات أن نعرّف الفهرس خطوة بخطوة، ابتداءً من اختيار قاعدة المعطيات، وانتهاءً بتحديد الأعمدة وخصائص الفهرس؛



<واجهة تحديد قاعدة المعطيات والجدول>



<واجهة اختيار الأعمدة>

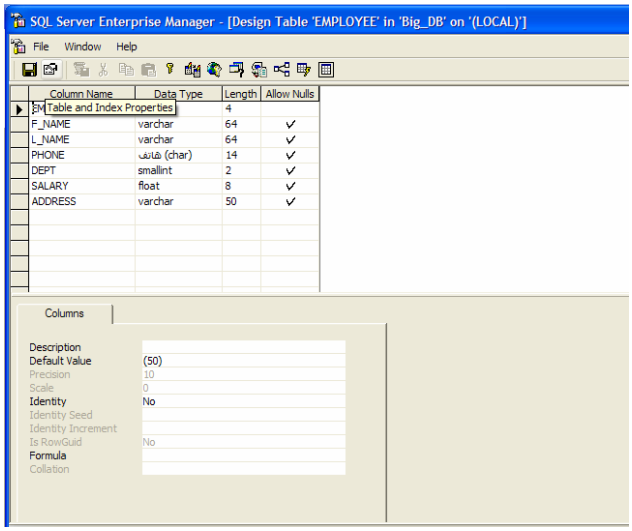


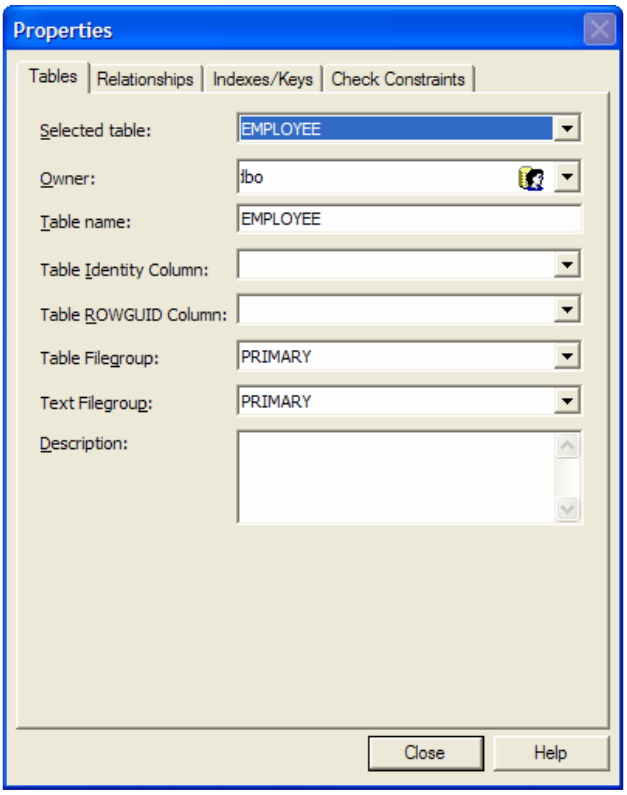
<واجهة خصائص الفهرس>

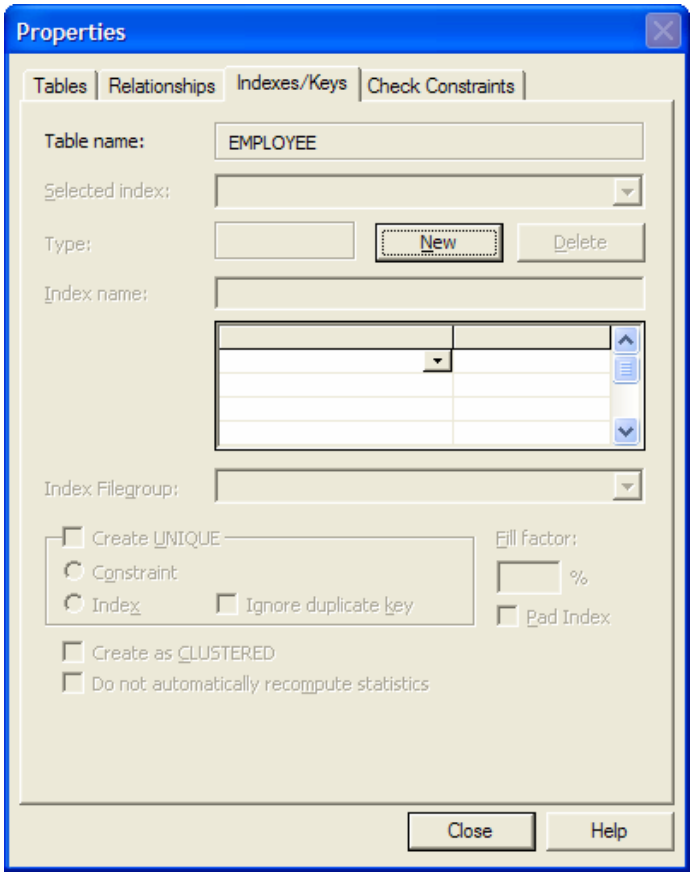
<واجهة تسمية الفهرس>

- من خلال خصائص الجدول نفسه في قاعدة المعطيات:

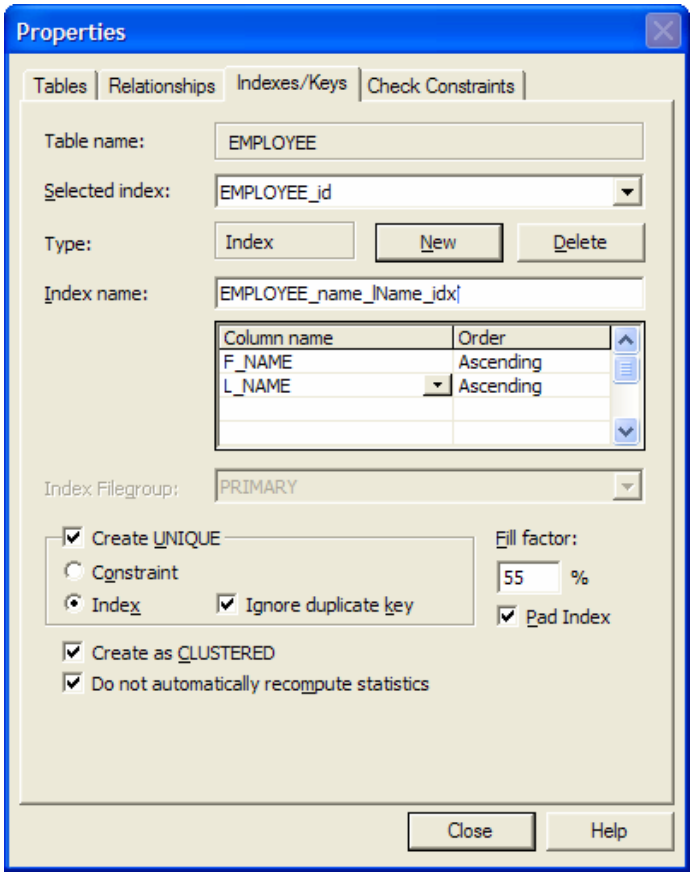
يمكننا إنشاء الفهرس أيضاً من خلال خصائص الجدول نفسه من ضمن Table Designer في الأداة Enterprise Manager؛




Properties
Tables
Relationships
Indexes/Keys
Check Constraints
Selected table:
EMPLOYEE
Owner:
ibo
Table name:
EMPLOYEE
Table Identity Column:
Table ROWGUID Column:
Table Filegroup:
PRIMARY
Text Filegroup:
PRIMARY
Description:
Close
Help



بحيث يمكننا إنشاء الفهرس وتحديد كافة خصائصة من خلال واجهة Indexes/Keys الفرعية، كما يوضح الشكل التالي:



إدارة الفهارس

يعمل SQL Server تلقائياً على إدارة كافة الفهارس المستخدمة، إذ يقوم تلقائياً بتعديل الفهارس بعد كل عملية إدخال أو تعديل أو حذف في المعطيات، كما يتم، وبشكل تلقائي، إنشاء كافة الإحصائيات التي يمكن استخدامها من قبل مُحسِّن الاستعلامات في SQL Server؛

يفضّل في بعض الأحيان أن يتم إجراء عمليات إدارة يدوية للفهارس، فعل سبيل المثال، ينبغي عندما نقوم بتحميل كمية كبية جداً من المعطيات على جدول معين، أن نحذف فهارس ذلك الجدول قبل عملية تحميل المعطيات، ثم نعيد بناء تلك الفهارس بعد الانتهاء من التحميل؛

يعمل SQL Server تلقائياً على إدارة كافة الفهارس المستخدمة، إذ يقوم تلقائياً بتعديل الفهارس بعد كل عملية إدخال أو تعديل أو حذف في المعطيات، كما يتم، وبشكل تلقائي، إنشاء كافة الإحصائيات التي يمكن استخدامها من قبل مُحسِّن الاستعلامات في SQL Server؛

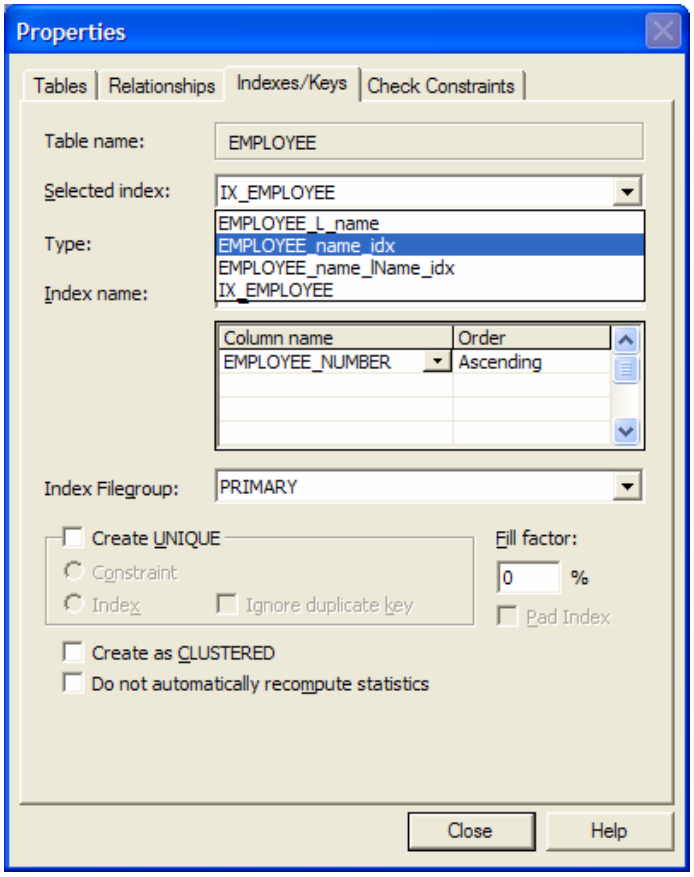
يفضّل في بعض الأحيان أن يتم إجراء عمليات إدارة يدوية للفهارس، فعل سبيل المثال، ينبغي عندما نقوم بتحميل كمية كبية جداً من المعطيات على جدول معين، أن نحذف فهارس ذلك الجدول قبل عملية تحميل المعطيات، ثم نعيد بناء تلك الفهارس بعد الانتهاء من التحميل؛

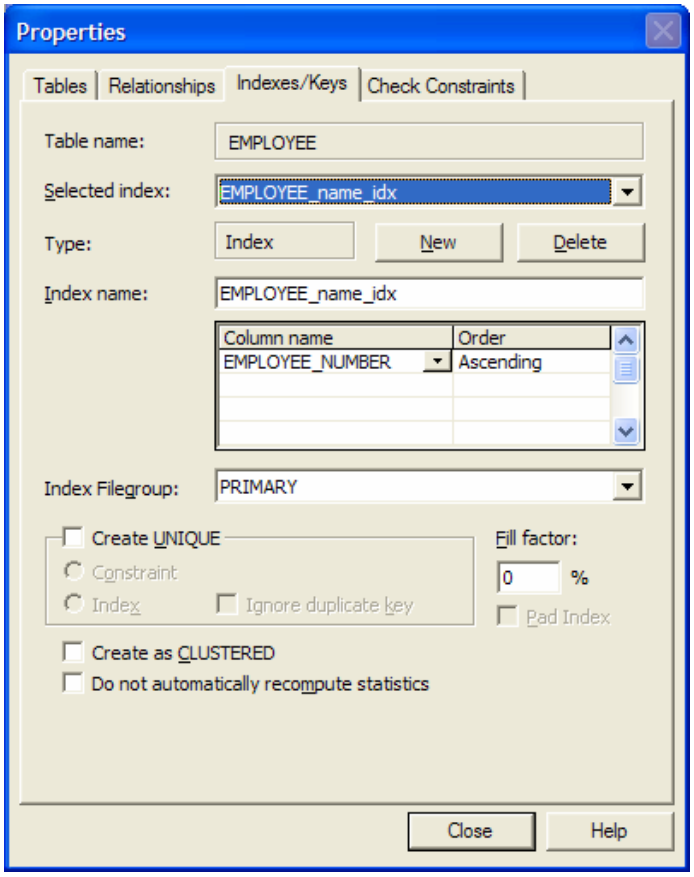
تعتبر العملية السابقة مفيدة من أجل تحسين الأداء، بحيث نتخلص من العبء المترتب على تعديل الفهارس عند كل عملية إضافة لمعطيات جديدة، وذلك من خلال إنشاء الفهرس دفعة واحدة بعد الانتهاء من إدخال كافة المعطيات إلى الجدول؛

يمكن –ومع مرور الزمن– أن تتجزأ الفهارس نفسها، هذا يحدث عندما يزداد حجم الفهرس ولا يتّسع للمزيد من المعطيات التي تستخدم للفهرسة، بحيث يؤدي ذلك إلى انقسام صفحة الفهرس إلى صفحتين. تؤدي تلك العملية إلى حدوث بعض المشاكل فيما يتعلق بالأداء المتوقع من الفهرس بحيث لا يتم –في معظم الأحيان– تخزين الصفحة الجديدة كتتمة للصفحة الأولى؛ سنناقش في الجلسات القادمة كيف نقوم بإدارة وتحسين أداء SQL Server وسنتطرق حينها إلى كيفية معالجة مسألة تجزُّء صفحات الفهرس.

حذف الفهارس

- يمكننا حذف الفهارس التي قمنا بإنشائها، من خلال الواجهة الفرعية Indexes/Keys في واجهة خصائص الجدول، وذلك من خلال اختيار الفهرس المناسب ثم حذفه:


Properties
Tables
Relationships
Indexes/Keys
Check Constraints
Table name:
EMPLOYEE
Selected index:
IX_EMPLOYEE
Type:
Index name:
EMPLOYEE_L_name
EMPLOYEE_name_idx
EMPLOYEE_name_lName_idx
IX_EMPLOYEE
Column name
Order
EMPLOYEE_NUMBER
Ascending
Index Filegroup:
PRIMARY
Create UNIQUE
Constraint
Index
Ignore duplicate key
Fill factor:
0
%
Pad Index
Create as CLUSTERED
Do not automatically recompute statistics
Close
Help



- يمكننا كذلك أن نحذف الفهارس باستخدام تعليمة Drop Index يليها اسم الفهرس أو أسماء الفهارس التي نرغب بحذفها مفصولة عن بعضها بفاصلة ",":

DROP INDEX *EMPLOYEE_name_idx,EMPLOYEE_L_name_idx,*

- مع العلم أنه لا يمكننا حذف الفهارس المنشأة على أساس المفتاح الأولي أو الفهارس الفريدة بنفس الأسلوب؛

ضمان تكامل المعطيات

- يعتبر مفهوم تكامل المعطيات أحد أهم الأسس التي ينبغي أن يأخذها مدير قاعدة المعطيات بعين الإعتبار، فهناك الكثير من المؤسسات التي تتخذ قرارات مصيرية اعتماداً على المعطيات المتوافرة لديها؛

- يعتبر إدخال المعطيات الخاطئ وبشكل غير مقصود، أحد أكبر المشاكل التي يمكن أن تعاني منها المؤسسات في معطياتها؛

- لا تعتبر الأخطاء السابقة من مسؤولية موظفي الإدخال، بحيث ينبغي على مدير قاعدة المعطيات الناجح أن يتجنب مثل هذا النوع من الأخطاء بنفسه، ويمكن أن يتم ذلك من خلال استخدام قيود التكامل التي يمنحها نظام إدارة قواعد المعطيات، والتي تعتبر في بعض الأحيان حلاً مناسباً.

- يعتبر مفهوم تكامل المعطيات أحد أهم الأسس التي ينبغي أن يأخذها مدير قاعدة المعطيات بعين الإعتبار، فهناك الكثير من المؤسسات التي تتخذ قرارات مصيرية اعتماداً على المعطيات المتوافرة لديها؛

- يعتبر إدخال المعطيات الخاطئ وبشكل غير مقصود، أحد أكبر المشاكل التي يمكن أن تعاني منها المؤسسات في معطياتها، فكثيرا ما يتم إدخال صفر عوضاً عن "O" أو بالعكس كما يمكن حدوث التباس ما بين حرف L وحرف I أو حتى الرقم 1 (l I1)، كما يمكن أن تحدث أخطاء أخرى من حيث التعبير عن الجنس على سبيل المثال، فالبعض يستخدم "ذكر" والبعض الآخر يستخدم "ذ" أو حتى يمكن استخدام "m" للدلالة على male، وغيرها من أخطاء إدخال المعطيات إلى النظام؛

- لا تعتبر الأخطاء السابقة من مسؤولية موظفي الإدخال، بحيث ينبغي على مدير قاعدة المعطيات الناجح أن يتجنب مثل هذا النوع من الأخطاء بنفسه، ويمكن أن يتم ذلك من خلال استخدام قيود التكامل التي يمنحها نظام إدارة قواعد المعطيات، والتي تعتبر في بعض الأحيان حلاً مناسباً؛

- سنركز من خلال الشرائح التالية على كيفية استخدام بعض القيود التي تضمن تكامل المعطيات، كالقواعد والقيم التلقائية؛

- يمكن –ومن وجهة نظر أخرى– أن نضمن تكامل المعطيات باستخدام عدة أساليب أخرى، أهمها وأكثرها شيوعاً، عملية التحكم بالدخل من **مستوى التطبيق**، أو حتى من خلال استخدام إجرائيات مخزّنة أو قوادح، إلا أننا سنركز الآن فقط على القيود التي يمكن استخدامها من قواعد أو قيم تلقائية.

أنماط تكامل المعطيات

- تعتمد كيفية تحقيق تكامل المعطيات على نمط تكامل المعطيات المستخدم، بحيث يمكن تصنيف أنماط تكامل المعطيات في ثلاثة أنواع:
  - تكامل المجال:

يعبر تكامل المجال عن مفهوم تحديد وتعيين القيم المتاحة لعمود ما، كما يشتمل على إمكانية السماح أو عدم السماح بالقيم الفارغة NULLs في ذلك العمود؛

يتم ضمان تكامل المعطيات اعتماداً على نمط تكامل المجال، من خلال استخدام أنماط المعطيات وتحديد استخدامية القيم الفارغة أو عدم استخدامها، بالإضافة إلى استخدام شروط الصحَّه أو ما يُعرف باسم قيود الاختبار؛

  - تكامل الكيان:

يتطلَّب مفهوم تكامل الكيان أن تكون كافة أسطر الجدول فريدة من نوعها، بحيث لا ينبغي تكرار أي سطر منها؛

يتم ضمان تكامل المعطيات اعتماداً على نمط تكامل الكيان، من خلال استخدام معرّف فريد، أو ما يُعرف باسم المفتاح الأولي؛

  - التكامل المرجعي:

يتطلَّب مفهوم التكامل المرجعي أن يتم ضمان القيم المستقلة ما بين الجداول المختلفة، بحيث يضمن أنّ حدوث أي تغير في معطيات جدول ما يؤدي بالضرورة إلى تغير تلك المعطيات في حال وجودها في جداول أخرى؛

يتم ضمان تكامل المعطيات اعتماداً على نمط التكامل المرجعي، من خلال استخدام ما يُعرف بعلاقة المفتاح الأولي/المفتاح الخارجي؛

طرائق تحقيق تكامل المعطيات

- تختلف طرائق تحقيق تكامل المعطيات إذ يمكن تصنيفها من خلال نوعين أساسيين، هما تكامل المعطيات التصريحي و تكامل المعطيات الإجرائي:
  - تكامل المعطيات التصريحي:

يعبّر تكامل المعطيات التصريحي عن استخدام القيود والقواعد والقيم التلقائية في سبيل ضمان تكامل المعطيات ضمن قاعدة المعطيات؛

تعتبر هذه الطريقة هي الأفضل لعدة أسباب تتعلق بسهولتها و بساطتها تقنياً، بالإضافة إلى سهولة إدارتها واتّساقها مع قاعدة المعطيات؛

  - تكامل المعطيات الإجرائي:

يعبّر تكامل المعطيات الإجرائي عن استخدام الإجرائيات المخزّنة والقوادح والرماز على مستوى التطبيق، في سبيل تحقيق ضمان تكامل المعطيات؛

تعتبر هذه الطريقة أكثر تعقيداً كما أنها يمكن أن تولّد عبئاً إضافياً مع العلم أنها تتيح إمكانيات أكبر ذات فعاليّة تتفوق عمّا يمكن إجراءه من خلال تكامل المعطيات التصريحي؛

القيود

- تعتبر القيود الطريقة الأولية المستخدمة لضمان تكامل قاعدة المعطيات، يمكن تصنيف القيود بعدّة أنواع، وهي:

لِيَ، المفتاح الخارجي، شرط التفرّد، شروط الاختبار، القيم التلقائية.

Comment [H1]: Primary Key, Foreign Key, Unique, Check, Default.

- مع العلم أنه يمكن اعتبار القيم التلقائية على أنها أغراض بحد ذاتها في قاعدة المعطيات، بالتالي يمكن تصنيفها كغرض أو كقيد.

- سنتناول في الشرائح التالية خصائص كل قيد من تلك القيود بالتفصيل.

القيود– المفتاح الأولي

- يستخدم المفتاح الأولي لضمان تحقيق تكامل الكيان، أو بأسلوب آخر، لضمان عدم تكرار الأسطر في جدول ما؛

- لا يسمح لأكثر من مفتاح أولي وحيد في كل جدول، كما يشترط في العمود –أو الأعمدة– المستخدمة للتعبير عن المفتاح الأولي ألاّ تسمح بالقيم الفارغة NULLs وألاّ تسمح بتكرار المعطيات فيها؛

- عندما نعرِّف مفتاحاً أولياً في جدول ما، يتم –وبشكل تلقائي– إنشاء فهرس عنقودي فريد على ذلك العمود أو الأعمدة؛

- يفضّل عندما يتم اختيار عمود ما ليعبّر عن المفتاح الأولي، أن يكون ذلك الخَيار أقصر ما يمكن، أو بأسلوب آخر، لا يستحسن أن يتم استخدام أكثر من عمود للتعبير عن المفتاح الأولي–مع العلم أن تلك العملية متاحة–؛

- يستخدم المفتاح الأولي لضمان تحقيق تكامل الكيان، أو بأسلوب آخر، لضمان عدم تكرار الأسطر في جدول ما؛

- لا يسمح لأكثر من مفتاح أولي وحيد في كل جدول، كما يشترط في العمود –أو الأعمدة– المستخدمة للتعبير عن المفتاح الأولي ألاّ تسمح بالقيم الفارغة NULLs وألاّ تسمح بتكرار المعطيات فيها؛

- عندما نعرِّف مفتاحاً أولياً في جدول ما، يتم –وبشكل تلقائي– إنشاء فهرس عنقودي فريد على ذلك العمود أو الأعمدة؛

- يفضّل عندما يتم اختيار عمود ما ليعبّر عن المفتاح الأولي، أن يكون ذلك الخَيار أقصر ما يمكن، أو بأسلوب آخر، لا يستحسن أن يتم استخدام أكثر من عمود للتعبير عن المفتاح الأولي–مع العلم أن تلك العملية متاحة–؛

- ينبغي الانتباه أثناء تعريف المفتاح الأولي من إمكانية حدوث تكرار مستقبلي في قيم المعطيات التي تحتويها الحقول المكوِّنه لذلك المفتاح، فعلى سبيل المثال، لا يفضل استخدام أعمدة "الاسم الأول" و"اسم الأب" و "الكنية" في جدول الموظفين مثلاً، للتعبير عن المفتاح الأولي، إذ أنه لا يمكن أن نضمن ألا يتم إدخال موظف جديد له نفس الاسم واسم الأب والكنية، وهذا وارد الحدوث على الرغم من ندرته، بالتالي يعتبر استخدام حقل رقم الموظف –للدلالة على المفتاح الأولي– مرشحاً أفضل، خاصةً إذا ما قمنا باستخدام خاصة التزايد المستمر من أجل توليد الأرقام الفريدة لذلك العمود؛

- فيما يلي عرض للمخطوط الذي يعبر عن إنشاء جدول بمفتاح أولي:

```
CREATE TABLE   EMPLOYEE (
        EMPLOYEE_NUMBER int IDENTITY (100, 10) CONSTRAINT EMP_PK
        PRIMARY KEY NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE char (14) NULL,
        DEPT   smallint,
        SALARY float NULL,
        PHOTO image NULL
)
GO
```

القيود– شرط التفرّد

- يُستخدم شرط التفرد لنفس الأغراض التي يستخدم المفتاح الأولي من أجلها، أي لضمان تحقيق تكامل الكيان من خلال عدم تكرار الأسطر في جدول ما؛

- يَستخدم شرط التفرد فهرس فريد على العمود أو الأعمدة التي يرتبط بها، وذلك لضمان عدم تكرار المعطيات؛

- يختلف شرط التفرد عن المفتاح الأولي في أنه يسمح بوجود القيم الفارغة NULLs، آخذين بعين الاعتبار أنه يسمح بورود وحيد للقيمة الفارغة في كافة الأسطر بحيث يعتبر تكرار القيمة NULL بمثابة تكرار أية قيمة أخرى؛

- يتم استخدام شرط التفرُّد عندما نرغب بضمان عدم تكرار المعطيات في أحد أعمدة الجدول التي لا تمثّل مفتاحاً أولياً، وذلك على اعتبار أنّ المفتاح الأولي فريد بالضرورة.

- يُستخدم شرط التفرد لنفس الأغراض التي يستخدم المفتاح الأولي من أجلها، أي لضمان تحقيق تكامل الكيان من خلال عدم تكرار الأسطر في جدول ما؛
- يَستخدم شرط التفرد فهرس فريد على العمود أو الأعمدة التي يرتبط بها، وذلك لضمان عدم تكرار المعطيات؛
- يختلف شرط التفرد عن المفتاح الأولي في أنه يسمح بوجود القيم الفارغة NULLs، آخذين بعين الاعتبار أنه يسمح بورود وحيد للقيمة الفارغة في كافة الأسطر بحيث يعتبر تكرار القيمة NULL بمثابة تكرار أية قيمة أخرى؛
- يتم استخدام شرط التفرُّد عندما نرغب بضمان عدم تكرار المعطيات في أحد أعمدة الجدول التي لا تمثّل مفتاحاً أولياً، وذلك على اعتبار أنّ المفتاح الأولي فريد بالضرورة. مثال على ذلك، يمكن أن نربط عمود "رقم الضمان الإجتماعي" في جدول الموظفين بشرط التفرُّد بحيث نضمن عدم تكرار المعطيات في هذا العمود؛

- فيما يلي عرض للمخطوط الذي يعبر عن إنشاء جدول بمفتاح أولي على أحد الأعمدة وشرط تفرُّد على عمود آخر:

```
CREATE TABLE   EMPLOYEE (
        EMPLOYEE_NUMBER int IDENTITY (100, 10) CONSTRAINT EMP_PK
        PRIMARY KEY NOT NULL,
        F_NAME varchar (64) NULL,
        L_NAME varchar (64) NULL,
        PHONE char (14) NULL,
        SOCIAL_NUMBER int CONSTRAINT EMP_SOCIAL_NUMBER_UNQ
        UNIQUE  NOT NULL,
        DEPT   smallint,
        SALARY float NULL,
        PHOTO image NULL
)
```

القيود- المفتاح الخارجي

- يُستخدم المفتاح الخارجي لضمان تحقيق التكامل المرجعي من خلال ضمان أنّ حدوث أي تغير في معطيات جدول ما يؤدي بالضرورة إلى تغير تلك المعطيات في حال وجودها في جداول أخرى؛

- تعبر علاقة المفتاح الأولي/المفتاح الخارجي عن علاقة تبعية لعمود في جدول ما لعمود في جدول آخر أو في الجدول نفسه، بحيث ينبغي أن يرتبط العمود الذي يعبّر عن المفتاح الخارجي بعمود يعبّر حصراً عن مفتاح أولي؛

- مثال؛

- خاصة التكامل المرجعي الشلالي:

تعتبر هذه الخاصة جديدة على SQL Server 2000 إذ أنها لم تتوافر في النسخ السابقة، تسمح هذه الخاصة بتعقب التغيرات التي تحدث على الجدول الأب ونقل تداعياتها إلى الجدول الابن.

- يُستخدم المفتاح الخارجي لضمان تحقيق التكامل المرجعي من خلال ضمان أنّ حدوث أي تغير في معطيات جدول ما يؤدي بالضرورة إلى تغير تلك المعطيات في حال وجودها في جداول أخرى؛

- تعبر علاقة المفتاح الأولي/المفتاح الخارجي عن علاقة تبعية لعمود في جدول ما لعمود في جدول آخر أو في الجدول نفسه، بحيث ينبغي أن يرتبط العمود الذي يعبّر عن المفتاح الخارجي بعمود يعبّر حصراً عن مفتاح أولي؛

- لنستخدم العمود DEPT في جدول الموظفين للتعبير عن رقم القسم الذي ينتمي إليه الموظف، بحيث يكون هذا العمود عبارة عن مفتاح خارجي مرتبط بالمفتاح الأولي لجدول الأقسام Department؛

إن أي محاولة لإضافة أو تعديل أي سطر في جدول الموظفين ستؤدي بالضرورة إلى إجراء اختبار لتأكيد أن قيمة العمود Dept الجديدة موجودة في جدول الأقسام، مع العلم أن عملية الاختبار هذه تتم من الطرفين، بحيث يمنع SQL Server إجراء أية محاولة حذف أو تعديل للأسطر التي تحتوي على المفتاح الرئيسي والتي يمكن أن تؤدي بدورها إلى حدوث خلل في تكامل المعطيات من جهة جدول المفتاح الخارجي؛

- فيما يلي عرض للمخطوط الذي يعبر عن كيفية إضافة المفتاح الخارجي إلى جدول الموظفين:

```
ALTER TABLE  EMPLOYEE
ADD CONSTRAINT employee_dept_fk
 FOREIGN KEY (DEPT)
REFERENCES DEPT(DEPARTMENT_NUMBER)
```

- التكامل المرجعي الشلالي:

– تعتبر هذه الخاصة جديدة على SQL Server 2000 إذ أنها لم تتوافر في النسخ السابقة، تسمح هذه الخاصة بتعقب التغيرات التي تحدث على الجدول الأب ونقل تداعياتها إلى الجدول الابن؛

– يتم إضافة عبارتين جديدتين إلى تعليمات إنشاء او تعديل الجدول هما ON DELETE و ON UPDATE وذلك لتسهيل استخدام الخاصة تلك، بحيث يمكن إطلاقها بحالتين، إما عند حذف أسطر من الجدول الأب أو عند تعديل أسطر ذلك الجدول؛

– يتم تفعيل خاصة التكامل المرجعي الشلالي بإضافة التعليمة CASCADE بعد كل من العبارتين السابقتين أثناء إنشاء أو تعديل جدول ما؛

– لكي نفهم الغرض من هذه الخاصة لابد لنا من استعراض مثال عليها، لنفترض حالة مثال جدولي القسم والموظف، وفيه توجد علاقة مفتاح أولي/مفتاح خارجي من جدول القسم إلى جدول الموظف، ويقصد بذلك أن كل قسم يمكن أن يكون فيه موظف أو عدّة موظفين. لنفترض الآن أن القسم 20 فيه 500 موظف؛

إن تعريف خاصة التكامل المرجعي الشلالي بعد تعليمة ON UPDATE على جدول الموظف تؤدي إلى تطبيق كافة التعديلات – في حال حدوثها في الجدول الأب أي جدول "القسم"– على كافة الأسطر المرتبطة بها في جدول الموظف، أي، لو قمنا بتغيير رقم القسم من 20 إلى 25، بالتالي سيتم تغيير كافة الأسطر الـــ 500 في جدول الموظف لتصبح قيمة العمود dept –الذي يمثل المفتاح الخارجي– 25 بدلاً من 20؛

إن تعريف خاصة التكامل المرجعي الشلالي بعد تعليمة ON DELETE على جدول الموظف تؤدي بدورها إلى تطبيق كافة التعديلات –في حال حدوثها في الجدول الأب– على كافة الأسطر المرتبطة بها في جدول الموظف، أي، لو قمنا بحذف القسم ذو الرقم 20 من جدول القسم، بالتالي سيتم حذف كافة الأسطر الـــ 500 في جدول الموظف؛

- كما نلاحظ من المثال السابق، تعتبر خاصة التكامل المرجعي الشلالي غاية في الفاعلية بالإضافة إلى أنها غاية في الخطورة، ولكن، وعلى الرغم من تلك الخطورة المتوقّعة، ينصح باستخدامها عند الحاجة إليها؛

- فيما يلي عرض للمخطوط الذي يعبر عن كيفية إضافة خاصة التكامل المرجعي الشلالي إلى جدول الموظفين:

```
ALTER  TABLE  EMPLOYEE
ADD CONSTRAINT employee_dept_fk
 FOREIGN KEY (DEPT)
REFERENCES DEPT(DEPARTMENT_NUMBER)

ON DELETE CASCADE
ON UPDATE CASCADE
```

القيود– قيد الاختبار

- يُستخدم قيد الاختبار من أجل حصر القيم التي يمكن إدخالها إلى عمود ما في جدول معين؛

- يُعبَّر عن قيد الاختبار على أنه عبارة بوليانية لا ينبغي أن تساوي القيمة false، وذلك لكي يتم تنفيذ عملية الإضافة أو التعديل على معطيات عمود ما؛

- يمكن أن تستخدم قيود الاختبار لضمان أن تأخذ المعطيات شكلاً معيناً، كما في حالة رقم الهاتف مثلاً، فمن خلال التعبير التالي يمكننا أن نضمن أن القيمة التي سيتم إدخالها في العمود لا بد أن تأخذ شكل ثلاثة أرقام ثم '-' ثم ستة أرقام، كما يلي:
[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9]

- مثال.

- يُستخدم قيد الاختبار من أجل حصر القيم التي يمكن إدخالها إلى عمود ما في جدول معين؛

- يُعبَّر عن قيد الاختبار على أنه عبارة بوليانية لا ينبغي أن تساوي القيمة false، وذلك لكي يتم تنفيذ عملية الإضافة أو التعديل على معطيات عمود ما؛

- يمكن أن تستخدم قيود الاختبار لضمان أن تأخذ المعطيات شكلاً معيناً، كما في حالة رقم الهاتف مثلاً، فمن خلال التعبير التالي يمكننا أن نضمن أن القيمة التي سيتم إدخالها في العمود لا بد أن تأخذ شكل ثلاثة أرقام ثم '-' ثم ستة أرقام، كما يلي:

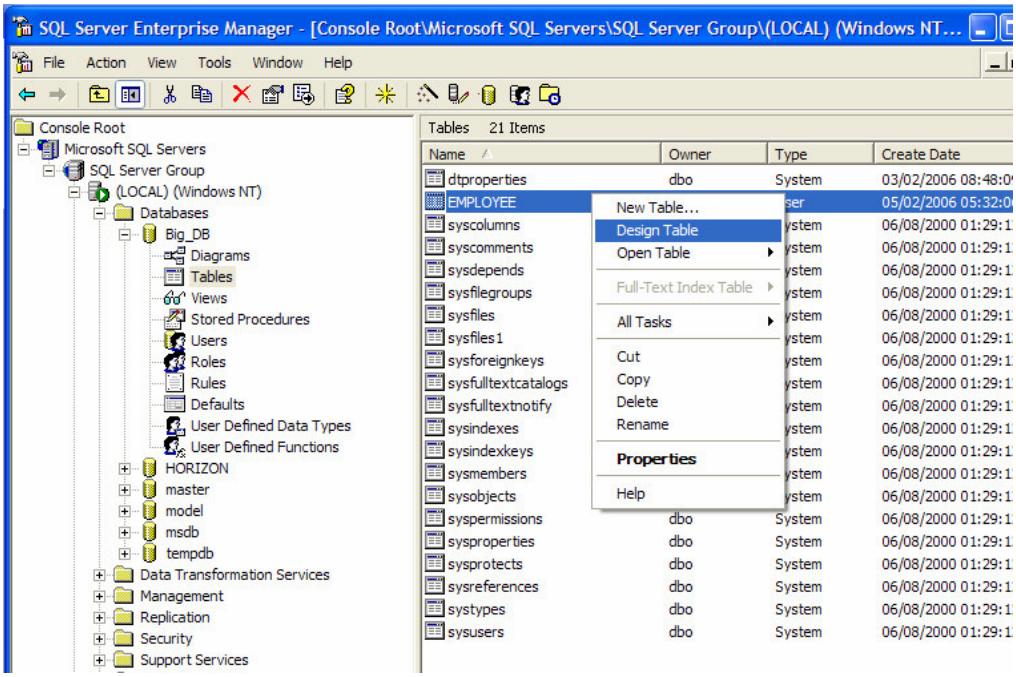
[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9]

- فيما يلي عرض للمخطوط الذي يعبر عن كيفية إضافة قيد اختبار إلى جدول معين:

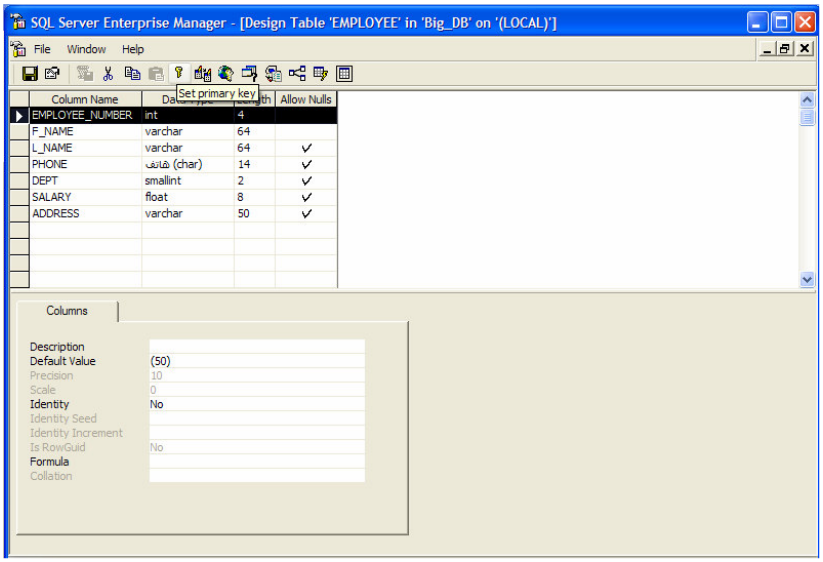
```
CREATE  TABLE   inventory
(
        item_code char(4) NOT NULL
        CONSTRAINT inventory_item_code_CHK
        CHECK (item_code LIKE '[0-9] [0-9] [0-9] [0-9]'),

        high_volume  int NOT NULL
        CONSTRAINT inventory_high_volume_CHK
        CHECK (high_volume > 0),

        low_volume  int NOT NULL
        CONSTRAINT inventory_low_volume_CHK
        CHECK (low_volume > 0),

        CONSTRAINT inventory_high_low_CHK
        CHECK (high_volume >= low_volume and high_volume - low_volume <
        1000),
)
```

إنشاء القيود باستخدام الأداة Enterprise Manager

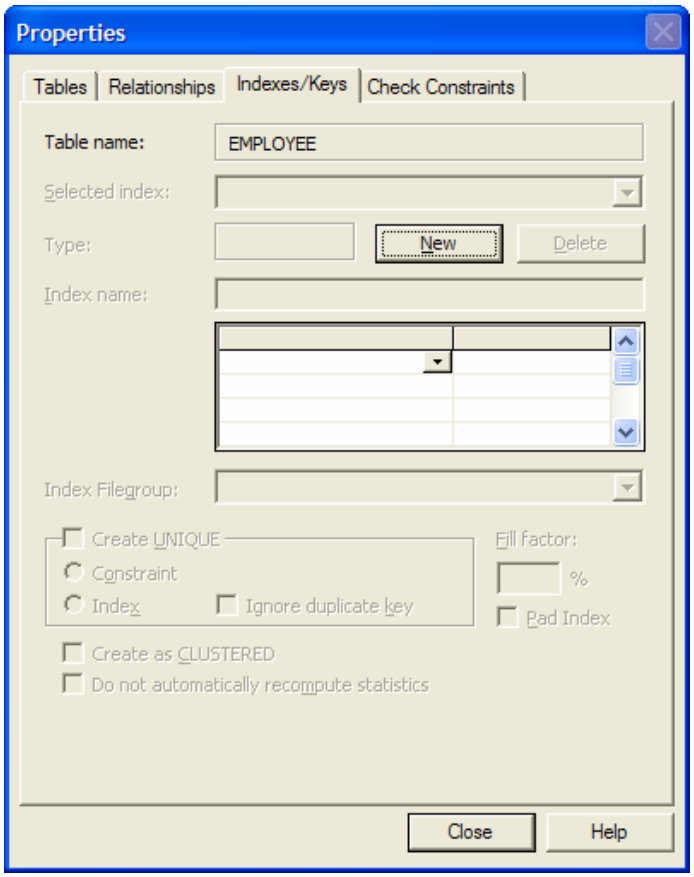
- لا تختلف عملية إضافة القيود كثيراً عن العمليات الأخرى التي يمكن إنشاؤها من خلال الأداة Enterprise Manager، فهي تتميز بالسهولة والبديهية؛
- يمكننا أن نقوم بإدارة القيود المفروضة على جدول ما من خلال الدخول إلى الواجهات الخاصة بتصميم الجدول في الأداة Enterprise Manager؛



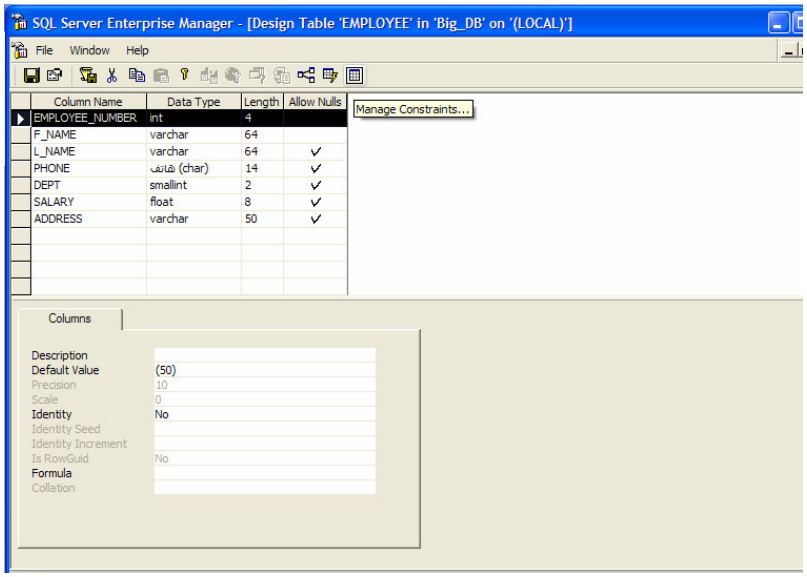
- يمكننا إضافة مفتاح أولي بشكل مباشر من خلال اختيار العمود أو الأعمدة التي نرغب بأن تكون المفتاح الأولي، ثم نضغط على أيقونة المفتاح "Set Primary Key"؛

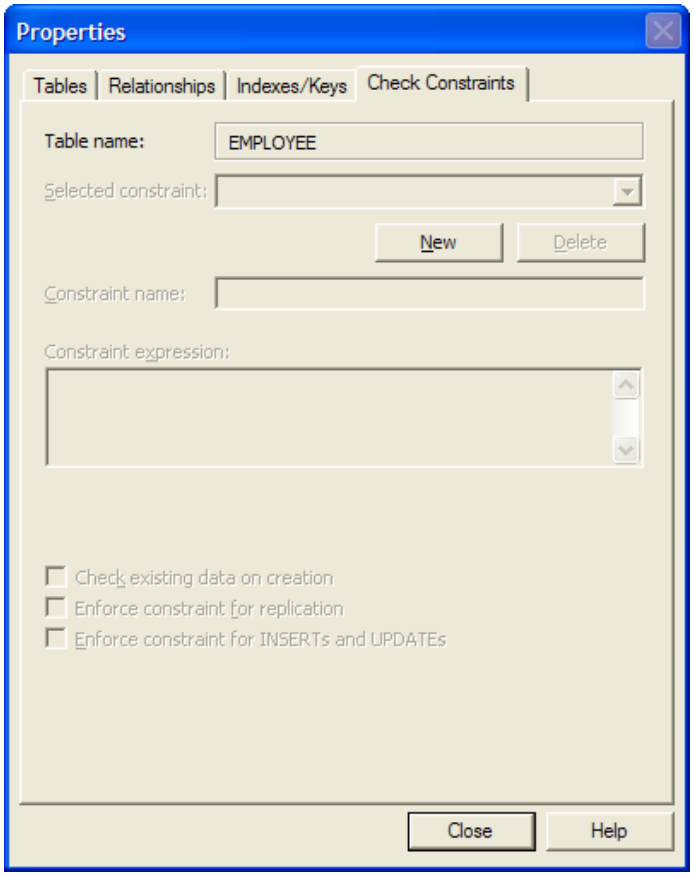


- يمكننا أيضاً إضافة المفتاح الأولي من خلال واجهة "Indexes/keys" الفرعية في واجهة خصائص الجدول:



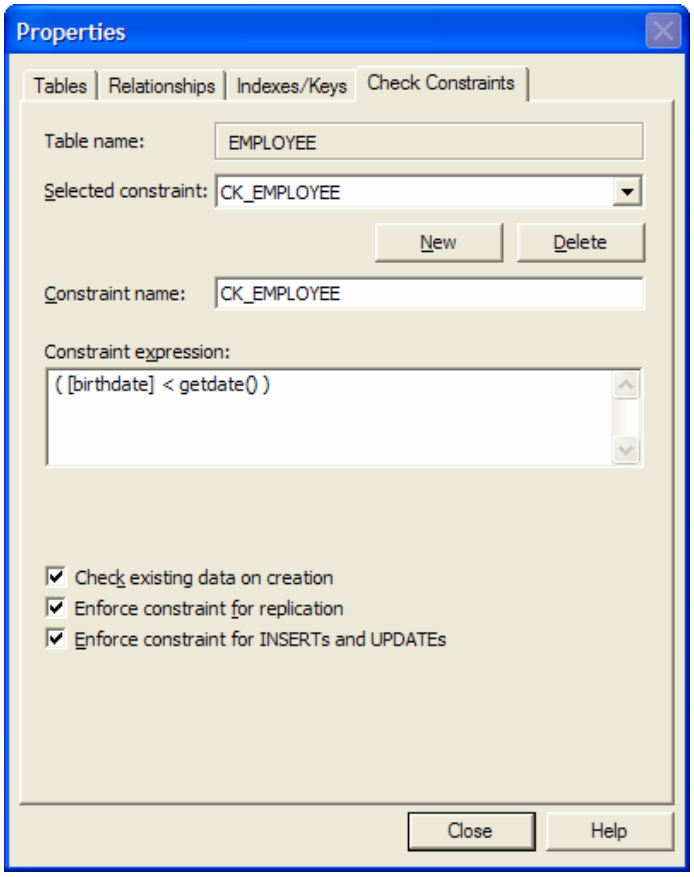
- يمكننا من زر إدارة القيود "Manage Constraints" أن ننتقل إلى الواجهة التي تتحكم بالقيود، إضافة قيود جديدة أو حذف قيود مفروضة مسبقاً؛


SQL Server Enterprise Manager - [Design Table 'EMPLOYEE' in 'Big_DB' on '(LOCAL)']
File
Window
Help
Column Name
Data Type
Length
Allow Nulls
EMPLOYEE_NUMBER
int
4
F_NAME
varchar
64
L_NAME
varchar
64
PHONE
هاتف (char)
14
DEPT
smallint
2
SALARY
float
8
ADDRESS
varchar
50
Manage Constraints...
Columns
Description
Default Value
(50)
Precision
10
Scale
0
Identity
No
Identity Seed
Identity Increment
Is RowGuid
No
Formula
Collation

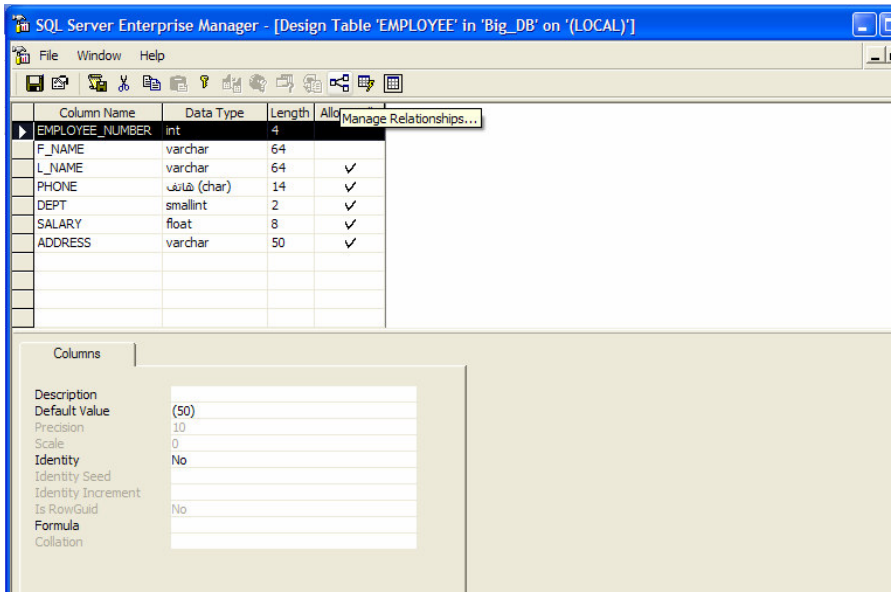


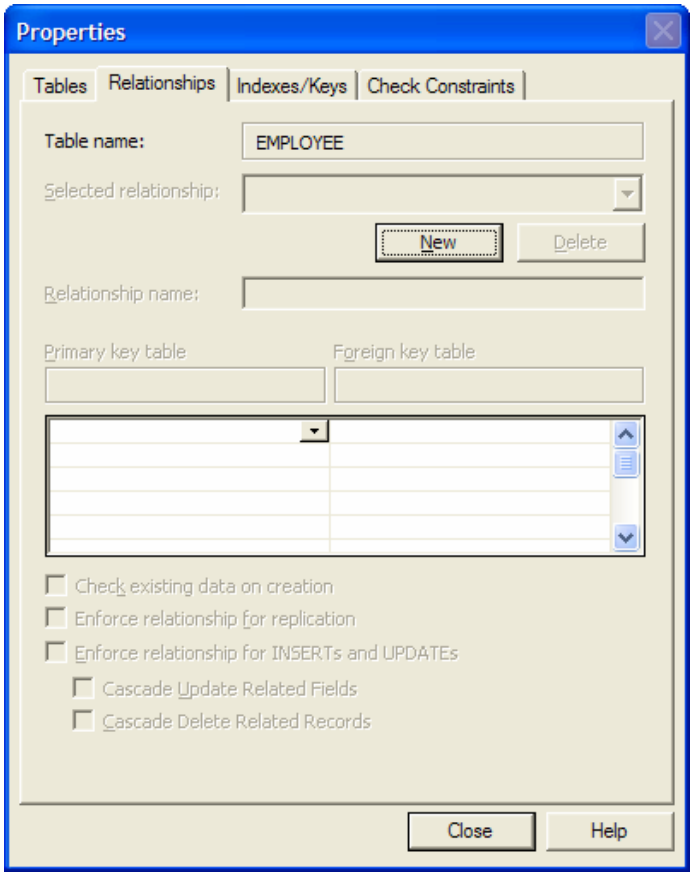
يمكننا كذلك الوصول إلى هذه الواجهة من خلال زر خصائص الجدول؛

نستطيع في هذه الواجهة أن نقوم بتعريف قيود جديدة على أعمدة الجدول الحالي، كما في المثال التالي، الذي يختبر أن يكون تاريخ الميلاد أصغر من تاريخ اليوم:

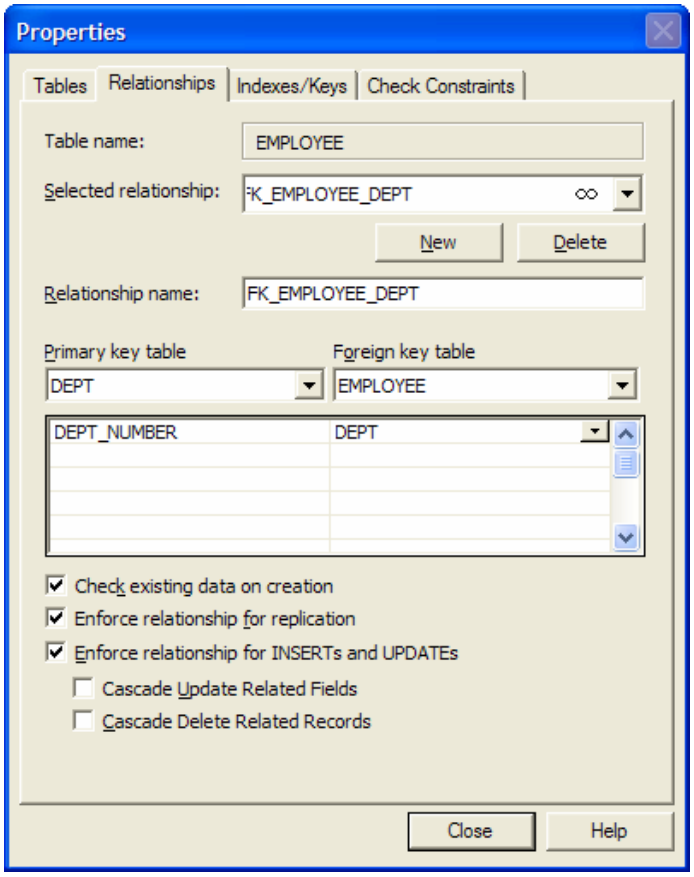


- يمكننا من خلال زر "Manage Relationships" أن نستعرض الواجهة التي يمكننا من خلالها تعريف المفاتيح الخارجية، مع العلم أنه يمكننا كذلك الوصول إلى هذه الواجهة من خلال زر خصائص الجدول:


SQL Server Enterprise Manager - [Design Table 'EMPLOYEE' in 'Big_DB' on '(LOCAL)']
File
Window
Help
Column Name
Data Type
Length
Allo
Manage Relationships...
EMPLOYEE_NUMBER
int
4
F_NAME
varchar
64
L_NAME
varchar
64
PHONE
هاتف (char)
14
DEPT
smallint
2
SALARY
float
8
ADDRESS
varchar
50
Columns
Description
Default Value
(50)
Precision
10
Scale
0
Identity
No
Identity Seed
Identity Increment
Is RowGuid
No
Formula
Collation



يمكننا من خلال هذه الواجهة أن نحدد العمود المصدر والعمود الهدف وذلك بعد تحديد جدول المفتاح الأولي وجدول المفتاح الخارجي لعلاقة مفتاح أولي/ مفتاح خارجي، كما يمكننا أيضاً أن نحدد فيما إذا كنا نرغب بتفعيل خاصة التكامل المرجعي الشلالي أم لا.



القواعد

- تستخدم القواعد لتحديد مجال القيم التي يمكن أن يتم تخزينها في عمود ما، وهي تشبه من حيث المبدأ مفهوم قيود الاختبـــار ولكنها تختلف عنها من حيث محدودية الوظيفة؛
- تعبر القاعدة عن غرض من أغراض قاعدة المعطيات، بحيث يتم أولاً إنشاؤها ثم يتم ربطها مع عمـــود أو نمـــط معطيـــات مستخدم معين؛
- مثال:

تعبر القاعدة التالية عن شكل معياري لأرقام الهاتف، وفيما يلي عرض للمخطوط الذي يعبر عن تلك القاعدة:

```
CREATE RULE phon_rule AS
@phone LIKE '(++[0-9][0-9][0-9]) [0-9][0-9][0-9] - [0-9][0-9][0-9] [0-9][0-9][0-9]'
```

بربط القاعدة السابقة إلى الأعمدة التي تعبر عن الهاتف في قاعدة المعطيات، يمكننا أن نضمن شكل ثابت ومتّسق لأرقام الهواتف المخزنة في قاعدة المعطيات، مع العلم أنه يمكننا أن نربط هذه القاعدة مع نمط معطيات كنّا قد قمنا بتعريفه مسبقاً، وبالتالي تصبح القاعدة الجديدة جزءاً من ذلك النمط؛

إنشاء وإدارة القواعد باستخدام تعليمات T-SQL

- يمكننا إنشاء القواعد من خلال التعليمة CREATE RULE كما مرّ معنا، يمكن أن تتضمن هذه التعليمة أي شيء يمكن أن يكتب بشكل صحيح بعد عبارة WHERE؛

- تتضمن عملية إدارة القواعد، إمكانية ربط تلك القواعد مع الأعمدة أو أنماط المعطيات من جهة، بالإضافة إلى إمكانية فك ذلك الارتباط من جهة أخرى؛

- تستخدم إجرائية النظام sp_bindrule لإجراء عمليات ربط القواعد مع الأعمدة أو الأنماط، كما تستخدم إجرائية النظام sp_unbindrule لفك ذلك الارتباط؛

- مثال:

ينبغي أولاً إنشاء القاعدة:

```
CREATE RULE color_rule AS
@color IN ('red', 'blue', 'green', 'black')
```

بعد الانتهاء من عملية إنشاء القاعدة يمكننا ربطها مع الأعمدة أو الأنماط:

الربط مع عمود:

```
sp_bindrule color_rule, 'myTable.colorColumn'
```

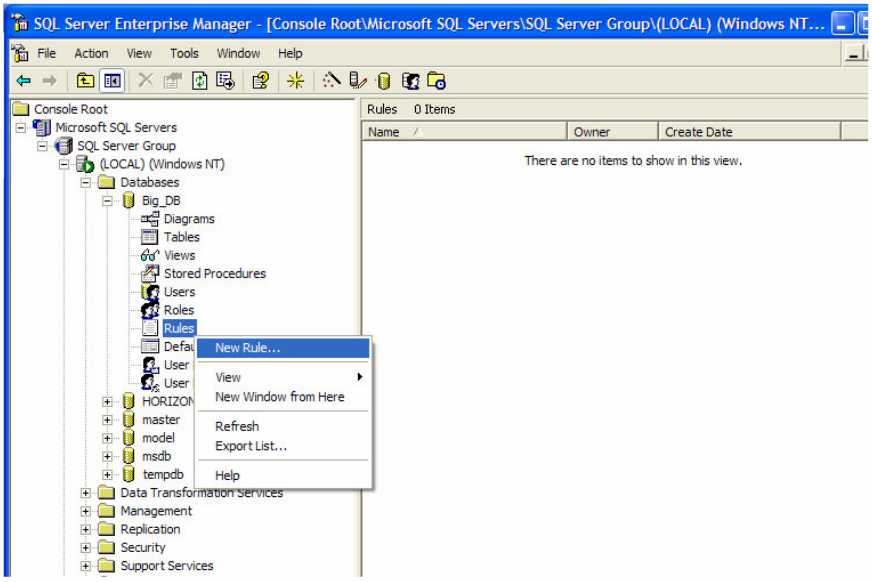
الربط مع نمط معطيات معرّف:

```
sp_bindrule color_rule, color
```

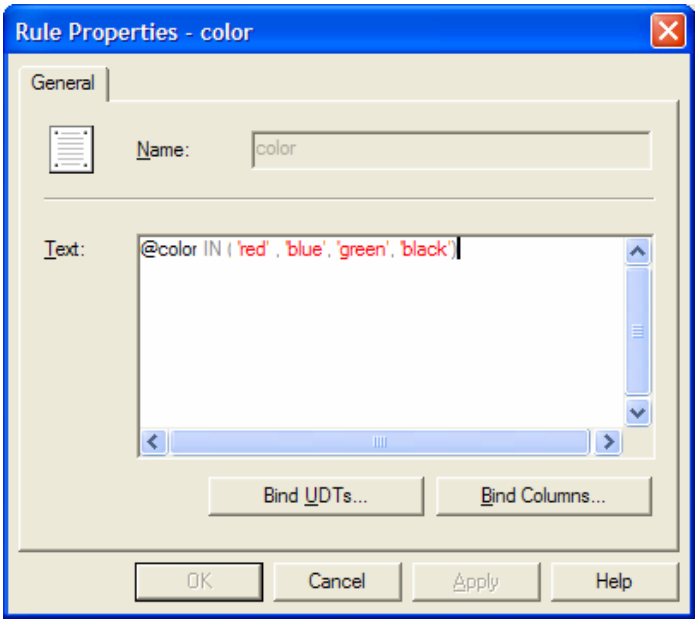
فك الارتباط:

```
sp_unbindrule color
```

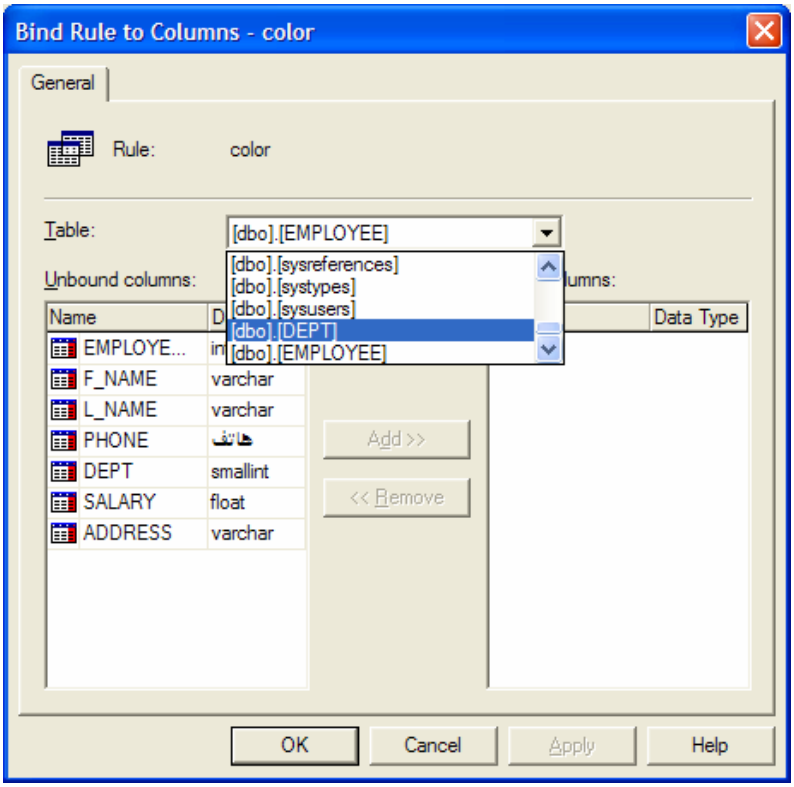
- يمكننا إنشاء القواعد من خلال التعليمة الأداة Enterprise Manager من خلال اختيار "New Rule" من قائمة المهمـــات السريعة للمجلد Rules في قاعدة المعطيات المناسبة؛



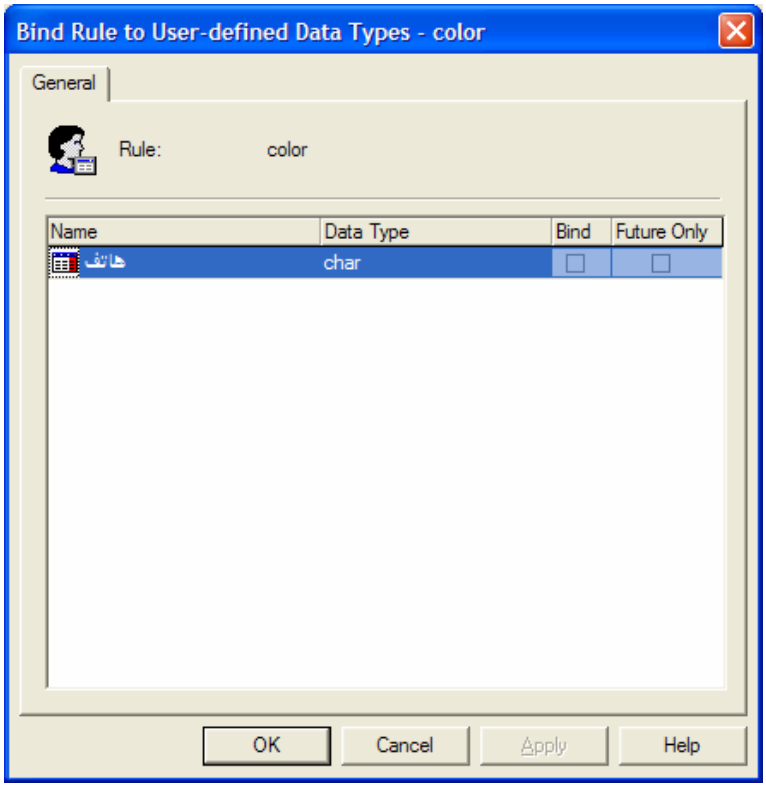
- ينبغي أن نقوم بتعبئة القيم في الأماكن المخصصة لها بحيث تطلب منّا الواجهة الخاصة بتعريف القاعدة الجديدة أن نحدد كلاً من اسم القاعدة وشروطها:



- يمكننا بعد ذلك أن نقوم بإجراء عمليات ربط القاعدة الجديدة مع الأعمدة أو الأنماط، وذلك من خلال الأزرار المخصصة لكل منها، بحيث تظهر واجهة خاصة تعرض كافة الجداول وأعمدتها ليتم الربط مع العمود المناسب؛



- كما تظهر واجهة خاصة تعرض كافة أنماط معطيات المستخدم المعرّفة ليتم الربط مع أحدها:



القيم التلقائية Defaults

- تستخدم القيم التلقائية لتحديد قيمة ليتم تخزينها في عمود ما بشكل تلقائي، وهي يمكن أن تكون شيء يمكن أن يؤدي إلى قيمة ثابته، كرقم ما أو تابع معين أو علاقة رياضية؛
- هناك نوعان من القيم التلقائية، القيم التلقائية المصرّحة والقيم التلقائية التي يمكن أن يتم ربطها بالأغراض؛
- تشبه القيم التلقائية المصرّحة إلى حد بعيد القيود، إذ يمكن ربطها بتعليمات CREATE أو ALTER TABLE؛
- سنعرض فيما يلي للمخطوط المعبر عن إنشاء قيمة تلقائية للعمود phone في جدول الموظف:

ALTER TABLE *employee* ADD
CONSTRAINT *emp_phone_default* DEFALT 'غير موجود' FOR phone

- فيما يلي المخطوط المستخدم لحذف القيمة التلقائية السابقة:

ALTER TABLE *employee* DROP CONSTRAINT *emp_phone_default*

- يتم بناء النوع الآخر من القيم التلقائية، أي القيم التلقائية التي يمكن أن يتم ربطها بالأغراض، بأسلوب أقــرب إلـــى مفهـــوم القواعد؛
  - يتم أولاً إنشاء القيمة التلقائية كغرض في قاعدة المعطيات؛
  - يتم لاحقاً ربطها بعمود أو بنمط مستخدم مسبق التعريف؛

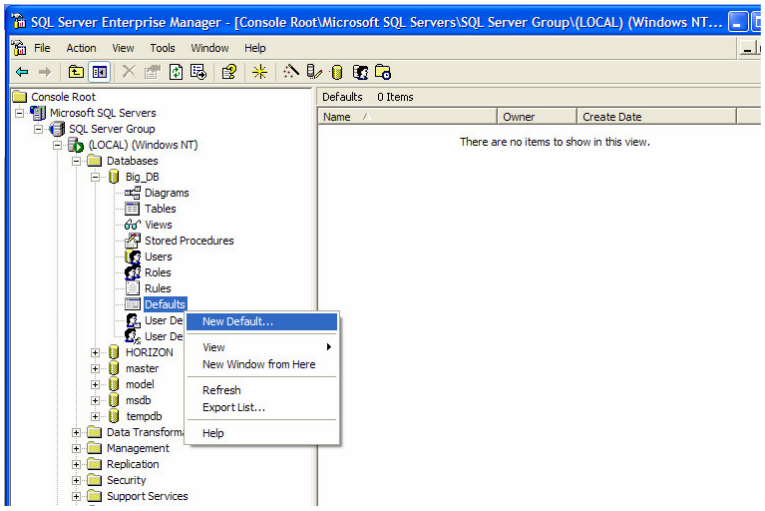
فيما يلي عرض لمثال عن هذا النوع من القيم التلقائية:

CREATE DEFAULT phone_def AS 'غير موجود'

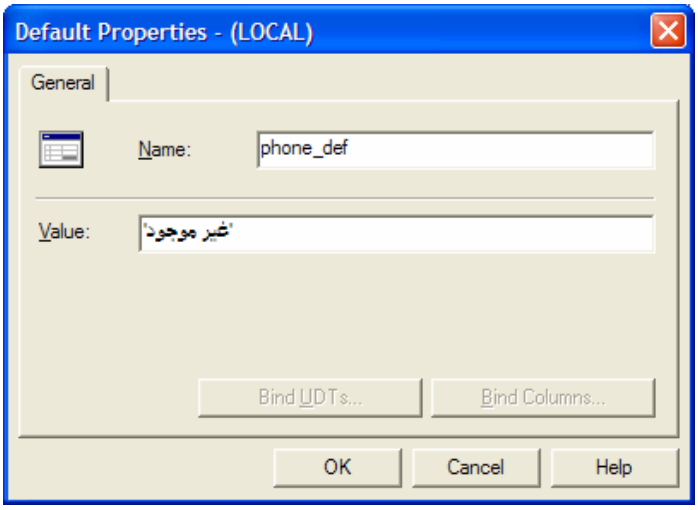
Sp_bindedefault phone_def, 'EMPLOYEE.PHONE'

إنشاء القيم التلقائية باستخدام الأداة Enterprise Manager

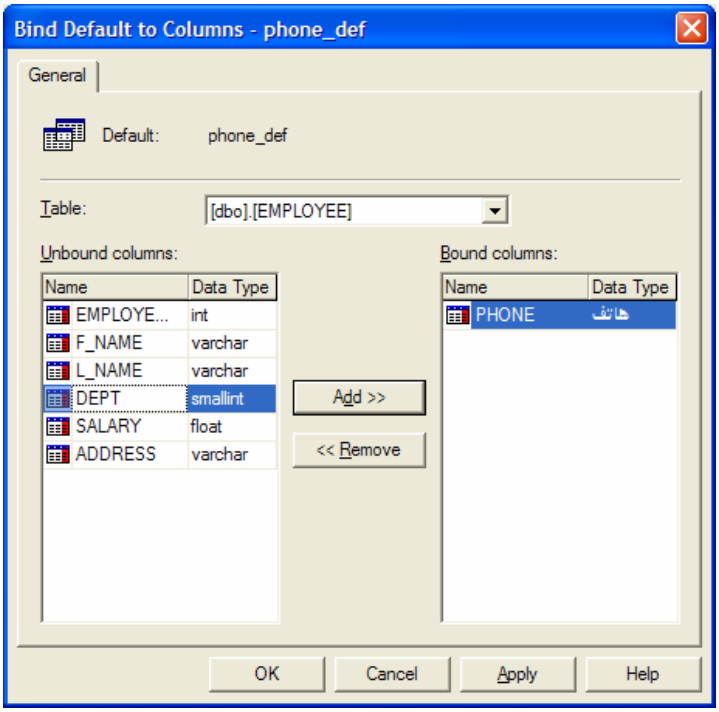
- يمكننا إنشاء القيم التلقائية من خلال التعليمة الأداة Enterprise Manager من خلال اختيار "New Default" من قائمـــة المهمات السريعة للمجلد Defaults في قاعدة المعطيات المناسبة؛



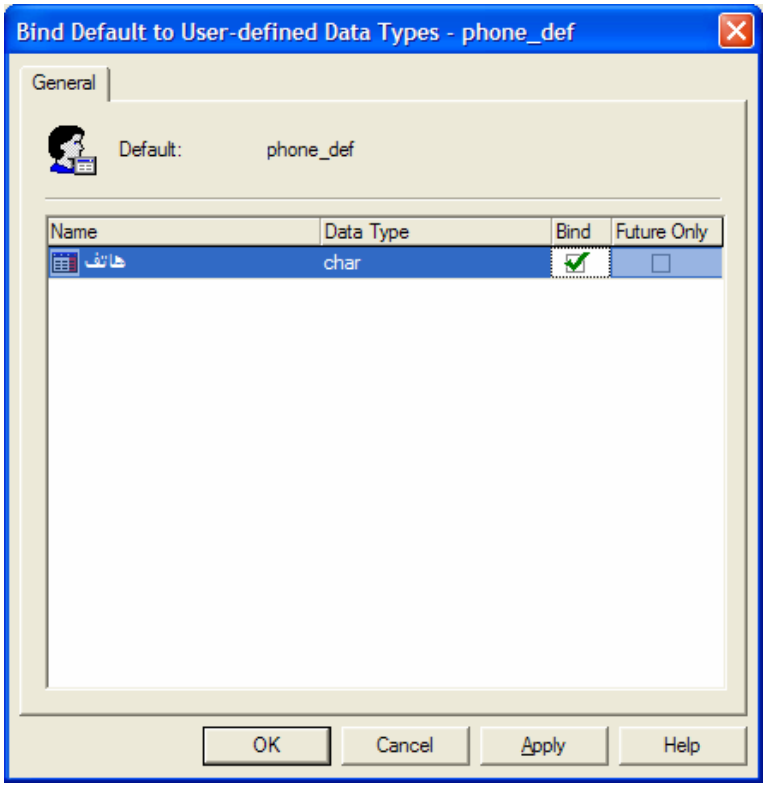
- ينبغي أن نقوم بتعبئة القيم في الأماكن المخصصة لها بحيث تطلب منّا الواجهة الخاصة بتعريف القيم التلقائية الجديدة أن نحدد كلاً من اسم القيمة التلقائية وقيمتها:



- يمكننا بعد ذلك أن نقوم بإجراء عمليات ربط القيمة التلقائية الجديدة مع الأعمدة أو الأنماط، وذلك من خلال الأزرار المخصصة لكل منها، بحيث تظهر واجهة خاصة تعرض كافة الجداول وأعمدتها ليتم الربط مع العمود المناسب:



- كما تظهر واجهة خاصة تعرض كافة أنماط معطيات المستخدم المعرّفة ليتم الربط مع أحدها:


Bind Default to User-defined Data Types - phone_def
General
Default:
phone_def
Name
Data Type
Bind
Future Only
هاتف
char
OK
Cancel
Apply
Help

عنوان الموضوع:

التخزين الاحتياطي واسترجاع المعطيات.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

سنناقش في هذه الجلسة كيف يمكننا القيام بعملية التخزين الاحتياطي وإدارتها، وسنتناول بالتفصيل أنواع تلك العملية ونماذجها والأغراض التي يمكن أن تُطبَّق عليها، كما سنناقش في هذه الجلسة أيضا كيف يمكننا القيام بعملية استرجاع المعطيات وإدارتها، وسنتناول أيضا أنواع تلك العملية ونماذجها وكيف يمكن تطبيقها.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- ما هي عملية التخزين الاحتياطي؟
- لماذا نقوم بعملية التخزين الاحتياطي؟
- ما هي خطة التخزين الاحتياطي واسترجاع المعطيات
- أنماط عمليات التخزين الاحتياطي وأنواعها:
  - التخزين الاحتياطي لكامل قاعدة المعطيات
  - التخزين الاحتياطي لأجزاء من قاعدة المعطيات
  - التخزين احتياطي للملفات ولمجموعات الملفات
  - التخزين الاحتياطي لملف سجلّ المناقلات
- نماذج التعافي:
  - نموذج التعافي التام
  - نموذج Bulk_logged Recovery
  - نموذج التعافي البسيط
- أجهزة التخزين الاحتياطي:
  - شرائط المعطيات
  - أجهزة الأقراص
  - أجهزة متعددة

- مجموعات وسائط التخزين
- عائلات وسائط التخزين
- كيفية إنشاء أجهزة التخزين الاحتياطي باستخدام T-SQL
- كيفية إنشاء أجهزة التخزين الاحتياطي باستخدام الأداة Enterprise Manager
- كيف يجري إنشاء نسخة احتياطية من قاعدة المعطيات باستخدام T-SQL أو باستخدام الأداة Enterprise Manager؟
- كيف يجري إنشاء نسخة احتياطية من ملف سجلّ المناقلات باستخدام T-SQL أو باستخدام الأداة Enterprise Manager؟
- كيف يجري إنشاء نسخة احتياطية من قواعد معطيات النظام؟
- استرجاع قاعدة المعطيات
- كيف يجري استرجاع قاعدة المعطيات باستخدام T-SQL أو باستخدام الأداة Enterprise Manager؟
- كيف يجري استرجاع ملف سجلّ المناقلات باستخدام T-SQL أو باستخدام الأداة Enterprise Manager؟

مقدمة

- تعتبر عملية التخزين الاحتياطي من إحدى أهم المهمات المرتبطة بإدارة قواعد المعطيات؛

- تعد عملية التخزين الاحتياطي واسترجاع المعطيات هي المفتاح الأساسي لأي مشروع قاعدة معطيات، وينبغي أخذها بعين الاعتبار في كافة مراحل عملية التطوير؛

- تُعرَّف عملية التخزين الاحتياطي بأنها عملية نَسخ جزئي أو كامل لقاعدة المعطيات، بحيث يمكن نقلها بعيداً عن بيئه المخدِّم ككل لأغراض أمنية؛

- تُعرَّف عملية استرجاع المعطيات بأنها عملية إعادة قاعدة المعطيات إلى حالة سابقة من خلال استخدام نسخة احتياطية جرى إنشاؤها مسبقاً. يمكن أيضاً استخدام عملية استرجاع المعطيات لنقل قاعدة المعطيات إلى مكان آخر.

تصميم خطة تخزين احتياطي واسترجاع للمعطيات

- يُمثل وضع خطة ملائمة، مفتاح النجاح في عملية التخزين الاحتياطي واسترجاع المعطيات؛

- ينبغي أخذ الوقت الكافي من أجل التخطيط لكافة الطرائق التي ينبغي استخدامها من أجل إجراء عمليات التخزين الاحتياطي، كما ينبغي التخطيط المسبق لكافة إجرائيات استرجاع المعطيات التي ينبغي اللجوء إليها وقت الحاجة؛

- إن وجود خطة فعّالة للخزن الاحتياطي ولاسترجاع المعطيات، تؤمن لمدير قاعدة المعطيات الاطمئنان تجاه الكوارث التي يمكن أن تحدث في المستقبل؛

- يجب التركيز على المعاملات التي يمكن أن تتأثر بها خطّة الخزن الاحتياطي؛

- هناك العديد من النقاط التي ينبغي على مدير قاعدة المعطيات أخذها بعين الاعتبار بعد الانتهاء من إعداد نسخ الخزن الاحتياطي.

- إن مفتاح النجاح في عملية التخزين الاحتياطي واسترجاع المعطيات هو في وجود خطة ملائمة؛
- ينبغي أخذ الوقت الكافي من أجل التخطيط لكافة الطرائق التي ينبغي استخدامها من أجل إجراء عمليات التخزين الاحتياطي، كما ينبغي التخطيط المسبق لكافة إجرائيات استرجاع المعطيات التي ينبغي اللجوء إليها وقت الحاجة؛
- إن وجود خطة فعّالة للخزن الاحتياطي ولاسترجاع المعطيات، تؤمن لمدير قاعدة المعطيات الاطمئنان تجاه الكوارث التي يمكن أن تحدث في المستقبل؛
- يمكن أن تتأثر خطّة الخزن الاحتياطي بعدّة معاملات، وهي:
  - حجم قاعدة المعطيات (أو قواعد المعطيات)؛
  - الوسائط المستخدمة لعملية التخزين الاحتياطي (قرص أو شريط ممغنط، محلياً أو على الشبكة)؛
  - توافرية قاعدة المعطيات (من التاسعة مساءً حتى السادسة صباحاً أو أنها مشغولة كل أيام الأسبوع على مدار الساعة)؛
  - كيفية تحميل المعطيات (تحميل أسبوعي أو عمليات ومناقلات تجري بشكل مستمر)؛
  - مدى عملية الاسترجاع (ما هي المدّة التي تتطلبها عملية استرجاع المعطيات).

- بعد الانتهاء من مرحلة تحليل متطلبات عملية الخزن الاحتياطي، يمكننا أن نبدأ بتنفيذ خطة التخزين الاحتياطي واسترجاع المعطيات التي نحتاجها بأخذ البيئة التي نعمل فيها بعين الاعتبار؛

- فعلى سبيل المثال، ليس من الضروري إجراء عمليات خزن احتياطي سريعة لسجل المناقلات في بيئة تمثّل مخزن معطيات، في حين تبرز أهمية ذلك الإجراء في بيئة تعالج المناقلات بشكل مباشر، كالنظام الذي يعالج مئات الطلبات في الساعة؛

- على كل حال، من الضروري أن نطرح السؤال التالي على دير قاعدة المعطيات:
  - هل تعتبر نفسك مديراً ناجحاً لقاعدة المعطيات؟

إذا كانت الإجابة بنعم، ننتقل إلى السؤال الثاني:

  - هل تعتقد أن مهمتك كمدير لقاعدة المعطيات فيما يتعلق بتخزين نسخ احتياطية من قاعدة المعطيات تنتهي عند إنشاء الخطة الملائمة للقيام بتلك العملية والحصول على النسخ الاحتياطية؟

بالتأكيد لا، فلا ينبغي مطلقاً **الافتراض** بأننا نمتلك نسخة احتياطية من قاعدة المعطيات وأنه يمكننا استرجاعها في أي وقت، إذ لابد من تجربة واختبار أية نسخة احتياطية نقوم بإنشائها؛

تقوم معظم المؤسسات –التي تدرك أهمية معطياتها وأهمية اتخاذ الاحتياطات اللازمة لتجنب الكوارث– بتدريب كوادرها على إجراء عمليات التخزين الاحتياطي واسترجاع المعطيات بشكل مستمر بحيث تضمن –إلى حدٍّ ما– جاهيزيتها لمواجهة الكوارث؛

ينبغي على مدير قاعدة المعطيات أن يدرك أمراً هاماً جداً، وهو ضرورة التوثيق، فلا يكفي أن يتمكن مدير قاعدة المعطيات من إجراء عمليات التخزين الاحتياطي واسترجاع المعطيات فحسب، بل ينبغي عليه أيضاً أن يوثّق كافة الإجراءات التي ينبغي اتخاذها لاسترجاع المعطيات إذا ما شاءت الأقدار واختارت قاعدة المعطيات أن تتعطل، عندما يكون مدير قاعدة المعطيات تلك في إجازة ما أو بعد أن يترك العمل في المؤسسة.

ما الهدف من عملية التخزين الاحتياطي؟

- ينبغي على مدير قاعدة المعطيات أن يكون حذراً في التعامل مع المسؤوليات الملقاة على عاتقه، فقد يُكلف الإهمال في موضوع القيام بعمليات التخزين الاحتياطي للمعطيات، مدير قاعدة المعطيات الكثير عند وقوع الكارثة

- لماذا نقوم بعملية التخزين الاحتياطي؟

تعدّ المعطيات حجر الأساس الذي تقوم عليه بعض المؤسسات، وتبرز هنا أهمية الحفاظ على تلك المعطيات من كافة الأخطار التي يمكن أن تحيق بها

- ينبغي أن يكون مدير قاعدة المعطيات الناجح على جاهزية تامة لاسترجاع معطياته مباشرة بعد وقوع المشكلة وبأقل قدر ممكن من الخسائر، إن لم يكن ذلك من دون خسائر على الإطلاق

- تختلف استراتيجيات الحفاظ على المعطيات، إذ يتوجب على مدير قاعدة المعطيات أن يقوم بحفظ النسخ الاحتياطية في مكان آمن بحيث يستطيع استرجاع تلك المعطيات عند الحاجة إليها

- ينبغي على مدير قاعدة المعطيات أن يكون حذراً في التعامل مع المسؤوليات الملقاة على عاتقه، بحيث أن الإهمال فيما يتعلّق بموضوع القيام بعمليات التخزين الاحتياطي للمعطيات قد يكلف مدير قاعدة المعطيات الكثير عند وقوع الكارثة

- يمكن أن يخطر ببالنا دائماً السؤال التالي: لماذا نقوم بعملية التخزين الاحتياطي؟

مرّ معنا سابقاً أنه من الممكن أن تُفلِس بعض المؤسسات إذا ما خسرت معطياتها، لذا تُعدّ المعطيات حجر الأساس الذي تقوم عليه تلك المؤسسات، وتبرز هنا أهمية الحفاظ على تلك المعطيات من كافة الأخطار التي يمكن أن تُحيق بها، سواء نجمت عن ضياع بعض المعطيات أو نجمت عن أخطاء استخدام، أو نجمت عن مشاكل أخرى كبرامج خبيثة أو فيروسات أو كوارث مختلفة كالزلازل أو الفيضانات أو نشوب الحرائق أو غيرها؛

- ينبغي أن يكون مدير قاعدة المعطيات الناجح على جاهزية تامة لاسترجاع معطياته مباشرة بعد وقوع المشكلة وبأقل قدر ممكن من الخسائر، إن لم يكن ذلك من دون خسائر على الإطلاق؛

- تختلف استراتيجيات الحفاظ على المعطيات، إذ يتوجب على مدير قاعدة المعطيات أن يقوم بحفظ النسخ الاحتياطية في مكان آمن بحيث يستطيع استرجاع تلك المعطيات عند الحاجة إليها، وتختلف الأساليب التي يمكن اتباعها في الحفظ الاحتياطي من مؤسسةٍ إلى أخرى، بحيث يمكن أن يكتفي البعض بنسخ المعطيات على أقراص وحفظها في مستودع الشركة، أو يمكن أن تتخذ مؤسسات أخرى احتياطات من نمط حفظ المعطيات في بناءٍ آخر تحسباً لما يمكن أن يحدث من كوارث، ويمكن أن يقوم البعض الآخر بحفظ تلك المعطيات في مدينةٍ أخرى، هذا بالإضافة إلى إمكانية تكرار تلك النسخ الاحتياطية على أكثر من موقع.

أنماط عمليات التخزين الاحتياطي

-تخزين احتياطي لكامل قاعدة المعطيات-

- يدعم SQL Server 2000 أربعة أنواع مختلفة من عمليات التخزين الاحتياطي؛

- تخزين احتياطي لكامل قاعدة المعطيات:
  - تقوم هذه العملية بإنشاء صورة عن قاعدة المعطيات منذ إنشائها ولغاية انتهاء عملية التخزين الاحتياطي الأخيرة قيد التنفيذ؛

  - تعتبر عملية التخزين الاحتياطي لكامل قاعدة المعطيات عملية ديناميكية تسمح بحدوث التغيرات على المعطيات خلال سير عملية التخزين الاحتياطي؛

  - ولكن، كيف يمكن أن تبقى قاعدة المعطيات متّسقة مع النسخة التي تم إنشاؤها، مع العلم أنه من الممكن أن تستغرق عملية التخزين الاحتياطي عدّة ساعات في بعض قواعد المعطيات الكبيرة وأن التغيرات مستمرة على قاعدة المعطيات الأصلية؟
- يدعم SQL Server 2000 أربعة أنواع مختلفة من عمليات التخزين الاحتياطي، وهي:

  - تخزين احتياطي لكامل قاعدة المعطيات:

تقوم هذه العملية بإنشاء صورة عن قاعدة المعطيات منذ إنشائها ولغاية انتهاء عملية التخزين الاحتياطي الأخيرة قيد التنفيذ، بحيث يقوم SQL Server 2000 بإنشاء رقم سجلّ تسلسلي يقوم من خلاله بتعليم بداية عملية النسخ الاحتياطي، ليباشر بعد ذلك عملية نسخ قاعدة المعطيات؛

تعتبر عملية التخزين الاحتياطي لكامل قاعدة المعطيات عملية ديناميكية تسمح بحدوث التغيرات على المعطيات خلال سير عملية التخزين الاحتياطي؛

توسّع:لا ضررة لقراءته

(رقم السجلّ التسلسلي عبارة عن رقم يرتبط بكل تسجيلة تجري كتابتها في سجلّ المناقلات بغرض متابعة وملاحقة التغيرات).

ولكن، كيف يمكن أن تبقى قاعدة المعطيات متّسقة مع النسخة التي تم إنشاؤها، مع العلم أنه من الممكن أن تستغرق عملية التخزين الاحتياطي عدّة ساعات في بعض قواعد المعطيات الكبيرة وأن التغيرات مستمرة على قاعدة المعطيات الأصلية؟

كما أشرنا سابقاً فإن SQL Server يقوم بإنشاء رقم سجلّ تسلسلي عند بداية عملية التخزين الاحتياطي، وأثناء قيامه بعملية نسخ قاعدة المعطيات يقوم في الوقت نفسه بتخزين التغيرات التي تحدث على قاعدة المعطيات الأصلية في سجل المناقلات –كالعادة–، وبعد الانتهاء من عملية النسخ يقوم بإضافة رقم سجلّ تسلسلي جديد إلى سجل المناقلات، وبعد ذلك يربط جزء السجل المحصور ما بين آخر رقم سجلّ تسلسلي والرقم التسلسلي الذي قبله، بكتلة الخزن الاحتياطية المنشأة، بحيث يمثل هذا الجزء الجديد من السجلّ كافة التغيرات التي طرأت على قاعدة المعطيات أثناء سير عملية التخزين الاحتياطي؛

عند إجراء عملية استرجاع للمعطيات، يجري استبدال قاعدة المعطيات الحالية بقاعدة المعطيات التي جرى إنشاؤها في النسخة الاحتياطية، بعد ذلك يجري تنفيذ جزء السجلّ المرفق مع النسخة الاحتياطية الأخيرة تلك ويجري تطبيق التغيرات التي تمت أثناء سير عملية التخزين الاحتياطي؛

نلاحظ مما سبق، أنه بازدياد عدد المناقلات التي تتم أثناء سير عملية التخزين الاحتياطي، يزداد حجم جزء السجل المرفق بتلك الكتلة، مما يؤثر بدوره أيضاً على الزمن الذي تتطلبه عملية التخزين الاحتياطي وعملية استرجاع المعطيات أيضاً، بالتالي يُنصح بإجراء هذا النمط من عمليات التخزين الاحتياطي في الفترة التي يكون فيها النشاط على قاعدة المعطيات أقل ما يمكن.

أنماط عمليات التخزين الاحتياطي

– تخزين احتياطي لأجزاء مختلفة من قاعدة المعطيات –

تقوم هذه العملية بإنشاء نسخ من قاعدة المعطيات اعتباراً من آخر عملية تخزين احتياطي كامل جرى إنشاؤها؛

تعتبر عملية التخزين الاحتياطي لأجزاء قاعدة المعطيات أسرع من عملية التخزين الاحتياطي لكامل قاعدة المعطيات، بحيث يتم فقط نسخ الامتدادات التي حصلت فيها تغيرات في المعطيات منذ عملية النسخ الاحتياطي الكامل الأخيرة؛

تسمح هذه الخدمة التي يقدّمها SQL Server بأن يجري توسيع الأداة لتصبح قادرة على دعم قواعد المعطيات الضخمة، وهذه ميزة هامة جداً ينبغي توافرها في نظم إدارة قواعد المعطيات؛

ينبغي قبل القيام بعملية استرجاع المعطيات المجزأة أن نقوم أولاً بعملية استرجاع كامل قاعدة المعطيات؛

مثال:

لنفترض أننا قمنا بإجراء عملية تخزين احتياطي لكامل قاعدة المعطيات ليلة الخميس، وبعد ذلك قمنا بـــإجراء عمليـــات تخـــزين احتياطي لأجزاء من قاعدة المعطيات بشكل يومي في منتصف الليل، ولنفترض الآن حدوث فشل ما في قاعـــدة المعطيـــات يـــوم الثلاثاء صباحاً، بالتالي ينبغي لكي نقوم باسترجاع المعطيات المفقودة أن نطبق الخطوات التالية:

- نقوم أولاً بإجراء عملية استرجاع معطيات لكامل قاعدة المعطيات التي تم تخزينها احتياطياً يوم الخميس الماضي؛
- ينبغي بعد ذلك استرجاع آخر نسخة من المعطيات الجزئية المخزّنة، أي النسخة المستخرجة يوم الاثين في منتصف الليل؛
- يمكن بعد ذلك تنفيذ استرجاع معطيات السجلات –في حال وجودها– للعودة بقاعدة المعطيات إلى الحالة الأخيرة التي كانـــت عليها قبل حدوث المشكلة.

- تخزين احتياطي لأجزاء مختلفة من قاعدة المعطيات:

أُتيحت هذه الميزة منذ الإصدار 7.0 من SQL Server، تقوم هذه العملية بإنشاء نسخ من قاعدة المعطيات اعتباراً من آخر عملية تخزين احتياطي كامل جرى إنشاؤها؛

تعتبر عملية التخزين الاحتياطي لأجزاء قاعدة المعطيات أسرع بالضرورة من عملية التخزين الاحتياطي لكامل قاعدة المعطيـــات، بحيث يجري فقط نسخ الامتدادات التي حصلت فيها تغيرات في المعطيات منذ عملية النسخ الاحتياطي الكامل الأخيرة؛

تسمح هذه الخدمة التي يقدّمها SQL Server بأن يجري توسيع هذه الأداة لتصبح قادرة على دعم قواعد المعطيات الضخمة، وهذه ميزة هامة جداً ينبغي توافرها في نظم إدارة قواعد المعطيات؛

ينبغي قبل القيام بعملية استرجاع المعطيات المجزأة أن نقوم أولاً بعملية استرجاع كامل قاعدة المعطيات؛

أنماط عمليات التخزين الاحتياطي

– تخزين احتياطي للملفات ولمجموعات الملفات–

صممت هذه الميزة لكي تخدم حالات التخزين الاحتياطي لقواعد المعطيات الضخمة؛

يمكن أن تجري عملية التخزين الاحتياطي، لملف تلو الآخر، أو لمجموعة ملفات تليها مجموعة ملفات أخرى؛
يمكننا من خلال هذه الطريقة الحصول على نسخة احتياطية من قاعدة معطيات ضخمة بأسلوب مرن؛

تظهر محدودية هذه الطريقة في التخزين الاحتياطي، عندما تحتوي قاعدة المعطيات على جدول ما مخزّن على مجموعة ملفات معينة، في حين يجري تخزين فهارس ذلك الجدول على مجموعة ملفات أخرى؛

تبرز أهمية هذا النوع من أنماط التخزين الاحتياطي، وخاصةً فيما يتعلق بتوفير الوقت عند إجراء عمليات استرجاع المعطيات في قواعد معطيات ضخمة.

تخزين احتياطي للملفات ولمجموعات الملفات:
صممت هذه الميزة لكي تخدم حالات التخزين الاحتياطي لقواعد المعطيات الضخمة؛
يمكن أن تجري عملية التخزين الاحتياطي، لملف تلو الآخر، أو لمجموعة ملفات تليها مجموعة ملفات أخرى، بالتالي، يمكن إجراء عمليات تخزين احتياطي لقاعدة معطيات ذات حجم يبلغ GB 500 ومكوّنة من خمسة ملفات كل منها حجمة GB 100، بأن يتم إجراء تلك العملية على كل ملف من تلك الملفات، بحيث يمكن أن يتم نسخ الملف الأول ليلة الجمعة و الملف الثاني ليلة السبت وهكذا...، يمكننا من خلال هذه الطريقة الحصول على نسخة احتياطية من قاعدة معطيات ضخمة بأسلوب مرن؛

تظهر محدودية هذه الطريقة في التخزين الاحتياطي، عندما تحتوي قاعدة المعطيات على جدول ما مخزّن على مجموعة ملفات معينة، في حين يتم تخزين فهارس ذلك الجدول على مجموعة ملفات أخرى؛

تعتبر الطريقة السابقة شائعة الاستخدام قي قواعد المعطيات، وذلك لأغراض تحسين الأداء، إلا أننا سنخسر إمكانية استخدام طريقة التخزين الاحتياطي لقاعدة المعطيات من خلال نَسخ الملفات أو مجموعات الملفات، إذ ينبغي إجراء تخزين احتياطي واسترجاع للمعطيات المخزنة على الجداول وفهارسها في آنٍ واحد، مع العلم أنه يمكننا أن نلجأ في مثل هذه الحالات إلى إجراء تخزين احتياطي لملف سجلّ المناقلات كما سيمر معنا في الشريحة التالية؛

كما مرّ معنا سابقاً، فإنه في كلتا حالتي التخزين الاحتياطي لكامل قاعدة المعطيات أو لأجزاء منها، فإنه لابد من إعادة تخزين كامل قاعدة المعطيات من أجل استرجاع المعطيات المفقودة في حالات ضياع المعطيات، إلا أنه يكفي بطريقة التخزين الاحتياطي للملفات ومجموعات الملفات أن يتم استرجاع الملف المتضرر فقط، وهنا تبرز أهمية هذا النوع من أنماط التخزين الاحتياطي، وخاصةً فيما يتعلق بتوفير الوقت عند إجراء عمليات استرجاع المعطيات في قواعد معطيات ضخمة؛

ينبغي ملاحظة أمر هام كتتمة لما سبق، ففي كثير من الأحيان لا يكفي أن نقوم باسترجاع ملف معطيات وحيد حتى ولو كان هو الملف الوحيد المتضرر، إذ من الممكن أن نكون قد قمنا بإجراء العديد من التغييرات على قاعدة المعطيات بعد إجراء عملية التخزين الاحتياطي لذلك الملف، بالتالي، يجب أن نمتلك ملف السجل الذي كان فعّالاً ويحتوي على كافة التغيرات التي تم إجراؤها على قاعدة المعطيات منذ القيام بعملية التخزين الاحتياطي الأخيرة، وبتنفيذ ذلك السجل بعد استرجاع ملف المعطيات يمكننا إعادة قاعدة المعطيات إلى الحالة المتَّسقة.

1.9-10-11.07 أنماط عمليات التخزين الاحتياطي

- تخزين احتياطي لملف سجلّ المناقلات-

- يتم من خلال التخزين الاحتياطي لملف سجلّ المناقلات نَسخ كافة المناقلات في ذلك السجلّ ثم حذف كافة المداخل في ذلك الملف -ما عدا المَدخل الفعّال- وذلك بغية توفير الحجم؛

- على اعتبار أن ملف سجلّ المناقلات يحتوي على تسجيل لكافة المناقلات التي تمَّ تنفيذها منذ القيام بعملية التخزين الاحتياطي الأخيرة، فبالتالي يمكننا القيام باسترجاع تلك الملفات أثناء القيام بعملية استرجاع النسخة الأخيرة المخزّنة من قاعدة المعطيات، وذلك من أجل إعادة قاعدة المعطيات إلى آخر حالة كانت عليها قبل أن تفشل.

تخزين احتياطي لملف سجلّ المناقلات:

- يتم من خلال التخزين الاحتياطي لملف سجلّ المناقلات نَسخ كافة المناقلات في ذلك السجلّ ثم حذف كافة المداخل في ذلك الملف -ما عدا المَدخل الفعّال- وذلك بغية توفير الحجم؛

- على اعتبار أن ملف سجلّ المناقلات يحتوي على تسجيل لكافة المناقلات التي تمَّ تنفيذها منذ القيام بعملية التخزين الاحتياطي الأخيرة، فبالتالي يمكننا القيام باسترجاع تلك الملفات أثناء القيام بعملية استرجاع النسخة الأخيرة المخزّنة من قاعدة المعطيات، وذلك من أجل إعادة قاعدة المعطيات إلى آخر حالة كانت عليها قبل أن تفشل؛

- عند القيام بإجراء عملية التخزين الاحتياطي لملف سجلّ المناقلات، فإنه يمكننا أن نقوم بإجراء عملية استرجاع للمعطيات -اعتماداً على ذلك الملف- لغاية فترة محددة؛

- ما أن يتم إجراء تخزين احتياطي لمف سجلّ المناقلات، حتى يتم اقتصاص الملف الفعلي وحذف المناقلات غير الفعّالة منه، وذلك لإتاحة المجال لمناقلات أخرى لكي يتك تخزينها في هذا السجلّ، تسمح هذه الطريقة بتجنب إجراء عمليات نمو تلقائية في ملف سجلّ المناقلات، بحيث يتم ترك مجال تخزين متاح للمناقلات الجديدة.

نماذج التعافي

- تعتبر نماذج التعافي الموجودة في SQL Server 2000 خدمه جديدة مضافة إلى هذا الإصدار من SQL Server؛
- يمكننا أن نختار نموذج تعافي لكل قاعدة معطيات من قواعد معطيات SQL Server، بحيث نستطيع من خلاله أن نحدد كم يمكن أن يتم القيام بعملية التخزين الاحتياطي للمعطيات وما هو الهامش المسموح به في خسارة المعطيات أثناء القيام بعملية تعافي من مشاكل فقدان المعطيات؛
- يدعم SQL Server 2000 ثلاثة أنواع من نماذج التعافي، سنقوم بشرح كل منها بالتفصيل، وهي:
  - التعافي التام

- Bulk_logged Recovery
- التعافي البسيط

نماذج التعافي
-التعافي التام-

- يعتبر التعافي التام هو النموذج الأكثر وُثوقية بين كافة نماذج التعافي الأخرى التي يقدّمها SQL Server وخاصّة فيما يتعلق بمخاطر فقدان المعطيات نتيجةً لفشل وسائط التخزين أو بسبب أخطاء مرتكبة من قبل المستخدمين أو التطبيقات. كما يزوّدنا هــذا النموذج بالمرونة المطلقة فيما يتعلق باسترجاع العمليات؛

- يرتبط تنفيذ نموذج التعافي التام بشكل كبير بعمليات التخزين الاحتياطي لكامل قاعدة المعطيات وكــذلك بعمليــات التخــزين الاحتياطي لسجلّ المناقلات.

- يعتبر التعافي التام هو النموذج الأكثر وُثوقية بين كافة نماذج التعافي الأخرى التي يقدّمها SQL Server وخاصّة فيما يتعلق بمخاطر فقدان المعطيات نتيجةً لفشل وسائط التخزين أو بسبب أخطاء مرتكبة من قبل المستخدمين أو التطبيقات. كما يزوّدنا هــذا النموذج بالمرونة المطلقة فيما يتعلق باسترجاع العمليات؛

- يرتبط تنفيذ نموذج التعافي التام بشكل كبير بعمليات التخزين الاحتياطي لكامل قاعدة المعطيات وكــذلك بعمليــات التخــزين الاحتياطي لسجلّ المناقلات؛

- يتم من خلال نموذج التعافي التام، إعادة تخزين قاعدة المعطيات ككل، واسترجاعها -تماماً-كما كانت في فترة زمنية محددة؛

- لكي يتم تطبيق هذا النموذج بالشكل المناسب، ينبغي أن يتم الاحتفاظ بكافة العمليات التي تمّت على قاعدة المعطيات، ويتضمن ذلك كافة التعليمات حتى SELECT INTO و BULK INSERT و bcp أو حتى CREATE INDEX؛

- تعتبر السيئة الوحيدة التي يمكن اعتبارها في هذا النموذج من نماذج التعافي، عي مشكلة نمو حجم ملف سجلّ المناقلات بشكل كبير، فيما عدا ذلك يعتبر هذا النموذج هو الأقل خطورة فيما يتعلق بضياع المعطيات.

نماذج التعافي
-Bulk_logged Recovery-

- يعتبر Bulk_logged Recovery بدوره نموذج تعافي آخر يسمح بإعادة تخزين كامل قاعدة المعطيات، بالإضافة إلى أنه يزوِّد أداءً أفضل للعمليات الضخمة بحيث يستهلك حيّز سجلّ أقل؛

- يتم من خلال نموذج Bulk_logged Recovery تسجيل العمليات الضخمة التي يمكن أن تجري على قاعدة المعطيات بأبسط طريقة ممكنة، مما يسمح بتحسين الأداء المتوقّع خلال تنفيذ تلك العمليات الضخمة

نماذج التعافي
-التعافي البسيط-

- يعبِّر نموذج التعافي البسيط عن النموذج الأكثر بدائية الذي يمكن استخدامه من أجل القيام بعمليات التخزين الاحتياطي واسترجاع المعطيات؛

- عادةً ما يكون ملف سجلّ المناقلات -في النموذج البسيط- مختزل إلى حدٍّ لا يكفي لإجراء عمليات تخزين احتياطي؛

- لا يدعم هذا النموذج إمكانيات إجراء تعافي كامل، بحيث لايمكن ضمان إعادة حالة قاعدة المعطيات إلى النقطة التي حصلت عندها المشكلة، كما لا يمكن أن يستخدم هذا النموذج لإعادة قاعدة المعطيات إلى حالتها في فترة زمنية محددة؛

- نستطيع من خلال هذا النموذج أن نستعيد المعطيات الضائعة من خلال استرجاع آخر نسخة مخزنة سواء كانت لكامل قاعدة المعطيات أو لجزء منها، بحيث ينبغي بعد ذلك القيام بعمليات تعافي يدوية، ونتيجةً لذلك، لا يفضل استخدام نموذج التعافي البسيط في معظم قواعد المعطيات التجارية؛

- يعتبر هذا النموذج مناسباً للاستخدام في قواعد المعطيات التي لا يتم إجراء عدد كبير من المناقلات عليها نسبياً، وكمثال على ذلك، تستخدم معظم قواعد المعطيات التي تستعمل لأنظمة دعم القرار هذا النوع من نماذج التعافي، بحيث تكون فيها نسبة عمليات التعديل منخفضة وليس بالضرورة أن نستخدم نماذج معقدة من أجل إجراء عمليات التخزين الاحتياطي على هذا النوع من قواعد المعطيات.

أجهزة التخزين الاحتياطي

- تستخدم أجهزة التخزين الاحتياطي لتأمين مكان مناسب أو شيء ملائم لتخزين نُسَخ المعطيات المنشأة؛
- تدعم الأداة SQL Server 2000 كلاً من أجهزة التخزين الاحتياطي الدائمة والمؤقتة؛
- تعبر أجهزة التخزين الاحتياطي الدائمة عن أجهزة مسبقة الإنشاء يمكن استخدامها لأكثر من مرّة في تخزين النُسخ الاحتياطية المنشأة؛
- تعبر أجهزة التخزين الاحتياطي المؤقتة –كما يبدو من اسمها– عن أجهزة تُعرَّف أثناء إنشاء النسخة الاحتياطية وتستخدم لمرّة واحدة فقط؛
- يمكن تخزين النُسخ الاحتياطية المنشأة إما على ملف أوعلى شريط أو على غيره من الوسائط.

أجهزة التخزين الاحتياطي
–شرائط المعطيات–

- مرّ معنا أن الهدف من استخدام أجهزة التخزين الاحتياطي، هو لتأمين مكان مناسب أو شيء ملائم لتخزين نسَخ المعطيات المنشأة؛
- تستخدم شرائط المعطيات لتخزين نسخ المعطيات إلى شرائط؛
- ينبغي أن تتصل هذه الأجهزة مباشرةً مع المخدِّم، كما يمكن أن تتصل عدّة نسخ احتياطية بعدّة سوّاقات على التفرع، ما يمكن أن يؤدي إلى تحسين الأداء؛
- تتمتع شرائط المعطيات بكونها آمنة وقابلة للنقل وقابلة للتوسُّع، فقابلية التوسع أمرٌ هام مع نمو قاعدة المعطيات، وبما أنها قابلة للنقل فيمكن تخزينها في أماكن آمنه بعيداً عن السرقات أو الأضرار الأخرى؛
- يدعم SQL Server توليد ملفات التخزين الاحتياطي –الخاصة بالتخزين على الشرائط– بهيئة معروفة يطلق عليها اسم Microsoft Tape Format (MTF)، بالتالي يمكن أن تتشارك كلا من نسخ SQL Server و نسخ Windows NT و Windows 2000 بنفس شريط التخزين؛

- يمكننا من خلال الأداة SQL Server أن نقوم بجدولة عمليات تخزين احتياطي على شرائط.

أجهزة التخزين الاحتياطي

-أجهزة الأقراص-

- تتكوّن أجهزة الأقراص من ملف يُخزَّن عادةً في مجلّد ما على القرص الصلب المحلّي، مع العلم أنه لايبغي أن يكون هذا القرص هو نفسه القرص المستخدم لتخزين المعطيات الأصلية؛

- تتمتع أجهزة الأقراص بعدة محاسن فيما يتعلق بالسرعة والوثوقية، ولكننا ينبغي أن نكون حذرين أثناء التعامل مع هذا النوع من أجهزة التخزين الاحتياطي، إذ أن النُسخ المولَّدة يتم تخزينها محلياً على نفس القرص، وهذا يمكن أن يؤدي إلى الكثير من المشاكل، فمَن يضمن ألا ينفجر المخدم بأكملة لسبب من الأسباب؟

- مثال:

تم إجراء اختبار عملي للقيام بعملية تخزين احتياطي لقاعدة معطيات يبلغ حجمها MB 15، بحيث يتم تخزينها على قرص مخدم آخر يتم الوصول إليه من خلال شبكة Ethernet بسرعة نقل تبلغ 10Mbps، وقد استغرقت العملية تلك ككل 47,241 ثانية، في حين استغرقت تلك العملية 4,766 ثانية عندما تمّ إجراءها على نفس القرص المحلي؛

أجهزة التخزين الاحتياطي

- القرص أم الشريط-

- لنطرح الآن السؤال التالي: أيهما أفضل للاستخدام، القرص أم الشريط؟
  - تقليدياً، تستخدم الشرائط من أجل تخزين النسخ الاحتياطية من قواعد المعطيات؛

  - على الرغم من أن استخدام الشريط لتخزين النسخ الاحتياطية، أكثر أمناً من استخدام القرص، إلا أنه يتمتع بسيئة تتعلق بالبطء الناتج عن استخدامه سواءً للتخزين الاحتياطي أو لاسترجاع المعطيات؛

  - كانت السيئة الأساسية التي حدّت من استخدام طريقة الأقراص لتخزين النسخ الاحتياطية ترتبط بالكلفة العالية للأقراص،إلا أنه يتوافر حالياً أقراص ذات سعة كبيرة وسرعة عالية وموثوقية أكبر، ولكننا ما نزال نعاني من مشكلة أمن الأقراص –غير القابلة النقل– بحد ذاتها.

- لنطرح الآن السؤال التالي: أيهما أفضل للاستخدام، القرص أم الشريط؟

  - تقليدياً، تستخدم الشرائط من أجل تخزين النسخ الاحتياطية من قواعد المعطيات؛

- على الرغم من أن استخدام الشريط لتخزين النسخ الاحتياطية، أكثر أمناً من استخدام القرص، إلا أنه يتمتع بسيئة تتعلق بالبطء الناتج عن استخدامه سواءً للتخزين الاحتياطي أو لاسترجاع المعطيات، هذا بالإضافة إلى أن الشرائط تتطلب المزيد من الإجراءات حتى يتم استخدامها، وذلك فيما يتعلق بتحميل الشريط المناسب لعملية التخزين الاحتياطي أو لاسترجاع المعطيات؛

- في الماضي، كانت السيئة الأساسية التي حدّت من استخدام طريقة الأقراص لتخزين النسخ الاحتياطية ترتبط –وبشكل مباشر– بالكلفة العالية للأقراص في ذلك الوقت، مع العلم أن هذه المشكلة مازالت موجودة عندما نتحدث عن قواعد معطيات ضخمة، إلا أنه، ومع التطور التكنولوجي، ازداد توافر الأقراص ذات السعة الكبيرة والسرعة العالية والموثوقية الأكبر، ولكننا ما نزال نعاني من مشكلة أمن الأقراص –غير القابلة النقل– بحد ذاتها؛

- اقتراح:

يمكننا أن نتّبع الطريقة التالية للاستفادة من كلٍّ من طريقة التخزين الاحتياطي على الأقراص وطريقة التخزين الاحتياطي على الشرائط، بحيث نقوم كخطوة أولى بإجراء عملية تخزين احتياطي محلية على القرص الحالي، بعد الانتهاء من هذه العملية يمكننا الانتقال إلى الخطوة التالية بحيث يمكننا أن نبدأ بنسخ تلك النسخة الاحتياطية المنشأة، إلى شريط أو أكثر محلياً أو عبر الشبكة، مع العلم أن هذه العملية تتم بشكل مستقل عن الأداة SQL Server، إذ ليس بالضرورة أن يتم إيقافه عن عمله فلا يوجد أي ارتباط بينه وبين النسخة المخزّنة على القرص والتي تم انشاؤها في الخطوة الأولى.

- ملاحظة:

يزودنا SQL Server 2000 بأجهزة تخزين احتياطي تُعرف باسم Named Pipe Devices وهي تلعب دور طرف ثالث في عملية التخزين الاحتياطي، كما أنها تُستخدم عادةً عندما نقوم بإجراء عملية تخزين احتياطي عن بعد، على شريط معين.

أجهزة التخزين الاحتياطي

–أجهزة متعددة–

- لتسهيل عملية دعم قواعد المعطيات الضخمة، يدعم SQL Server إمكانية إجراء عمليات تخزين احتياطي على عدّة أجهزة، وهذا ما يُشار إليه عادةً باسم عملية التخزين الاحتياطي التفرعية؛

- نستطيع من خلال اتباع طريقة التخزين الاحتياطي على أجهزة مختلفة أن نختزل الكثير من زمن التخزين الاحتياطي أو استرجاع المعطيات بحيث يتم إجراء تلك العمليات على عدّة أجهزة في آنٍ واحد؛

- ينبغي عند استخدام طريقة التخزين الاحتياطي التفرعية أن تكون الأجهزة على كافة المواقع متماثلة من حيث نوع الوسائط المستخدمة للتخزين، بالإضافة إلى أنه لا ينبغي إجراء مزج مُختلف مع عمليات تخزين احتياطي أخرى؛

- عند تهيئة أحد أجزاء مجموعة النسخة الاحتياطية التفرعية، فستفقد بالضرورة كافة الأجـزاء الأخـرى قيمتهـا، وستـصبح محتوياتها غير قابلة للاستخدام.

- لتسهيل عملية دعم قواعد المعطيات الضخمة، يدعم SQL Server إمكانية إجراء عمليات تخزين احتياطي على عدّة أجهزة، وهذا ما يُشار إليه عادةً باسم عملية التخزين الاحتياطي التفرعية؛

- نستطيع من خلال اتباع طريقة التخزين الاحتياطي على أجهزة مختلفة أن نختزل الكثير من زمن التخـزين الاحتيـاطي أو استرجاع المعطيات بحيث يتم إجراء تلك العمليات على عدّة أجهزة في آنٍ واحد؛

- ينبغي عند استخدام طريقة التخزين الاحتياطي التفرعية أن تكون الأجهزة على كافة المواقع متماثلة من حيث نوع الوسـائط المستخدمة للتخزين، بالإضافة إلى أنه لا ينبغي إجراء مزج مُختلف مع عمليات تخزين احتياطي أخرى؛

- بعد أن يتم استخدام جهاز ما في عملية التخزين الاحتياطي التفرعية، يتم اعتباره كجزء من مجموعة تُكوِّن النسخة الاحتياطية، ولا يمكن استخدامه لوحده من دون بقية الأجزاء المكوِّنة لتلك النسخة، كما لا يمكن استخدام ذلك الجهاز لوحدة مرّة أخرى من أجل إجراء عمليات تخزين احتياطي ما لم يتم إعادة تهيئته من جديد وحذف محتوياته السابقة؛

- عند تهيئة أحد أجزاء مجموعة النسخة الاحتياطية التفرعية، فستفقد بالضرورة كافة الأجـزاء الأخـرى قيمتهـا، وستـصبح محتوياتها غير قابلة للاستخدام؛

- يمكن أن يكون كلاً من أجهزة التخزين الاحتياطي الدائمة أو المؤقتة، جزءاً من مجموعة النسخة الاحتياطية التفرعية؛

- فيما يلي عرض لمخطوط يُعبر عن إجراء عملية نسخ احتياطي إلى عدّة أجهزة دائمة:

BACKUP DATABASE *databaseName* **TO** *backup1, backup2, backup3*

مجموعات وعائلات وسائط التخزين

تعبِّر "**مجموعات وسائط التخزين**" و "**عائلات وسائط التخزين**" عن مصطلحات تستخدم للدلالة علـى مكوِّنـات عمليـة التخـزين الاحتياطي، بحيث تعبِّر عائلات وسائط التخزين عن كافة الوسائط المستخدمة من قبل جهاز واحد؛

مثال:

لنفترض أنه لدينا مجموعة تخزين احتياطي مكوّنة من أربعة أجهزة شرائط، وكل جهاز منها يتطلّب بدوره تحميل خمسة أشـرطة أخرى لكي تكتمل النسخة الاحتياطية. تعبِّر كافة الأشرطة الخمسة –المستخدمة من قبل الجهاز الأول– عن عائلة وسائط تخزين؛

يطلق اسم **الوسيط الأساسي** على أول مكوّن من مكوّنات عائلة وسائط التخزين، كما يطلق على بقية المكوِّنات اسم **الوسائط التابعة.**
يطلق على كافة عائلات وسائط التخزين مجتمعةً اسم **مجموعة وسائط التخزين؛**

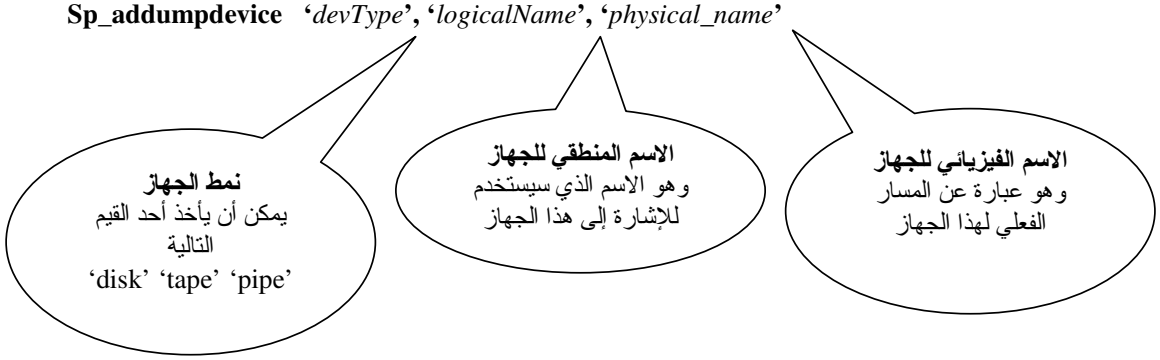
بالتالي تتكون مجموعة وسائط التخزين في المثال السابق من 20 شريط.

إنشاء أجهزة التخزين الاحتياطي باستخدام T-SQL

- يمكننا استخدام إجرائية النظام sp_addumpdevice() لإنشاء جهاز تخزين احتياطي دائم؛

- فيما يلي عرض لمخطوط يبين الإجرائية السابقة ومعملاتها:

Sp_addumpdevice '*devType*', '*logicalName*', '*physical_name*'



- فيما يلي مثال يعرض إنشاء قرص مع تحديد اسمه ومساره:

Exec Sp_addumpdevice '*disk*', '*bigDB_backup*', '*E:\myBackups\bigDB_backup_01-05-06.bak*'

- فيما يلي مثال يعرض إنشاء شريط مع تحديد اسمه ومساره:

Exec Sp_addumpdevice '*tape*', '*bigDB_backup_tape*', '\\.\tape0'

- فيما يلي مثال يعرض المخطوط الذي يمكن استخدامه لإنشاء قرص مؤقت مع تحديد مساره:

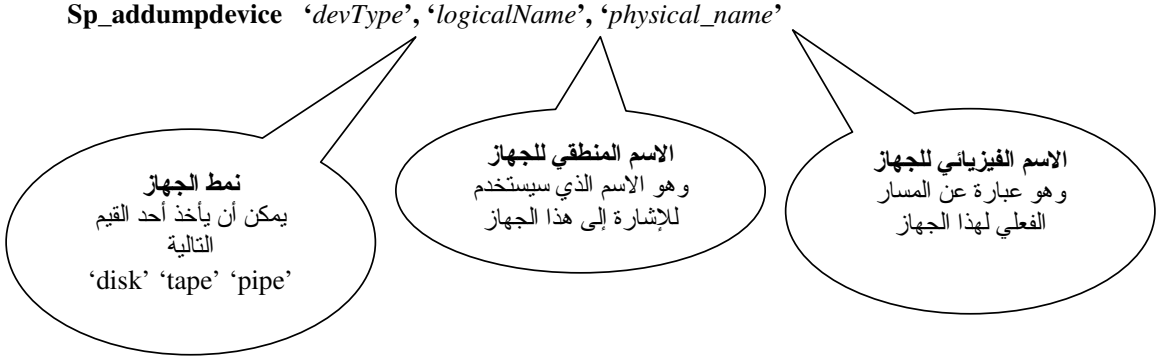
BACKUP DATABASE *databaseName* **TO DISK = '***D:\myBackups\temp_dev.bak***'**

- يمكننا من خلال استخدام أجهزة التخزين الاحتياطي المؤقته أن نقوم ببناء اسم ملف النسخة الاحتياطية –التي نعمل على بنائها– بشكل ديناميكي. ويتم ذلك عادة من خلال ربط "ختم زمني" باسم النسخة الاحتياطية المنشأة.

- يمكننا استخدام إجرائية النظام sp_addumpdevice() لإنشاء جهاز تخزين احتياطي دائم. يعود استخدام هذه الإجرائية إلى أيام الإصدار SQL Server 7.0؛

- فيما يلي عرض لمخطوط يبين الإجرائية السابقة ومعملاتها:

Sp_addumpdevice **'***devType***', '***logicalName***', '***physical_name***'**



- فيما يلي مثال يعرض إنشاء قرص مع تحديد اسمه ومساره:

Exec Sp_addumpdevice **'***disk***', '***bigDB_backup***', '***E:\myBackups\bigDB_backup_01-05-06.bak***'**

- فيما يلي مثال يعرض إنشاء شريط مع تحديد اسمه ومساره:

Exec Sp_addumpdevice **'***tape***', '***bigDB_backup_tape***', '\\.\tape0'**

- ملاحظة:

لا يتم إنشاء ملفات جهاز التخزين الاحتياطي الفيزيائية على القرص باستخدام الإجرائية sp_addumpdevice()، في حين يتم ذلك مباشرةً بعد أول مرّة يتم فيها استخدام ذلك الجهاز؛
كذلك لا يتم إسناد اسم فيزيائي لذلك الجهاز حتى يتم استخدامه للمرة الأولى.

- فيما يلي مثال يعرض المخطوط الذي يمكن استخدامه لإنشاء قرص مؤقت مع تحديد مساره:

BACKUP DATABASE *databaseName* **TO DISK = '***D:\myBackups\temp_dev.bak***'**

- يمكننا من خلال استخدام أجهزة التخزين الاحتياطي المؤقته أن نقوم ببناء اسم ملف النسخة الاحتياطية –التي نعمل على بنائها– بشكل ديناميكي. ويتم ذلك عادة من خلال ربط "ختم زمني" باسم النسخة الاحتياطية المنشأة وذلك للدلالة تماماً على الوقت الذي تمت فيه عملية التخزين الاحتياطي، وكذلك لكي نضمن عدم وَصل النسخة الجديدة المنشأة بملف موجود.
يأخذ اسم الملف الجديد المنشأ –والذي يعبر عن النسخة الاحتياطية الجديدة– الشكل التالي:

dbname_db_YYMMDDHHMM.bak

إنشاء أجهزة التخزين الاحتياطي باستخدام الأداة Enterprise Manager

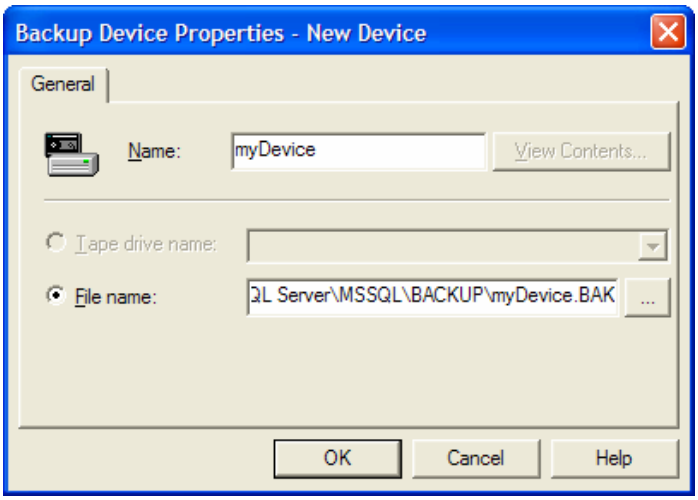
يمكننا استخدام الأداة Enterprise Manager لإنشاء أجهزة النسخ الاحتياطية من خلال واجهات تخاطبية سهلة الاستخدام؛

نستطيع القيام بالمهمة السابقة من خلال عقدة "Management" في شجرة الأداة Enterprise Manager، بعد ذلك نختار "New Backup Device" من قائمة المهمات السريعة للأيقونة "Backup" بحيث تظهر واجهة خاصة والتي نستطيع من خلالها أن نحدد الاسم المنطقي للجهاز الذي نقوم بإنشائه بالإضافة إلى المسار الفيزيائي لذلك الجهاز؛

نستطيع في الواجهة السابقة أيضاً أن نجد خَيار يسمح لنا بتحديد اسم الشريط –في حال كنا نرغب بتعريف الجهاز على أنه شريط–، مع العلم أنه لايمكن استخدام هذه الميزة ما لم يتوافر فعليّاً شريط يتّصل بالمخدّم.

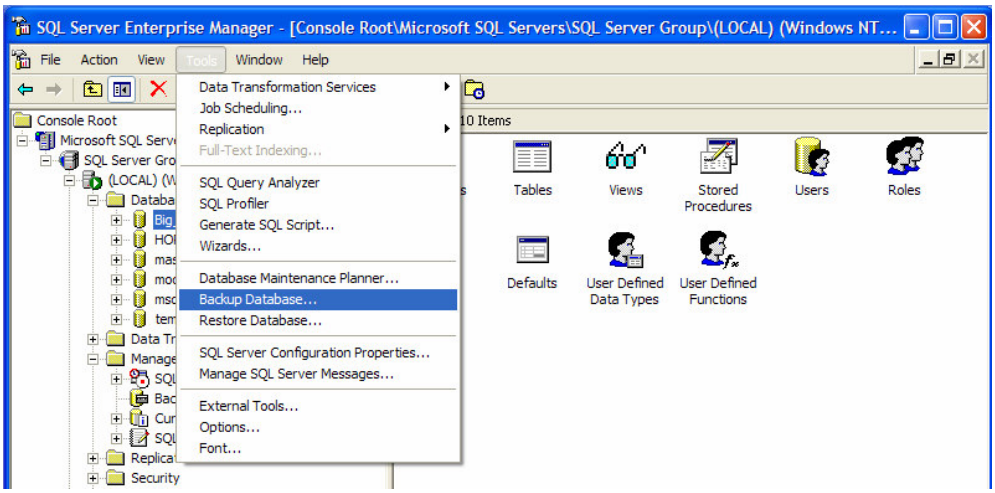
يمكننا استخدام الأداة Enterprise Manager لإنشاء أجهزة النسخ الاحتياطية من خلال واجهات تخاطبية سهلة الاستخدام؛

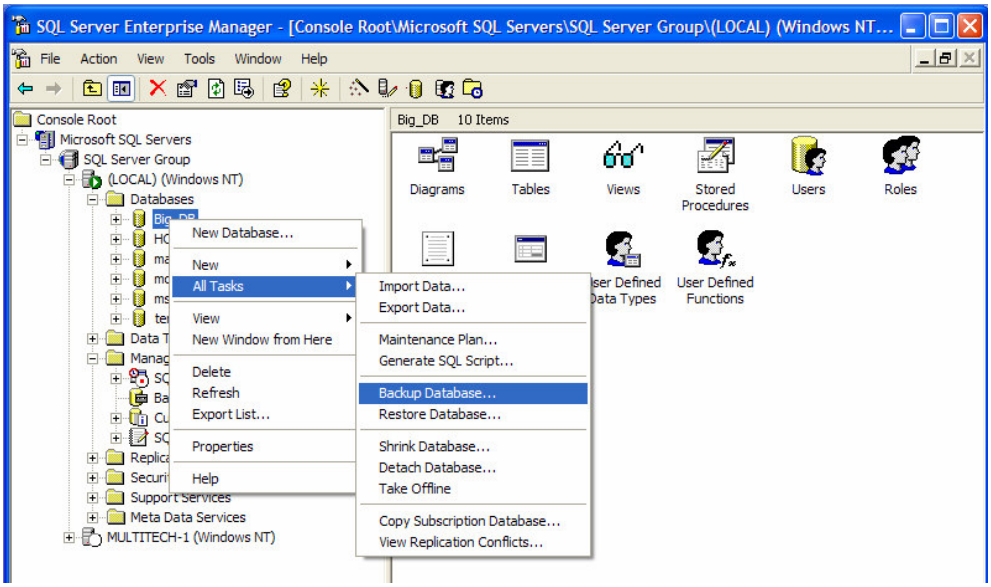
نستطيع القيام بالمهمة السابقة من خلال عقدة "Management" في شجرة الأداة Enterprise Manager، بعد ذلك نختار "New Backup Device" من قائمة المهمات السريعة للأيقونة "Backup" بحيث تظهر الواجهة الموضحة بالشكل التالي:

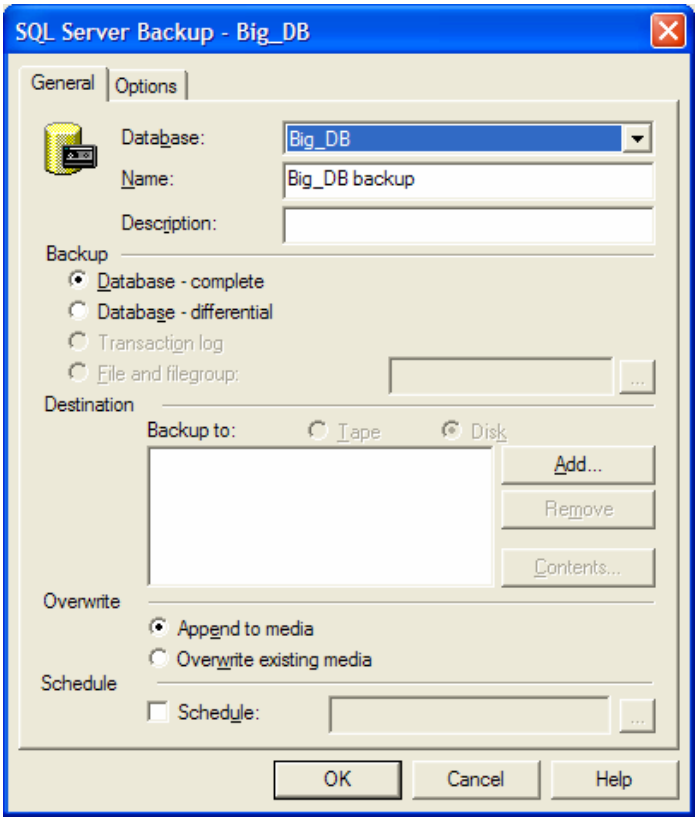


والتي نستطيع من خلالها أن نحدد الاسم المنطقي للجهاز الذي نقوم بإنشائه بالإضافة إلى المسار الفيزيائي لذلك الجهاز؛
نلاحظ في الواجهة السابقة أيضاً وجود خَيار يسمح لنا بتحديد اسم الشريط –في حال كنا نرغب بتعريف الجهاز على أنه شريط–، مع العلم أنه لايمكن استخدام هذه الميزة ما لم يتوافر فعليّاً شريط يتّصل بالمخدّم.

يمكننا أيضاً الوصول إلى الواجهة السابقة من خلال طريق آخر، بحيث يمكننا أن نختار "Backup Database" من قائمة Tools في الأداة Enterprise Manager أو يمكننا اختيار قاعدة معطيات معينة ثم نختار All Tasks من قائمة المهمات السريعة الخاصة بها ثم نختار "Backup Database"، لتظهر بعد ذلك واجهة خاصة بعمليات التخزين الاحتياطي التي يمكن تطبيقها على قواعد المعطيات؛



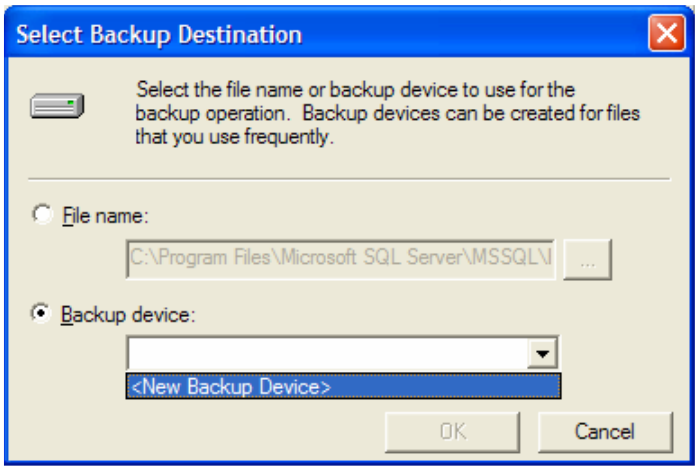


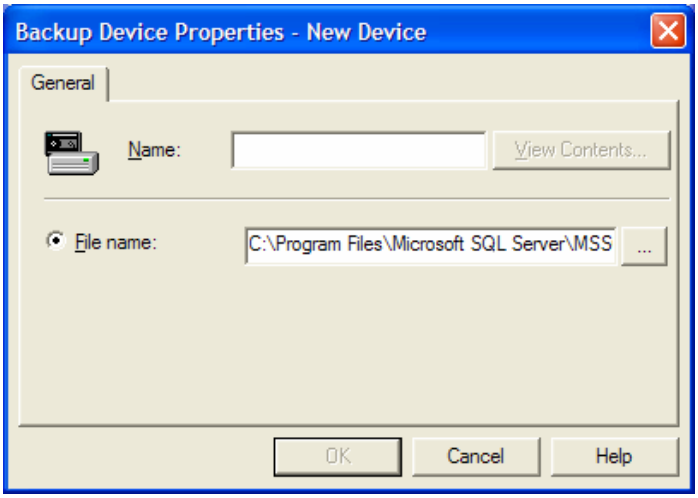


يمكننا بعد ذلك، ومن خلال واجهة SQL Server Backup أن نصل إلى الواجهة الخاصة بتعريف جهاز تخزين احتياطي جديد، وذلك من خلال الضغط على زر ADD

ثم اختيار Backup Device

ثم اختيار إضافة جهاز جديد، لتظهر الواجهة التي نبحث عنها:





إنشاء نسخة احتياطية من قاعدة المعطيات

بعد أن ناقشنا في الشرائح السابقة أنماط عمليات التخزين الاحتياطي ونماذجها بالإضافة إلى أجهزة ووسائط التخزين التي يمكن استخدامها لتلك العملية، لابد الآن أن ننتقل إلى الجزء الأهم ألا وهو عمليات التخزين الاحتياطي الفعليّة؛

نستطيع في SQL Server 2000 أن نستخدم إحدى طريقتين لإنشاء النسخ الاحتياطية، بحيث يمكننا استخدام تعليمات خاصة ومخطوطات T-SQL، أو من خلال واجهات تخاطبية تسمح لنا بإنشاء أو جدولة عمليات التخزين الاحتياطي ببساطة وسهولة؛

يمكننا كذلك أن نقوم بجدولة خطط لإجراء عمليات التخزين الاحتياطي من خلال وكيل SQL Server؛

إنشاء نسخة احتياطية من قاعدة المعطيات باستخدام T-SQL

نستطيع في SQL Server 2000 أن نستخدم مخطوطات T-SQL خاصة من أجل القيام بعملية التخزين الاحتياطي، بحيث تستخدم التعليمة BACKUP DATABASE لذلك الغرض؛

تعبّر التعليمة التالية عن عملية تخزين احتياطي لكامل قاعدة المعطيات:

BACKUP DATABASE *database_name* **TO** *bk_dev*

اسم قاعدة المعطيات التي نرغب بحفظ نسخة احتياطية من معطياتها

فيما يلي عرض لمثال تفصيلي حول المخطوط المسؤول عن القيام بإنشاء نسخة احتياطية من قاعدة المعطيات:

```
BACKUP DATABASE { database_name | @database_name_var }
[ <file_or_filegroup> [,…n] ]
TO <backup_device> [,…n]
[ WITH
[ BLOCKSIZE = { blocksize | @blocksize_variable } ]
[ [ , ] DESCRIPTION = { 'text' | @text_variable } ]
[ [ , ] DIFFERENTIAL ]
[ [ , ] EXPIREDATE={date | @date_var } | RETAINDAYS ={days | @days_var }]
[ [ , ] PASSWORD ={ password | @password_variable } ]
[ [ , ] FORMAT | NOFORMAT ]
[ [ , ] { INIT | NOINIT } ]
[ [ , ] MEDIADESCRIPTION = { 'text' | @text_variable } ]
[ [ , ] MEDIANAME= { media_name | @media_name_variable } ]
[ [ , ] MEDIAPASSWORD = { media_password | @media_password_variable } ]
[ [ , ] NAME= { backup_set_name | @ backup_set _name_variable } ]
[ [ , ] { NOSKIP | SKIP } ]
[ [ , ] { NOREWIND | REWIND } ]
[ [ , ] { NOUNLOAD | UNLOAD } ]
[ [ , ] RESTART]
[ [ , ] STATS [ = percentage ] ]
]
}
< backup_device > ::=
{
{ logical_backup_device_name | @ logical_backup_device_name_var }
|
{ DISK | TAPE } =
{ 'physical_backup_device_name' | @ physical_backup_device_name_var }
}
< file_or_filegroup> ::=

FILE = { logical_file_name | @ logical_file_name_var }
|
FILEGROUP = { logical_filegroup_name | @ logical_filegroup_name_var }
```

الشرح المرافق:

المُعامل	الوصف
BLOCKSIZE	الحجم الفيزيائي لكتلة التخزين(بايت)، مع العلم أن هذا خَيار أن SQL Server يختار قيمة تلقائية تناسب الجهاز الذي يُستخدم لعملية التخزين الاحتياطي.
DESCRIPTION	عبارة عن وصف نصي لمجموعة النسخة الاحتياطية.
DIFFERENTIAL	للتعبير عن أن عملية التخزين الاحتياطي التي يتم إنشاؤها هي عملية تخزين احتياطي جزئية.
EXPIREDATE	التاريخ الذي تنتهي عنده صلاحية النسخة الاحتياطية ويُسمح عنده بالكتابة فوق تلك النسخة.
RETAINDAYS	هو خَيار مماثل لـــ expiredate السابق ولكنه يختلف عنه بحيث يتم تحديد عدد أيام الصلاحية عوضاً عن التاريخ.
PASSWORD	يمثّل كلمة مرور للنسخة الاحتياطية، بحيث ينبغي تزويد كلمة المرور تلك عند القيام بعملية استرجاع تلك النسخة الاحتياطية.
FORMAT | NOFORMAT	يتم إعادة كتابة جهاز التخزين الاحتياطي بالإضافة إلى ترويسة وسيط وخِّي الحذر عند التعامل مع هذا الخَيار إذ أنه يؤدي إلى ضياع محتويات التخزين المحددة ككل، وبتفعيل هذا الخَيار يتم تلقائياً تفعيل كلاً من skip و عنها فيما يلي، مع العلم أن الخاصّة format لا تقوم بإعادة كتابة ترويسة جهاز التخزين الاحتياطي ما لم يتم تعريف الخاصة INIT أيضاً.
INIT | NOINIT	تستخدم noinit للدلاله على أنه سيتم وَصل النسخة الجديدة في جهاز التخزين الاحتياطي المحدد، وتستخدم init للدلالة على القيام بإعادة كتابة النسخة الجديدة.
MEDIADESCRIPTION	نصّي لمجموعة وسائط التخزين بالكامل.
MEDIANAME	عبارة عن اسم منطقي لمجموعة وسائط التخزين المنشأة، مع العلم أنه تم استخدام هذا الاسم أثناء القيام بعملية استرجاع النسخة.
MEDIAPASSWORD	يمثّل كلمة مرور لمجموعة وسائط التخزين بالكامل، بحيث ينبغي تزويد كلمة المرور تلك عند القيام بعملية استرجاع تلك المجموعة.
NAME	اسم مجموعة التخزين الاحتياطي المنشأة.
NOSKIP | SKIP	ينبغي قراءة عناوين الشرائط، أو اهمال العملية، مع العلم أن noskip هو الخيار التلقائي الذي يجبر بإجراء عملية القراءة تلك.

NOREWIND | REWIND	ل ينبغي إعادة (أو عكس) الشريط أم لا.
NOUNLOAD | UNLOAD	ل ينبغي فكّ عملية تحميل الشريط أم لا.
RESTART	يستخدَم هذا الخَيار لإعادة تشغيل عملية تخزين احتياطي متوقّفة اعتباراً من نقطة التوقف، وذلك بغرض اختصار الوقت.
STATS	عبارة عن نسبة تستخدم لقياس تقدّم عملية النسخ الاحتياطي، ة التلقائية تساوي 10%.

ملاحظة:

يمكن استخدام التعليمة DUMP عوضاً عن التعليمة BACKUP، بحيث يتيح SQL Server 2000 توافقية مع النسخ القديمة، إذ كانت هذه التعليمة هي المسؤولة عن إنشاء النسخة الاحتياطية من قاعدة المعطيات في SQL Server 7.0؛

فيما يلي عرض لمثال تفصيلي حول المخطوط المسؤول عن القيام بإنشاء نسخة احتياطية من قاعدة المعطيات:

```
BACKUP DATABASE { database_name | @database_name_var }
[ <file_or_filegroup> [,…n] ]
TO <backup_device> [,…n]
[ WITH
[ BLOCKSIZE = { blocksize | @blocksize_variable } ]
[ [ , ] DESCRIPTION = { 'text' | @text_variable } ]
[ [ , ] DIFFERENTIAL ]
[ [ , ] EXPIREDATE={date | @date_var } | RETAINDAYS ={days | @days_var }]
[ [ , ] PASSWORD ={ password | @password_variable } ]
[ [ , ] FORMAT | NOFORMAT ]
[ [ , ] { INIT | NOINIT } ]
[ [ , ] MEDIADESCRIPTION = { 'text' | @text_variable } ]
[ [ , ] MEDIANAME= { media_name | @media_name_variable } ]
[ [ , ] MEDIAPASSWORD = { media_password | @media_password_variable } ]
[ [ , ] NAME= { backup_set_name | @ backup_set _name_variable } ]
[ [ , ] { NOSKIP | SKIP } ]
[ [ , ] { NOREWIND | REWIND } ]
[ [ , ] { NOUNLOAD | UNLOAD } ]
[ [ , ] RESTART]
[ [ , ] STATS [ = percentage ] ]
]
}
< backup_device > ::=
{
```

{ *logical_backup_device_name* | @ *logical_backup_device_name_var* }
|
{ DISK | TAPE } =
{ '*physical_backup_device_name*' | @ *physical_backup_device_name_var* }
}
< file_or_filegroup> ::=
{
FILE = { *logical_file_name* | @ *logical_file_name_var* }
|
FILEGROUP = { *logical_filegroup_name* | @ *logical_filegroup_name_var* }
}

الشرح المرافق:

المُعامل	الوصف
BLOCKSIZE	الحجم الفيزيائي لكتلة التخزين(بايت)، مع العلم أن هذا خَيار غير إجباري وأن SQL Server يختار قيمة تلقائية تناسب الجهاز الذي يُستخدم لعملية التخزين الاحتياطي.
DESCRIPTION	عبارة عن وصف نصي لمجموعة النسخة الاحتياطية.
DIFFERENTIAL	للتعبير عن أن عملية التخزين الاحتياطي التي يتم إنشاؤها هي عملية تخزين احتياطي جزئية.
EXPIREDATE	ديد التاريخ الذي تنتهي عنده صلاحية النسخة الاحتياطية ويُسمح عنده بالكتابة فوق تلك النسخة.
RETAINDAYS	وهو خَيار مماثل لـــ expiredate السابق ولكنه يختلف عنه بحيث يتم تحديد عدد أيام الصلاحية عوضاً عن التاريخ.
PASSWORD	يمثّل كلمة مرور للنسخة الاحتياطية، بحيث ينبغي تزويد كلمة المرور تلك عند القيام بعملية استرجاع تلك النسخة الاحتياطية.
FORMAT | NOFORMAT	عيل هذا الخَيار يتم إعادة كتابة جهاز التخزين الاحتياطي بالإضافة إلى ترويسة وسيط التخزين، ينبغي توخّي الحذر عند التعامل مع هذا الخَيار إذ أنه يؤدي إلى ضياع محتويات مجموعة وسائط التخزين المحددة ككل، وبتفعيل هذا الخَيار يتم تلقائياً تفعيل كلاً من skip و init التي سنتحدث عنها فيما يلي، مع العلم أن الخاصّة format لا تقوم بإعادة كتابة ترويسة وسيط التخزين ولا جهاز التخزين الاحتياطي ما لم يتم تعريف الخاصة INIT أيضاً.
INIT | NOINIT	تخدم noinit للدلاله على أنه سيتم وَصل النسخة الجديدة في جهاز التخزين الاحتياطي المحدد، وتستخدم init للدلالة على القيام بإعادة كتابة النسخة الجديدة.
MEDIADESCRIPTION	عبارة عن وصف نصّي لمجموعة وسائط التخزين بالكامل.
MEDIANAME	عبارة عن اسم منطقي لمجموعة وسائط التخزين المنشأة، مع العلم أنه سيتم استخدام هذا الاسم أثناء القيام بعملية استرجاع النسخة.

MEDIAPASSWORD	يمثّل كلمة مرور لمجموعة وسائط التخزين بالكامل، بحيث ينبغي تزويد كلمة المرور تلك عند القيام بعملية استرجاع تلك المجموعة.
NAME	اسم مجموعة التخزين الاحتياطي المنشأة.
NOSKIP | SKIP	لتحديد إذا ما كان ينبغي قراءة عناوين الشرائط، أو اهمال العملية، مع العلم أن noskip هو الخيار التلقائي الذي يجبر بإجراء عملية القراءة تلك.
NOREWIND | REWIND	، إعادة (أو عكس) الشريط أم لا.
NOUNLOAD | UNLOAD	، فكّ عملية تحميل الشريط أم لا.
RESTART	يستخدَم هذا الخَيار لإعادة تشغيل عملية تخزين احتياطي متوقّفة اعتباراً من نقطة التوقف، وذلك بغرض اختصار الوقت.
STATS	عبارة عن نسبة تستخدم لقياس تقدّم عملية النسخ الاحتياطي، مع العلم أن القيمة التلقائية تساوي 10%.

أمثلة:

مثال1:

عملية تخزين احتياطي لكامل قاعدة المعطيات big_db:

```
USE MASTER
BACKUP DATABASE big_DB TO backup_dev
WITH
NOUNLOAD,
NAME= 'BIG_DB full database backup'
DECRIPTION = 'A full database backup for BIG_DB '
```

مثال2:

عملية تخزين احتياطي جزئية لقاعدة المعطيات big_db :

```
USE MASTER
BACKUP DATABASE big_DB TO DISK 'E:\backup\myBackup.bak'
WITH DIFFERENTIAL
```

مثال3:

عملية تخزين احتياطي كاملة لقاعدة المعطيات big_db على عدّة أجهزة تخزين:

```
USE MASTER
BACKUP DATABASE big_DB
ev1, backup_dev2, backup_dev3, backup_dev4
WITH
```

NAME= 'BIG_DB full backup'
DECRIPTION = 'A full database backup for BIG_DB striped to four devices '
FORMAT,
MEDIANAME = '*big_db_stip*'

خاصة من أجل القيام بعمليـــة التخـــزين T-SQL أن نستخدم مخطوطات SQL Server 2000نستطيع في BACKUP DATABASEالاحتياطي، بحيث تستخدم التعليمة لذلك الغرض؛

تعبّر التعليمة التالية عن عملية تخزين احتياطي لكامل قاعدة المعطيات:

BACKUP DATABASE *database_name* **TO** *bk_dev*

اسم قاعدة المعطيات التي نرغب
بحفظ نسخة احتياطية من معطياتها

إنشاء نسخة احتياطية من قاعدة المعطيات باستخدام الأداة Enterprise Manager
-باستخدام Backup Wizard-

تعمل الأداة Enterprise Manager ببساطة على توليد المخطوطات اللازمة لإجراء عملية التخزين الاحتياطي السابقة ولكن من خلال واجهات تخاطبية يقوم من خلالها المستخدم أو مدير قاعدة المعطيات بإدخال كافة الخيارات التي يرغب بتوصيفها؛
تتميز هذه الطريقة -بغض النظر عن سهولة استخدام الواجهات التخاطبية مقارنةً مع كتابة المخطوطات- بإمكانية إعداد أعمال للتنفيذ، ويقصد بذلك إمكانية تنفيذ عملية التخزين الاحتياطي مباشرةً أو جدولة ذلك العمل بحيث يتم تنفيذه في وقت محدد؛

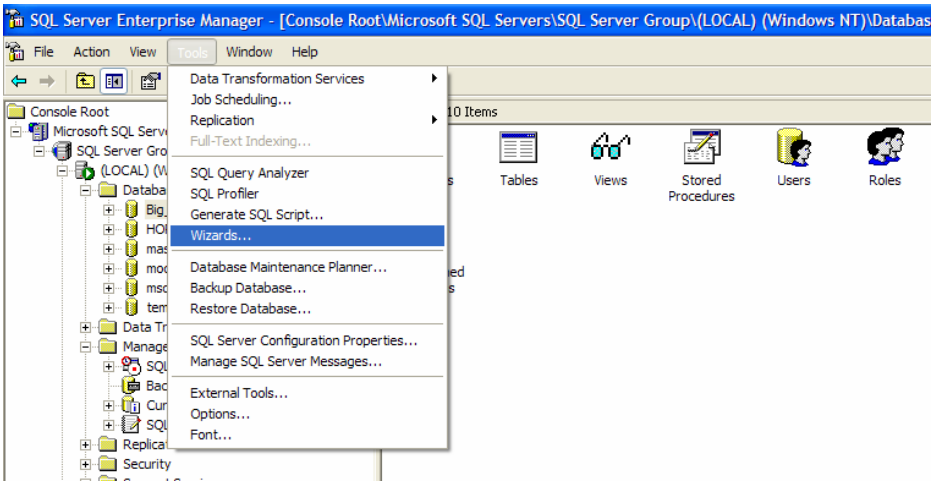
نستطيع -كما نعلم- أن نستخدم الأداة Enterprise Manager للقيام بعملية التخزين الاحتياطي بعدّة أساليب وطُرق، بحيث يمكننا مثلاً أن نصل إلى الواجهات الخاصة بعملية التخزين الاحتياطي تلك من خلال استخدام لوحة المهمات Taskpad View، وبالمرور فوق سهم Maintenance يمكننا أن نختار 'Backup Database' من القائمة المتولّدة؛

كما يمكننا اختيار إنشاء النسخ الاحتياطية اعتماداً على Wizard خاصة من بين المجموعات المتاحة والتي يمكن الوصول إليها من خلال القائمة Tools، وذلك باختيار Backup Wizard من قائمة Management، ثم التنقل بين الواجهات المتاحة والتي تسمح بتحديد كافة خصائص عملية التخزين الاحتياطي.

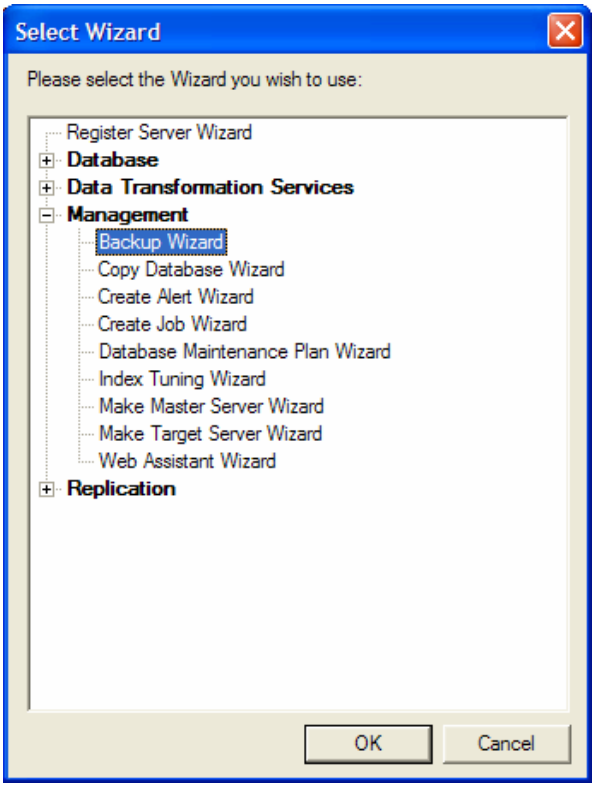
تعمل الأداة Enterprise Manager ببساطة على توليد المخطوطات اللازمة لإجراء عملية التخزين الاحتياطي السابقة ولكن من خلال واجهات تخاطبية يقوم من خلالها المستخدم أو مدير قاعدة المعطيات بإدخال كافة الخيارات التي يرغب بتوصيفها؛
تتميز هذه الطريقة -بغض النظر عن سهولة استخدام الواجهات التخاطبية مقارنةً مع كتابة المخطوطات- بإمكانية إعداد أعمال للتنفيذ، ويقصد بذلك إمكانية تنفيذ عملية التخزين الاحتياطي مباشرةً أو جدولة ذلك العمل بحيث يتم تنفيذه في وقت محدد؛

نستطيع -كما نعلم- أن نستخدم الأداة Enterprise Manager للقيام بعملية التخزين الاحتياطي بعدّة أساليب وطُرق، بحيث يمكننا مثلاً أن نصل إلى الواجهات الخاصة بعملية التخزين الاحتياطي تلك من خلال استخدام لوحة المهمات Taskpad View، وبالمرور فوق سهم Maintenance يمكننا أن نختار 'Backup Database' من القائمة المتولّدة؛

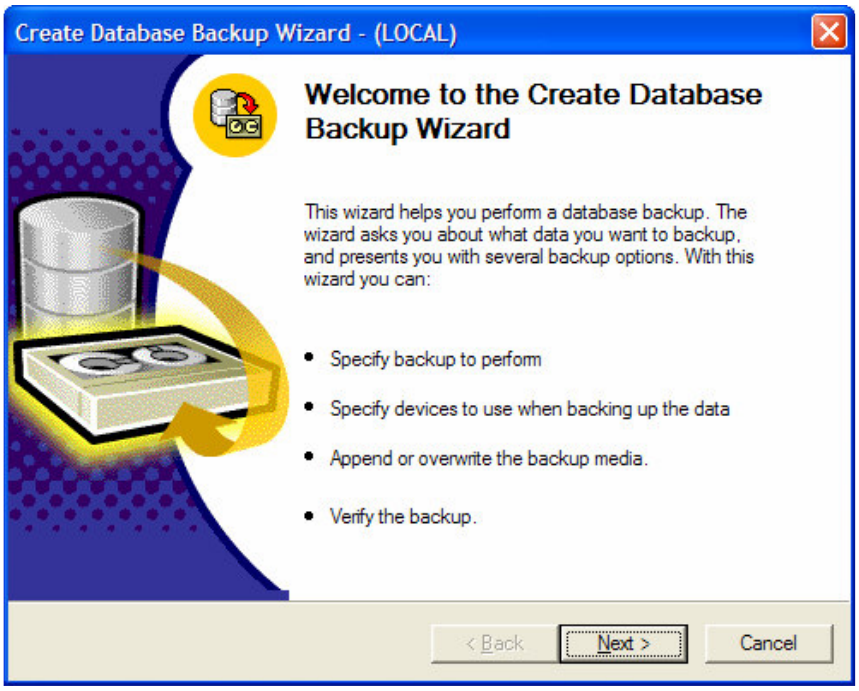
كما يمكننا اختيار إنشاء النسخ الاحتياطية اعتماداً على Wizard خاصة بذلك:

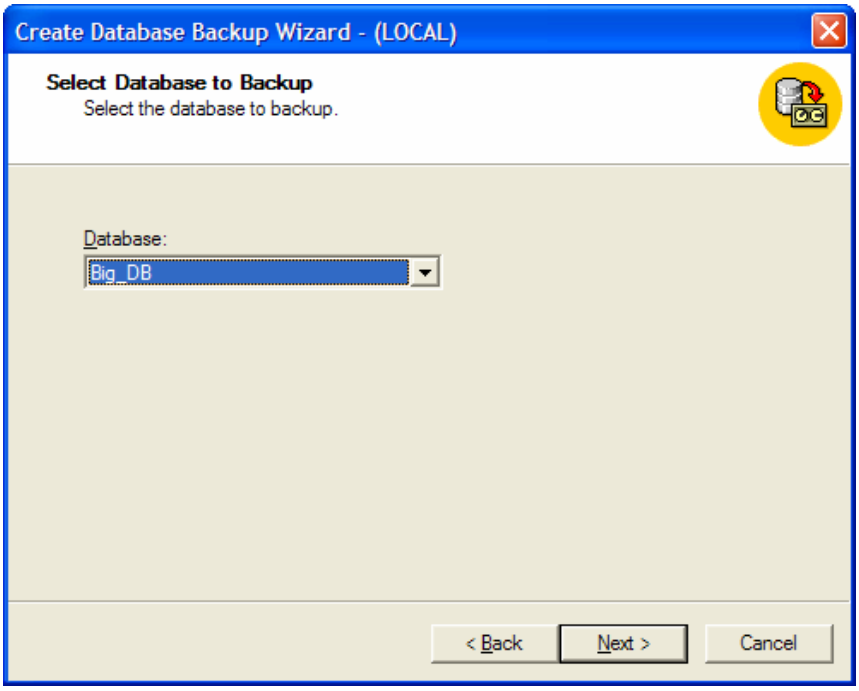


وذلك باختيار Backup Wizard من قائمة Management:

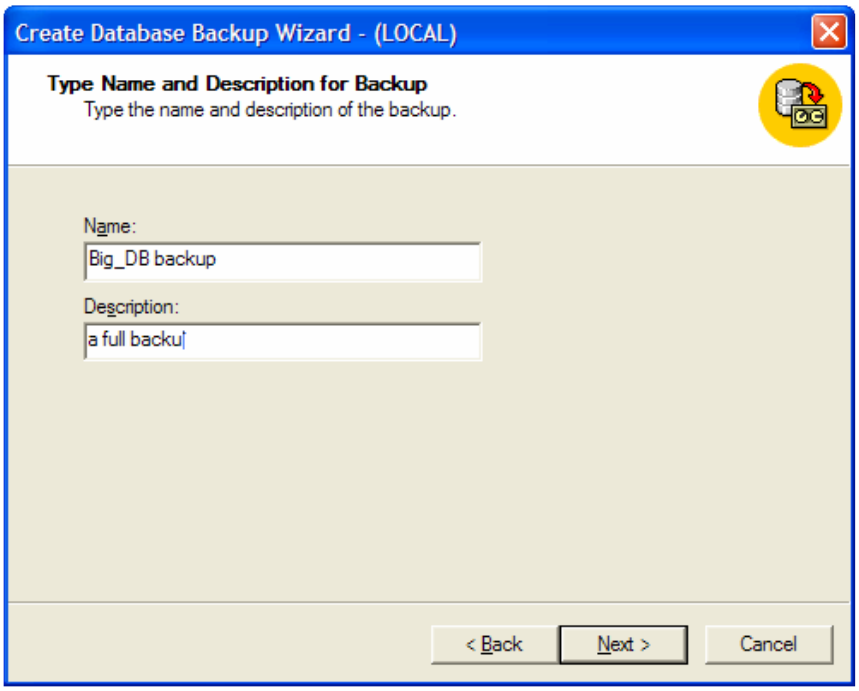


ثم التنقل بين الواجهات المتاحة:

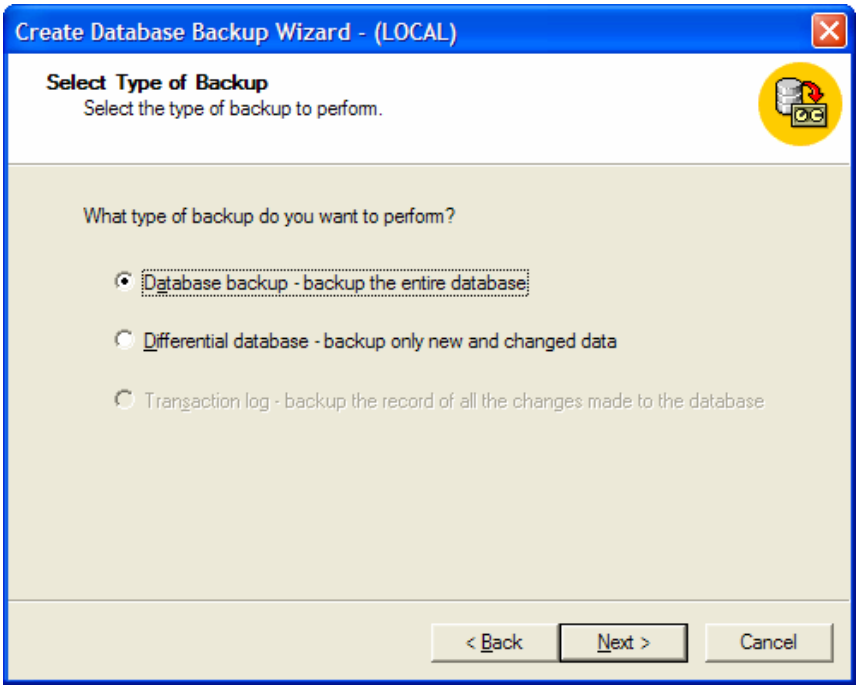




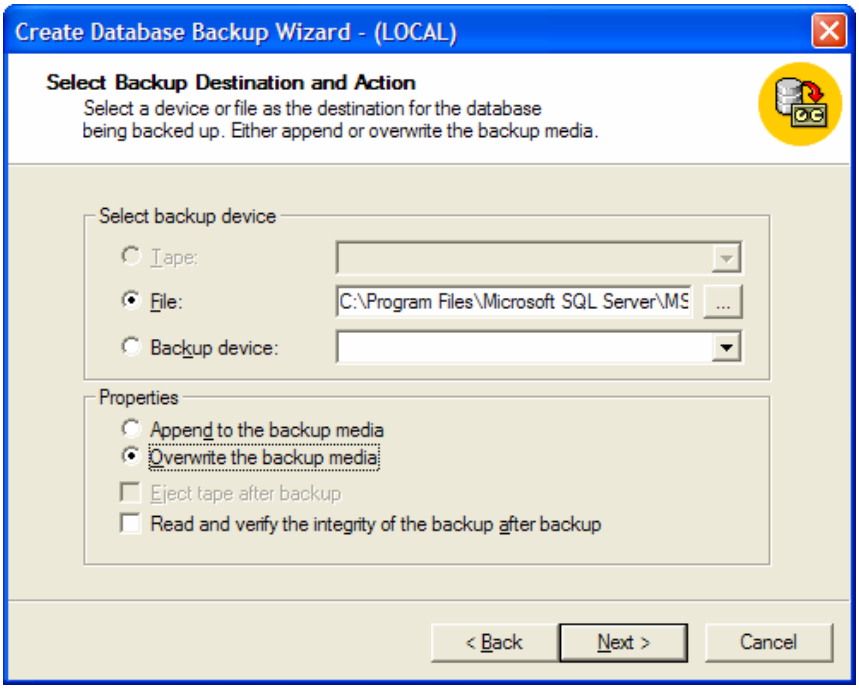
<واجهة اختيار قاعدة المعطيات التي نرغب بحفظ نسخة احتياطية منها>



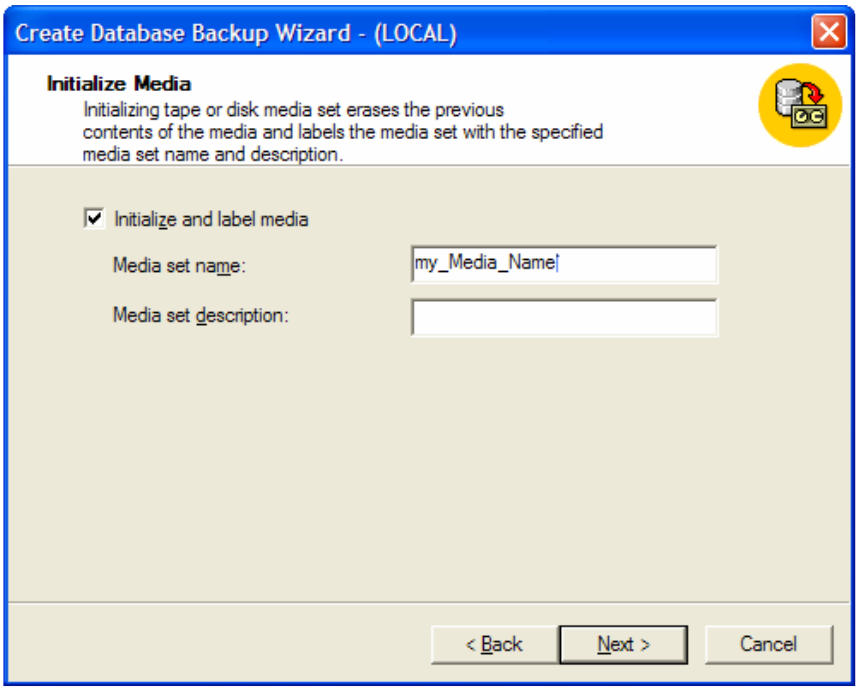
<واجهة تحديد اسم النسخة الاحتياطية والوصف>



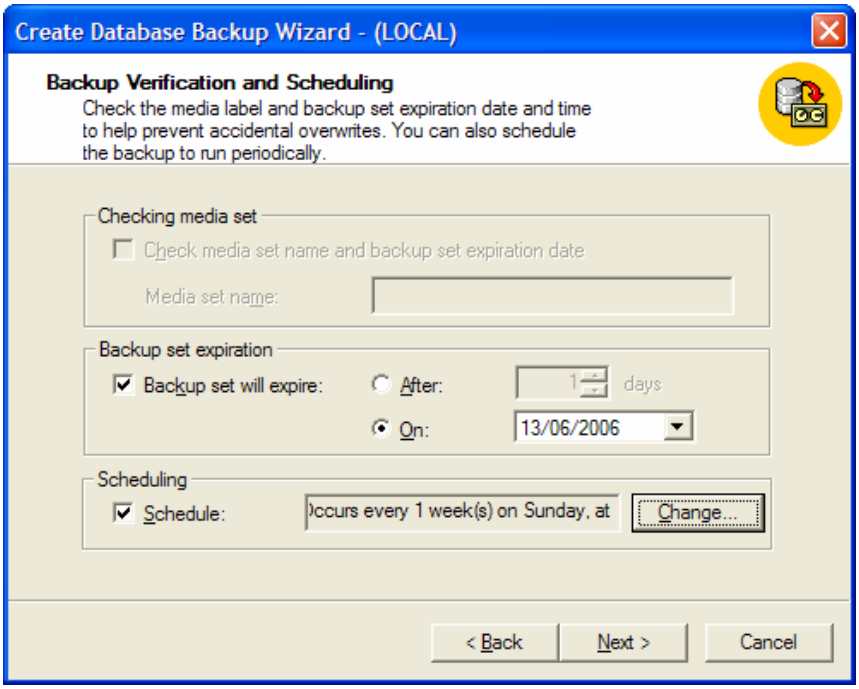
< واجهة تحديد نوع النسخة الاحتياطية



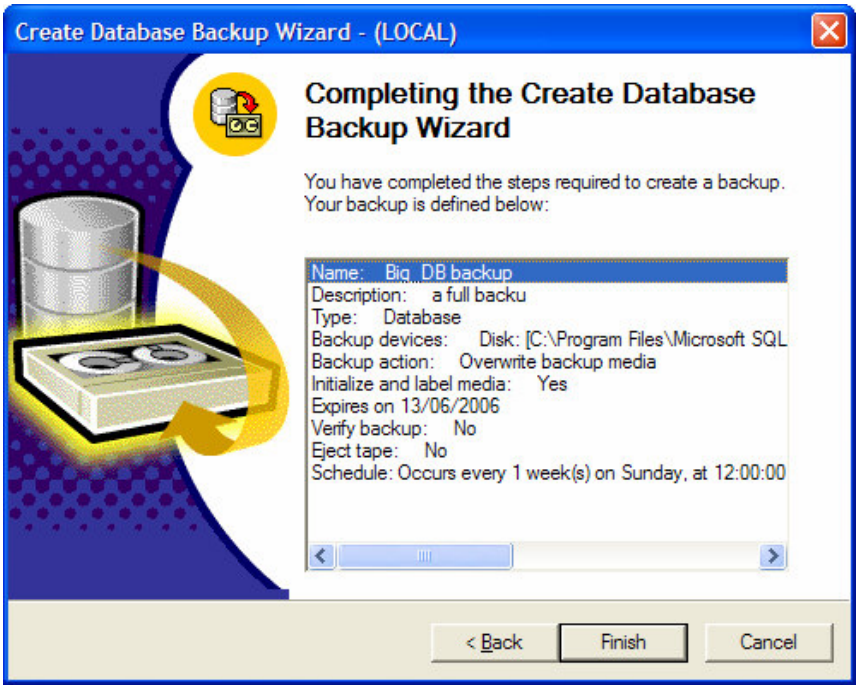
<حواجهة تحديد جهاز التخزين الاحتياطي و خصائص النسخة الاحتياطية >



<حواجهة تحديد اسم ووصف مجموعة وسائط التخزين المستخدمة>



<واجهة تحديد صلاحية النسخة الاحتياطية >



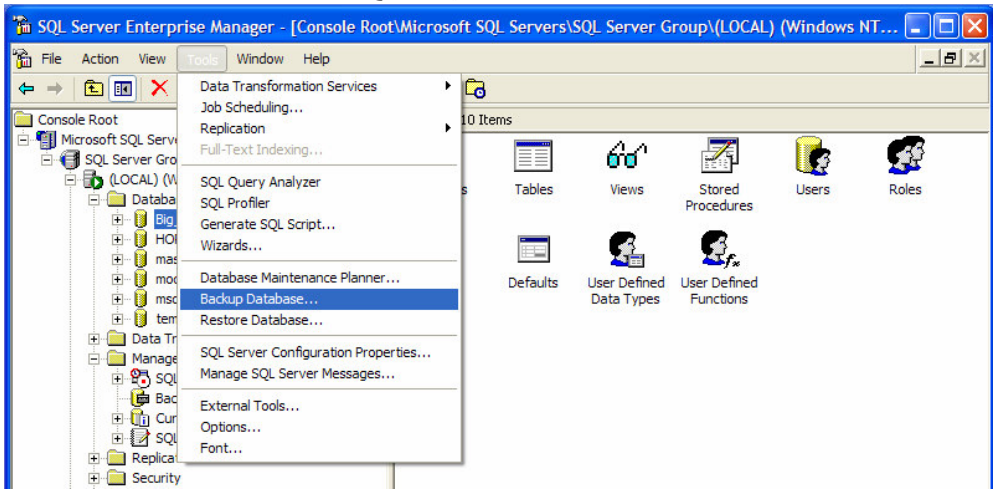
إنشاء نسخة احتياطية من قاعدة المعطيات باستخدام الأداة Enterprise Manager

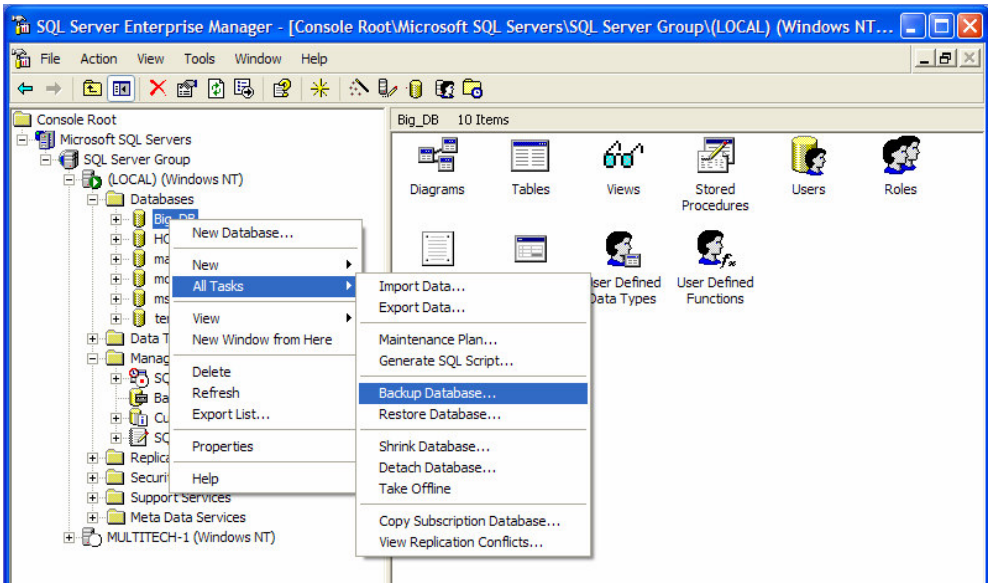
-باستخدام واجهة SQL Server Backup-

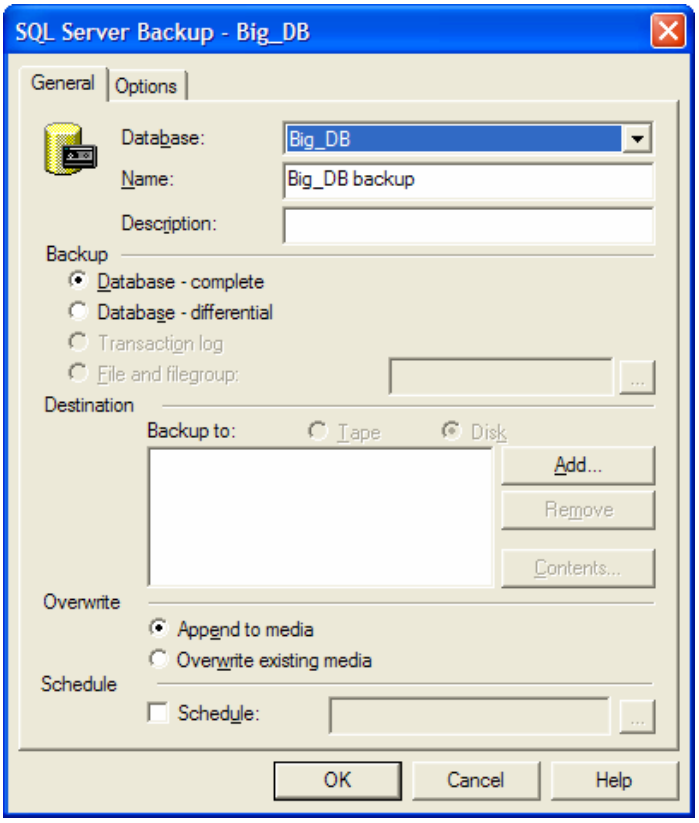
- يمكننا الوصول إلى واجهة SQL Server Backup الخاصة بعملية التخزين الاحتياطي من خلال قائمة الأدوات أو من خلال قائمة المهمات السريعة لقاعدة المعطيات نفسها كما مرّ معنا في الشرائح السابقة.

- نستطيع من خلال هذه الواجهة أن نقوم بعملية التخزين الاحتياطي لقاعدة معطيات يتم تحديدها، بالإضافة إلى أن هذه الواجهة تدعم إمكانية توصيف كافة الخصائص التي تتعلق بهذه العملية؛

يمكننا الوصول إلى واجهة SQL Server Backup الخاصة بعملية التخزين الاحتياطي من خلال قائمة الأدوات أو من خلال قائمة المهمات السريعة لقاعدة المعطيات نفسها كما مرّ معنا في الشرائح السابقة:

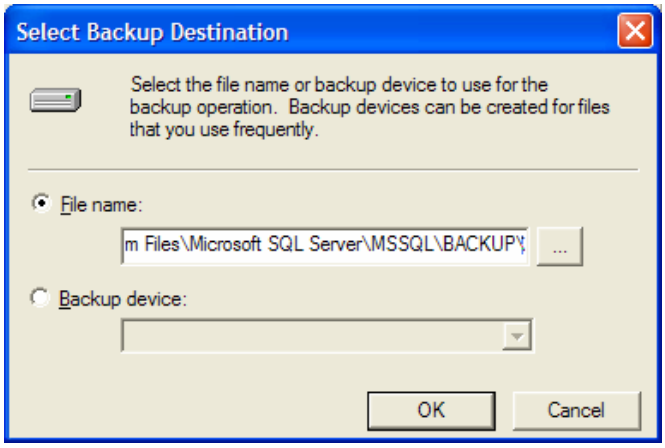



SQL Server Enterprise Manager - [Console Root\Microsoft SQL Servers\SQL Server Group\(LOCAL) (Windows NT...
File Action View Tools Window Help
Console Root
Microsoft SQL Servers
SQL Server Group
(LOCAL) (Windows NT)
Databases
Big_DB 10 Items
Diagrams
Tables
Views
Stored Procedures
Users
Roles
User Defined Functions
New Database...
New
All Tasks
View
New Window from Here
Delete
Refresh
Export List...
Properties
Help
Import Data...
Export Data...
Maintenance Plan...
Generate SQL Script...
Backup Database...
Restore Database...
Shrink Database...
Detach Database...
Take Offline
Copy Subscription Database...
View Replication Conflicts...
Support Services
Meta Data Services
MULTITECH-1 (Windows NT)


SQL Server Backup - Big_DB
General
Options
Database:
Big_DB
Name:
Big_DB backup
Description:
Backup
Database - complete
Database - differential
Transaction log
File and filegroup:
Destination
Backup to:
Tape
Disk
Add...
Remove
Contents...
Overwrite
Append to media
Overwrite existing media
Schedule
Schedule:
OK
Cancel
Help

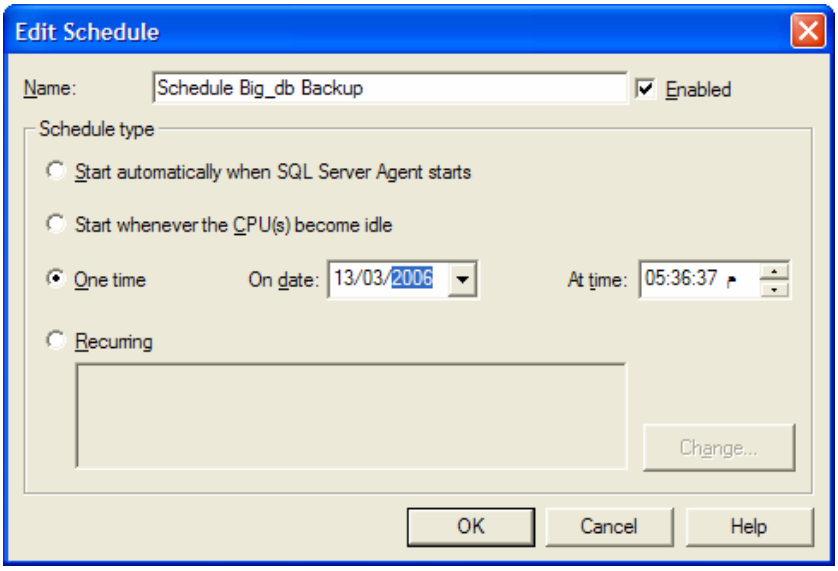
نستطيع من خلال هذه الواجهة أن نقوم بعملية التخزين الاحتياطي لقاعدة معطيات يتم تحديدها، بالإضافة إلى أن هذه الواجهة تدعم إمكانية توصيف كافة الخصائص التي تتعلق بهذه العملية، كما يمكننا أيضاً أن نعمل من خلالها على تعريف جهاز تخزين احتياطي –كما مر معنا في شرائح سابقة–؛

يمكننا بالضغط على زر ADD في هذه الواجهة أن نحدد المسار الفيزيائي الذي نرغب بتخزين النسخة الاحتياطية فيه:

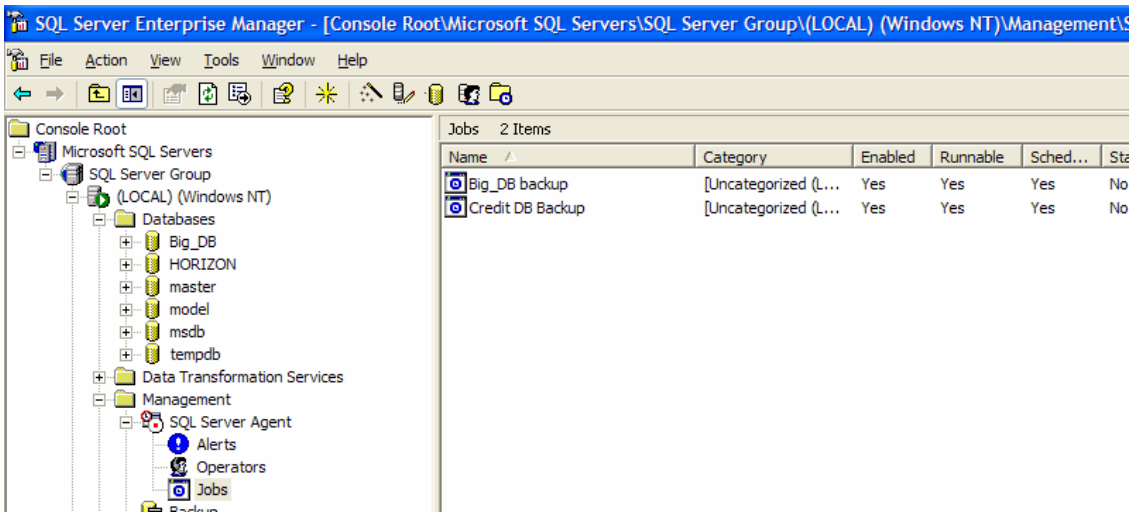


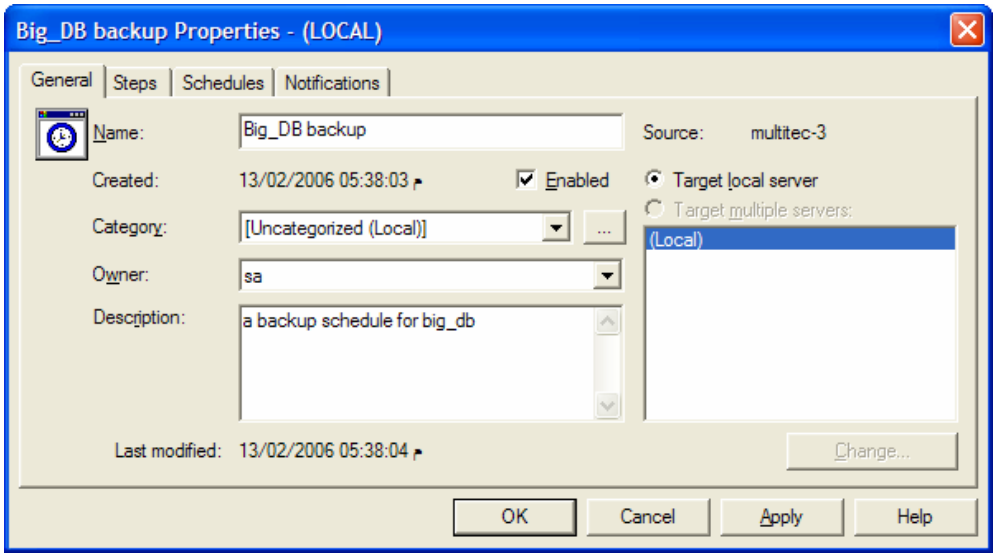
مع العلم أن وسيط التخزين التلقائي المستخدم هو القرص، وأنه لا يمكن تغييره ما لم يتم تعريف شريط على المخدِّم، بالتالي لا يمكننا تغيير إعدادت كل ما يتعلق بالشرائط في الواجهة؛

يمكننا أيضاً أن نقوم بجدولة تلك المهمة من خلال تحديد التاريخ الذي نريد تنفيذ عملية التخزين الاحتياطي فيه:



نستطيع مراقبة المهمة –أو المهمات– التي قمنا بإعدادها من خلال واجهة "Jobs" التي يمكن الوصول إليها من خلال وكيل SQL Server ضمن عقدة 'Management' في شجرة Enterprise Manager، بالإضافة إلى أنه يمكننا تعديل تلك المهمات المجدولة وتغيير خصائصها من خلال تلك الواجهة:





إنشاء نسخة احتياطية من ملف سجلّ المناقلات باستخدام T-SQL

- بما أن SQL Server يقوم بكتابة كافة المناقلات التي يتم تنفيذها إلى ملف سجلّ المناقلات، لذلك تعد عملية إجراء تخزين احتياطي لهذا الملف عملية هامة جداً؛

- لا تنحصر الغاية من إجراء نسخة احتياطية من ملف سجلّ المناقلات في إدارة عملية التعافي فقط، بحيث أنّ إجراء تلك العملية يؤدي إلى اقتصاص جزء السجلّ –الذي تمّ تخزين نسخة احتياطيّة منه– من السجلّ الأصلي، مما يؤدي بالضرورة إلى تحسين الأداء بشكل عام من خلال إتاحة إمكانية تخزين مناقلات جديدة في ذلك السجلّ مكان المناقلات التي تم حذفها منه؛

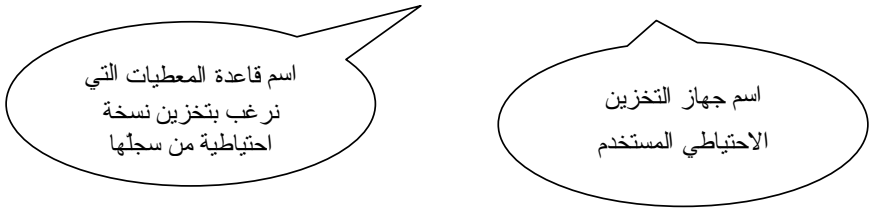
ملاحظة:

يقوم SQL Server من خلال اتّباع استراتيجية التعافي البسيط باقتصاص جزء السجلّ –الذي يحتوي على مناقلات قديمة– من دون القيام بحفظ نسخة احتياطية منه، بالتالي لايمكن استرجاع المعطيات من خلال تنفيذ السجلّ في هذه الحالة؛

- يمكننا القيام بعملية التخزين الاحتياطي لملف سجلّ المناقلات بعدّة طرق، بحيث يمكننا أن نستخدم تعليمات مكتوبة بلغة T-SQL للقيام بهذه العملية؛

- لا تختلف تعليمات T-SQL المستخدمة لحفظ نسخة احتياطية من ملف سجلّ المناقلات كثيراً عن تلك التعليمات المستخدمة لحفظ نسخة احتياطية من قاعدة المعطيات نفسها، وفيما يلي المخطوط الأولي الذي يمكن استخدامه للتعبير عن هذه العملية:

BACKUP LOG *my_database* **TO** *my_device*



- سنناقش فيما يلي كافة الخصائص التي يمكن استخدامها مع تعليمة التخزين الاحتياطي لملف سجلّ المناقلات:

BACKUP LOG { *database_name* | @*database_name_var* }
TO <backup_device> [,…n]
[**WITH**
[{ **NO_LOG** | **TRUNCATE_ONLY** }]
[[,] **NO_TRUNCATE**]
[[,] { **NORECOVERY** | **STANDBY** = *undo_file_name* }]
]
}

الشرح المرافق

TRUNCATE_ONLY	صتين متماثلتان تماماً ولهما نفس الوظيفة، بحيث يتم عند استخدام منهما مع تعليمة backup log اقتصاص الجزء غير الفعّال من سجلّ المناقلات تنفيذ عملية التخزين الاحتياطي لذلك الملف. ينصح عادةً بإجراء عملية تخزين احتياطي لكامل قاعدة المعطيات قبل القيام بإجراء عملية تخزين احتياطي لملف سجلّ المناقلات مع تفعيل هذه الخاصة.
NO_TRUNCATE	هي الميزة المعاكسة تماماً للخاصة السابقة.
NORECOVERY و **STANDBY** = *undo_file_name*	تعتبر هاتين الميزتين جديدتان مع SQL Server 2000 بحيث كانتا جزءاً من عملية استرجاع ملف سجلّ المناقلات في الإصدار SQL Server 7.0؛ عندما يتم استخدام الخاصة norecovery فإنه يتم تخزين نسخة احتياطية من السجلّ في عادة المعطيات في وضع التعافي، وهذا ما يُستخدم عادةً في خطط التعافي في المخدمات التي تعمل بشكل مستقل؛ تشبه الخاصة standby الخاصة السابقة أيضاً، إلا أنها تترك قاعدة المعطيات متاحة في وضع القراءة فقط أثناء القيام بعملية التخزين الاحتياطي للسجلّ.

أمثلة:

مثال1:

- فيما يلي عرض للمخطوط المستخدم للقيام بعملية تخزين احتياطي لسجلّ مناقلات:

```
USE MASTER
BACKUP LOG BIG_DB TO my_device
WITH
NOINIT,
NAME = 'A LOG BACKUP FOR THE BIG_DB DATABASE',
NOFORMATE;
```

مثال2:

- فيما يلي عرض للمخطوط المستخدم للقيام بحذف محتويات سجلّ مناقلات محدد بالكامل:

```
USE MASTER
BACKUP LOG BIG_DB WITH NO_LOG
```

مثال3:

- فيما يلي عرض للمخطوط المستخدم للقيام بعملية تخزين احتياطي لسجلّ مناقلات قاعدة معطيات لا يمكن الوصول إليها بشكل مباشر:

```
USE MASTER
BACKUP LOG BIG_DB TO my_device
WITH NO_TRUNCATE
```

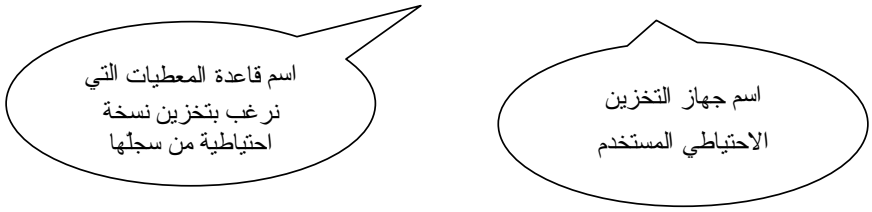
- بما أن SQL Server يقوم بكتابة كافة المناقلات التي يتم تنفيذها إلى ملف سجلّ المناقلات، لذلك تعد عملية إجراء تخزين احتياطي لهذا الملف عملية هامة جداً؛

- لا تنحصر الغاية من إجراء نسخة احتياطية من ملف سجلّ المناقلات في إدارة عملية التعافي فقط، بحيث أنّ إجراء تلك العملية يؤدي إلى اقتصاص جزء السجلّ –الذي تمّ تخزين نسخة احتياطيّة منه– من السجلّ الأصلي، مما يؤدي بالضرورة إلى تحسين الأداء بشكل عام من خلال إتاحة إمكانية تخزين مناقلات جديدة في ذلك السجلّ مكان المناقلات التي تم حذفها منه؛

- يمكننا القيام بعملية التخزين الاحتياطي لملف سجلّ المناقلات بعدّة طرق، بحيث يمكننا أن نستخدم تعليمات مكتوبة بلغة T-SQL للقيام بهذه العملية؛

- لا تختلف تعليمات T-SQL المستخدمة لحفظ نسخة احتياطية من ملف سجلّ المناقلات كثيراً عن تلك التعليمات المستخدمة لحفظ نسخة احتياطية من قاعدة المعطيات نفسها، وفيما يلي المخطوط الأولي الذي يمكن استخدامه للتعبير عن هذه العملية:

BACKUP LOG *my_database* **TO** *my_device*



إنشاء نسخة احتياطية من ملف سجلّ المناقلات باستخدام الأداة Enterprise Manager

يمكننا استخدام نفس الواجهات –التي تم استخدامها لإنشاء نسخة احتياطية من قاعدة المعطيات– من أجل القيام بإنشاء نسخة احتياطية من ملف سجلّ المناقلات باستخدام الأداة Enterprise Manager؛

ينبغي أن نعيّن على خَيار"Transaction Log" ثم نحدد جهاز التخزين الاحتياطي الذي نرغب باستخدامه، بالإضافة إلى أنه يمكننا أن نجدول تلك المهمة لوقت لاحق من خلال الواجهة المخصصة لعملية الجدولة تلك؛

ينبغي الانتباه إلى خاصة "Append to media" المُختارة بشكل تلقائي، والتي تعمل على إضافة النسخ الاحتياطية الجديدة إلى النسخ الاحتياطية القديمة التي تم إنشاؤها مسبقاً؛

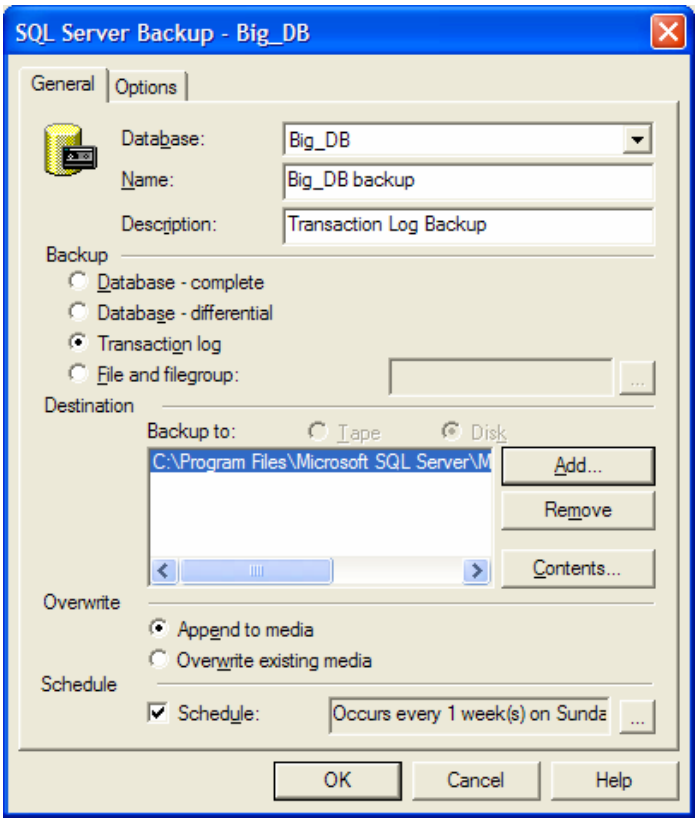
ينبغي أيضا أن نأخذ الواجهة الفرعية Options –والموجودة ضمن الواجهة السابقة– بعين الاعتبار بحيث نلاحظ فيها أنه قد تمّ اختيار إزالة المناقلات غير الفعّالة من سجلّ المناقلات بعد القيام بإجراء عملية التخزين الاحتياطي له بشكل تلقائي، مع العلم أنّ إزالة ذلك الخَيار يماثل كتابة NO_TRUNCATE؛

ملاحظة:

إذا لم نستطع القيام بعملية التخزين الاحتياطي لملف سجلّ المناقلات من خلال الواجهة السابقة لأن خَيار"Transaction Log" لم يكن فعّالاً، فيعود ذلك إلى أن استراتيجية التعافي المستخدمة هي التعافي البسيط بالتالي ينبغي تحويل أسلوب التعافي المستخدم إلى نوع آخر.

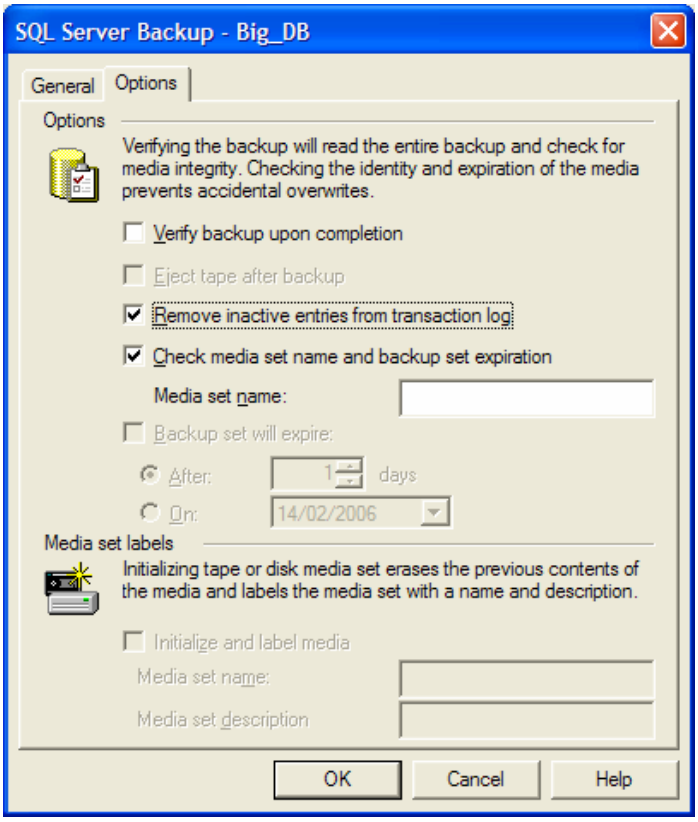
يمكننا استخدام نفس الواجهات –التي تم استخدامها لإنشاء نسخة احتياطية من قاعدة المعطيات– من أجل القيام بإنشاء نسخة احتياطية من ملف سجلّ المناقلات باستخدام الأداة Enterprise Manager؛

ينبغي أن نعيّن على خَيار"Transaction Log" ثم نحدد جهاز التخزين الاحتياطي الذي نرغب باستخدامه، بالإضافة إلى أنه يمكننا أن نجدول تلك المهمة لوقت لاحق من خلال الواجهة المخصصة لعملية الجدولة تلك؛



ينبغي الانتباه إلى خاصة "Append to media" المُختارة بشكل تلقائي، والتي تعمل على إضافة النسخ الاحتياطية الجديدة إلى النسخ الاحتياطية القديمة التي تم إنشاؤها مسبقاً، بحيث لابد لنا من الانتباه ألا نقوم بتغيير هذا الخَيار إلى حالة الكتابة فوق النسخ السابقة وخاصّة عندما نستخدم عمليات نسخ احتياطي مجدولة؛

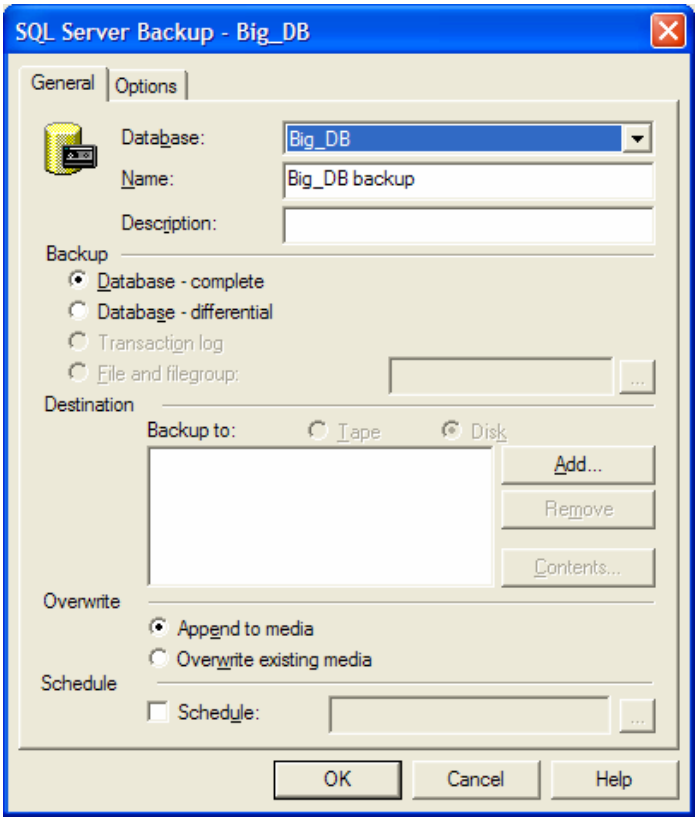
ينبغي أيضا أن نأخذ الواجهة الفرعية Options –والموجودة ضمن الواجهة السابقة– بعين الاعتبار بحيث نلاحظ فيها أنه قد تمّ اختيار إزالة المناقلات غير الفعّالة من سجلّ المناقلات بعد القيام بإجراء عملية التخزين الاحتياطي له بشكل تلقائي، مع العلم أنّ إزالة ذلك الخَيار يماثل كتابة NO_TRUNCATE التي مرّت معنا في الشريحة السابقة؛



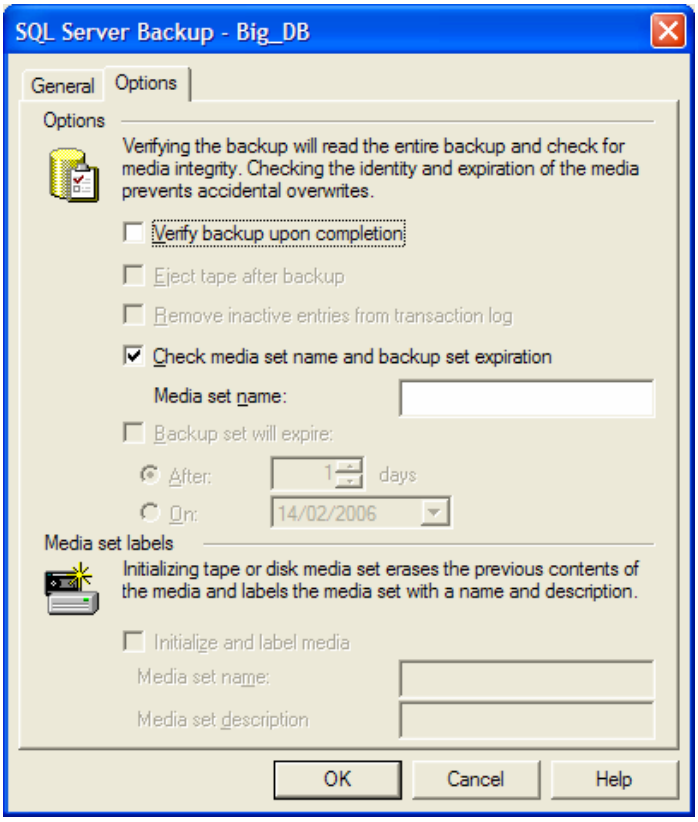
ملاحظة:

إذا لم نستطع القيام بعملية التخزين الاحتياطي لملف سجلّ المناقلات من خلال الواجهة السابقة لأن خَيار"Transaction Log" لـــم يكن فعّالاً، فيعود ذلك إلى أن استراتيجية التعافي المستخدمة هي التعافي البسيط بالتالي ينبغي تحويل أسلوب التعافي المستخدم إلى نوع آخر(تعافي كامل على سبيل المثال)، ويتم ذلك من خلال واجهة خصائص قاعدة المعطيات:

<الواجهات التي لا يمكننا من خلالها القيام بإجراء عملية التخزين الاحتياطي لملف سجلّ المناقلات>



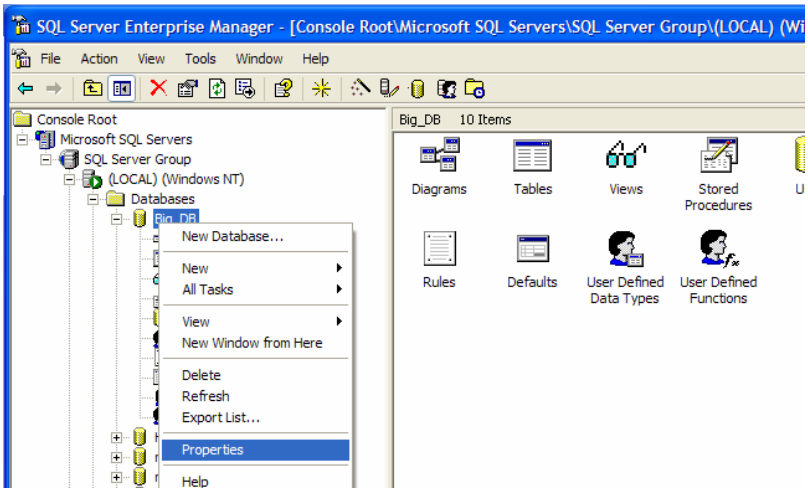
نلاحظ في هذه الواجهة أن خَيار Transaction Log غير فعّال.



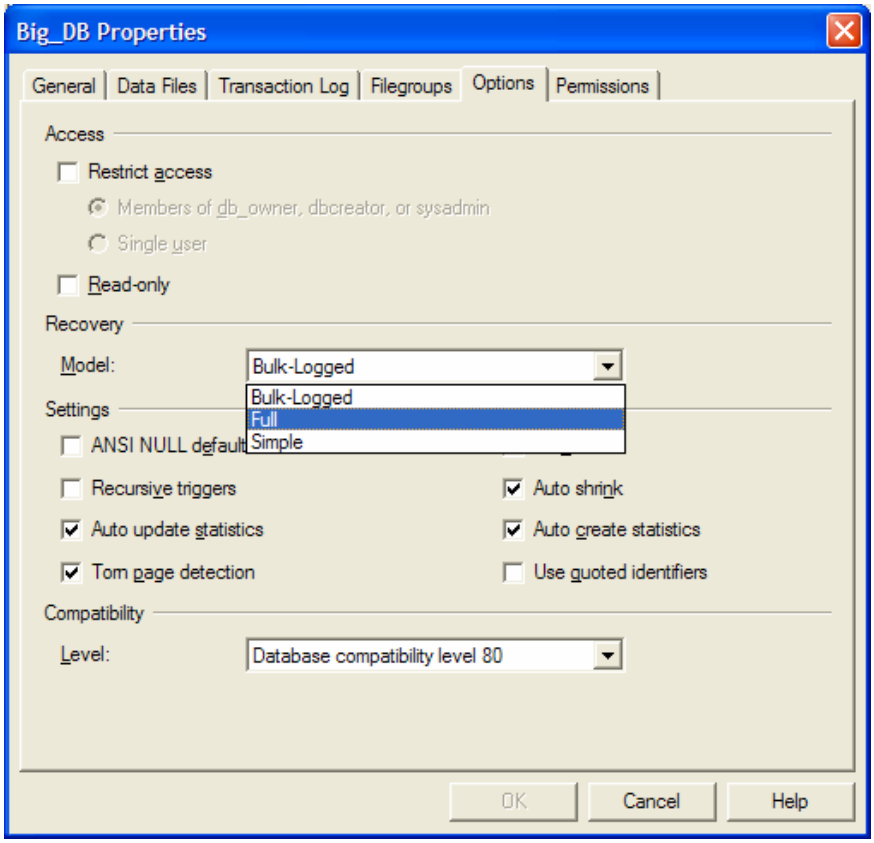
نلاحظ في هذه الواجهة أن الخصائص التي ترتبط بإجراء عملية التخزين الاحتياطي على ملف سجلّ المناقلات غير فعّاله.

كيفية تغيير نوع نموذج التعافي المستخدم:

من واجهة خصائص قاعدة المعطيات:



ومن الواجهة الفرعية "Options":



نختار نموذج التعافي المناسب.

إنشاء نسخة احتياطية من قواعد معطيات النظام

ينبغي أيضا أن نقوم بإجراء عمليات تخزين احتياطي لقواعد معطيات النظام التي مرّت معنا في جلسات سابقة، بحيث تعتبر مسألة الحفاظ على نسخة احتياطية من master و msdb و model بنفس أهمية الحفاظ على نسخة احتياطية من قاعدة المعطيات التي نعمل عليها، مع العلم أنه ليس من الضروري أن نحافظ على أية نسخة احتياطية من قاعدة المعطيات tempdb على اعتبار أنها مؤقتة و لا تحتوي على معطيات دائمة لكي يتم تخزينها؛

أُعدّت كل من قاعدة المعطيات master و msdb بشكل تلقائي لكي يتم إجراء عمليات التخزين الاحتياطي عليها بالاعتماد على نموذج التعافي البسيط، علماً أنه من الممكن استخدام نموذج التعافي الكامل، إلا أنه ليس ضرورياً على اعتبار أن قاعدتي المعطيات صغيرتا الحجم؛

أُعدّت قاعدة المعطيات model بشكل تلقائي لكي يتم إجراء عمليات التخزين الاحتياطي عليها بالاعتماد على نموذج التعافي الكامل، إذ أنها تعتبر الأساس الذي يعتمد عليه المستخدم في بناء قواعد المعطيات، مع العلم أن أي تغيير في نموذج التعافي المطبّق على قاعدة المعطيات هذه سيؤثّر بالضرورة على كافة قواعد معطيات المستخدم التي سيتم إنشاءها؛

استرجاع قاعدة المعطيات

-استعراض محتويات النسخ الاحتياطية قبل استرجاعها-

لابد من الإشارة قبل أن نبدأ بالحديث عن استرجاع قاعدة المعطيات أن هذه العملية بسيطة وسهلة الاستخدام تماماً مثل عملية إنشاء النسخ الاحتياطية؛

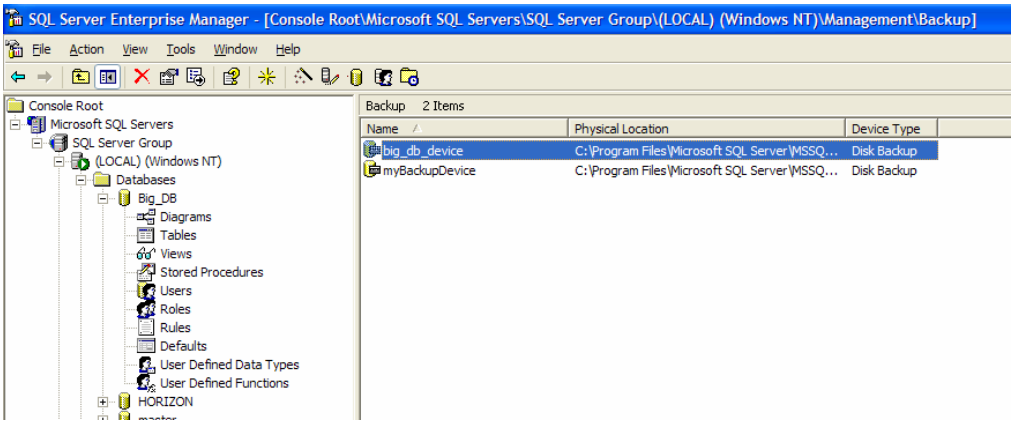
قبل البدء بعملية استرجاع المعطيات لابد لنا من القيام باختبار أن جهاز التخزين الاحتياطي الذي نتعامل معه هو الجهاز الصحيح والذي يحتوي على النسخة الاحتياطية المناسبة التي نريد استرجاعها؛

يمكننا القيام بتلك العملية بعدة أساليب:

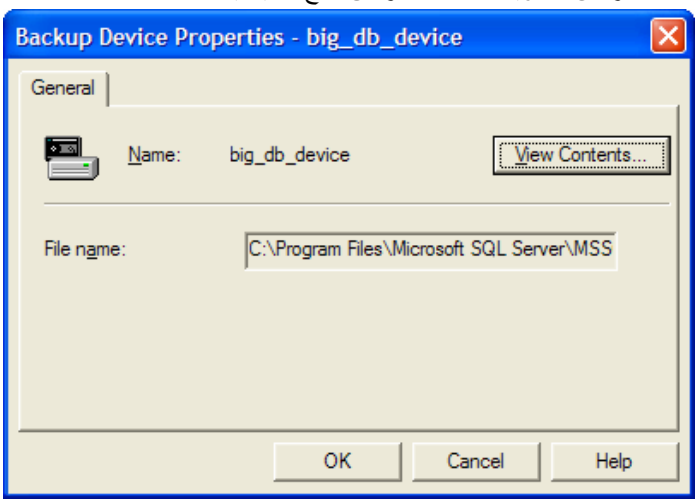
- من خلال الأداة Enterprise Manager؛
- من خلال تنفيذ مخطوط T-SQL مناسب.

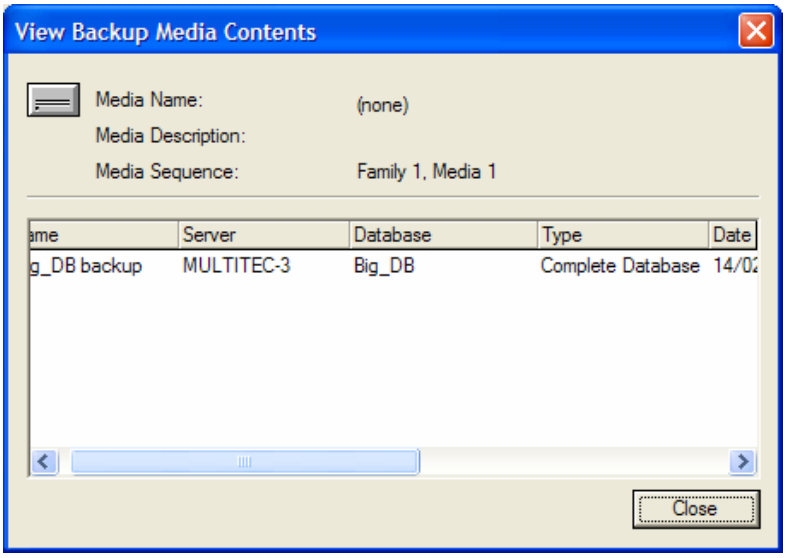
لابد من الإشارة قبل أن نبدأ بالحديث عن استرجاع قاعدة المعطيات أن هذه العملية بسيطة وسهلة الاستخدام تماماً مثل عملية إنشاء النسخ الاحتياطية؛

قبل البدء بعملية استرجاع المعطيات لابد لنا من القيام باختبار أن جهاز التخزين الاحتياطي الذي نتعامل معه هو الجهاز الصحيح والذي يحتوي على النسخة الاحتياطية المناسبة التي نريد استرجاعها، يمكننا التأكد من تلك العملية من خلال الأداة Enterprise Manager وذلك بالنقر مرتين على جهاز التخزين الاحتياطي ثم اختيار عرض المحتويات؛



بحيث نستطيع من خلال ذلك استعراض محتويات ذلك الجهاز من نسخ احتياطية سابقة؛





يمكننا أيضاً القيام بتلك العملية من خلال تنفيذ مخطوط T-SQL مناسب كما يلي:

RESTORE HEADER ONLY

يمكننا من خلال التعليمة السابقة أن نستعرض فقط معلومات ترويسة ملف النسخة الاحتياطية أو مجموعة النسخ الاحتياطية المحددة، ويشتمل ذلك على الاسم والوصف ونمط وسيط التخزين والطريقة التي تم اتباعها في عملية التخزين الاحتياطي ووقت وتاريخ تلك العملية وحجم النسخة الاحتياطية بالإضافة إلى رقمها التسلسلي؛

RESTORE FILELISTONLY

يمكننا من خلال هذه التعليمة أن نستعرض معلومات حول قاعدة المعطيات الأصلية أو ملفات سجلّ المناقلات التي تم تخزينها فـــي النسخة الاحتياطية المحددة؛

RESTORE LABELONLY

تستخدم هذه التعليمة فقط لاستعراض معلومات حول وسيط التخزين المستخدم؛

RESTORE VERIFYONLY

تستخدم هذه التعليمة للتأكد من أن كافة الملفات المكوِّنة لمجموعة التخزين الاحتياطي المستخدمة كاملة وسليمة وقابلة للقراءة؛

استرجاع قاعدة المعطيات باستخدام تعليمات T-SQL

لا تستخدم تعليمة RESTORE DATABASE من أجل استرجاع قاعدة معطيات متضررة فحسب، بل أنه من الممكن استخدامها لعدّة أغراض أخرى، كنقل ملفات المعطيات أو سجلات المناقلات أو استرجاع نسخة من قاعدة المعطيات باسم مختلف أولاسترجاع ملف أو "مجموعة ملفات" أو للقيام بعملية استرجاع جزئية للمعطيات وغيرها؛

فيما يلي عرض للقواعد التي تعبّر عن عملية استرجاع قاعدة المعطيات:

RESTORE DATABASE { *database_name* | @*database_name_var* }
[<file_or_filegroup> [,…n]]
[**FROM** <backup_device> [,…n]]
[**WITH**
[**PARTIAL**]
[**RESTRICTED_USER**]
[[,] **FILE** = { *file_number* | @*file_number_var* }]
[[,] **PASSWORD** = { *password* | @*password_var* }]
[[,] **MEDIANAME** = { *media_name* | @*media_name_var* }]
[[,] **MEDIAPASSWORD** = { *mediapassword* | @*mediapassword_var* }]
[[,] **MOVE** *'logical_file_name'* TO *'operating_system_file_name'*] [,…n]
[[,] **KEEP_REPLICATION**]
[[,] {**NORECOVERY** | **RECOVERY** | **STANDBY**=*undo_file_name*}]
[[,] {**NOREWIND** | **REWIND**}]
[[,] {**NOUNLOAD** | **UNLOAD**}]
[[,] **REPLACE**]
[[,] **RESTART**]
[[,] **STATS** =*percentage*]]

الشرح المرافق:

المُعامل	الوصف
<file_or_filegroup> [,…n]	"مجموعة الملفات" أوالملفات أو مجموعات الملفات التي ينبغي على SQL Server أن يقوم لنا هذه الميزة بإمكانية استرجاع الجزء المتضرر فقط من قاعدة المعطيات، ما يحسن من أداء ك ويزيد من سرعتها.
PARTIAL	دة مضافة إلى SQL Server 2000 تستخدم عادة مع عملية استرجاع كامل قاعدة المعطيات عة الملفات"، وهي تختلف عن عملية استرجاع الملف أو مجموعات الملفات بأنها تقوم جموعة الملفات الأولية بالإضافة إلى مجموعات الملفات الأخرى المحددة.
RESTRICTED_USER	ولوج إلى نسخة قاعدة المعطيات الجديدة التي سيتم استرجاعها، بحيث يسمح بدخول فقط sysa و db_owner و db_creator .

FILE	لتحديد أية مجموعة تخزين احتياطي ينبغي القيام باسترجاعها من بين المجموعات الأخرى الموجودة في جهاز التخزين الاحتياطي، بحيث يشار إليها برقم كمعامل يدل ترتيب تلك المجموعة في الجهاز، مثلاً 2 يدل على المجموعة الثانية في الجهاز المحدد مسبقاً.
PASSWORD	يمثّل كلمة مرور للنسخة الاحتياطية، بحيث ينبغي تزويد كلمة المرور تلك عند القيام بعملية استرجاع تلك النسخة الاحتياطية.
MEDIANAME	عبارة عن اسم منطقي لمجموعة وسائط التخزين المنشأة، مع العلم أنه سيتم استخدام هذا الاسم أثناء القيام بعملية استرجاع النسخة.
MEDIAPASSWORD	مثّل كلمة مرور لمجموعة وسائط التخزين بالكامل، بحيث ينبغي تزويد كلمة المرور تلك عند القيام بعملية استرجاع تلك المجموعة.
NOREWIND | REWIND	لتحديد فيما إذا كان ينبغي إعادة (أو عكس) الشريط أم لا.
NOUNLOAD | UNLOAD	لتحديد فيما إذا كان ينبغي فكّ عملية تحميل الشريط أم لا.
RESTART	يستخدَم هذا الخَيار لإعادة تشغيل عملية تخزين احتياطي متوقّفة اعتباراً من نقطة التوقف، ذلك بغرض اختصار الوقت.
STATS	بارة عن نسبة تستخدم لقياس تقدّم عملية النسخ الاحتياطي، ع العلم أن القيمة التلقائية تساوي 10%.
MOVE	سمح هذا الخَيار باسترجاع الملف المحدد وتخزينة على مسار فيزيائي جديد، بحيث تبرز أهمية هذا الخَيار عندما نرغب بنقل قاعدة لمعطيات أو سجلّ المناقلات إلى قرص جديد، كما يُعتبرخَياراً بالغ الأهمية عندما تواجهنا مشاكل مع القرص الذي يتم تخزين قاعدة المعطيات يه حالياً ونرغب بنقل قاعدة المعطيات تلك إلى قرص آخر بالسرعة القصوى.
KEEP_REPLICATION	ذا الخَيار لحماية إعدادات عملية تكرار قاعدة المعطيات إذا ا قمنا بمحاولة استرجاعها إلى مخدّم آخر.
REPLACE	سمح لنا هذا الخَيار بإجراء عملية استرجاع قاعدة المعطيات وإعادة كتابتها فوق قاعدة المعطيات الأصلية.

من أجل استرجاع قاعدة معطيات متضررة فحسب، بل أنه من الممكن RESTORE DATABASEلا تستخدم تعليمة استخدامها لعدّة أغراض أخرى، كنقل ملفات المعطيات أو سجلات المناقلات أو استرجاع نسخة من قاعدة المعطيات باسم مختلف أولاسترجاع ملف أو "مجموعة ملفات" أو للقيام بعملية استرجاع جزئية للمعطيات وغيرها؛

استرجاع ملف سجلّ المناقلات باستخدام T-SQL

- تُستخدم تعليمة RESTORE LOG لاسترجاع نسخة ملف سجلّ المناقلات الاحتياطية، أو بشكل أدق من ذلك، ينبغي بعد استرجاع قاعدة معطيات معينة أن نقوم بتطبيق كافة النسخ الاحتياطية من سجلات المناقلات المرتبطة بنسخة قاعدة المعطيات تلك؛
- تحتوي النسخة الاحتياطية لملف سجلّ المناقلات على كافة المناقلات التي تم تنفيذها على قاعدة المعطيات بشكل تسلسلي، بالتالي ينبغي أن يتم استرجاع هذا الملف من خلال تطبيق تلك المناقلات بنفس الترتيب الذي تم تخزينها فيه؛
- بالنسبة للقواعد التي تعبّر عن عملية استرجاع قاعدة المعطيات، نورد فيما يلي عرض للقواعد التي تعبّـر عـن عمليـة استرجاع ملف سجلّ المناقلات لقاعدة معطيات محددة:

RESTORE LOG { *database_name* | @*database_name_var* }
TO <backup_device> [,…n]
[**WITH**
[[,] { **NORECOVERY**| **RECOVERY**| **STANDBY** = *undo_file_name* }]
[[,] **STOPAT = {***date_time* **|** @*date_time_var* **}**
| [,] **STOPATMARK =** *'mark_name'* **[AFTER** *datetime***]**
| [,] **STOPBEFOREMARK =** *'mark_name'* **[AFTER** *datetime***]**

الشرح المرافق

الخاصة	الوصف
NORECOVERY	على أنه ينبغي على SQL Server ألا يقوم بأي عملية تراجع عن أية مناقلة غير مؤكّدة في سجلّ المناقلات، أي أنه ينبغي ترك قاعدة المعطيات في نمط التعافي.
RECOVERY	عكس العملية السابقة، أي أنه ينبغي على SQL Server أن يقوم بإجراء عملية ع عن أية مناقلة غير مؤكّدة في سجلّ المناقلات، بالتالي يسمح باستخدام قاعدة المعطيات مباشرة بعد انتهاء عملية استرجاع ملف سجلّ المناقلات، مع العلم أنه الخَيار التلقائي.
STANDBY	هذا الخَيار –إلى حدٍّ ما– حالة NONRECOVERY إلا أنه يترك قاعدة المعطيات في حالة القراءة فقط.
STOPAT	يأخذ معاملاً يعبر عن التاريخ والوقت الذي ينبغي عنده إيقاف عملية استرجاع ملف سجلّ المناقلات.
STOPATMARK	وهي خاصة جديدة مضافة إلى SQL Server 2000 تسمح بإيقاف عملية استرجاع ملف سجلّ المناقلات عند نقطة محددة في المناقلة؛

وهي خاصة جديدة مضافة إلى SQL Server 2000 تسمح بإيقاف عملية استرجاع ملف سجلّ المناقلات قبل نقطة محددة في المناقلة؛	**STOPBEFOREMARK**

أمثلة عن استرجاع المعطيات باستخدام T-SQL
(نفذ الأمثلة التالية عملياً على قواعد معطيات تجريبية على جهازك)

مثال1:

- عرض للمخطوط المستخدم للقيام استرجاع كامل لقاعدة معطيات محددة.

مثال2:

- عرض للمخطوط المستخدم للقيام استرجاع كامل لقاعدة معطيات محددة ولملف سجلّ المناقلات التابع لها.

مثال3:

- عرض للمخطوط المستخدم للقيام بعملية تعافي إلى نقطة محددة وفي وقت محدد.

مثال4:

- عرض للمخطوط المستخدم للقيام بعملية استرجاع ملف معين.

مثال1:

- فيما يلي عرض للمخطوط المستخدم للقيام استرجاع كامل لقاعدة معطيات محددة:

```
USE MASTER
RESTORE DATABASE BIG_DB
FROM big_db_backup
WITH
FILE =1,
RECOVERY;
```

مثال2:

- فيما يلي عرض للمخطوط المستخدم للقيام استرجاع كامل لقاعدة معطيات محددة ولملف سجلّ المناقلات التابع لها:

```
USE MASTER
RESTORE DATABASE BIG_DB
FROM big_db_backup
```

```
WITH
FILE =1,
NORECOVERY

GO

RESTORE LOG BIG_DB
FROM big_db_LOG_backup
WITH
FILE =1,
NORECOVERY

GO

RESTORE LOG BIG_DB
FROM big_db_LOG_backup
WITH
FILE =2,
RECOVERY
```

مثال3:

- فيما يلي عرض للمخطوط المستخدم للقيام بعملية تعافي إلى نقطة محددة وفي وقت محدد:

```
USE MASTER
RESTORE DATABASE BIG_DB
FROM big_db_backup
WITH
FILE =1,
NORECOVERY

GO

RESTORE LOG BIG_DB
FROM big_db_LOG_backup
WITH
FILE =1,
NORECOVERY

GO

RESTORE LOG BIG_DB
```

FROM *big_db_LOG_backup*
WITH
FILE =2,
RECOVERY
STOPAT= 'September 25, 2006 09:00 AM'

مثال4:

- فيما يلي عرض للمخطوط المستخدم للقيام بعملية استرجاع ملف معين:

USE MASTER
RESTORE DATABASE BIG_DB
FILE = BIG_DB_FILE5
FROM *big_db_backup*
WITH
NORECOVERY

GO

RESTORE LOG BIG_DB
FROM *big_db_LOG_backup*
WITH
FILE =1,
RECOVERY

القيام بعملية استرجاع المعطيات باستخدام الأداة Enterprise Manager

- إن ارتكاب الأخطاء أثناء القيام بعملية استرجاع المعطيات قد تكلف الكثير فيما يتعلق بالمشاكل التي يمكن أن تتولّد عنها؛

- تؤمن الأداة Enterprise Manager واجهات تخاطبية سهلة الاستخدام تساعد على إنجاز عمليات استرجاع المعطيات بأسلوب سهل وبسيط، مما يؤدي إلى انخفاض امكانية حدوث أخطاء –أثناء هذه العملية– بنسبة كبيرة؛

- يمكننا من خلال واجهة استرجاع المعطيات أن نحدد قاعدة معطيات معينة نرغب بإعادة تخزينها، أو أن نحدد اسم جديد لقاعدة المعطيات التي نرغب بإنشائها بالاعتماد على النسخة الاحتياطية التي نملكها؛

- يتم اختيار نموذج الاسترجاع الكامل لقاعدة المعطيات بشكل تلقائي، مع العلم أنه يمكننا أن نتحدد أننا نرغب باسترجاع ملف أو استرجاع "مجموعة ملفات" أو استرجاع المعطيات من قرص؛

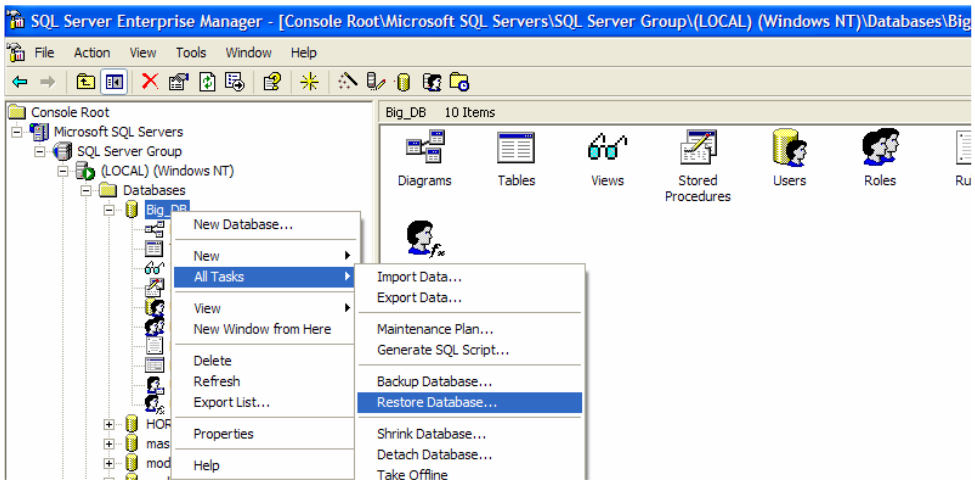
- يمكننا كذلك أن نحدد أية نسخة احتياطية نرغب باستخدامها في عملية استرجاع المعطيات، مع العلم أن آخر نسخة احتياطية مُنشأة لقاعدة المعطيات المحددة، هي التي يتم اختيارها بشكل تلقائي؛

- يمكننا أن نحدد كذلك من خلال هذه الواجهة، النقطة التي ينبغي توقّف عملية استرجاع المعطيات عندها عندما نقوم بإجراء عملية استرجاع لملف سجلّ مناقلات معين؛

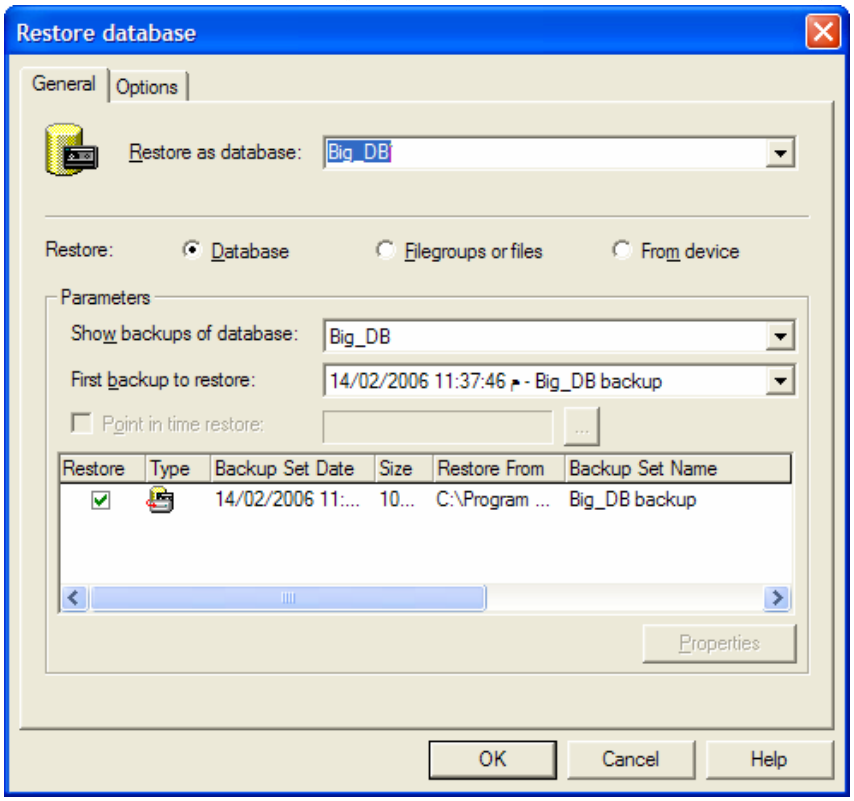
- تعرض لنا تلك الواجهة أيضاً قسما يمكننا أن نستعرض من خلاله كافة النسخ الاحتياطية التي قمنا بتوليدها مسبقاً لقاعدة المعطيات المحددة، مع العلم أن الأداة Enterprise Manager ستوفر علينا عملية انتقاء النسخة الاحتياطية، بحيث تنتقي بشكل تلقائي النسخة الاحتياطية اللازمة لإجراء عملية الاسترجاع تلك،

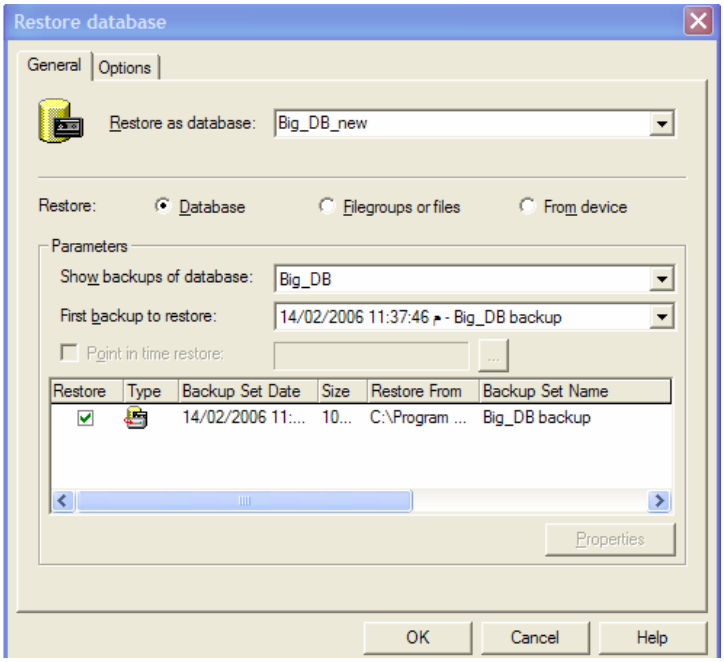
- نستطيع من خلال الواجهة الفرعية Options أن نحدد خصائص إضافية أخرى لعملية استرجاع المعطيات.

- إن ارتكاب الأخطاء أثناء القيام بعملية استرجاع المعطيات قد تكلف الكثير فيما يتعلق بالمشاكل التي يمكن أن تتولّد عنها؛
- تؤمن الأداة Enterprise Manager واجهات تخاطبية سهلة الاستخدام تساعد على إنجاز عمليات استرجاع المعطيات بأسلوب سهل وبسيط، مما يؤدي إلى انخفاض امكانية حدوث أخطاء –أثناء هذه العملية– بنسبة كبيرة؛
- كما مرّ معنا مسبقاً عندما استعرضنا كيفية الوصول إلى الواجهات الخاصة بعملية التخزين الاحتياطي، يمكننا –وبنفس الطريقة– الوصول إلى الواجهات الخاصة بعملية استرجاع المعطيات، وذلك إما من خلال "لائحة المهمات" Task pad أو من خلال القائمةTools أو من خلال خصائص قاعدة المعطيات نفسها ثم اختيار Restore Database من ضمن القائمة All Tasks.



- يمكننا من خلال واجهة استرجاع المعطيات أن نحدد قاعدة معطيات معينة نرغب بإعادة تخزينها، أو أن نحدد اسم جديد لقاعدة المعطيات التي نرغب بإنشائها بالاعتماد على النسخة الاحتياطية التي نملكها؛





- يتم اختيار نموذج الاسترجاع الكامل لقاعدة المعطيات بشكل تلقائي، مع العلم أنه يمكننا أن ننحدد أننا نرغب باســترجاع ملف أو استرجاع "مجموعة ملفات" أو استرجاع المعطيات من قرص؛

- يمكننا كذلك أن نحدد أية نسخة احتياطية نرغب باستخدامها في عملية استرجاع المعطيات، مع العلــم أن آخــر نــسخة احتياطية مُنشأة لقاعدة المعطيات المحددة، هي التي يتم اختيارها بشكل تلقائي؛

- يمكننا أن نحدد كذلك من خلال هذه الواجهة، النقطة التي ينبغي توقّف عملية استرجاع المعطيات عندها عندما نقوم بإجراء عملية استرجاع لملف سجلّ مناقلات معين؛

- تعرض لنا تلك الواجهة أيضاً قسما يمكننا أن نستعرض من خلاله كافة النسخ الاحتياطية التي قمنا بتوليدها مسبقاً لقاعــدة المعطيات المحددة، مع العلم أن الأداة Enterprise Manager ستوفر علينا عملية انتقاء النسخة الاحتياطية، بحيث تنتقي بــشكل تلقائي النسخة الاحتياطية اللازمة لإجراء عملية الاسترجاع تلك،
  - مثال:

لنفترض أننا قمنا بنشاء نسخة احتياطية من قاعدة معطيات معينة وذلك على النحو التالي:

1. عملية نسخ احتياطي لكامل قاعدة المعطيات؛
2. عملية نسخ احتياطي لثلاثة سجلات مناقلات؛
3. عملية نسخ احتياطي جزئية لقاعدة المعطيات؛

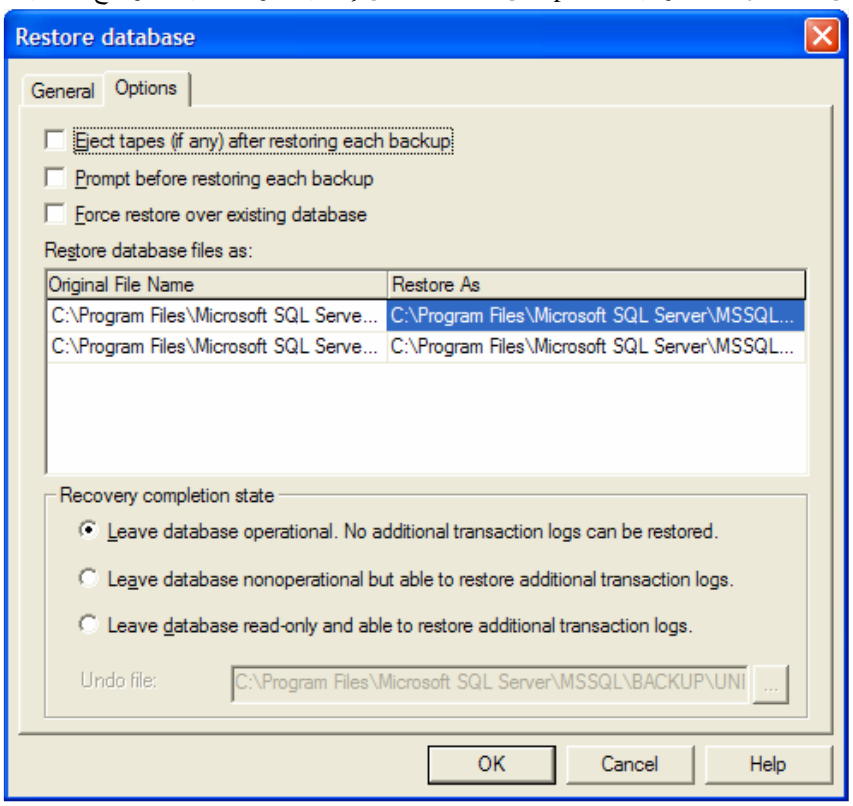
4. عملية نسخ احتياطي لملفي سجلّ مناقلات.

تعتبر الخطة الأفضل والتي ينبغي تنفيذها من أجل القيام بعملية استرجاع قاعدة المعطيات تلك، هي بإجراء العمليات التالية:

1. استرجاع النسخة الاحتياطية لكامل قاعدة المعطيات؛
2. ثم استرجاع النسخة الاحتياطية الجزئية لقاعدة المعطيات؛
3. ثم استرجاع آخر ملفي سجلّ مناقلات تم تخزينهما.

كان الغرض من المثال السابق، التأكيد على أن الأداة Enterprise Manager ستعمل تلقائيا على اختيار هذا السيناريو –الأمثل– لعملية الاسترجاع.

- نستطيع من خلال الواجهة الفرعية Options أن نحدد خصائص إضافية أخرى لعملية استرجاع المعطيات، ومنها:



- إمكانية السماح بسحب الشرائط بعد الانتهاء من إجراء عملية استرجاع قاعدة المعطيات وذلك عند استخدام الشرائط كوسائط تخزين؛
- ضرورة إعلام المستخدم قبل المباشرة بإجراء عملية استرجاع قاعدة المعطيات؛
- تطبيق عملية استرجاع قاعدة المعطيات وتخزين النتائج الجديدة فوق قاعدة المعطيات الأصلية؛

- إمكانية إعادة تخزين ملفات قاعدة المعطيات الجديدة التي سيتم استرجاعها، في مسار فيزيائي جديد يمكن تحديده (ملاحظة: تذكّر خاصة MOVE في تعليمة RESTORE DATABASE)؛
- إمكانية تحديد إحدى الخيارات التالية RECOVERY أو NORECOVERY أو STANDBY (راجع الشريحة التي تتحدث عن كيفية القيام بعملية استرجاع قاعدة المعطيات باستخدام تعليماتT-SQL).

الفصل الثاني عشر والثالث عشر

عنوان الموضوع:

Transact-SQL (1)

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

نبدأ في هذا القسم باستعراض لغة Transact-SQL الخاصة بنظام إدرة قواعد المعطيات SQL Server.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- التعليميات الأساسية في لغة الاستعلام المهيكلة T-SQL
- المناظير المفهرسة (Indexed Views)
- التجميعات الفائقة (Hyper Aggregaes)

الإستعلامات المُضمنة/الجزئية (Subqueries)

- سنسعى من خلال هذه الجلسة لدراسة لغة Transact SQL وعرض مكوّناتها؛

- سنركز بشكل أساسي –فيما يلي من شرائح– على بعض تعليمات DML (Data Manipulation Language) وعلى البنى البرمجية الموجودة في T-SQL بالإضافة إلى تعليمات DDL (Data Definition Language) المختلفة المتاحة؛

- سنحاول أن نستعرض معظم مكونات T-SQL بالإضافة إلى المكونات المعيارية للغة SQL المعروفة بشكل مختصر، كما سنشرح بالتفصيل كيف يتم بناء واستخدام أغراض متعددة في SQL Server كالإجرائيات المعرّفة أو التوابع المخزّنة أو المناظير وغيرها.

لغة SQL المعيارية

"من ANSI SQL إلى T-SQL"

- تطورت لغة SQL المعيارية كثيراً منذ أن تم إصدارها لأول مرة، بحيث تم إضافة العديد من التحسينات عليها خلال السنوات الأخيرة فيما يتعلق بتوفير تعليمات المعيارية تسمح بإدارة وتعريف المعطيات أو النفاذ إليها أو تعديلها؛

- يطلق على الإصدارات المعيارية من SQL اسم ANSI SQL؛

- تدعم مايكروسوفت من خلال SQL Server العديد من الخصائص الإضافية التي تسمح بتوسيع إمكانيات لغة SQL المعيارية؛

- يطلق على التعديلات المُضافة إلى لغة SQL المعيارية من قبل مايكروسوفت والتي تدعم SQL Server، اسم Transact SQL أو T-SQL.

- تطورت لغة SQL المعيارية كثيراً منذ أن تم إصدارها لأول مرة، بحيث تم إضافة العديد من التحسينات عليها خلال السنوات الأخيرة، فيما يتعلق بتوفير تعليمات معيارية تسمح بإدارة وتعريف المعطيات أو النفاذ إليها أو تعديلها؛

- يطلق على الإصدارات المعيارية من SQL اسم ANSI SQL، ويعتبر SQL-99 هو أحدث تلك الإصدارات، مع العلم أن SQL-92 كان أحد أكبر الإصدارات المعيارية التي تم نشرها؛

- يتوافق SQL Server 2000 مع الإصدار المعياري ANSI SQL-92، أي أنه يدعم كافة الميزات الأساسية الموجودة في ذلك الإصدار بالإضافة إلى أنه يدعم بعض الخصائص المتوافرة مع الإصدار ANSI-99، كخاصة تعريف الأدوار Roles على سبيل المثال؛

- تدعم مايكروسوفت من خلال SQL Server العديد من الخصائص الإضافية التي تسمح بتوسيع إمكانيات لغة SQL المعيارية، كإمكانية استخدام بعض التعليمات التي تتعلق بنظام التشغيل الذي يعمل عليه SQL Server أو إمكانية استخدام بعض التوسعات على لغة SQL، والتي يمكن حصرها –بشكل عام– من خلال النقاط التالية:
  - برامج من جهة المخدِّم، كالإجرائيات المخزّنة أوالتوابع أوالقوادح؛
  - كيفية التحكم بتنفيذ العبارات؛
  - أنماط معطيات جديدة، بالإضافة إلى أنماط معطيات المستخدم المعرّفة؛
  - أنواع مختلفة من أساليب ضمان تكامل المعطيات المعرّفة، كالقواعد والقوادح والقيم التلقائية؛
  - توابع مضمّنة إضافية.

- يطلق على التعديلات المُضافة إلى لغة SQL المعيارية من قبل مايكروسوفت والتي تدعم SQL Server، اسم Transact SQL أو T-SQL.

T-SQL

"مقدمة"

- تطورت لغة T-SQL عند الانتقال من SQL Server 6.5 إلى SQL Server 7.0 بشكل واضح، إذ تم في ذلك الحين إضافة العديد من المزايا الجديدة، إلاّ أن الانتقال إلى SQL Server 2000 لم يقدّم العديد من التغيرات الملحوظة في مجال تحديث مكونات T-SQL عن الإصدار السابق له، فيما عدا إضافة بعض أنماط المعطيات وإمكانية إنشاء توابع مستخدِم معرّفة وإنشاء مناظير مفهرسة؛

- سنتناول فيما يلي من شرائح، ميزات تلك الخصائص الجديدة مع العلم أننا سنقوم لاحقاً باستعراض بعض مكونات لغة SQL المعيارية ومكونات لغة T-SQL بالتفصيل.

ميزات T-SQL الجديدة في SQL Server 2000

أنماط المعطيات الجديدة – bigint

- يقدم SQL Server 2000 ثلاثة أنماط معطيات جديدة وهي:
  - bigint
  - sql_variant
  - table

- نمط معطيات bigint:
  - يعبر نمط المعطيات bigint عن نمط معطيات رقمي بحجم تخزين يصل إلى 8 بايت، أي يمكن أن يخزن الأرقام ما بين (9223372036854775808-)= -2^{63}
إلى (9223372036854775807)=2^{63-1}؛
  - يمكن استخدام هذا النمط عندما يُتوقع ضرورة استخدام قيم رقمية لا يستطيع النمط int التعبير عنها، كما أنه يمكن قواعدياً أن يتم استخدام النمط bigint في أي مكان يُسمح فيه باستخدام النمط int؛
  - يزوّدنا SQL Server 2000 بتابعين جديدين يُستخدمان للتعامل مع القيم الرقمية من نمط bigint، وهما:
    - COUNT_BIG( ): وهو تابع يشبه في عمله تماماً التابع COUNT إلا أنه يعيد نتيجة من نمط bigint. يُفضل استخدام هذا التابع عندما نتوقع نتيجة لا يمكن التعبير عنها من خلال النمط int؛
    - ROWCOUNT_BIG( ): وهو تابع يشبه في عمله تماماً التابع @@ROWCOUNT إلا أنه يعيد عدد الأسطر –التي تأثرت بآخر استعلام تم تنفيذه– مرمّزة بنمط bigint. يُفضل استخدام هذا التابع عندما نتوقع نتيجة كبيرة في عدد الأسطر المتأثرة بالاستعلام المُنفذ بحيث لا يمكن التعبير عنها من خلال النمط int؛

ميزات T-SQL الجديدة في SQL Server 2000

أنماط المعطيات الجديدة – sql_variant

- يشبه هذا النمط، نمط المعطيات variant المستخدم في Microsoft Visual Basic، كما يشبه كذلك نمط المعطيات DBTYPE_VARIANT المستخدم في OLE DB؛

- يمكننا استخدام هذا النمط في عدّة أماكن، وهي: أثناء تعريف الأعمدة أو في المتحولات أو في المعاملات أو كقيمة خرج لتابع مستخدِم معرَّف؛

- عند استخدام هذا النمط أثناء تعريف الأعمدة فإن الغرض من ذلك يكون من أجل التعبير عن الحالات التي يمكن فيها تخزين قيم مختلفة في نفس العمود، كأن يُستخدم مثلاً أحد الأعمدة ليخزن محارف في أسطر ما وقيماً رقمية في أسطر أخرى؛

- إن حجم التخزين الأعظمي لنمط المعطيات sql_variant يساوي 8016 بايت، على الرغم من أننا لا نقوم صراحةً بتحديد حجم ما لهذا النمط، إذ أنه يتحدد تلقائياً عند استخدامه مع عمود أو متحول، مع العلم أنه يمكننا استخدام الأعمدة من نمط sql_variant كمفاتيح فريدة أو كفهارس، طالما لم يتجاوز حجم ذلك العمود 900 بايت.

ميزات T-SQL الجديدة في SQL Server 2000

أنماط المعطيات الجديدة – table

- يزوّدنا SQL Server 2000 بنمط معطيات جديد آخر، وهو نمط المعطيات table؛

- يمكن استخدام هذا النمط كمتحول محلي في تابع مستخدِم معرَّف أو إجرائية مخزّنة، كما يمكن استخدامه كخرج لتابع ما؛

- لا يمكن لأعمدة هذا النوع من الأنماط أن تكون كذلك من النمط table، كما أنه لا يمكن استخدام النمط table كمعامل إجرائية أو تابع؛

- إن قواعد إنشاء هذا النوع من أنماط المعطيات تشبه تماماً قواعد إنشاء جداول قاعدة المعطيات، في حين ينبغي إضافة ترويسة مناسبة للدلالة على المتحول الذي يتمتع بهذا النمط، فيما يلي عرض لمثال يوضح كيفية تعريف متحول على أنه من نمط table:

DECLARE @variable **TABLE** (*column definition* | *table_constraint* [, …])

- نلاحظ من المثال السابق أنه يمكننا فرض قيود على أعمدة نمط المعطيات table، إلا أن القيود المسموح بها هي المفتاح الأولي والمفتاح الفريد وقيود الاختبار والقيم التلقائية، كما أن الأعمدة يمكن أن تكون ذات قيم فارغة أو لا كما يمكن ربطها بخاصة التزايد التلقائي، إلا أنه لا يمكن استخدام قيد المفتاح الخارجي على أعمدة هذا النوع من أنماط المعطيات؛

- مثال:

```
DECLARE @title_info TABLE (    title_id int,
title varchar(64),
pub_date datetime,
price money null
    )
```

ميزات T-SQL الجديدة في SQL Server 2000

توابع المستخدِم المعرَّفة

- كان SQL Server دائماً يدعم عدداً من التوابع المضمَّنة بغية توسيع إمكانيات لغة T-SQL، إلا أن تلك التوابع كانت مبرمجة ضمنياً في SQL Server ولم تتاح إمكانية إجراء تعديلات عليها؛

- ما تزال التوابع المضمَّنة غير قابلة للتعديل، إلا أن SQL Server 2000 يتيح حالياً إمكانية إنشاء توابع مستخدِم معرَّفة وتعديلها؛

- يمكن إنشاء توابع المستخدم المعرَّفة باستخدام عبارة CREATE FUNCTION، كما يمكن تعديلها باستخدام عبارة ALTER FUNCTION ويمكن حذفها باستخدام DROP FUNCTION، مع العلم أنه لا يمكن تعريف أكثر من تابع بنفس الاسم من أجل نفس المستخدم في قاعدة المعطيات؛

- يمكن لتوابع المستخدم المعرَّفة أن تأخذ من 0 إلى 1024 معامل دخل، كما يمكن أن تُعيد قيمة وحيدة أو أسطر جدول كامل كخرج؛

- يمكن استخدام توابع المستخدم المعرَّفة التي تُعيد قيمة وحيدة في أي مكان يتطلب قيمة ثابتة في الاستعلام؛

- يمكن استخدام توابع المستخدم المعرَّفة التي تُعيد قيمة عدّة أسطر في أي مكان يمكن فيه كتابة عبارة ما في الاستعلام.

ميزات T-SQL الجديدة في SQL Server 2000

المناظير المفهرسة

- يمكن اعتبار المناظير بشكل بسيط بأنها جدول افتراضي؛

- لا يتم إجراء تخزين فعلي للأسطر الناتجة عن منظار معين في أغراض محددة في قاعدة المعطيات، بحيث يتم إعدادها واسترجاعها مباشرةً من الجدول أو الجداول التي تنتمي إليها؛

- تعتبر المناظير بشكل عام على أنها استعلامات اختيار عادية؛

- تظهر المشاكل في التعامل مع المناظير، وبشكل واضح، بنتيجة العبء الإضافي المفروض على عملية الإنشاء الديناميكي للمنظار، خاصةً عندما تتكون من عدد كبير من الأسطر، أو تتطلب إجراء عمليات تجميعية متعددة لكي يتم إنشاءها؛

- يقدم SQL Server 2000 إمكانية إنشاء فهارس على المناظير، بحيث يمكننا تحسين أداء المناظير المعقدة بإنشاء فهارس عنقودية عليها (راجع جلسة الفهارس)؛

- تختلف المناظير المفهرسة بشكل كبير عن المناظير الاعتيادية، إذ أنه ما أن يتم إنشاء منظار مفهرس حتى يتم تنفيذ ذلك المنظار واسترجاع المعطيات الناتجة عنه وتخزينها فيزيائياً وفهرستها، تماماً كما يحصل مع جدول ذو فهرس عنقودي؛

- يتم تلقائيا تعديل محتويات المنظار المفهرس من معطيات، مباشرة بعد تطبيق تلك التعديلات على المعطيات الأصلية في الجداول التي تم إنشاء ذلك المنظار منها؛

- تُحسِّن المناظير المفهرسة –وبشكل واضح– من أداء التطبيقات التي تَستخدم باستمرار استعلامات متعددة أو عمليات تجميعية كثيرة على جداول ذات معطيات ضخمة.

تعليمات الإضافة والحذف والتعديل والاستعلام

- توفّر لغة SQL القياسية أربعة تعليمات أساسية تستخدم لاسترجاع أو تعديل المعطيات الموجودة في الجداول؛
- نستطيع من خلال تعليمة SELECT أن نقوم باسترجاع المعطيات الموجودة في جدول أو أكثر؛
- نستطيع من خلال تعليمة INSERT أن نقوم بإضافة أسطر جديدة إلى جدول وحيد؛
- نستطيع من خلال تعليمة UPDATE أن نقوم بتعديل الأسطر الموجودة في جدول وحيد؛
- نستطيع من خلال تعليمة DELETE أن نقوم بإزالة الأسطر الموجودة في جدول وحيد؛
- سنستعرض في الشرائح التالية القواعد التي تميز كل من تلك التعليمات على حدى.

عبارة SELECT

SELECT [DISTINCT] [**TOP** *n* [**PERCENT**]] *column1* [**AS** *column_heading*] [, *column2* [**AS** *column_heading*] , …]
[**INTO** *new_table_name*]
FROM *table1* [[**AS**] *table_alias*]
R | {**LEFT** | **RIGHT** | **FULL**} [**OUTER**]] **JOIN** *table2* [[**AS**] *table_alias2*] **ON** (*join_conditions*)] […]
[**WHERE** *search_conditions*]
[**GROUP BY** *aggregate_free_expression*]
[**HAVING** *search_condition*]
[ORDER BY *order_expression* **[ASC | DESC]]**
[COMPUTE *row_aggrigate* (*column_name*) **[, …]**
[BY *column_name* **[,** *column_name* **] …]]**

- تعيد تعليمة SELECT وبشكل تلقائي كافة الأسطر التي توافق شروط البحث المحددة؛
- في حال تم استخدام DISTINCT مع تعليمة الاستعلام فإنه سيتم إزالة الأسطر المكررة بين الأسطر المسترجعة؛
- تستخدم العبارة WHERE من أجل فلترة الأسطر المسترجعة في الاستعلام، بحيث تُهمل أية أسطر لا تحقق شروط البحث المحددة في هذه العبارة؛
- يمكننا كذلك أن نستخدم عبارات OR أو AND المنطقية لكي نقوم بالربط بين أجزاء عبارة شروط البحث، كما يمكننا استخدام الأقواس "(" ")" للتعبير عن الأولويات المرغوبة؛
- تستخدم العبارة ORDER BY من أجل عرض الأسطر المسترجعة في الاستعلام بشكل مرتب وذلك حسب الأعمدة المحددة في تلك العبارة، ويكون الترتيب تصاعدياً بشكل تلقائي "ASC"، مع العلم أنه يمكننا عرض النتيجة مرتبة تنازلياً من خلال استخدام العبارة "DESC"؛
- تستخدم العبارة TOP لكي نحصر عدد الأسطر التي نقوم باسترجاعها، بحيث يمكننا أن نحدد عدد معين من الأعمدة أو نسبة مئوية من الأسطر المسترجعة في الاستعلام؛
- يمكننا استخدام العبارة WITH TIES مع العبارة TOP وذلك لكي نقوم باسترجاع الأسطر المكررة في حال وجودها.
- سنناقش فيما يلي القواعد الأساسية لعملية الاختيار:

SELECT [DISTINCT] [**TOP** *n* [**PERCENT**]] *column1* [**AS** *column_heading*] [, *column2* [**AS** *column_heading*] , …]
[**INTO** *new_table_name*]

FROM *table1* [[**AS**] *table_alias*]
R | {**LEFT** | **RIGHT** | **FULL**} [**OUTER**]] **JOIN** *table2* [[**AS**] *table_alias2*] **ON** (*join_conditions*)] […]
[**WHERE** *search_conditions*]
[**GROUP BY** *aggregate_free_expression*]
[**HAVING** *search_condition*]
[ORDER BY *order_expression* **[ASC | DESC]**]
[COMPUTE *row_aggrigate* (*column_name*) **[, …]**
[BY *column_name* **[,** *column_name* **] …]**]

الشرح المرافق

- تعيد تعليمة SELECT وبشكل تلقائي كافة الأسطر التي توافق شروط البحث المحددة؛
- في حال تم استخدام DISTINCT مع تعليمة الاستعلام فإنه سيتم إزالة الأسطر المكررة بين الأسطر المسترجعة؛
- ملاحظة: لا يفضل استخدام العبارة DISTINCT بشكل متكرر في الاستعلامات التي لا تحتاج لهذا النوع من العبارات، وذلك لأنها ستؤثر بالضرورة على أداء وسرعة الاستعلام سلباً، خاصةً وأنها تقوم بنسخ الأسطر المسترجعة إلى جدول خاص من أجل إزالة التكرار وإعادة النتيجة النهائية.
- تستخدم العبارة WHERE من أجل فلترة الأسطر المسترجعة في الاستعلام، بحيث تُهمل أية أسطر لا تحقق شروط البحث المحددة في هذه العبارة؛
- يوضح الجدول التالي قائمة العمليات التي يمكن استخدامها في عبارة شروط البحث:

لاسترجاع كافة الأسطر التي تساوي قيمة محددة	=
لاسترجاع كافة الأسطر التي لا تساوي قيمة محددة	<>
استرجاع كافة الأسطر ذات قيمة أكبر من قيمة محددة	>
لاسترجاع كافة الأسطر ذات قيمة أصغر من قيمة محددة	<
لاسترجاع كافة الأسطر ذات قيمة أكبر أو تساوي قيمة محددة	>=
لاسترجاع كافة الأسطر ذات قيمة أصغر أو تساوي قيمة محددة	<=
لاسترجاع كافة الأسطر التي تحتوي على قيمة تقع ما بين قيمتين محددتين	BETWEEN exp1 AND exp2
لاسترجاع كافة الأسطر التي تحقق قيمها أحد العناصر المحددة في العبارة	IN (element1, element2, …)
لاسترجاع كافة الأسطر التي تحتوي على قيم محرفية تشبه القيمة المحددة، مع العلم أن الرمز "%" يعبر عن إمكانية تكرار أي عدد ممكن من المحارف، وأن الرمز "_" يعبر عن إمكانية وجود محرف وحيد فقط، وأنه يمكن استخدام عن أن المحارف التي تجري مقارنتها يمكن أن تشبه أية محرف محدد بين القوسين السابقين.	LIKE string_expression

- يمكننا كذلك أن نستخدم عبارات OR أو AND المنطقية لكي نقوم بالربط بين أجزاء عبارة شروط البحث، كما يمكننا استخدام الأقواس "(" ")" للتعبير عن الأولويات المرغوبة؛

أمثلة:

مثال1:

فيما يلي عرض للمخطوط المستخدم للقيام بعملية استرجاع لكافة الكتب من نوع business والتي يقل ثمنها عن 1000 ل.س:

```
SELECT title, type, price
FROM titles
WHERE type= 'business' AND price < 1000
```

النتيجة:

title	type	price
الاقتصاد العالمي	business	700
الاقتصاد والتجارة الدولية	business	800
البورصة وإدارة الأعمال	business	400

(3 row(s) affected)

مثال2:

فيما يلي عرض للمخطوط المستخدم للقيام بعملية استرجاع لكافة الكتب من نوع business والتي يقل ثمنها عن 1000 ل.س أو الكتب من نوع business والتي تزداد مبيعاتها عن 10000 ل س:

```
SELECT title, type, price, sales
FROM titles
WHERE type= 'business' AND ( price < 1000 or sales > 10000 )
```

النتيجة:

title	type	price	sales
الاقتصاد العالمي	business	700	15000
الاقتصاد والتجارة الدولية	business	800	22000
البورصة وإدارة الأعمال	business	400	9000

(3 row(s) affected)

مثال3:

فيما يلي عرض للمخطوط المستخدم للقيام بعملية استرجاع لكافة الكتب من نوع business والتي يقل ثمنها عــن 1000 ل.س أو الكتب التي تزداد مبيعاتها عن 10000 ل س:

```
SELECT title, type, price, sales
FROM titles
WHERE type= 'business' AND price < 1000 or sales > 10000
```

النتيجة:

title	type	price	sales
الاقتصاد العالمي	business	700	15000
الاقتصاد والتجارة الدولية	business	800	22000
البورصة وإدارة الأعمال	business	400	9000
المعلوماتية الحديثة	IT	400	50000
sql server 2000	IT	1200	15000

(5 row(s) affected)

- تستخدم العبارة ORDER BY من أجل عرض الأسطر المسترجعة في الاستعلام بشكل مرتب وذلك حسب الأعمدة المحددة في تلك العبارة، ويكون الترتيب تصاعدياً بشكل تلقائي "ASC"، مع العلم أنه يمكننا عرض النتيجة مرتبة تنازلياً من خلال استخدام العبارة "DESC"؛
- ينبغي استخدام العبارة ORDER BY بشكل صريح عندما نرغب بترتيب نتيجة الاستعلام، بحيث لا يمكن ضمان ترتيــب نَ هذه العبارة حتى مع استخدام DISTINCT أو GROUP BY؛

[2F]Comment: سنتحدث عن هذه العبارة لاحقاً

أمثلة:

مثال4:

فيما يلي عرض للمخطوط المستخدم للقيام بعملية استرجاع لكافة المؤلفين من مدينة "دمشق" بشكل مرتب تصاعدياً حــسب الاســم الأول ثم الكنية:

```
SELECT FNAME, LNAME
FROM AUTHERS
WHERE CITY= 'دمشق'
ORDER BY FNAME , LNAME
```

النتيجة:

FNAME LNAME

---------- ----------

حامد سامر

دياب سامر

حسن سناء

خليل عامر

سعيد عمر

أحمد فراس

معين يامن

سؤال1:

ماذا تتوقع أن تصبح النتيجة السابقة إذا ما أضفنا للاستعلام السابق العبارة DESC على الكنية، كما يلي:

```
SELECT FNAME, LNAME
FROM AUTHERS
WHERE CITY= 'دمشق'
ORDER BY FNAME ASC , LNAME DESC
```

سؤال2:

ماذا تتوقع أن تصبح النتيجة السابقة إذا ما أضفنا للاستعلام السابق العبارة LIKE، كما يلي:

```
SELECT FNAME, LNAME
FROM AUTHERS
WHERE CITY= 'دمشق'AND FNAME LIKE '%س_م'
ORDER BY FNAME , LNAME DESC
```

[3F]Comment: "السين" أولا ثم "_" ثم "ميم" ثم "%"

مثال5: فيما يلي إعادة للمخطوط المستخدم في المثال4 بطريقة أخرى للتعبير عن تتالي الأعمدة في العبارة ORDER BY:

```
SELECT FNAME, LNAME
FROM AUTHERS
WHERE CITY= 'دمشق'
ORDER BY 1,2
```

بحيث تعبر الأرقام عن ترتيب تتالي الأعمدة في عبارة الاستعلام.

- تستخدم العبارة TOP لكي نحصر عدد الأسطر التي نقوم باسترجاعها، بحيث يمكننا أن نحدد عدد معين من الأعمدة أو نسبة مئوية من الأسطر المسترجعة في الاستعلام؛

مثال6:

فيما يلي عرض للمخطوط المستخدم للقيام بعملية استرجاع لأعلى 5 كتب كلفةً مرتبة تنازلياً حسب السعر:

SELECT TOP *5 title, type, price*
FROM *titles*
ORDER BY *PRICE* **DESC**

النتيجة:

TITLE	TYPE	PRICE
مبادئ عمل الحاسوب	IT	1500
sql server 2000	IT	1200
الاقتصاد والتجارة الدولية	business	800
الاقتصاد العالمي	business	700
المعلوماتية الحديثة	IT	400

ملاحظة:

ينبغي الانتباه إلى أن العبارة TOP لا تسرّع الاستعلام، وذلك لأنها تعيد عدد الأسطر المطلوب بعد تنفيذ ذلك الاستعلام بالكامل.

- يمكننا استخدام العبارة WITH TIES مع العبارة TOP وذلك لكي نقوم باسترجاع الأسطر المكـــررة في حال وجودها؛

مثال7:

فيما يلي إعادة للمثال6 للقيام بعملية استرجاع أعلى 5 كتب كلفةً مرتبة تنازلياً حسب السعر ولكن الآن مع استخدام العبارة WITH TIES:

SELECT TOP 5 WITH TIES *title, type, price*
FROM *titles*
ORDER BY *PRICE* **DESC**

النتيجة:

TITLE	TYPE	PRICE
مبادئ عمل الحاسوب	IT	1500
sql server 2000	IT	1200
الاقتصاد والتجارة الدولية	business	800

الاقتصاد العالمي	business	700
البورصة وإدارة الأعمال	business	400
المعلوماتية الحديثة	IT	400
البرمجة	IT	400

- مع العلم أن التكرار يشمل فقط آخر سطر في الاستعلام، وأنه لا يمكن استخدام العبارة WITH TIES بدون العبارة ORDER BY.

SELECT INTO

CT [DISTINCT] [**TOP** *n* [**PERCENT**]] *column1* [**AS** *column_heading*] [, *column2* [**AS** *column_heading*] , …]
[**INTO** *new_table_name*]
FROM *table1* [[**AS**] *table_alias*]
LEFT | RIGHT | FULL} [**OUTER**]] **JOIN** *table2* [[**AS**] *table_alias2*] **ON** (*join_conditions*)] […]
[**WHERE** *search_conditions*]
[**GROUP BY** *aggregate_free_expression*]
[**HAVING** *search_condition*]
[ORDER BY *order_expression* **[ASC | DESC]**]
[COMPUTE *row_aggrigate* (*column_name*) **[, …]**
[BY *column_name* **[,** *column_name* **] …]**]

- يمكننا من خلال العبارة INTO المضمّنة في التعليمة SELECT أن نقوم بإنشاء جدول جديد وتخزين نتيجة الاستعلام فيه؛
- يتم إنشاء الجدول الجديد بنفس الأعمدة وأنماط المعطيات التي تم استخدامها في نتيجة الاستعلام؛
- يتم تحديد اسم الجدول الجديد الذي سيتم إنشاءه مباشرة بعد عبارة INTO، مع العلم أنَّ إضافة الرمز '#' أو الرمز '##' قبل اسم ذلك الجدول يشير إلى أنه جدول مؤقت.

- مثال؛

[DISTINCT] [**TOP** *n* [**PERCENT**]] *column1* [**AS** *column_heading*] [, *column2* [**AS** *column_heading*] , …]
[**INTO** *new_table_name*]
FROM *table1* [[**AS**] *table_alias*]
[[**INNER** | {**LEFT** | **RIGHT** | **FULL**} [**OUTER**]] **JOIN** *table2* [[**AS**] *table_alias2*] **ON** (*join_conditions*)] […]
[**WHERE** *search_conditions*]
[**GROUP BY** *aggregate_free_expression*]
[**HAVING** *search_condition*]
[ORDER BY *order_expression* **[ASC | DESC]**]
[COMPUTE *row_aggrigate* (*column_name*) **[, …]**
[BY *column_name* **[,** *column_name* **] …]**]

- يمكننا من خلال العبارة INTO المضمّنة في التعليمة SELECT أن نقوم بإنشاء جدول جديد وتخزين نتيجة الاستعلام فيه؛

- يتم إنشاء الجدول الجديد بنفس الأعمدة وأنماط المعطيات التي تم استخدامها في نتيجة الاستعلام؛

- يمكننا استخدام معظم خيارات عملية SELECT، سواء كانت ORDER BY أو GROUP BY أو التوابع التجميعية، باستثناء عبارة COMPUTE؛

- إذا وُجد في الاستعلام عمود ما ناتج عن تابع تجميعي أو عبارة عن عمود مُنشأ من خلال عمليات وصل محارف باستخدام التوابع المحرفية، فإنه ينبغي تحديد اسم ذلك العمود لكي يتم إنشاءه في الجدول الجديد، مع العلم أنه ينبغي إتباع أساليب التسمية المعروفة (راجع جلسة إنشاء الجداول)؛

- يتم تحديد اسم الجدول الجديد الذي سيتم إنشاءه مباشرة بعد عبارة INTO، مع العلم أنَّ إضافة الرمز '#' أو الرمز '##' قبل اسم ذلك الجدول يشير إلى أنه جدول مؤقت، ويتم بناءه في قاعدة المعطيات tempdb، وأنه لا بد للمستخدم الذي ينفذ هذا المخطوط من أن يمتلك سماحية إنشاء الجداول إذا ما أراد القيام بإنشاء جدول دائم؛

- **مثال:**

فيما يلي عرض للمخطوط المستخدم للاستعلام عن وسطي أسعار الكتب بحسب أنواعها المتوافرة، بحيث يتم تخزين ذلك الاستعلام في جدول جديد:

SELECT *type, avg(price)* **AS** *'Price Average'*
INTO *#type_avgPrices*
FROM *titles*
GROUP BY *type*

order by *'Price Average'* **desc**

SELECT * FROM *#type_avgPrices*

النتيجة:

type	Price Average
IT	875
business	633

(2 row(s) affected)

UNION 1

- نستطيع من خلال العبارة UNION أن نقوم بإجراء تجميع منطقي للأسطر الناتجة عن استعلامين أو أكثر، بحيث ينبغي على كل مجموعة أن تحتوي على نفس العدد من الأعمدة بالإضافة إلى أنه ينبغي أن تكون تلك الأعمدة متوافقة من حيث نمط المعطيات المستخدم في كل منها على الترتيب، وإلا فإنه لا بد من إجراء تحويلات مناسبة لكي يتحقق ذلك التوافق؛

- سيتم استخدام ترويسة الأعمدة الموصّفة في أول استعلام من الاستعلامات المكوّنة للعبارة UNION لتكوين الأسماء المعبّرة عن أعمدة النتيجة النهائية لعملية التجميع المنطقي تلك؛

- تقوم العبارة UNION تلقائياً بإزالة الأسطر المكررة من نتيجة الاستعلام النهائية، مع العلم أنـــه يمكننـــا إضـــافة العبـــارة UNION ALL لاسترجاع كافة الأسطر، حتى المكرر منها؛

- يمكننا ترتيب النتيجة النهائية باستخدام ORDER BY بعد آخر عبارة استعلام من مكونات العبارة UNION.

- نستطيع من خلال العبارة UNION أن نقوم بإجراء تجميع منطقي للأسطر الناتجة عن استعلامين أو أكثر، بحيث ينبغي على كل مجموعة أن تحتوي على نفس العدد من الأعمدة بالإضافة إلى أنه ينبغي أن تكون تلك الأعمدة متوافقة من حيث نمط المعطيات المستخدم في كل منها على الترتيب، وإلا فإنه لا بد من إجراء تحويلات مناسبة لكي يتحقق ذلك التوافق؛

- سيتم استخدام ترويسة الأعمدة الموصّفة في أول استعلام من الاستعلامات المكوّنة للعبارة UNION لتكوين الأسماء المعبّرة عن أعمدة النتيجة النهائية لعملية التجميع المنطقي تلك؛

- **مثال:**

فيما يلي عرض للمخطوط المستخدم للاستعلام عن أسماء الكتّاب والناشرين في مدينة دمشق:

```
(
SELECT fname 'الاسم', city 'المدينة'
FROM authers
WHERE city = 'دمشق'
)
UNION
(
SELECT fname 'الاسم الأول', city 'المدينة'
FROM publishers
WHERE city = 'دمشق'
)
ORDER BY fname
```

النتيجة:

الاسم	المدينة
سامر	دمشق
سامي	دمشق
سناء	دمشق
عامر	دمشق
عمر	دمشق
فراس	دمشق
نادر	دمشق
يامن	دمشق

(8 row(s) affected)

- تقوم العبارة UNION تلقائياً بإزالة الأسطر المكررة من نتيجة الاستعلام النهائية، مع العلم أنــه يمكننــا إضــافة العبــارة UNION ALL لاسترجاع كافة الأسطر، حتى المكرر منها؛

- **مثال:**

فيما يلي عرض للمخطوط المستخدم للاستعلام عن أسماء الكتّاب والناشرين في مدينة دمشق:

```
(
SELECT fname 'الاسم', city 'المدينة'
FROM authers
WHERE city = 'دمشق'
)
UNION ALL
(
SELECT fname 'الاسم الأول', city 'المدينة'
FROM publishers
WHERE city = 'دمشق'
)
ORDER BY fname
```

النتيجة:

الاسم	المدينة
سامر	دمشق
سامر	دمشق
سامي	دمشق
سناء	دمشق
عامر	دمشق
عمر	دمشق
فراس	دمشق
نادر	دمشق
يامن	دمشق

(9 row(s) affected)

- يمكننا ترتيب النتيجة النهائية باستخدام ORDER BY بعد آخر عبارة استعلام من مكونات العبارة UNION، مع العلم أن مكونات ORDER BY من أعمدة ينبغي أن تتوافق مع الأعمدة الموصّفة في عبارة الاستعلام الأولى، كما يمكننا أيضاً أن نستخدم الأرقام للتعبير عن ترتيب العمود الذي نرغب بإجراء عملية الفرز على أساسه.

```
(
SELECT fname 'الاسم', city 'المدينة'
FROM authers
```

```
WHERE city = 'دمشق'
)
UNION ALL
(
SELECT fname 'الاسم الأول', city 'المدينة'
FROM publishers
WHERE city = 'دمشق'
)
ORDER BY 1
```

GROUP BY and HAVING

SELECT [DISTINCT] [**TOP** *n* [**PERCENT**]] *column1* [**AS** *column_heading*] [, *column2* [**AS** *column_heading*] , …]
[**INTO** *new_table_name*]
FROM *table1* [[**AS**] *table_alias*]
[[**INNER** | {**LEFT** | **RIGHT** | **FULL**} [**OUTER**]] **JOIN** *table2* [[**AS**] *table_alias2*] **ON** (*join_conditions*)] […]
[**WHERE** *search_conditions*]
[**GROUP BY** *aggregate_free_expression*]
[**HAVING** *search_condition*]
[ORDER BY *order_expression* **[ASC | DESC]**]
[COMPUTE *row_aggrigate* (*column_name*) **[, …]**
[BY *column_name* **[,** *column_name* **] …]**]

- تستخدم كل من العبارتين GROUP BY و HAVING حصراً مع التوابع التجميعية؛
- تسمح العبارة GROUP BY بإجراء عمليات حسابية تجميعية على مجموعات جزئية مختلفة من الجداول؛
- عندما يحتوي الاستعلام على العبارة WHERE فإنه يتم تنفيذها قبل العبارة GROUP BY، أي أنه يتم فلترة الاســتعلام أولاً وبعد ذلك يتم إجراء عملية التجميع وحساب النتيجة المطلوبة؛
- تسمح لنا العبارة HAVING بتحديد عدد الأسطر المسترجعة من عبارة التجميع، أي أنه يتم فلترة الاستعلام بعد إجراء عملية التجميع وحساب النتيجة المطلوبة.

SELECT [DISTINCT] [**TOP** *n* [**PERCENT**]] *column1* [**AS** *column_heading*] [, *column2* [**AS** *column_heading*] , …]
[**INTO** *new_table_name*]
FROM *table1* [[**AS**] *table_alias*]
{**LEFT** | **RIGHT** | **FULL**} [**OUTER**]] **JOIN** *table2* [[**AS**] *table_alias2*] **ON** (*join_conditions*)] […]
[**WHERE** *search_conditions*]
[**GROUP BY** *aggregate_free_expression*]
[**HAVING** *search_condition*]
[ORDER BY *order_expression* **[ASC | DESC]]**
[COMPUTE *row_aggrigate* **(***column_name***) [, …]**
[BY *column_name* **[,** *column_name* **] …]]**

- تستخدم كل من العبارتين GROUP BY و HAVING حصراً مع التوابع التجميعية؛
- تسمح العبارة GROUP BY بإجراء عمليات حسابية تجميعية على مجموعات جزئية مختلفة من الجداول؛

- **مثال:**

فيما يلي عرض للمخطوط المستخدم للاستعلام عن وسطي أسعار الكتب بحسب أنواعها المتوافرة:

```
SELECT type, AVG(price)
FROM titles
GROUP BY type
```

النتيجة:

```
type
---------------------- -----------
business               633
COOK                   275
IT                     875
PSYCHOLOGY             275

(4 row(s) affected)
```

- عندما يحتوي الاستعلام على العبارة WHERE فإنه يتم تنفيذها قبل العبارة GROUP BY، أي أنه يتم فلترة الاستعلام أولاً وبعد ذلك يتم إجراء عملية التجميع وحساب النتيجة المطلوبة؛

- **مثال:**

فيما يلي عرض للمخطوط المستخدم للاستعلام عن وسطي أسعار الكتب بحسب أنواعها المتوافرة وذلك لناشر محدد:

SELECT *type*, **AVG**(*price*)
FROM *titles*
WHERE *publisher_id = 150*
GROUP BY *type*

النتيجة:

```
type
---------------------- -----------
IT                     400
PSYCHOLOGY             275

(2 row(s) affected)
```

- تسمح لنا العبارة HAVING بتحديد عدد الأسطر المسترجعة من عبارة التجميع، أي أنه يتم فلترة الاستعلام بعد إجراء عملية التجميع وحساب النتيجة المطلوبة، وذلك وفق قيود يتم تحديدها بعد عبارة HAVING؛

مثال:

فيما يلي عرض للمخطوط المستخدم للاستعلام عن وسطي أسعار الكتب بحسب أنواعها، والتي يزداد وسطي سعرها عن 500 ل.س:

SELECT *type*, **AVG**(*price*)
FROM *titles*
GROUP BY *type*
HAVING AVG(*price*) > *500*

النتيجة:

```
type
---------------------- -----------
business               633
IT                     875

(2 row(s) affected)
```

ROLLUP

- تزوّدنا العبارة ROLLUP بعمليات تجميعية أو ما يُعرف بعمليات تجميعية فائقة ضمن عبارة GROUP BY نفسها؛
- تستخدم العبارة ROLLUP لاستخلاص التجميعات التراكمية ضمن مجموعة أسطر، أو بأسلوب آخر، تقوم العبارة ROLLUP بإنشاء تجميعات من اليمين إلى اليسار مروراً بكافة الأعمدة المحددة في عبارة GROUP BY، بحيث يتم تطبيق التابع التجميعي على كل مجموعة من الأصغر إلى الأكبر؛

- **مثال**.

- تزوّدنا العبارة ROLLUP بعمليات تجميعية أو ما يُعرف بعمليات تجميعية فائقة ضمن عبارة GROUP BY نفسها؛
- تستخدم العبارة ROLLUP لاستخلاص التجميعات التراكمية ضمن مجموعة أسطر، أو بأسلوب آخر، تقوم العبارة ROLLUP بإنشاء تجميعات من اليمين إلى اليسار مروراً بكافة الأعمدة المحددة في عبارة GROUP BY، بحيث يتم تطبيق التابع التجميعي على كل مجموعة من الأصغر إلى الأكبر؛

- **مثال**:

سنقوم أولاً باستعراض قائمة بالكتب ومجموع مبيعاتها مجمّعة بحسب الناشر ثم نوع الكتاب:

```
SELECT    publisher_id, type, SUM(sales) AS 'SUM_SALES'
FROM      titles
GROUP BY publisher_id, type
```

النتيجة:

publisher_id	*type*	SUM_SALES
150	business	46000
150	IT	75000
170	IT	4000
170	PSYCHOLOGY	3400
230	COOK	41400

سنضيف الآن العبارة ROLLUP إلى المخطوط السابق:

```
SELECT    publisher_id, type, SUM(sales) AS 'SUM_SALES'
FROM      titles
GROUP BY publisher_id, type WITH ROLLUP
```

النتيجة:

publisher_id	*type*	SUM_SALES
150	business	46000
150	IT	75000
150	NULL	121000
170	IT	4000
170	PSYCHOLOGY	3400
170	NULL	7400
230	COOK	41400
230	NULL	41400
NULL	NULL	169800

مجموع مبيعات كل تصنيف كتاب لكل ناشر
ومجموع مبيعات كل ناشر
ومجموع المبيعات ككل

- نلاحظ أن عملية التجميع تمت أولاً بالنسبة إلى publisher_id, type ثم بالنسبة إلى publisher_id ثم بالنسبة إلى كامل الأسطر ككل.

CUBE

- تزوّدنا العبارة CUBE بعمليات تجميعية أو ما يُعرف بعمليات تجميعية فائقة وعمليات إسنادات ترافقية ضمن عبارة GROUP BY نفسها؛

تختلف العبارة CUBE عن ROLLUP، فهي لا تقوم بإنشاء تجميعات من اليمين إلى اليسار فقط، في حين أنها تقوم بإنشاء أية تجميع يمكن أن ينشأ بين كافة الأعمدة الموصّفة في عبارة GROUP BY، بحيث يتحدد عدد التركيبات أو التجميعات الفائقة التي يمكن إنشاؤها بعدد الأعمدة الموصّفة في نتيجة الاستعلام.

- **مثال1**؛
- **مثال2**.
- تزوّدنا العبارة CUBE بعمليات تجميعية أو ما يُعرف بعمليات تجميعية فائقة ضمن عبارة GROUP BY نفسها؛
- تستخدم العبارة CUBE لاستخلاص التجميعات التراكمية ضمن مجموعة أسطر، أو بأسلوب آخر، تقوم العبارة CUBE بإنشاء تجميعات من اليمين إلى اليسار مروراً بكافة الأعمدة المحددة في عبارة GROUP BY، بحيث يتم تطبيق التابع التجميعي على كل مجموعة من الأصغر إلى الأكبر؛

- **مثال1**:

سنقوم أولاً باستعراض قائمة بالكتب ومجموع مبيعاتها مجمّعة بحسب الناشر ثم نوع الكتاب:

```
SELECT    publisher_id, type, SUM(sales) AS 'SUM_SALES'
FROM      titles
GROUP BY publisher_id, type
```

النتيجة:

publisher_id	*type*	SUM_SALES
150	business	46000
150	IT	75000
170	IT	4000
170	PSYCHOLOGY	3400
230	COOK	41400

سنضيف الآن العبارة CUBE إلى المخطوط السابق:

```
SELECT    publisher_id, type, SUM(sales) AS 'SUM_SALES'
FROM      titles
GROUP BY publisher_id, type WITH CUBE
```

النتيجة:

publisher_id	*type*	SUM_SALES
150	business	46000
150	IT	75000
150	NULL	121000
170	IT	4000
170	PSYCHOLOGY	3400
170	NULL	7400
230	COOK	41400
230	NULL	41400
NULL	NULL	169800
NULL	business	46000
NULL	COOK	41400
NULL	IT	79000
NULL	PSYCHOLOGY	3400

مجموع مبيعات كل تصنيف كتاب لكل ناشر

ومجموع مبيعات كل ناشر
ومجموع المبيعات ككل
مجموع مبيعات كل تصنيف كتاب

- **مثال2**:

سنقوم باستعراض قائمة بأنواع تصنيفات الكتب المتاحة ومجموع مبيعات كل منها:

```
SELECT   type, SUM(sales) AS 'SUM_SALES'
FROM     titles
GROUP BY type
```

النتيجة:

type	SUM_SALES
business	46000
COOK	41400
IT	79000
PSYCHOLOGY	3400

سنضيف الآن العبارة CUBE إلى المخطوط السابق:

```
SELECT   type, SUM(sales) AS 'SUM_SALES'
FROM     titles
GROUP BY type  WITH CUBE
```

النتيجة:

type	SUM_SALES
business	46000
COOK	41400
IT	79000
PSYCHOLOGY	3400
NULL	169800

مجموع مبيعات كل تصنيف كتاب

ومجموع مبيعات كافة التصنيفات

GROUPING Function

- تستخدم العبارتان CUBE و ROLLUP القيمة NULL للتعبير عن قيم تجميعية لعمود ما في نتيجة الاستعلام، كما مرّ معنا في الأمثلة السابقة، إلا أن هذا يمكن أن يؤدي إلى حدوث اختلاط ما بين قيم الـــ NULL المولّدة من خلال CUBE أو ROLLUP و قيم الـــ NULL التي تعبّر فعلاً عن قيم فارغة؛

- يعيد التابع GROUPING القيمة 1 عندما تكون قيمة العمود ناتجة عن تابع تجميعي، في حين يعيد القيمة 0 عندما يحتوي ذلك العمود على القيمة الفارغة؛

- **مثال1**

- **مثال2**

- تستخدم العبارتان CUBE و ROLLUP القيمة NULL للتعبير عن قيم تجميعية لعمود ما في نتيجة الاستعلام، كما مرّ معنا في الأمثلة السابقة، إلا أن هذا يمكن أن يؤدي إلى حدوث اختلاط ما بين قيم الـــ NULL المولّدة من خلال CUBE أو ROLLUP و قيم الـــ NULL التي تعبّر فعلاً عن قيم فارغة؛

- يعيد التابع GROUPING القيمة 1 عندما تكون قيمة العمود ناتجة عن تابع تجميعي، في حين يعيد القيمة 0 عندما يحتوي ذلك العمود على القيمة الفارغة؛

- **مثال1**:

سنقوم أولاً باستعراض قائمة بالكتب ومجموع مبيعاتها مجمّعة بحسب الناشر ثم نوع الكتاب:

```
SELECT   CASE WHEN GROUPING(publisher_id) =1 THEN
          'ALL PUBLISHERS'
         ELSE
          CONVERT(VARCHAR(20),publisher_id) END AS 'Pub_id',
         CASE WHEN GROUPING(type) =1 THEN
           'ALL TYPES'
         ELSE
        type END AS 'type',
         SUM(sales) AS 'SUM_SALES'
FROM       titles
GROUP BY publisher_id, type WITH CUBE
```

النتيجة:

Pub_id	*type*	SUM_SALES
150	business	46000
150	IT	75000
150	ALL TYPES	121000
170	IT	4000
170	PSYCHOLOGY	3400
170	ALL TYPES	7400
230	COOK	41400
230	ALL TYPES	41400
ALL PUBLISHERS	NULL	169800
ALL PUBLISHERS	business	46000
ALL PUBLISHERS	COOK	41400
ALL PUBLISHERS	IT	79000
ALL PUBLISHERS	PSYCHOLOGY	3400

- **مثال2**:

سنقوم باستعراض قائمة بأنواع تصنيفات الكتب المتاحة ومجموع مبيعات كل منها:

```
SELECT  CASE
WHEN GROUPING(type)=1 THEN 'ALL' ELSE   type
END as 'type_name',
SUM(sales) as 'SUM_SALES'
```

```
FROM      titles
GROUP BY type WITH CUBE
```

النتيجة:

type_name	SUM_SALESS
business	46000
COOK	41400
IT	79000
PSYCHOLOGY	3400
ALL	169800

ربط الجداول

- تسمح لنا T-SQL بالقيام بعملية ربط بين الجداول مع بعضها البعض باستخدام عملية JOIN؛
- عادة ما يتم الربط بين الجداول اعتماداً على عمود معين، بحيث غالباً ما يمثل ذلك العمود علاقة مفتاح أولي ومفتاح خارجي؛
- يمكن التعبير عن عملية الربط بين الجداول بطريقتين مختلفتين:
  - إما من خلال العبارة WHERE وهو الأسلوب الأكثر شيوعاً والأقدم؛
  - أو من خلال العبارة FROM، بحيث تتوافق هذه الطريقة مع المعيار ANSI-92.
- **مثال1:**

ربط بين جدولين اعتماداً على المفاتيح، باستخدام العبارة **WHERE**:

```
SELECT fname, lname, SUM(sales) AS 'sum'
FROM publishers P, titles T
WHERE P.pub_id = T.publisher_id
GROUP BY fname, lname
```

النتيجة:

fname	lname	sum
سامر	الأحمدي	41400
أمين	الحلو	7400
نادر	الرفاعي	121000

- **مثال2:**

ربط بين جدولين اعتماداً على المفاتيح، باستخدام العبارة **FROM**:

```
SELECT fname, lname, SUM(sales) AS 'sum'
FROM publishers P INNER JOIN titles T ON P.pub_id = T.publisher_id
GROUP BY fname, lname
```

النتيجة:

fname	lname	sum
سامر	الأحمدي	41400
أمين	الحلو	7400
نادر	الرفاعي	121000

- تختلف أنواع الربط المستخدمة بين الجداول، ويستخدم كل نوع لغرض محدد:
  - INNER JOIN:

يعتمد INNER JOIN على القيم المتماثلة تماماً، أثناء القيام بعملية الربط بين الجداول، بحيث يتم استرجاع الأسطر المتماثلة تماماً فقط؛

مثال:

راجع المثال2.

  - OUTER JOIN:

توجد ثلاثة أنواع للـــ OUTER JOIN وهي: LEFT و RIGHT و FULL OUTER JOIN، وفيه يتم إنشاء عملية الـــربط بين القيم الموجودة في أحد أعمدة الربط مع كافة القيم الموجودة في عمود الربط المقابل، وذلك حسب نوع الربط الخارجي المحدد؛

مثال3:

ربط خارجي من **اليسار** بين جدولين اعتماداً على المفاتيح، باستخدام العبارة FROM، وفيه يتم ربط كل قيمة من العمود اليساري مع مقابلاتها في العمود اليميني، بحيث يتم عرض كافة قيم العمود اليساري حتى ولو لم ترتبط بقيم مقابلة:

```
SELECT fname, lname, SUM(sales) AS 'sum'
FROM publishers  P LEFT OUTER JOIN titles T ON P.pub_id = T.publisher_id
GROUP BY fname, lname
```

النتيجة:

fname	lname	sum
سامر	الأحمدي	41400
أمين	الحلو	7400
نادر	الرفاعي	121000
سامي	الزين	NULL

NULL سعد حسام

وتفسير ذلك المثال:

أي أنه لا يوجد لكل من "سامي الزين" و "حسام سعد" من جدول publishers أية أسطر ترتبط بها في جدول titles.

مثال4:

ربط خارجي من **اليمين** بين جدولين اعتماداً على المفاتيح، باستخدام العبارة FROM، وفيه يتم ربط كل قيمة من العمود اليميني مع مقابلاتها في العمود اليساري، بحيث يتم عرض كافة قيم العمود اليميني حتى ولو لم ترتبط بقيم مقابلة:

```
SELECT fname, lname, SUM(sales) AS 'sum'
FROM publishers  P RIGHT OUTER JOIN titles T ON P.pub_id = T.publisher_id
GROUP BY fname, lname
```

النتيجة:

```
fname       lname           sum
----------- --------------- ---------------------
NULL NULL           1000
سامر    الأحمدي         41400
أمين            الحلو           7400
نادر            الرفاعي         121000
```

وتفسير ذلك المثال:

أي أنه يوجد جدول titles أسطر ليس لها مقابلات في جدول publishers.

- CROSS JOIN:

يُعرف هذا النوع من الارتباطات بين الجداول باسم الجداء الديكارتي، وفيه يتم ربط كل سطر من الجدول الأول بكافة الأسطر في الجدول الثاني؛

ينبغي الحذر أثناء التعامل مع هذا النوع من الارتباطات، خاصةً وأنه من الممكن أن يزداد حجم النتيجة بشكل كبير جداً حتى ولـــو كانت الجداول صغيرة نسبياً، فعلى سبيل المثال، إجراء جداء ديكارتي بين جدول بـــ500 سطر وجدول بـــ 1000 سطر ســـتكون النتيجة 500×1000= 500,000 سطر!!!؛

مثال5:

استرجاع أول 30 سطر من الجداء الديكارتي بين الجدولين المستخدمين في الأمثلة السابقة:

```
SELECT TOP 30 p.fname, t.type
FROM publishers p CROSS JOIN titles t
```

النتيجة:

```
fname  type
------------ --------------------
حسام    business
حسام    business
```

حسام business
حسام IT
حسام IT
حسام IT
حسام IT
حسام COOK
حسام COOK
حسام PSYCHOLOGY
حسام PSYCHOLOGY
حسام METH
نادر business
نادر business
نادر business
نادر IT
نادر IT
نادر IT
نادر IT
نادر COOK
نادر COOK
نادر PSYCHOLOGY
نادر PSYCHOLOGY
نادر METH
سامي business
سامي business
سامي business
سامي IT
سامي IT
سامي IT

- **ملاحظة:**

يمكن التعبير عن الدمج الخارجي اليساري و اليميني من خلال الرمزين =* و *= على الترتيب، بحيث يتم استخدامها فقط فـــي حالة إجراء الدمج الخارجي في عبارة where؛

مثال:

SELECT *fname, lname*, **SUM**(*sales*) **AS** 'sum'
FROM *publishers* P, *titles* T
WHERE *P.pub_id* ***=** *T.publisher_id*
GROUP BY *fname, lname*

الاستعلامات المضمّنة subqueries

- يمكننا تعريف الاستعلام المضمّن بكل بساطة على أنه استعلام محتوى في استعلام آخر؛
- يمكن كتابة الاستعلام المضمّن في أي مكان يسمح فيه بكتابة عبارة في الاستعلام الأساسي، مع العلم أنه توجد بعض القيـــود على الاستعلام المضمّن، بحيث ينبغي أن يعيد قيمة وحيدة، أو سطر وحيد في حالات خاصة؛
- ينبغي أن يتم كتابة الاستعلام المضمّن ما بين قوسين "(" ")"؛
- يمكننا استخدام استعلام جزئي يعيد عمود وحيد وأكثر من سطر وذلك ضمن عبارة IN؛
- يمكننا استخدام استعلام جزئي يعيد عمود وحيد وأكثر من سطر وذلك ضمن عبارة EXISTS؛
- من الممكن أيضاً أن نقوم باستخدام استعلامات جزئية في عبارة الاستعلام الأساسية.

- يمكننا تعريف الاستعلام المضمّن بكل بساطة على أنه استعلام محتوى في استعلام آخر؛
- يمكن كتابة الاستعلام المضمّن في أي مكان يسمح فيه بكتابة عبارة في الاستعلام الأساسي، مع العلم أنه توجد بعض القيـــود على الاستعلام المضمّن، بحيث ينبغي أن يعيد قيمة وحيدة، أو سطر وحيد في حالات خاصة؛
- ينبغي أن يتم كتابة الاستعلام المضمّن ما بين قوسين "(" ")"؛

- مثال:

فيما يلي عرض للمخطوط الذي يمثل مجموعة الكتب للناشر "نادر الرفاعي":


```
SELECT title, type, price
FROM titles
WHERE publisher_id = (
SELECT pub_id FROM publishers WHERE fname= 'نادر' AND lname='الرفاعي'
)
```

النتيجة:

title	type	price
الاقتصاد العالمي	business	700
الاقتصاد والتجارة الدولية	business	800
البورصة وإدارة الأعمال	business	400
مبادئ عمل الحاسوب	IT	1500
المعلوماتية الحديثة	IT	400
sql server 2000	IT	1200

- سيعيد المثال السابق رسالة خطأ في حال وجود أكثر من سطر كنتيجة للاستعلام المضمّن، أي في هذه الحالة في حال وجود ناشِرين باسم "نادر الرفاعي".

- يمكننا استخدام استعلام جزئي يعيد عمود وحيد وأكثر من سطر وذلك ضمن عبارة IN كما يلي:

مثال:

فيما يلي عرض للمخطوط الذي يمثل مجموعة الكتب العائدة للناشرين في مدينتي دمشق وحلب:

```
SELECT title, type, price, city
FROM titles as t INNER JOIN publishers as p ON p.pub_id = t.publisher_id
WHERE publisher_id IN (
SELECT pub_id FROM publishers WHERE city='دمشق' OR city='حلب'
)
```

النتيجة:

title	type	price	city
الاقتصاد العالمي	business	700	دمشق
الاقتصاد والتجارة الدولية	business	800	دمشق

البورصة وإدارة الأعمال	business	400	دمشق
مبادئ عمل الحاسوب	IT	1500	دمشق
المعلوماتية الحديثة	IT	400	دمشق
sql server 2000	IT	1200	دمشق
فن الطهو	COOK	200	حلب
المأكولات البحرية	COOK	350	حلب

- يمكننا استخدام استعلام جزئي يعيد عمود وحيد وأكثر من سطر وذلك ضمن عبارة EXISTS كما يلي:

مثال:

فيما يلي عرض للمخطوط الذي يمثل مجموعة الكتب العائدة للناشرين في مدينة دمشق:

SELECT *title, type, price*
FROM *titles* t **WHERE EXISTS** (
SELECT * **FROM** *publishers p* **WHERE** *t.publisher_id = p.pub_id* **and** *city='دمشق'*
)

النتيجة:

title	type	price
الاقتصاد العالمي	business	700
الاقتصاد والتجارة الدولية	business	800
البورصة وإدارة الأعمال	business	400
مبادئ عمل الحاسوب	IT	1500
المعلوماتية الحديثة	IT	400
sql server 2000	IT	1200

إن الاستعلام الجزئي في المثال السابق لا يعيد في الحقيقة أسطراً على الرغم من وجود الرمز "*" ، إنما يعيد قيمة True أو False اعتماداً على وجود أو عدم وجود نتيجة لذلك الاستعلام، وذلك بسبب العبارة EXISTS؛

- **ملاحظة:**

يطلق على الاستعلام الجزئي في المثال السابق اسم الاستعلام الجزئي المترابط، وذلك نظراً لأنه يشير ضمنياً في عبارة WHERE الموجودة فيه، إلى جدول في الاستعلام الأساسي، (*t.publisher_id = p.pub_id*). يقوم SQL Server بتنفيذ الاستعلام الجزئي من أجل كل سطر من أسطر الاستعلام الأساسي، وعند أول وجود لنتيجة أو أسطر من الاستعلام الداخلي فلن يتابع SQL Server تنفيذ ذلك الاستعلام الجزئي وذلك بسبب تحقق شرط EXISTS؛

على الرغم من أن الاستعلام السابق يمكن تحقيقه باستخدام العبارة IN، إلا أنه يتم استخدام هذه الطريقة في الاستعلامات التي تتطلب مقارنة أكثر من عمود في آن واحد، نظراً لأن ذلك لا يمكن تحقيقه من خلال العبارة IN؛

- من الممكن أيضاً أن نقوم باستخدام استعلامات جزئية في عبارة الاستعلام الأساسية نفسها كما في المثال التالي:

مثال:

فيما يلي عرض لمخطوط يعرض مجموعة كتب مع أسعارها، ونلاحظ في النتيجة وجود أحد الكتب بدون تحديد سعر، بالتالي سنقوم بتغيير الاستعلام نفسه لنعرض وسطي الأسعار ككل لكافة الكتب التي لا تمتلك سعر محدد:

SELECT TOP 12 *title, type, price*
FROM *titles*
order by *type*

النتيجة:

title	type	price
الاقتصاد العالمي	business	700
الاقتصاد والتجارة الدولية	business	800
البورصة وإدارة الأعمال	business	400
فن الطهو	COOK	200
المأكولات البحرية	COOK	350
مبادئ عمل الحاسوب	IT	1500
المعلوماتية الحديثة	IT	400
sql server 2000	IT	1200
البرمجة	IT	400
حرب طروادة	METH	NULL
ابن خلدون	PSYCHOLOGY	350
علم الاجتماع في سطور	PSYCHOLOGY	200

الاستعلام الجديد:

SELECT *title, type*, **ISNULL**(*price* ,(**SELECT AVG**(*price*) **FROM***titles*))**AS** price
FROM *titles*

النتيجة الجديدة:

title	type	price
الاقتصاد العالمي	business	700

الاقتصاد والتجارة الدولية	business	800
البورصة وإدارة الأعمال	business	400
فن الطهو	COOK	200
المأكولات البحرية	COOK	350
مبادئ عمل الحاسوب	IT	1500
المعلوماتية الحديثة	IT	400
sql server 2000	IT	1200
البرمجة	IT	400
حرب طروادة	METH	590
ابن خلدون	PSYCHOLOGY	350
علم الاجتماع في سطور	PSYCHOLOGY	200

تمرين:

لنفترض أننا نرغب باستعراض الكتب التي ينخفض سعرها عن وسطي أسعار كافة الكتب الموجودة، اكتب الاستعلام المناسب لتلك العملية؟

الحل:

يمكن أن يخطر ببالنا مباشرة أن نحاول كتابة ذلك الاستعلام كما يلي:

Select *title* **from** *titles* **where** *price* **<** **avg**(*price*)

ولكن هذه التعليمة خاطئة، إذ لا يمكن كتابة تابع تجميعي في عبارة WHERE، بالتالي لابد هنا من كتابة استعلام جزئي لكي نستطيع الحصول على النتيجة المطلوبة، كما يلي:

رسالة الخطأ الناتجة:

Server: Msg 147, Level 15, State 1, Line 1

An aggregate may not appear in the WHERE clause unless it is in a subquery contained in a HAVING clause or a select list, and the column being aggregated is an outer reference.

Select *title* **from** *titles* **where** *price* **< (SELECT avg**(*price*) **from** *titles*)

النتيجة

title	price
البورصة وإدارة الأعمال	400
المعلوماتية الحديثة	400
البرمجة	400
فن الطهو	200
المأكولات البحرية	350
ابن خلدون	350
علم الاجتماع في سطور	200

اقتراح للطلاب:

يمكن استعراض أمثلة أكثر تنوعاً حول هذا الموضوع بالعودة إلى الـ Help الخـــاص بـــ SQL Server: SQL Server Books Online الذي يتم تنصيبه مع الأداة نفسها، راجع جلسة تنصيب SQL Server.

إضافة أسطر جديدة باستخدام التعليمة INSERT

- فيما يلي عرض بسيط للقواعد التي تعرف تعليمة الإضافة:

INSERT [INTO] *table_or_view* [(*column_list*)] VALUES (*data_values*)

- تستخدم التعليمة INSERT أسطر جديدة إلى جدول وحيد؛
- ينبغي أن يتوافق عدد القيم المدخلة مع عدد الأعمدة المحددة، كما ينبغي أن تتوافق الأنماط أيضاً.
- يمكننا تجنب تحديد أسماء الأعمدة إذا ما كنّا نقوم بإدخال كافة القيم سويةً؛
- يمكننا تحديد عدد معين من الأعمدة وإدخال قيمها فقط، بشرط أن تكون بقية الأعمدة تسمح بقيم فارغة NULLS أو تمتلك قيم تلقائية أو تتمتع بخاصة التزايد التلقائي، مع العلم أنه ينبغي الحفاظ على ترتيب الأعمدة في الجدول عند القيام بعملية الإدخال؛

- أمثلة عن أخطاء إدخال، ورسائل الخطأ التي توافقها:

- إضافة عدة أسطر إلى جدول ما باستخدام عبارة SELECT عوضاً عن VALUES.

- فيما يلي عرض بسيط للقواعد التي تعرف تعليمة الإضافة:

INSERT [INTO] *table_or_view* [(*column_list*)] VALUES (*data_values*)

- تستخدم التعليمة INSERT أسطر جديدة إلى جدول وحيد؛

- مثال:

إضافة سطر جديد إلى جدول titles:

INSERT INTO *titles*(*title, type, price, sales, pubDate, publisher_id*)
VALUES ('SQL Server Step by Step', 'IT', 1000, 0, null , 150)

- ينبغي أن يتوافق عدد القيم المدخلة مع عدد الأعمدة المحددة، كما ينبغي أن تتوافق الأنماط أيضاً.
- يمكننا تجنب تحديد أسماء الأعمدة إذا ما كنّا نقوم بإدخال كافة القيم سويةً؛
- يمكننا تحديد عدد معين من الأعمدة وإدخال قيمها فقط، بشرط أن تكون بقية الأعمدة تسمح بقيم فارغة NULLS أو تمتلك قيم تلقائية أو تتمتع بخاصة التزايد التلقائي، مع العلم أنه ينبغي الحفاظ على ترتيب الأعمدة في الجدول عند القيام بعملية الإدخال؛

أمثلة عن أخطاء إدخال، ورسائل الخطأ التي توافقها:

مثال 1:

عدم توافق في عدد القيم المدخلة والأعمدة الموصفة:

```
INSERT INTO titles(title, type, price, sales, pubDate, publisher_id )
VALUES ('SQL Server Step by Step', 'IT', 1000, 0 , 150 )
```

رسالة الخطأ:

```
Server: Msg 109, Level 15, State 1, Line 2
There are more columns in the INSERT statement than values specified in the VALUES
clause. The number of values in the VALUES clause must match the number of
columns specified in the INSERT statement.
```

مثال 2:

عدم توافق في نمط القيم المدخلة والأعمدة الموصفة:

```
INSERT INTO titles(title, type, price, sales, pubDate, publisher_id )
VALUES ('SQL Server Step by Step', 1000, 'IT', 0, null, 150 )
```

رسالة الخطأ:

```
Server: Msg 245, Level 16, State 1, Line 1
```

Syntax error converting the varchar value 'IT' to a column of data type int.

مثال3:

اسم عمود خاطئ:

```
INSERT INTO titles(title, type, priceee, sales, pubDate, publisher_id )
VALUES ('SQL Server Step by Step', 'IT', 1000, 0, null, 150 )
```

رسالة الخطأ:

```
Server: Msg 207, Level 16, State 1, Line 1
Invalid column name 'priceee'.
```

- لا نستطيع أن نقوم بإضافة أكثر من سطر وحيد من خلال استخدام عبارة VALUES مع عبارة INSERT، بحيث ينبغي كتابة عدة عبارات لإضافة عدة أسطر، إلا أنه يمكننا استخدام العبارة SELECT مع العبارة INSERT لإضافة عدة أسطر في آن واحد، ويتم ذلك كما يلي:
  - أولا ينبغي أن نحدد الجدول والأعمدة التي نرغب بإضافة المعطيات إليها، وذلك بعد عبارة INSERT؛
  - ثم ينبغي كتابة الاستعلام المناسب الذي يعيد نفس عدد الأعمدة المحددة ونفس نمط معطياتها.

مثال:

لنفترض وجود الجدول titles_archive، وهو يماثل تماماً بنية الجدول titles، ويستخدم لتخزين أرشيف معطيات. سنقوم الآن بكتابة المخطوط الذي يسمح لنا بإضافة مجموعة أسطر –تم استعلامها من الجدول titles– دفعة واحدة، إلى الجدول titles_archive:

```
INSERT INTO titles_archive
SELECT * FROM titles WHERE type = 'business'
```

ملاحظة:

لن تتم العملية السابقة بنجاح إذا ما قمنا باستخدام عبارة * SELECT و كان يوجد أحد الأعمدة في الجدول الهدف يتمتع بخاصة التزايد التلقائي، إذ لا يمكننا في هذه الحالة إضافة قيم جدول المصدر لذلك العمود، وستظهر الرسالة التالية:

```
Server: Msg 8101, Level 16, State 1, Line 1
An explicit value for the identity column in table 'titles1' can only be specified when a
column list is used and IDENTITY_INSERT is ON.
```

بالتالي ينبغي إيقاف تلك الخاصة ثم إضافة الأسطر ثم إعادة تشغيلها، مع العلم أن ذلك الحل لا يضمن نجاح العملية خاصةً ومع وجود قيود من نوع مفتاح أولي مثلاً على ذلك العمود.

تعديل الأسطر باستخدام التعليمة UPDATE

- تستخدم التعليمة UPDATE لتعديل الأسطر الموجودة في جدول وحيد، وتعتبر هذه العملية غير قابلة للعكس؛

- فيما يلي عرض بسيط للقواعد التي تعرف تعليمة التعديل:

UPDATE table_name
SET { column_name = *expression* } [,...n]
[**WHERE** < search_condition >]

- يتم في تعليمة الـــ UPDATE أولاً، تحديد الجدول الذي نرغب بتعديله، الأعمدة التي نرغب بتعديلها، القيم الجديدة، والأسطر التي ينبغي أن نطبق عليها العملية؛

- في حال تم إهمال العبارة WHERE في التعليمة UPDATE فإنه سيتم تطبيق التغيرات على كامل أسطر الجدول؛

- يمكننا كذلك أن نقوم بإجراء عملية ربط JOIN ضمن عملية التعديل؛

- يمكننا أيضاً أن نقوم بإجراء استعلامات جزئيّة في عملية التعديل؛

- تستخدم التعليمة UPDATE لتعديل الأسطر الموجودة في جدول وحيد، وتعتبر هذه العملية غير قابلة للعكس؛

- فيما يلي عرض بسيط للقواعد التي تعرف تعليمة الإضافة:

UPDATE table_name
SET { column_name = *expression* } [,...n]
[**WHERE** < search_condition >]

- يتم في تعليمة الـــ UPDATE أولاً، تحديد الجدول الذي نرغب بتعديله، الأعمدة التي نرغب بتعديلها، القيم الجديدة، والأسطر التي ينبغي أن نطبق عليها العملية؛

مثال:

UPDATE *titles*
SET price = *1000*
WHERE id = 15

- في حال تم إهمال العبارة WHERE في التعليمة UPDATE فإنه سيتم تطبيق التغيرات على كامل أسطر الجدول؛

- يمكننا كذلك أن نقوم بإجراء عملية ربط JOIN ضمن عملية التعديل؛

مثال:

سنقوم بزيادة سعر كافة كتب الناشر رقم 150 بنسبة 5%:

UPDATE *titles*

SET *price = price * 1.05*
FROM *publishers p* **join** *titles* **on** *pub_id = publisher_id*
WHERE *p.pub_id =150*

- يمكننا أيضاً أن نقوم بإجراء استعلامات جزئيّة في عملية التعديل؛

مثال:

سنقوم بزيادة سعر كافة كتب الناشرين في مدينة دمشق بنسبة 5%:

UPDATE *titles*
SET *price = price * 1.05*
WHERE publisher_id **IN** (**SELECT** *pub_id* **FROM** *publishers* **WHERE** *city='دمشق'*
)

حذف الأسطر باستخدام التعليمة DELETE

- تستخدم التعليمة UPDATE لإزالة الأسطر الموجودة في جدول وحيد؛
- فيما يلي عرض بسيط للقواعد التي تعرف تعليمة الحذف:

DELETE [FROM] *table_name*

[WHERE < search_condition >]

- في حال تم إهمال العبارة WHERE في التعليمة DELETE فإنه سيتم حذف كامل أسطر الجدول؛
- يمكننا كذلك أن نقوم بإجراء عملية ربط JOIN ضمن عملية التعديل؛
- يمكننا أيضاً أن نقوم بإجراء استعلامات جزئيّة في عملية التعديل؛
- تستخدم التعليمة UPDATE لإزالة الأسطر الموجودة في جدول وحيد؛
- فيما يلي عرض بسيط للقواعد التي تعرف تعليمة الحذف:

DELETE [FROM] *table_name*

[WHERE < search_condition >]

مثال:

DELETE FROM *titles*
WHERE price <= 100

- في حال تم إهمال العبارة WHERE في التعليمة DELETE فإنه سيتم حذف كامل أسطر الجدول؛

- يمكننا كذلك أن نقوم بإجراء عملية ربط JOIN ضمن عملية التعديل؛

مثال:

سنقوم بحذف كتب الناشر رقم 190:

```
DELETE FROM titles
FROM publishers p join titles on pub_id = publisher_id
WHERE p.pub_id =190
```

- يمكننا أيضاً أن نقوم بإجراء استعلامات جزئيّة في عملية التعديل؛

مثال:

سنقوم بحذف كافة كتب الناشرين في مدينة دمشق:

```
DELETE FROM titles
WHERE publisher_id IN ( SELECT pub_id  FROM publishers WHERE city='دمشق'
)
```

عنوان الموضوع:

Transact-SQL (2)

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

نتابع في هذا القسم استعراض لغة Transact-SQL الخاصة بنظام إدرة قواعد المعطيات SQL Server.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- توابع SQL Sever المحرفية والرياضية
- توابع التاريخ والوقت
- توابع المعطيات السامية
- توابع النظام
- توابع الأمن
- التوابع التجميعية
- البنى البرمجية

توابع SQL Server

- أضافت مايكروسوفت أكثر من 30 تابع جديد إلى الإصدار SQL Server 7.0 إلى جانب الكمية الكبيرة من التوابع الموجودة مسبقاً، كما أضافت أيضاً العديد من التوابع الأخرى إلى الإصدار SQL Server 2000 لدعم خصائص جديدة وتأمين إمكانيات إضافية للتوابع؛

- القواعد التي تُعرّف استخدام التوابع:

FUNCTION_NAME([*parameter1* [,*parameter2* [,…]]])

اسم التابع

معاملات التابع:
يمكن أن تكون المعاملات إما عبارة (اسم عمود أو تابع آخر) أو ثابت أو رماز خاص

- يعيد التابع قيمة معينة يعتمد نمطها على نوع التابع المستخدم؛

- يمكن تصنيف التوابع في عدّة مجموعات حسب وظيفتها، وهي:

  - التوابع المحرفيّة
  - التوابع الرياضية
  - توابع التاريخ والزمن
  - توابع المعطيات السامية
  - توابع النظام
  - توابع الأمن
  - توابع النص والصورة
  - توابع مجموعة الأسطر
  - توابع Niladic
  - التوابع التجميعية

- أضافت مايكروسوفت أكثر من 30 تابع جديد إلى الإصدار SQL Server 7.0 إلى جانب الكمية الكبيرة من التوابع الموجودة مسبقاً، كما أضافت أيضاً العديد من التوابع الأخرى إلى الإصدار SQL Server 2000 لدعم خصائص جديدة وتأمين إمكانيات إضافية للتوابع؛

- تقدم العديد من التوابع المُضافة خدمات يمكن الحصول عليها بعدّة طرق، فعلى سبيل المثال نستطيع من خلال التابع OBJECT_ID( ) أن نحصل على مُعرِّف غرض معين نعرف اسمه مسبقاً، مع العلم أنه بإمكاننا البحث عن معرّف ذلك الغرض من الجدول sysobjects؛

- على الرغم من وجود توابع ليس من الضروري استخدامها، كبعض التوابع الرياضية، فعلى سبيل المثال يمكننا حساب الجذر التربيعي من دون استخدام التابع المخصص لتلك العملية، إلا أن استخدام هذا التابع يُعدّ أكثر فعاليّة؛

- فيما يلي عرض للقواعد التي تُعرّف استخدام التوابع:

FUNCTION_NAME([*parameter1* [,*parameter2* [,…]]] **)**

اسم التابع

معاملات التابع:
يمكن أن تكون المعاملات إما عبارة (اسم عمود أو تابع آخر) أو ثابت أو رماز خاص

- يعيد التابع قيمة معينة يعتمد نمطها على نوع التابع المستخدم؛

- يمكن تصنيف التوابع في عدّة مجموعات حسب وظيفتها، وهي:
  - التوابع المحرفيّة؛
  - التوابع الرياضية؛
  - توابع التاريخ والزمن؛
  - توابع المعطيات السامية؛
  - توابع النظام؛
  - توابع الأمن؛
  - توابع النص والصورة؛
  - توابع مجموعة الأسطر؛

- توابع Niladic؛
- التوابع التجميعية.

سنستعرض في الشرائح التالية لمحة عن أهم تلك التوابع مع العلم أنه يمكن العودة للتوثيق الخاص بـــ SQL Server (Books On Line) للحصول على معلومات إضافية.

التوابع المحرفيّة

- تسمح لنا التوابع المحرفيّة بإجراء عمليات الدمج والتعاقب والمسح على سلاسل المحارف؛
- يعرض الجدول التالي قائمة بكافة التوابع المحرفيّة التي يوفرها SQL Server:

التابع	الوصف
ASCII (*char*)	ـــ ASCII للمحرف الأول من جهة اليسار لمعامل دخل هذا التابع؛ مثال: SELECT ASCII('ABC') النتيجة: 65
CHAR(*int*)	المحرفي للرقم المُدخل بحسب ترميز ASCII؛ مثال: SELECT CHAR('65') النتيجة: A
CHARINDEX(*char_pattern, char,* [*int_start*])	لمحرف المحدد في المعامل الأول بالنسبة إلى مجموعـــة المحـــارف ل الثاني، كما يمكننا أن نبدأ البحث اعتباراً من الرقم الـــذي يحـــدده صفر في حال عدم وجود تطابق؛ مثال: SELECT CHARINDEX('ABC','ABCDE ABC',2) النتيجة: 7
DIFFERENCE(*char1,char2*)	اوح بين 0 و 4 للدلالة على مدى الاختلاف اللفظي بـــين سلـــسلتي تبر القيمة 4 أن السلسلتين متقاربتان لفظياً بشكل كبير؛

	مثال: SELECT DIFFERENCE('university' , 'universe') النتيجة: 4
LEFT(*char, int*)	قيمة محرفية بطول يساوي الرقم المحدد، من يسار السلسلة المُدخلة؛ مثال: SELECT left('university', 3) النتيجة: uni
LEN(*char*)	تابع يعيد طول سلسلة محرفية، باستثناء الفراغات في آخر السلسلة؛
LOWER(*char*)	حوّل نمط المحارف اللاتينية في السلسة المُدخلة إلى الوضع الصغير؛
LTRIM(*char*)	تابع يعيد سلسة محرفية بعد اقتصاص الفراغات الموجودة في بداية تلك السلسلة؛
NCHAR(*int*)	المحرفي للرقم المدخل بحسب ترميز Unicode؛ مثال: SELECT NCHAR('1602') النتيجة: ق
PATINDEX(*char_pattern, char*)	المحرف المحدد في المعامل الأول بالنسبة إلى مجموعة المحارف ة في المعامل الثاني؛ أو صفر في حال عدم وجود تطابق؛
REPLACE(*char1, char2, char3*)	أي ورود للمعامل المحرفي الثاني في السلسة char1، بالمعامل مثال: SELECT REPLACE('ABC ABC AAA','BC','X') النتيجة: AX AX AAA
QUOTENAME(*char*, [*char_quote*])	دخل محاطة بقوسين من النوع []، وذلك يستخدم عندما نريد استخدام عبير عن معامل له نفس اسم محجوز معين. فإذا أردنا –على سبيل عمود معين يمثل تاريخ، بالاسم date، فإنه ينبغي أن نعبر عن ذلك مات بالشكل [date] لأن هذا الاسم يمثل كلمة محجوزة للأداة SQL المحرف الذي نرغب باستخدامه للإحاطة بسلسلة الدخل، من خلال ختياري؛

	مثال: SELECT QUOTENAME('NAME','"') النتيجة: 'NAME'
REPLICATE(*char, int*)	ر السلسلة المحددة في المعامل الأول بعدد مرات يــساوي المعامــل
REVERSE(*char*)	يستخدم لعكس المحارف في المحددة في معامل الدخل؛
RIGHT(*char, int*)	تابع يقوم بنسخ قيمة محرفية بطول يساوي الرقم المحدد، من يمين السلسلة المُدخلة؛
RTRIM(*char*)	محرفية بعد اقتصاص الفراغات الموجودة في نهاية تلك السلسلة؛
SOUNDEX(*char*)	أربعة محارف للدلالة على مدى الاختلاف اللفظي بــين سلــسلتي مثال: SELECT soundex('university') , soundex('universe') النتيجة: U516 U516
SPACE(*int*)	تابع يعيد سلسلة محرفية مكونة من فراغات تساوي الرقم المُدخل؛
STR(*float,* [*length,* [*decimal*]])	تابع يستخدم لتحويل قيمة رقمية إلى محرفية. طول القيمة الخرج التلقائي يساوي عشرة محارف، وعدد الخانات التلقائي بعد الفاصلة يساوي صفر؛ مثال: select str(40.04) النتيجة: 40
STUFF(*char1, start, length, char2*)	تابع يستخدم لوصل طول محدد من char2 إلى char1 اعتباراً من موقع محدد؛ مثال: select stuff ('ABC',2,3,'DEFG') النتيجة: ADEFG
SUBSTRING(*char, start, length*)	باص طول محدد من سلسلة محارف اعتباراً من موقع محدد؛ مثال: select substring('ABCDEFGH',2,3) النتيجة: BCD
UNICODE(*char*)	الــ Unicode للمحرف الأول من جهة اليسار لمعامل دخل هذا

التابع؛ مثال: SELECT UNICODE('قلب') النتيجة: 1602	
نمط المحارف اللاتينية في السلسة المُدخلة إلى الوضع الكبير؛	UPPER(*char*)

- يمكننا استخدام العملية + للوصل بين المحارف؛

- مثال:

سنستعرض فيما يلي للمخطوط الذي يستخرج الحرف الأول من اسم وكنية المؤلفين في جدول authors:

SELECT substring(*fname*,1,1)+'.'+**substring**(*lname*,1,1) as 'Initials'
FROM *authors*

النتيجة:

Initials

س.ح
ع.س
ع.خ
س.ح
ي.م
ف.أ
س.د

التوابع الرياضيّة

- تسمح لنا التوابع الرياضية بإجراء عمليات حسابية اعتماداً على قيم مُدخلة في معاملات تلك التوابع، وتُعيد قيماً رقمية كنتيجة لتلك الحسابات؛

- لم يقدم الإصدار SQL Server 2000 أية توابع رياضية جديدة عن الإصدار SQL Server 7.0؛

- يعرض الجدول التالي قائمة بكافة التوابع الرياضية التي يوفرها SQL Server:

الوصف	التابع
تابع يعيد القيمة المُطْلَقة للرقم المُدخل؛	ABS(*numeric*)
تابع يعيد قيمة arccosine للرقم المُدخل؛	ACOS(*float*)
تابع يعيد قيمة arcsine للرقم المُدخل؛	ASIN(*float*)
تابع يعيد قيمة arc tan للرقم المُدخل؛	ATAN(*float*)
تابع يعيد قيمة arc tan لظل الزاويتين المحددتين بالرقمين المُدخلين؛	ATAN2(*float1*, *float2*)
تابع يعيد أصغر قيمة صحيحة أكبر أو تساوي الرقم المُدخل؛	CEILING(*numeric*)
تابع يعيد قيمة التجيب للرقم المُدخل؛	COS(*float*)
تابع يعيد قيمة arc cotangent للرقم المُدخل؛	COT(*float*)
تابع يعمل على تحويل القيمة المدخلة من راديان إلى درجات؛	DEGREES(*numeric*)
تابع يعيد exp الرقم المُدخل؛	EXP(*float*)
تابع يعيد أكبر قيمة صحيحة أصغر أو تساوي الرقم المُدخل؛	FLOOR(*numeric*)
تابع يعيد اللوغاريتم الثنائي للرقم المُدخل؛	LOG(*float*)
تابع يعيد اللوغاريتم العِشري للرقم المُدخل؛	LOG10(*float*)
تابع يعيد القيمة π؛	PI( )
تابع يقوم برفع الرقم الأول إلى القوة المحددة بالرقم الثاني؛	POWER(*numeric1, numeric2*)
تابع يعمل على تحويل القيمة المدخلة من درجات إلى راديان؛	RADIANS(*numeric*)
تابع يعطي قيمة عشوائية بين الصفر والواحد، يعبر المعامل seed عن الأساس الذي يمكن الاعتماد عليه في بناء القيمة العشوائية؛	RAND([seed])
تابع يقوم بعمليات تقريب مختلفة للرقم المُدخل؛	ROUND(*numeric, length, func*)

	مثال: SELECT ROUND(123.4545, 2) النتيجة: 123.4500 مثال: SELECT ROUND(123.45, 2) النتيجة: 100,00 مثال: SELECT ROUND(150.75, 0) النتيجة: 151.00
SIGN(*numeric*)	يعيد 1 عندما يكون الرقم المُدخل موجباً و −1 عندما يكون سالباً و 0 عندما يكون صفر؛
SIN(*float*)	تابع يعيد قيمة الجيب للرقم المُدخل؛
SQUARE(*float*)	تابع يعيد قيمة مربّع للرقم المُدخل؛
SQRT(*float*)	تابع يعيد قيمة الجذر التربيعي للرقم المُدخل؛
	تابع يعيد قيمة ظل للرقم المُدخل؛

- كما يمكننا استخدام العمليات +، −، *، /، % بين التعابير الحسابية؛

توابع التاريخ والزمن

- تسمح لنا توابع التاريخ والزمن بالتحكم بالعمليات التي يمكننا تنفيذها على الأغراض من نمط datetime بحيث يكون دخلها تاريخ أو وقت أما خرجها فيمكن أن يكون قيماً محرفية أو رقمية أو تاريخ أو وقت؛

- تأخذ بعض أنواع توابع التاريخ والوقت معاملاً خاصاً يطلق عليه اسم جزء التاريخ، وهو يمثل جزءاً محدداً من النمط DATETIME بحيث يمكن أن يكون السنة أو الشهر أو اليوم أو الساعة أو الدقيقة أو الثانية أو جزء الثانية بالإضافة إلى قيم أخرى سنقوم بتوضيحها في الجدول التالي:

جدول جزء التاريخ:

جزء التاريخ	الاختصار	م التي يأخذها	الوصف
year	**yy, yyyy**	1753-9999	السنة
quarter	**qq, q**	1-4	ربع السنة (الفصل)
month	**mm, m**	1-12	الشهر
dayofyear	**dy, y**	1-366	ترتيب اليوم في السنة
day	**dd, d**	1-31	اليوم
week	**wk, ww**	1-53	الأسبوع
weekday	**dw**	1-7	ترتيب اليوم في الأسبوع
hour	**hh**	0-23	الساعة
minute	**mi, n**	0-59	الدقيقة
second	**ss, s**	0-59	الثانية
millisecond	**ms**	0-999	الجزء المئوي من الثانية

- يعرض الجدول التالي قائمة بكافة توابع التاريخ والزمن التي يوفرها SQL Server:

التابع	الوصف
DATEADD(*datepart, int, date*)	تابع يقوم بإضافة قيمة محددة بالمعامل الثاني إلى جزء التاريخ المحدد في المعامل الأول وذلك للتاريخ المحدد في المعامل الثالث، لـ DATETIME؛ مع العلم أن جزء التاريخ (أي المعامل الأول) يمكن أن يأخذ أحد القيم الموضّحة في جدول "جزء التاريخ"؛ مثال: SELECT DATEADD(YEAR ,2, '2000-09-25 14:00:000') النتيجة: 2006-09-25 14:05:00.000 مثال: SELECT DATEADD(DAY ,2, '2000-09-25 14:00:000') النتيجة: 2000-09-27 14:00:00.000 مثال: SELECT DATEADD(HOUR ,2, '2000-09-25 14:00:000') النتيجة: 2000-09-25 16:00:00.000

DATEDIFF(*datepart, date1, date2*)	تابع يعيد الفرق بين تاريخين محددين اعتماداً على جزء التاريخ المحدد، أي الفرق بالسنوات أو الأشهر أو الأيام أو الساعات أو الدقائق أو الثواني أو أجزاء الثانية بالإضافة إلى قيم أخرى (انظر جدول جزء التاريخ)، والخرج من نمط رقمي؛ مثال: SELECT DATEDIFF(HOUR,'2000-09-25 14:00:000', '2000-10-25 14:00:000') النتيجة: 720 مثال: SELECT DATEDIFF(MONTH,'2000-09-25 14:00:000', '2000-10-25 14:00:000') النتيجة: 1
DATENAME(*datepart, date*)	تابع يعيد اسم جزء التاريخ المحدد، إذا كان بالإمكان تسمية ذلك الجزء (أي اليوم أو الشهر)؛ مثال: SELECT DATENAME (WEEKDAY,'2000-09-25 14:00:000') النتيجة: Monday مثال: SELECT DATENAME (MONTH,'2000-09-25 14:00:000') النتيجة: September كما أنه يعيد قيمة جزء التاريخ المحدد؛ مثال: SELECT DATENAME (YEAR,'2000-09-25 14:00:000') النتيجة: 2000
TEPART(*datepart, date*)	تابع ترتيب جزء التاريخ المحدد من التاريخ المُدخل؛

مثال:
فيما يلي ترتيب الشهر في السنة المحددة بالتاريخ التالي:
SELECT DATEPART (MONTH,'2000-sep-25 14:00:000')
النتيجة:
9
مثال:
فيما يلي ترتيب الأسبوع بالتاريخ التالي:
SELECT DATEPART (WEEK,'2000-09-25 14:00:000')

النتيجة:
40
مثال:
فيما يلي ترتيب اليوم في الشهر بالتاريخ التالي:
SELECT DATEPART (DAY,'2000-09-25 14:00:000')
النتيجة:
25
مثال:
فيما يلي ترتيب اليوم في السنة بالتاريخ التالي:
SELECT DATEPART (DAYOFYEAR,'2000-09-25 14:00:000')
النتيجة:
269
مثال:
فيما يلي ترتيب اليوم في الأسبوع المحدد بالتاريخ التالي:
SELECT DATPART (WEEKDAY,'2000-09-25 14:00:000')

النتيجة:
2
في المثال السابق على أن ترتيب اليوم في 2تدل النتيجة
التاريخ المختار هو الثاني في ذلك الأسبوع، ويعتمد ذلك على
اليوم المعتمد لكي يكون بداية الأسبوع؛
يتحدد يوم بداية الأسبوع من خلال المعامل
بحيث يأخذ قيمة رقمية تدل على ذلك @@DATEFIRST
عن 2 عن يوم الاثنين، والقيمة 1اليوم، بحيث تعبر القيمة
عن الجمعة؛7الثلاثاء ... والقيمة
يمكننا تحديد يوم بداية الأسبوع كما يلي:
SET DATEFIRST 6
بحيث اعتبرنا يوم السبت هو يوم بداية الأسبوع، بالتالي، ومن
معاينة المثال التالي، يمكننا أن نستنتج أن يوم 01/01/2006
هو اليوم الثاني في الأسبوع، أي يوم الأحد (تأكد من ذلك
باستخدام تاريخ الحاسب لديك):

```
SELECT @@DATEFIRST 'Date First',
       DATEPART(dw, '2006-01-01')  'Date Order'

Date First      Date Order
----------      ----------
 6               2
```

تابع يعيد قيمة اليوم من الشهر من التاريخ المُدخل؛	DAY(*date*)
تابع يعيد تاريخ ووقت اليوم، أي التاريخ والوقت الحالي بحسب ما هو معرَّف على ساعة المخدِّم؛	GETDATE()
تابع يعيد تاريخ ووقت اليوم بحسب التوقيت العالمي أي بحسب توقيت غرينتش؛	GETUTCDATE()
تابع يعيد قيمة الشهر من التاريخ المُدخل؛	MONTH(*date*)
تابع يعيد قيمة السنة من التاريخ المُدخل؛	YEAR(*date*)

- يمكننا استخدام العمليات +، – لإضافة أو إنقاص يوم من تاريخ محدد:

مثال:

SELECT GETDATE( )'Today', **GETDATE**( )+5 'After 5 Days'

النتيجة:

```
Today                          After 5 Days
------------------------------ ------------------------------
2005-02-02 17:01:24.950        2005-02-07 17:01:24.950
```

توابع المعطيات السامية

- تستخدم توابع المعطيات السامية لاسترجاع معلومات خاصة تتعلق بقاعدة المعطيات والجداول بحد ذاتها، كاسترجاع أسماء الأعمدة أو أسماء الجداول أو مفاتيح الفهارس وغيرها؛

- تتعامل معظم توابع المعطيات السامية بشكل مباشر مع جداول النظام (راجع جلسة قواعد معطيات النظام)؛

- يُنصح عادةً باستخدام توابع المعطيات السامية بدلاً من استعلام جداول النظام بشكل مباشر، خاصة فيما يتعلق بالإصدارات المختلفة من SQL Server، إذ أن استخدام توابع SQL Server الخاصة بعمليات الاستعلام تلك يضمن عدم حدوث أخطاء فيما إذا حدثت تعديلات على جداول النظام؛

- يعرض الجدول التالي قائمة ببعض توابع المعطيات السامية التي يوفرها SQL Server:

الوصف	التابع
تابع يعيد طول عمود في جدول؛	COL_LENGTH(*table, column*)

	مثال: SELECT COL_LENGTH('authers','fname') النتيجة: 50
COL_NAME(*table_id, column_id*)	تابع يعيد اسم عمود بعد تحديد معرّف ذلك العمود ومعرّف الجدول الذي يحتويه؛
DB_ID([*db_name*])	تابع يعيد معرّف قاعدة المعطيات المحددة، أو معرّف قاعدة المعطيات الحالية؛
DB_NAME([*db_id*])	تابع يعيد اسم قاعدة المعطيات ذات المعرّف المحدد، أو اسم قاعدة المعطيات الحالية؛
FILE_ID(*file_name*)	تابع يعيد معرّف ملف معطيات محدد؛
FILE_NAME(*file_id*)	تابع يعيد اسم ملف معطيات ذو معرّف محدد؛
FILEGROUP_ID(*filegroup_name*)	تابع يعيد معرّف مجموعة ملفات محددة؛
FILEGROUP_NAME(*filegroup*_id)	تابع اسم مجموعة ملفات محددة؛
OBJECT_ID(*object_name*)	تابع يعيد معرّف غرض محدد؛
OBJECT_NAME(*object_id*)	تابع يعيد اسم غرض ذو معرّف محدد؛

للمزيد من المعلومات حول توابع المعطيات السامية، يمكن الإطّلاع على التوثيق الخاص بـــ SQL Server (Books On Line)؛

توابع النظام

- تستخدم توابع النظام لاسترجاع معلومات خاصة تتعلق بقيم وخَيارات وإعدادات SQL Server؛
- يعرض الجدول التالي قائمة ببعض توابع النظام التي يوفرها SQL Server:

التابع	الوصف
APP_NAME( )	بيق الذي يقوم بتنفيذ هذا التابع؛ مثال: MS SQLEM - Data Tools SQL Query Analyzer
CAST(*expression* **AS** *dataType*)	اماً التابع CONVERT، ومهمته تحويل تعبير معين إلى نمط معطيات آخر؛

[4F]Comment: يوجد صورتين في أسفل الشريحة تابعتان لهذا الموضوع

مثال: DECLARE @myval decimal (5, 2) SET @myval = 100.05 SELECT CAST(@myval AS varbinary(20)) النتيجة 0x0502000115270000	
DB_ID([*db_name*])	تابع يعيد معرّف قاعدة المعطيات المحددة، أو معرّف قاعدة المعطيات الحالية؛
COALESCE(*expr1, [expr2, …]*)	تابع يعيد أول عبارة لا تساوي NULL في قائمة العبارات المحددة كمعاملات دخل؛
CONVERT(*dataType[(length)], expression, style*)	تابع يقوم بتحويل نمط العبارة المحددة إلى نمط المعطيات المُختار، اعتماداً على النموذج المحدد، ويستخدم عادةً من أجل تحويلات التاريخ والوقت؛

مثال:
عرض عمود السعر بنمط محرفي:

```
select title, type,convert(varchar(20),price)+' S.P' as 'price'
from titles
where type='business'
```

النتيجة:

title	type	price
الاقتصاد العالمي	business	771 S.P
الاقتصاد والتجارة الدولية	business	882 S.P
البورصة وإدارة الأعمال	business	441 S.P

مثال:
عرض التاريخ بهيئات مختلفة:
mon dd yyyy hh:miAM (or PM)

```
select convert(char, getdate(),100)
```

النتيجة:
Mar 25 2005 2:47PM

مثال:
mm/dd/yyyy

```
select convert(char, getdate(),101)
```

النتيجة:
03/25/2005

مثال:
yyyy.mm.dd

```
select convert(char, getdate(),102)
```

النتيجة:
2005.03.25
مثال:
Mon dd, yyyy

```
select convert(char, getdate(),107)
```

النتيجة:
Mar 25, 2005

مثال:
hh:mm:ss

```
select convert(char, getdate(),108)
```

النتيجة:
14:56:23

مثال:
yy/mm/dd

```
select convert(char, getdate(),11)
```

النتيجة:
05/03/25

مثال:
التوقيت الهجري:
dd mon yyyy hh:mi:ss:mmmAM

select convert(char, getdate(),130)

النتيجة
3:06:40:553 1427 صفر 3PM
مثال:
dd/mm/yy hh:mi:ss:mmmAM

select convert(char, getdate(),131)

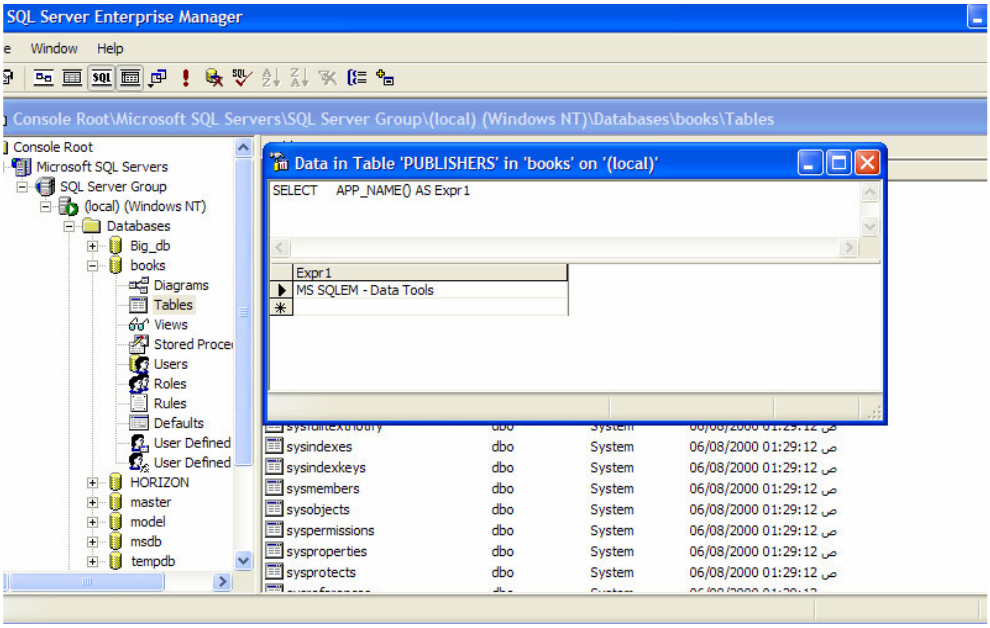
النتيجة
3:07:26:590 1427/02/3PM

DATALENGTH(*expression*)	تابع يعيد طول العبارة المحددة، بحيث يتضمن ذلك حتى الفراغات؛ datalength((select title from titles where id = 2)) select datalength(' abc ')
HOST_ID( )	تابع يعيد معرّف الحاسب المضيف؛
HOST_NAME( )	تابع يعيد اسم الحاسب المضيف؛
ISDATE(*char*)	تابع يعيد القيمة 1 إذا كان المعامل المحرفي المُدخل يعبر عن أحد أشكال DATETIME الصحيحة، ويعيد القيمة 0 في خلاف ذلك؛
ISNULL(*expression, value*)	القيمة value المحددة في المعامل الثاني، فيما إذا كانت العبارة المحددة في المعامل الأول تساوي قيمة فارغة NULL؛ select title, type, isnull(price, 0) as 'Price' title type price ---------- ---------- ----------

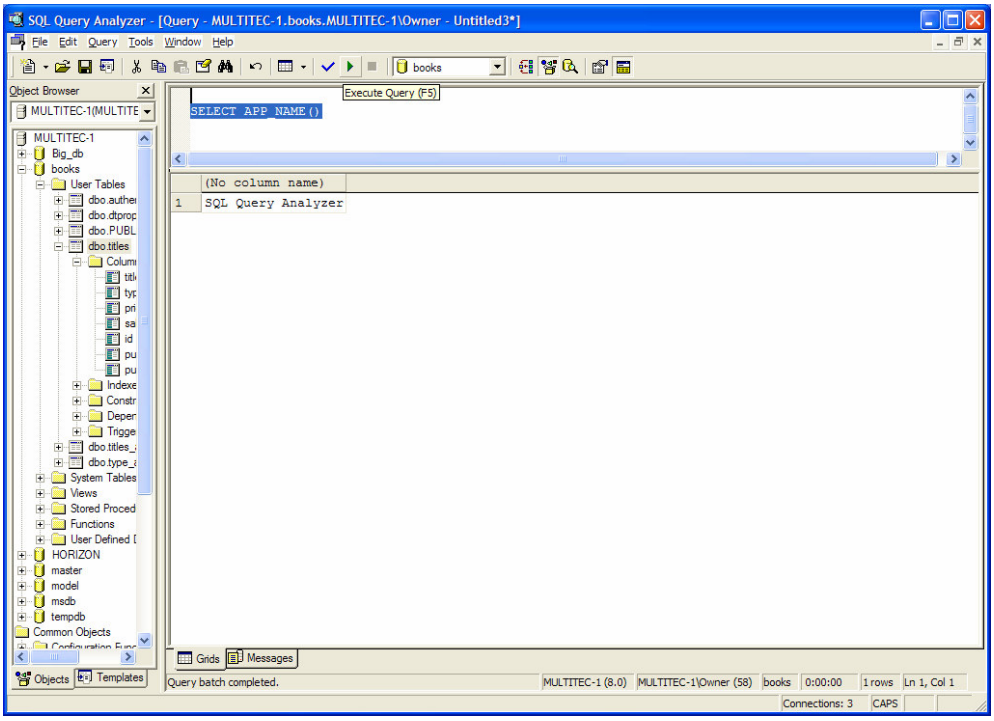
	الاقتصاد العالمي business 771 الاقتصاد والتجارة الدولية business 882 البورصة وإدارة الأعمال business 441 مبادئ عمل الحاسوب IT 1653 المعلوماتية الحديثة IT 441 sql server 2000 IT 1323 البرمجة IT 400 فن الطهو COOK 200 المأكولات البحرية COOK 350 ابن خلدون PSYCHOLOGY 350 علم الاجتماع في سطور PSYCHOLOGY 200 حرب طروادة METH 0 SQL Server Step by Step IT 1102 مثال2: select title,type, isnull(cast(price as char),'NA')as 'Price' from titles النتيجة: title type price ---------- ---------- ---------- الاقتصاد العالمي business 771 الاقتصاد والتجارة الدولية business 882 البورصة وإدارة الأعمال business 441 مبادئ عمل الحاسوب IT 1653 المعلوماتية الحديثة IT 441 sql server 2000 IT 1323 البرمجة IT 400 فن الطهو COOK 200 المأكولات البحرية COOK 350 ابن خلدون PSYCHOLOGY 350 علم الاجتماع في سطور PSYCHOLOGY 200 حرب طروادة METH NA SQL Server Step by Step IT 1102
ISNUMERIC(*char*)	تابع يعيد القيمة 1 إذا كان بالإمكان تحويل المعامل المحرفي المُدخل إلى رقم، ويعيد القيمة 0 في خلاف ذلك؛
NULLIF(*expr1, expr2*)	تابع يعيد القيمة NULL إذا تساوى المعاملان المُدخلان؛

للمزيد من المعلومات حول توابع النظام، يمكن الإطّلاع على التوثيق الخاص بـــ SQL Server (Books On Line)؛

ملحق:



استخدام Enterprise Manager لتنفيذ التابع APP_NAME()



استخدام Query Analyzer لتنفيذ التابع APP_NAME()

توابع الأمن

- تستخدم توابع الأمن لاسترجاع معلومات خاصة تتعلق بالمستخدمين والأدوار؛
- يعرض الجدول التالي قائمة ببعض توابع النظام التي يوفرها SQL Server:

التابع	الوصف
HAS_DBACCESS(*database*)	تابع يعيد معلومات فيما إذا كان المستخدم الحالي يمتلك سماحيات الولوج إلى قاعدة المعطيات المحددة؛
IS_MEMBER(*group* | *role*)	تابع يعيد القيمة 1 فيما إذا كان المستخدم الحالي عضواً في المجموعة أو الدور المحدد، الولوج إلى قاعدة المعطيات المحددة، ويعيد القيمة 0 في خلاف ذلك؛
IS_SRVROLEMEMBER(*role* [, *login*])	فيما إذا كان المستخدم الحالي مخول باستخدام دور المخدم المحدد، ويعيد القيمة ؛ مع العلم أنه بالإمكان تحديد معرّف مستخدم معين صراحةً؛

USER_ID([*username*])	تابع يعيد معرّف المستخدم الحالي، أو المستخدم المحدد؛
USER_NAME([*user_id*])	تابع يعيد اسم المستخدم الحالي، أو اسم المستخدم ذو المعّرف المحدد؛

للمزيد من المعلومات حول توابع الأمن، يمكن الإطّلاع على التوثيق الخاص بـــ SQL Server (Books On Line)؛

التوابع التجميعية

- تختلف التوابع التجميعية عن توابع SQL Server الأخرى بأنها تقوم بإجراء عمليات على مجموعة من الأسطر في عمـــود معين وتُعيد نتيجة تجميعية لتلك الأسطر؛

- يعرض الجدول التالي قائمة بأهم التوابع التجميعية التي يوفرها SQL Server:

التابع	الوصف
AVG([ALL | DISTINCT] *expression*)	تابع يعيد وسطي القيم المحددة بالعبارة المُدخلة؛
COUNT([ALL | DISTINCT] *expression* |*)	تابع يحصي عدد القيم التي لا تساوي NULL في العبارة المحددة، مع العلم أنه يتم أخذ القيم الفارغة بعين الاعتبار عندما يُستخدم الرمز *؛
COUNT_BIG ([ALL | DISTINCT] *expression* |*)	تابع يشبه التابع COUNT تماماً، إلا أن الخرج من نمط bigint؛
GROUPING(column*name*)	لأسطر التي يتم إضافتها على عبارة group by بعد استخدام cube أو rollup، ويعيد القيمة 0 في خلاف ذلك؛
MAX([ALL | DISTINCT] *expression*)	تابع يعيد القيمة الأكبر في العبارة المحددة؛
MIN([ALL | DISTINCT] *expression*)	تابع يعيد القيمة الأصغر في العبارة المحددة؛
SUM([ALL | DISTINCT] *expression*)	تابع يعيد مجموع القيم في العبارة المحددة؛

- عندما يتم استخدام القيمة DISTINCT مع التوابع السابقة، فإنه سيتم تجميع القيم غير المكررة، مع العلم أن القيمة ALL هي القيمة التلقائية؛
- لا يتم أخذ القيم الفارغة بعين الاعتبار ما لم يتم تحديد الرمز * في التابع التجميعي:

مثال:

يعرض المخطوط التالي قائمة بعدد كامل الكتب وعدد الكتب المُسعَّرة وعدد الكتب ذات الأسعار المختلفة فقط:

```
SELECT
COUNT(*) 'ALL BOOKS',
COUNT(price) 'ONLY PRICED BOOKS',
COUNT(DISTINCT price) 'DISTINCT PRICES'
FROM titles
```

النتيجة:

ALL BOOKS	ONLY PRICED BOOKS	DISTINCT PRICES
13	12	9

مثال:

يعرض المخطوط التالي قائمة بأعلى سعر وأقل سعر ومعدل وسطي أسعار الكتب الموجودة:

```
SELECT
MAX(price) 'MAX PRICE',
MIN(price)  'MIN PRICE',
AVG(price) 'AVERAGE PRICE '
FROM titles
```

النتيجة:

MAX PRICE	MIN PRICE	AVERAGE PRICE
1653	200	676

مثال:

يعرض المخطوط التالي قائمة بأعلى سعر وأقل سعر ومعدل وسطي أسعار الكتب الموجودة مع أخذ الكتب غير المسعَّرة بعين الاعتبار، بحيث يُعتبر سعر ذلك الكتاب يساوي 0:

```
SELECT
MAX(price) 'MAX PRICE',
MIN(price)  'MIN PRICE',
AVG(ISNULL(price,0)) 'AVERAGE PRICE '
FROM titles
```

النتيجة:

MAX PRICE	MIN PRICE	AVERAGE PRICE
1653	200	624

تُستخدَم عادة التوابع التجميعية مع عبارة GROUP BY كما في المثال التالي:

مثال:

```
SELECT type, AVG(price) 'AVERAGE PRICE '
FROM titles
GROUP BY type
```

النتيجة:

type	AVERAGE PRICE

business	698
COOK	275
IT	983
METH	NULL
PSYCHOLOGY	275

- استخدام العبارة **COMPUTE** والعبارة **COMPUTE BY**

تستخدم كلتا العبارتين compute و compute by للحصول على معلومات أكثر تفصيلاً ضمن عبارة الاختيار نفسها؛

تُستَخدم العبارة COMPUTE لتطبيق تابع تجميعي على عمود معين وإظهار النتيجة ضمن عبارة الاستعلام نفسها، كما في المثال التالي:

مثال:

سنعرض من خلال المخطوط التالي قائمة بأسماء الكتب وتصنيفاتها وأسعارها، بالإضافة إلى وسطي الأسعار في آخر القائمة:

SELECT *title,type,price*
FROM *titles*
COMPUTE AVG(*price*)

النتيجة:

title	type	price
الاقتصاد العالمي	business	771
الاقتصاد والتجارة الدولية	business	882
البورصة وإدارة الأعمال	business	441
مبادئ عمل الحاسوب	IT	1653
المعلوماتية الحديثة	IT	441
sql server 2000	IT	1323
البرمجة	IT	400
فن الطهو	COOK	200
المأكولات البحرية	COOK	350
ابن خلدون	PSYCHOLOGY	350
علم الاجتماع في سطور	PSYCHOLOGY	200

حرب طروادة	METH	NULL
SQL Server Step by Step	IT	1102

avg
==========
676

تُستَخدم العبارة COMPUTE BY مع العبارة ORDER BY وذلك لتطبيق تابع تجميعي على كل مجموعة جزئيــة وإظهــار النتيجة ضمن تلك المجموعة في عبارة الاستعلام نفسها، كما في المثال التالي:

مثال:

سنعرض من خلال المخطوط التالي قائمة بأسماء الكتب وتصنيفاتها وأسعارها، بالإضافة إلى أكبر سعر في كل مجموعــة جزئيــة متكوّنة بالإضافة إلى وسطي الأسعار في آخر القائمة:

```
SELECT title,type,price
FROM titles
ORDER BY type
COMPUTE MAX(price) BY type
COMPUTE AVG(price)
```

النتيجة:

title	type	price
الاقتصاد العالمي	business	771
الاقتصاد والتجارة الدولية	business	882
البورصة وإدارة الأعمال	business	441

max
==========
882

title	type	price
فن الطهو	COOK	200
المأكولات البحرية	COOK	350

max
==========
350

title	type	price
SQL Server Step by Step	IT	1102
البرمجة	IT	400
المعلوماتية الحديثة	IT	441
sql server 2000	IT	1323
مبادئ عمل الحاسوب	IT	1653

max
==========
1653

title	type	price
حرب طروادة	METH	NULL

max
==========

title	type	price
علم الاجتماع في سطور	PSYCHOLOGY	200
ابن خلدون	PSYCHOLOGY	350

max
==========
350

avg
==========
676

للمزيد من المعلومات حول التوابع التجميعية، يمكن الإطّلاع على التوثيق الخاص بـــ SQL Server (Books On Line).

البنى البرمجية

- تُصنّف عادةً اللغات التي تتعامل مع أنظمة إدارة قواعد المعطيات، في ثلاثة مجموعات رئيسية:
  - DML(Data Manipulation Language)، وهي اللغة التي تسمح بقراءة وتعديل المعطيات، كتعليمات SELECT, INSERT, UPDATE, DELETE؛
  - DDL(Data Definition Language)، وهي اللغة التي تسمح ببناء وتعديل البنى التخزينية، كتعليمات CREATE TABLE و ALTER TABLE؛
  - DCL(Data Control Language)، وهي اللغة التي تسمح بتعريف ومنح سماحيات الولوج إلى المعطيات، كتعليمات GRANT, REVOKE, DENY.

- تتضمن T-SQL المزيد من العبارات والتعليمات المفيدة الأخرى، كالعبارات التي تسمح بربط تعليمات DML مع بعــضها البعض ضمن إجرائية مُخزّنة ما؛

- **التعليمة IF:**

تسمح لنا التعليمة IF باختبار شرط معين وتنفيذ تعليمات معينة اعتماداً على تحققه؛

فيما يلي عرض للقواعد التي تعبّر عن التعليمة IF:

IF *Boolean_expression*
    Statement_block
ELSE
    Statement_block

تأخذ التعليمة IF معاملاً وحيداً هو عبارة عن تعبير بولياني ينبغي أن يُساوي القيمة true أو false؛
تعبر العبارة Statement_block في القواعد السابقة عن كتلة من التعليمات محاطة بتعليمتي BEGIN و END، كما أنه يمكنها أن تعبر عن تعليمة وحيدة، بالتالي ليس من الضروري في هذه الحالة استخدام BEGIN و END؛
إذا تحقق الشرط البولياني بالتالي سيتم تنفيذ كتلة التعليمات الأولى، وإذا لم يتحقق فسيتم تنفيذ الكتلة الثانية؛

- مثال:

سنقوم من خلال المخطوط التالي بحذف الجدول titles بعد اختبار وجوده في قاعدة المعطيات، (مع العلم أنه ينبغي الانتبــاه إلــى ضرورة وجود نسخة احتياطية من قاعدة المعطيات التي نقوم بإجراء الاختبارات عليها، خاصةً عنــدما نهــتم بمحتوياتهــا مــن المعطيات. راجع جلسة التخزين الاحتياطي واسترجاع المعطيات)

أحد توابع المعطيات المترفّعة، يُعيد معلومات حول معرّف الغرض المحدد في المعامل الأول وحول الخاصة المحددة في المعامل الثاني

```
IF OBJECTPROPERTY(OBJECT_ID('titles'), 'istable') = 1
BEGIN
        PRINT "Dropping table titles"
        DROP TABLE titles
END
ELSE
        PRINT "Table doesn't exists"
```

- **التعليمة WHILE و BREAK و CONTINUE:**

تسمح لنا التعليمة WHILE بتنفيذ مجموعة تعليمات بشكل متكرر مادام شرط تنفيذ تلك الحلقة محققاً؛

فيما يلي عرض للقواعد التي تعبّر عن التعليمة WHILE:

```
WHILE Boolean_expression
        Statement_block
```

تدل العبارة Statement_block في القواعد السابقة عن كتلة من التعليمات محاطة بتعليمتي BEGIN و END، كما أنه يمكنها أن تحتوي على التعليمتين BREAK و CONTINUE بحيث تؤدي الأولى عند تنفيذها إلى كسر الحلقة والخروج منها، أما الثانية فتعيد تنفيذ شرط الحلقة من البداية وتكرر التعليمات من جديد بغض النظر عن بقية التعليمات الموجودة في تلك الكتلة؛

- مثال:

سنقوم من خلال المخطوط التالي بزيادة أسعار الكتب في الجدول titles بنسبة 5% بحيث نستمر في الزيادة مادام هنالك كتباً سعرها أقل من 150 ل.س، كما أنه ينبغي الاستمرار في الزيادة حتى نصل لسعر وسطي أكبر من 1000 ل.س للكتاب إلا أننا سنكسر الحلقة في حال وصل سعر أحد الكتب إلى 2000 ل.س:

```
WHILE ( SELECT AVG(price) FROM titles) < 1000
BEGIN
        UPDATE titles SET price = price * 1.05

        IF (SELECT COUNT(*) FROM tiltes WHERE price < 150) > 0
                CONTINUE

        IF (SELECT MAX(price) FROM tiltes) > 2000
                BREAK

END
```

ملاحظة:

ظهرت الرسالة التالية في نتيجة تنفيذ العمليات السابقة:

Warning: Null value is eliminated by an aggregate or other SET operation.

(13 row(s) affected)

بحيث لم يتم تعديل الأسطر التي تحتوي القيمة NULL في عمود السعر؛

تمرين:

أضف التعديلات المناسبة إلى المخطوط السابق ليأخذ القيم الفارغة بعين الاعتبار بحيث يتم اعتبار تلك الأعمدة ذات سعر يساوي 50 ل.س لنتمكن من تطبيق التعديلات المطلوبة عليها؟

- **التعليمة CASE:**

تعتبر التعليمة CASE من تعليمات لغة SQL المعيارية، وهي عبارة عن تعليمة تقوم بمعالجة عدّة شروط وتعيد نتيجة وحيدة؛

يمكن استخدام العبارة CASE في أي مكان يُسمح فيه باستخدام قيمة أو عبارة ثابتة، أي في عبارة SELECT أو WHERE أو GROUP BY أو ORDER BY؛

يمكن استخدام التعليمة CASE بطريقتين مختلفتين، بالتالي يمكن التعبير عن قواعد تلك التعليمة CASE بأسلوبين:

```
CASE expression
        WHEN  value1  THEN result1
        [ WHEN  value2 THEN  result2 ]
        […]
        [ELSE resultN ]
        END
```

```
CASE
        WHEN  boolean_expression1 THEN expression1
        [ [ WHEN  boolean_expression2 THEN  expression2 ] [...] ]
        [ELSE expressionN ]
        END
```

تعتبر العبارة CASE فعّالة جداً في عمليات استبدال قيم المعطيات في الأعمدة، كما في المثال التالي:

- مثال:

سنقوم من خلال المخطوط التالي باستخدام العبارة CASE لعرض وصف حول الكتب التي تتوافر في الجدول titles وذلك بحسب تصنيفها، كما سنعرض الأسلوبين المختلفين للتعامل مع العبارة CASE:

```
SELECT title AS 'Book Title',
CASE type
        WHEN 'business'  THEN 'Business Related Books'
        WHEN 'IT'  THEN 'Information Technology Related Books'
        WHEN 'MYTH'  THEN 'Mythology Related Books'
        ELSE 'Other Books'
END AS 'Type', 'Price Rated to be '+

CASE
        WHEN price < 500  THEN 'Low'
        WHEN price BETWEEN 500  AND 2000 THEN 'Average'
        WHEN price > 2000 THEN 'High'
        ELSE 'Unknown'
END AS 'Price Rate'
FROM titles
```

النتيجة:

Book Title	Type	Price Rate
الاقتصاد العالمي	Business Related Books	Price Rated to be Average
الاقتصاد والتجارة الدولية	Business Related Books	Price Rated to be Average
البورصة وإدارة الأعمال	Business Related Books	Price Rated to be Average
مبادئ عمل الحاسوب	Information Technology Related Books	Price Rated to be Average
المعلوماتية الحديثة	Information Technology Related Books	Price Rated to be Average
sql server 2000	Information Technology Related Books	Price Rated to be High
البرمجة	Information Technology Related Books	Price Rated to be Low

فن الطهو	Other Books	Price Rated to be Low
المأكولات البحرية	Other Books	Price Rated to be Average
ابن خلدون	Other Books	Price Rated to be Average
علم الاجتماع في سطور	Other Books	Price Rated to be Low
حرب طروادة	Other Books	Price Rated to be Unknown
SQL Server Step by Step	Information Technology Related Books	Price Rated to be Average

يمكننا كذلك استخدام العبارة CASE لتعديل المعطيات، بحيث تعمل مثل التعليمة SET في UPDATE؛

- مثال:

سنقوم من خلال المخطوط التالي باستخدام العبارة CASE مع العبارة UPDATE لتعديل أسعار الكتب، بحيث نزيد سعر كتب الاقتصاد بنسبة 10% وكتب المعلوماتية بنسبة 8% وبقية لكتب بنسبة 5%:

```
UPDATE titles
SET price =
            CASE type
WHEN 'business'  THEN price*1.10
WHEN 'IT'  THEN price*1.08
            ELSE  price*1.05
END
```

- **تعليمات أخرى:**

تتوافر في SQL Server 2000 المزيد من التعليمات التي تستخدم لمعالجة البنى البرمجية، كالتعليمة GOTO التي تستخدم للقفز بين التعليمات (لا ينصح عادة باستخدام هذه التعليمة من وجهة نظر هندسية)، أو التعليمة WAITFOR التي تستخدم لتأخير التنفيذ بتأخير زمني محدد، أو التعليمة EXECUTE التي تستخدم لتنفيذ أو استدعاء الإجرائيات المخزّنة؛
بالإضافة إلى تعليمات أخرى يمكن استعراضها بتفصيل أكبر من خلال Books On Line؛

البنى البرمجية
(المتحولات)

- **المتحولات المحلّية:**

تسمح لنا المتحولات المحلية بتخزين القيم بشكل مؤقت، وتُستخدم بشكل كبير في الإجرائيات المخزّنة؛

يتم عادةً التصريح عن المتحول المحلي باستخدام العبارة DECLARE، كما يتم أيضاً تحديد نمط معطيات مناسب لذلك المتحـــوّل بحيث يمكن أن يكون ذلك النمط من أنماط النّظام أو من أنماط المستخدم المعرّفة (راجع جلسة بناء قاعدة المعطيات)؛

يتم عادةً الإشارة إلى المتحول المحلي من خلال الرمز @ بحيث يسبق اسم المتحول مباشرةً والذي يمكـــن أن يـــصل إلـــى 128 محرف؛

يُسند إلى المتحولات المحليّة القيمة الفارغة NULL كقيمة ابتدائية، كما أنه يمكن إسناد القيم إلى المتحولات من خلال العبارة SET أو SELECT، مع العلم أن الأولى تستخدم لإسناد قيمة ما لمتحول وحيد، في حين تُستخدم العبارة SELECT لإسناد القيم إلـــى متحول أو أكثر في آن واحد؛

- مثال:

```
DECLARE @user_msg varchar(255)
SET @user_msg = 'Empty Message'

PRINT @user_msg

SELECT @user_msg = 'There are '+
        CONVERT(varchar(5), (SELECT COUNT(*) FROM titles)) + ' books in the
titles table '
PRINT @user_msg
```

النتيجة:

```
Empty Message
There are 13 books in the titles table
```

لا يمكن أن يحتوي المتحول المحلي على أكثر من قيمة وحيدة، بحيث أن عملية الإسناد ستؤدي إلى تخزين القيمة الجديـــدة مكـــان القيمة السابقة، كما أن إجراء عملية إسناد من خلال عبارة SELECT معينة تُعيد مجموعة أسطر سيؤدي إلى تخزين قيمة السطر الأخير –من تلك الأسطر المسترجعة– في المتحول، مع العلم أنه في حال عدم استرجاع أي سطر فإنه لن يتم إسناد القيمة الفارغة إلى المتحول، بحيث سيحافظ على القيمة الأخيرة التي كان يمتلكها؛

- **المتحولات العامّة:**

استُخدِم مصطلح المتحولات العامة في الإصدارات السابقة من SQL Server للإشارة إلى مجموعة خاصة من التوابع، إلا أن هذه التسمية لم تكن بالمثالية، بحيث أن مصطلح المتحوّل العام يعبّر عن متحوّل خاص يتم تعريفة بحيث يمكن الوصول إليـــه مـــن أي إجراء أو تابع، إلا أن هذا لم يكن الهدف من "المتحولات العامة" التي نتحدّث عنها، لذلك فقد تم إلغاء هذه التسمية حالياً ليصبح اسم تلك المتحولات، توابع؛

تستخدم هذه التوابع للتعبير عن معلومات خاصة بالأداة SQL Server نفسها، بحيث يمكن استخدامها لأغراض مختلفة كاختبار رمز الخطأ الصادر عن آخر تعليمة تم تنفيذها على سبيل المثال؛

يتم عادةً الإشارة إلى هذا النوع من التوابع من خلال الرمز "@@"، مع العلم أنه يمكننا استخدام هذا الرمز للتصريح عن المتحولات المحليّة تماماً مثل الرمز"@"؛

تكمن أهمية هذه التوابع من خلال المعلومات التي تؤمّنها، بحيث أن بعض تلك التوابع يحتوي على معلومات لا يمكن الوصول إليها، كما أن بعضها الآخر يزوّدنا بمعلومات يصعب الوصول إليها بطريقة أخرى؛

سنعرض في الجدول التالي قائمة ببعض تلك التوابع، بحيث يمكننا استعراض بقية التوابع من خلال (Books On Line):

اسم التابع	الوصف
@@CURSOR_ROWS	عدد الأسطر في آخر CURSOR تم استخدامه؛
@@DATEFIRST	أول يوم في الأسبوع، بحيث يدل الرقم 7 على يوم الأحد و 1 على الاثنين وهكذا...
@@ERROR	رقم الخطأ الصادر عن آخر تعليمة تم تنفيذها، يفيد هذا التابع كثيراً لمتابعة الأخطاء المرتكبة في الإجرائيات المخزّنة أو القوادح؛
@@LANGID	رقم معرّف اللغة المستخدمة من خلال الاتصال الأخير المنشأ مع قاعدة المعطيات؛
@@LANGUAGE	اسم اللغة المستخدمة من خلال الاتصال الأخير المنشأ مع قاعدة المعطيات؛
@@PROCID	رقم معرّف الإجرائية المخزّنة التي يتم تنفيذها حالياً؛
@@ROWCOUNT	عدد الأسطر التي تأثرت بآخر تعليمة تم تنفيذها؛
@@SPID	رقم معرّف الاتصال الحالي المُنشأ؛
@@CONNECTIONS	عدد محاولات الولوج التي تمت على SQL Server منذ عملية التشغيل الأخيرة؛
@@TOTAL_ERRORS	عدد المرات التي حدث فيها أخطاء أثناء القراءة أو الكتابة، منذ عملية التشغيل الأخيرة لـ SQL Server؛
@@TOTAL_READ	عدد عمليات القراءة الفيزيائية، منذ عملية التشغيل الأخيرة لـ SQL Server؛
@@TOTAL_WRITE	عدد عمليات الكتابة الفيزيائية، منذ عملية التشغيل الأخيرة لـ SQL Server؛
@@DBTS	قيمة الخَتْم الأخير المُستخدم في قاعدة المعطيات الحالية؛
@@MAX_CONNECTIONS	عدد الاتصالات الأعظمي التي يمكن أن يدعمها الإصدار المنصَّب حالياً؛
@@MAX_PRECISION	الدقة الأعظمية التي يدعمها SQL Server في القيم ذات نمط المعطيات الرقمي أو العشري؛
@@VERSION	رقم إصدار SQL Server المُستخدم؛

@@SERVERNAME	اسم المخدّم المستخدم؛

CURSORS

- تعتبر لغة SQL –بعكس معظم لغات البرمجة الأخرى– على أنها لغة معالجة مجموعات، بحيث نقوم من خلالها باسترجاع مجموعات من الأسطر، وتعديل أو حذف مجموعات أخرى من الأسطر، بحيث تتحدد مجموعات الأسطر تلك بشروط البحث الموصّفة في الاستعلام؛

- اعتاد المبرمجون و لسوء الحظ على تطبيق العمليات على المعطيات على مستوى التسجيلات، إلا أن القيام بمثل هذه العمليات على معطيات SQL Server من خلال معالجة مجموعة أسطر ناتجة عن استعلام وحيد، وبشكل مباشر، لا يعد بالأمر البسيط، كما أنه يمكن أن يعتبر في بعض الحالات مستحيلاً؛

- فعلى سبيل المثال، لنفترض أننا نرغب بتعديل أسعار بعض الكتب في الجدول titles، بحيث نزيد سعر كافة الكتب التي يقل سعرها عن 500 ل.س بنسبة 15%، وننقص سعر كافة الكتب التي يزيد سعرها عن 500 ل.س بنسبة 10%، وننقص سعر كافة كتب الاقتصاد بنسبة 25%؛

لتطبيق التعديلات السابقة، نحتاج – من وجهة نظر معالجة لمجموعة أسطر– للقيام بثلاثة عمليات UPDATE منفصلة، كما يلي:

```
UPDATE titles
SET price = price * 1.15
WHERE price < 500
```

```
UPDATE titles
SET price = price * 0.9
WHERE price >= 500
```

```
UPDATE titles
SET price = price * 0.75
WHERE type = 'business'
```

نلاحظ مباشرة وجود مشكلة منطقية في المخطوط السابق، فلنفترض وجود كتاب سعره 480 ل.س، بالتالي سيتحقق شرط عملية التعديل الأولى وسيتم زيادة سعره بنسبة 15% ليصبح السعر الجديد لذلك الكتاب مساوياً لـ 552 ل.س، بالتالي، وعند تنفيذ عملية التعديل الثانية، سيخضع هذا الكتاب نفسه لشرط تلك العملية أيضاً، وسيصبح سعره الجديد مساوياً للقيمة 496.8 ...!!! مما يؤدي إلى حدوث خلل منطقي في المعطيات المخزّنة، ولكن ماذا لو كان ذلك الكتاب مصنّف من ضمن كتب الاقتصاد أيضاً، هنا أصبحت المشكلة أكثر تعقيداً ☹؛

ولكن، وبافتراض أننا لا نعرف بوجود العبارة CASE، فكيف نستطيع أن نتجنب ذلك الخطأ وأن نحل تلك المشكلة باستخدام أسلوب معالجة مجموعات الأسطر؟

سنناقش فيما يلي عدّة خيارات تسمح بحل المشكلة السابقة:

- يمكننا من وجهة نظر معينة، أن نقوم بنسخ الأسطر المختلفة إلى ثلاثة جداول مؤقتة، ومن ثم تعديل الأسطر المتغيرة في الجدول titles من خلال الدمج ما بين الجداول المؤقتة الثلاثة السابقة؛

لا يعتبر هذا الحل مثالياً، إذ أننا نقوم من خلاله أولاً بإجراء ثلاثة عمليات اختيار على الجدول titles لنقل المعطيات إلى الجداول المؤقتة بحيث نقوم ثانياً بإجراء عمليات INSERT لإضافة تلك المعطيات إلى الجداول المؤقتة، بعد ذلك نقوم بإجراء ثلاثة عمليات مسح للجدول titles من أجل نقل التعديلات إليه من الجداول المؤقتة؛

يعتبر هذا الحل مكلف جداً، وعلى الرغم من أن استبدال الجداول المؤقتة بمتحولات من نمط table قد يحسن من الأداء، إلا أننا ما نزال نحتاج للقيام بستة عمليات مسح للجدول titles لكي نحقق هدفنا؛

- يمكننا من وجهة نظر أخرى، أن نقوم بإضافة عمود من نمط bit يحتوي على القيمة 0 بشكل تلقائي، بحيث نخزن فيه القيمة 1 في الأسطر التي طرأ عليها تعديل في الجدول titles، ما يسمح لنا بعدم إجراء تعديلات أخرى على نفس الأسطر تلك عندما نقوم بإجراء بقية التعديلات؛

تحلّ هذه الطريقة المشكلة التي نواجهها، إلا أننا ما نزال نقوم بإجراء ثلاثة عمليات مسح منفصلة للجدول titles بالإضافة إلى عملية مسح رابعة لإلغاء قيم العمود الجديد بعد الانتهاء من العملية السابقة تلك؛

- يعتبر إجراء المعالجة السابقة باستخدام cursors أسهل وأبسط بكثير، إذ يمكننا من خلالها فحص كل سطر على حدى وتطبيق التعديلات المناسبة عليه اعتمادا على المعيار الذي يحققه؛
- نستطيع باستخدام CURSOR أن نقوم بالتنقل ما بين مكوّنات كل سطر من الأسطر، أو حفظ معطيات أحد الأعمدة في متحولات خاصة، أو تطبيق حسابات مختلفة وتنفيذ عمليات التعديل على الأسطر من خلال عملية مسح وحيده للجدول.

Cursors Syntax

- تعريف الـ Cursor:

يتم التصريح عن الـ Cursor من أجل عبارة اختيار محددة؛

فيما يلي القواعد التي تعبر عن المخطوط المعياري للتصريح عن الـ Cursors:

DECLARE *cursor_name* **[INSENSITIVE] [SCROLL]CURSOR**
FOR *select_statement*
[FOR { READ ONLY | UPDATE [OF *column_name* [,...n]] }]

فيما يلي القواعد التي تعبر عن أسلوب التصريح عن الـ Cursors في لغة T-SQL:

DECLARE *cursor_name* **CURSOR**
[LOCAL | GLOBAL]
[FORWARD_ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC | FAST_FORWARD]
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
[TYPE_WARNING]
FOR *select_statement*
[FOR UPDATE [OF *column_name* **[,...*n*]]]**

لا يمكننا استخدام العبارات COMPUTE أو COMPUTE BY أو BROWSE أو INTO ضمن عبارة الاستعلام *select_statement* في القواعد السابقة؛

- سنعرض فيما يلي مثالاً بسيطاً للمخطوط المستخدم للتصريح عن Cursor، وسنناقش مكونات ذلك المخطوط:

DECLARE *titles_cur* **CURSOR**
FOR SELECT *title, price ,type* **FROM** *titles*
FOR UPDATE OF *price*

- يوصف المثال السابق، Cursor محدد من خلال تعليمة اختيار مطبّقة على جدول وحيد؛
- ينبغي في كافة الاستعلامات أن يتم تحديد عدد الأسطر التي ينبغي أن يعالجها الـ Cursor من خلال عبارة WHERE، بحيث لا يفضل السماح للـCursor بمعالجة أسطر إضافية غير ضرورية، إذ أن زيادة عدد الأسطر تلك تؤثر سلباً على أداء الـCursor؛
- بما أننا نسعى من خلال المثال السابق لتعديل معطيات في جدول titles، بالتالي فإننا نعرف الـ Cursor ونصرّح أنه سيستخدم من أجل عمليات التعديل(**FOR UPDATE**)؛
- على الرغم من أن الـ Cursors تبنى بشكل تلقائي على أنها مخصصة لعمليات التعديل، ما لم يُحدد خلاف ذلك، إلا أنه يفضل التعريف صراحةً بأن الـ Cursor المُنشأ مخصص لتعديل المعطيات؛
- إذا لم نرغب بتعديل محتويات الأسطر في الـ Cursor الذي نستخدمة، بالتالي يمكننا التصريح عن ذلك الـ cursor على أنه للقراءة فقط (**FOR READ ONLY**)، مما يسمح للأداة SQL Server باتّباع استراتيجيات قَفْل مناسبة لضمان تلك العملية؛
- يتحدد من خلال *column_list* العمود أو الأعمدة التي سيتم تطبيق التعديلات عليها في الجدول المحدد، مع العلم أن عدم التصريح عن هذا المُعامل سيسمح للـ Cursor الجديد المُنشأ بأن يُجري التعديلات على كافة أعمدة ذلك الجدول؛
- تخضع عملية تسمية الـ Cursor إلى قواعد التسمية المعروفة.

- الـــ cursors المحلية والعامة:
  - تسمح خاصة جديدة مُضافة إلى الإصدار SQL Server 7.0 بإمكانية تعريف الـــ Cursor على أنه محلي أو عام؛
  - تبقى الـــ cursors العامّة معرّفة مادام الاتصال مع قاعدة المعطيات مستمراً، وذلك إما لغاية حذفها بـــشكل صـــريح أو لغاية حذفها ضمنياً نتيجةً لإنهاء الاتصال القائم؛
  - يتحدد مجال الـــ Cursor المحلي بالمكنان الذي تم تعريفه فيه، أي الإجرائية المخزّنة أو القادح على سبيل المثال؛
  - تبقى الـــ cursors المحلية معرّفة لغاية حذفها صراحةً أو انتهاء تنفيذ الإجرائية أو القادح الذي يحتويها؛
  - إذا لم يتم تحديد نوع الـــ Cursor بشكل صريح أثناء تعريفة، فإن النوع التلقائي الذي سيتم اختياره هو Cursor محلّي.

- الـــ cursors الساكنة والكثيفة:
  - تعتبر التسميتان السابقتان للـــCursor متماثلتان تماماً، بحيث تستخدم الأولى في قواعد T-SQL بينما تستخدم الثانية في قواعد SQL المعيارية؛
  - يتم تخزين المعطيات التي سيستخدمها الـــ Cursor في جدول مؤقت في قاعدة المعطيات tempdb، وعند تعريف الـــ Cursor ليكون ساكناً أو كثيفاً، فإننا نقوم بنسخ المعطيات التي ستتم معالجتها بغض النظر عن إمكانية حـــدوث تعـــديلات علـــى المعطيات الأصلية؛
  - تستخدم هذه الخاصّة عندما نرغب ببناء Cursor للقراءة فقط، ولا نريد لمعطيات ذلك الـــ Cursorأن تتـــأثر بتبـــدلات المعطيات المخزنة في قاعدة المعطيات أثناء سير عملية المعالجة من قبل الـــCursor ؛
  - يختلف الCursor الساكن عن الكثيف بأن الخاصة SCROLL هي الخاصة المفعّلة بشكل تلقائي فيه، في حين تتفعّـــل الخاصة FORWARD_ONLY في الـــ Cursorالكثيف بشكل تلقائي (سنناقش هذه الخاصة لاحقاً).

- خاصة KEYSET:
  - عند استخدام الخاصة KEYSET أثناء التصريح عن الـــ Cursor فسيتم نسخ قيم مفتاحية للأسطر التي تحقق معايير عبارة الاختيار المحددة بدلاً من نسخ كامل الأسطر تلك إلى الجدول المؤقت في قاعدة المعطيات tempdb؛
  - ينبغي لكي نستطيع استخدام هذه الخاصة أن يتوافر فهرس فريد على الجدول الذي نقوم بمعالجته؛
- الـــ Cursor المتغيّر:
  - لا تعتبر العلاقة في الـــ Cursorالمتغير ثابتة مع المعطيات، فعلى سبيل المثال إذا ما تمت إضافة أسطر جديدة بعد بناء الـــCursor فإنه سيتم جلب تلك الأسطر ليتم معالجتها في عملية استدعاء إضافية. وبالمثل، فأية أسطر يتم حـــذفها مـــن الجـــدول الرئيسي سيتم حذفها كذلك من الـــ Cursor قيد المعالجة؛
- خاصة FORWARD_ONLY:
  - وهي الخاصة التلقائية التي يتم تحديدها لـــ Cursor معين ما لم يتم تحديد أي خَيـــار أو نـــوع آخـــر، ويعتبـــر الـــــ Cursorالذي يتمتع بهذه الخاصة على أنه Cursor متغيّر يمكنه فقط أن يجلب السطر التالي لتتم معالجته؛
  - لا يمكن المرور على الأسطر أكثر من مرّة عندما نستخدم هذه الخاصة للـــ Cursor؛

- يعتبر هذا النوع من الـ Cursors هو الأسرع، إلا أنه لا يمكن أن تتم مقارنته مع تعليمة الـ SELECT من حيث الأداء؛

- فتح الـCursors :
  - يتم مباشرةً عند فتح الـ Cursor، إنشاء عملية الاختيار المضمّنة فيه؛
  - يتم في الـ Cursor الساكن أو الكثيف إنشاء جدول مؤقت في قاعدة المعطيات tempdb مباشرةً بعد تنفيذ عبارة OPEN؛
  - عندما يتم إنشاء الـCursor وفتحه، يتم تهيئة مؤشر ذلك الـ Cursor بحيث يتوضّع فوق السطر الأول؛
  - يمكننا استعراض عدد الأسطر التي يتكون منها الـ Cursor من خلال التابع Cursor_ROWS@@، بحيث تدل القيمة 1- على أن الـCursor المستخدم هو Cursor متغيّر أو له الخاصة FORWARD_ONLY؛
  - عندما يتم إغلاق الـCursor وإعادة فتحة، فإنه سيتم إعادة تنفيذ تعليمة SELECT المضمّنة فيه.

- التصريح عن المتحولات:
  - كيف تتم معالجة القيم المُسترجعة من الـ Cursor ؟
  - بإمكاننا وببساطة أن نعرض كل سطر للمستخدم كنتيجة معالجة ضمن الـCursor ، إلا أن ذلك قد لا يفي بالغرض دائماً، بحيث من الأفضل استرجاع المعطيات التي تعبر عن النتيجة التي يرغب بها المستخدم، بالتالي فلابد في بعض الحالات من استخدام متحولات محلية لتخزين معطيات بعض الأعمدة فيها بحيث نقوم بإجراء المعالجة المناسبة عليها وإعادتها بالشكل الملائم للمستخدم؛
  - يفضل عادةً استخدام أسماء متحولات في لـ Cursor شبيهة بأسماء الأعمدة التي تتم معالجتها، وذلك لتسهيل تذكر الغرض منها؛
  - يمكننا تعريف المتحولات داخل الـ Cursor بالطريقة التي استطلعناها مسبقاً (راجع شريحة تعريف المتحولات)؛

- جلب الأسطر:
  - بعد الانتهاء من عملية فتح الـ Cursor، يصبح بالإمكان البدء بعملية قراءة الأسطر من ذلك الـCursor وذلك من خلال التعليمة FETCH:

```
FETCH FROM titles_cur INTO @title, @price, @type
```

  - تقوم التعليمة FETCH بشكل تلقائي بجلب السطر التالي من الـCursor وتخزينه في المتحولات المحددة؛
  - في حال استخدام التعليمة FETCH مع عدد من المتحولات لا يساوي عدد الأعمدة المستعلمة في عبارة SELECT المستخدمة عند التصريح عن الـCursor ، فإنه سيتم إعادة رسالة الخطأ التالية:

```
Server: Msg 16924, Level 16, State 1, Line 21
Cursorfetch: The number of variables declared in the INTO list must match that of
selected columns.
```

- الـــ Cursor المُنزلق:

  - إذا قمنا باستخدام العبارة SCROLL أثناء التصريح عن الـــCursor ، فإننا بذلك نصرّح عن Cursor مُنزلـــق، وهـــو عبارة عن Cursor مزوّد بخاصة تمكّننا من الولوج إلى مكوناته من الأسطر بالطريقة التي نرغب بها، أي يمكننا تصفحه بطريقة الولوج المباشر إلى السطر الذي نريده من خلال التعليمة FETCH ؛
  - إن الاستخدام التلقائي للتعليمةFETCH في Cursor مُنزلق يؤدي إلى جلب السطر التالي كما هو الحـــال فـــي الأمثلـــة السابقة؛
  - نستطيع أن نستخدم خواص التعليمة FETCH لجلب الأسطر من عدّة مواقع، كما في الأمثلة التالية:

  - جلب السطر التالي:

FETCH NEXT **FROM** titles_cur **INTO** @title, @price, @type

  - جلب السطر السابق:

FETCH PRIOR **FROM** titles_cur **INTO** @title, @price, @type

  - جلب السطر الأول في مجموعة أسطر الـــ Cursor:

FETCH FIRST **FROM** titles_cur **INTO** @title, @price, @type

  - جلب السطر الأخير في مجموعة أسطر الـــ Cursor:

FETCH LAST **FROM** titles_cur **INTO** @title, @price, @type

  - جلب سطر ذو موقع محدد في مجموعة أسطر الـــ Cursor:

FETCH ABSOLUTE 10 **FROM** titles_cur **INTO** @title, @price, @type

  - تصفح الأسطر والالتفاف حولها في مجموعة أسطر الـــ Cursor:

FETCH RELATIVE -4 **FROM** titles_cur **INTO** @title, @price, @type

للانتقال أربعة أسطر للخلف بين أسطر الـ Cursor

FETCH RELATIVE 3 **FROM** titles_cur **INTO** @title, @price, @type

للانتقال ثلاثة أسطر للأمام بين أسطر الـ Cursor

ملاحظة:

- ينبغي عند استخدام تصفح الأسطر في أي نوع من أنواع الـــ Cursors الانتباه إلى الحالات التي نحاول فيها جلب سطر غير موجود أساساً؛

- يمكننا تجنب المشكلة السابقة من خلال استخدام شروط خاصة لاختبار وجود الأسطر التي نحاول جلبها؛
- يعيد التابع @@FETCH_STATUS القيمة 0 عندما تنجح عملية جلب السطر المحدد، ويعيد القيمة 1- عندما تفشل تلك العملية أو عندما يكون رقم السطر المحدد خارج مجال مجموعة أسطر الـ Cursor، ويعيد أيضاً القيمة 2- عندما يُفقد السطر الذي نحاول جلبه؛

مثال شامل:

- سنعرض فيما يلي المثال الذي طرحناه في الشريحة السابقة، وسنقوم بحل المشكلة التي كانت تواجهنا فيه من خلال استخدام الـCursor:

النص: لنفترض أننا نرغب بتعديل أسعار بعض الكتب في الجدول titles، بحيث نزيد سعر كافة الكتب التي يقل سعرها عن 500 ل.س بنسبة 15%، وننقص سعر كافة الكتب التي يزيد سعرها عن 500 ل.س بنسبة 10%، وننقص سعر كافة كتب الاقتصاد بنسبة 25%؛

الحل:

```
DECLARE titles_cur CURSOR
FOR SELECT title, price ,type FROM titles
FOR UPDATE  OF price

DECLARE @title varchar(50), @price int, @type varchar(50)

OPEN titles_cur

FETCH  FROM titles_cur INTO @title, @price, @type

IF(@@FETCH_STATUS = -1)
BEGIN
        PRINT 'NO BOOKS'
        CLOSE titles_cur
        DEALLOCATE titles_cur
```

التصريح عن الـCursor

التصريح عن المتحولات المحلية المستخدمة فى الـCursor

فتح الـCursor

جلب قيم السطر التالي

اختبار حالة عدم وجود أسطر فى الـCursor

إغلاق الـCursor وإلغاء الحيز التخزيني المحجوز له

```
        RETURN
END
WHILE (@@FETCH_STATUS = 0)
BEGIN
   IF  @type = 'business'
        begin
UPDATE titles SET price=price*0.75 WHERE CURRENT OF titles_cur
PRINT 'Business book updated'
end
   ELSE
   BEGIN
IF @price < 500
begin
UPDATE titles SET price = price * 1.15 WHERE CURRENT OF titles_cur
PRINT 'book priced less than 500 SP updated'
                        end
ELSE
begin
UPDATE titles SET price = price * 0.9 WHERE CURRENT OF titles_cur
PRINT 'book priced more than 500 SP updated'
                        end
   END
  FETCH  FROM titles_cur INTO @title, @price
END

IF (@@FETCH_STATUS = -2)
        RAISERROR('Attempt to fetch a row failed',16,1)
CLOSE titles_cur
DEALLOCATE titles_cur
```

الحلقة الرئيسية

عرض رسالة خطأ للتعبير عن حدوث مشكلة أثناء استرجاع السطر التالى

إغلاق الـ Cursor وإلغاء الحيز التخزيني المحجوز له بعد الانتهاء من تنفيذ كامل العملية

النتيجة:

```
(1 row(s) affected)
Business book updated
(1 row(s) affected)
Business book updated
(1 row(s) affected)
Business book updated
(1 row(s) affected)
book priced more than 500 SP updated
(1 row(s) affected)
book priced more than 500 SP updated
(1 row(s) affected)
book priced more than 500 SP updated
```

```
(1 row(s) affected)
book priced less than 500 SP updated
(1 row(s) affected)
book priced less than 500 SP updated
(1 row(s) affected)
book priced more than 500 SP updated
(1 row(s) affected)
book priced more than 500 SP updated
(1 row(s) affected)
book priced less than 500 SP updated
(1 row(s) affected)
book priced more than 500 SP updated
(1 row(s) affected)
book priced more than 500 SP updated
```

الفصل السادس عشر

عنوان الموضوع:

بناء وإدارة الإجرائيات المخزّنة وتوابع المستخدم المعرّفة والمناظير والقوادح.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

سنناقش في هذه الجلسة كيفية إنشاء وإدارة عدة أنواع من أغراض قواعد المعطيات, بحيث سنتناول بالتفصيل كلا من الإجرائيات المخزّنة وتوابع المستخدم المعرّفة والمناظير والقوادح؛

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- بناء وإدارة المناظير في SQL Server
  - تعريف
  - استخدامات المناظير
  - كيفية إنشاء المناظير:
    - إنشاء المناظير باستخدام T-SQL
    - إنشاء المناظير باستخدام الأداة Enterprise Manager

- ما هي المناظير المجزّأة والمناظير المفهرسة؟
- بناء وإدارة الإجرائيات المخزّنة في SQL Server
  - تعريف
  - فوائد استخدام الإجرائيات المخزّنة
  - مساوئ الإجرائيات المخزّنة
  - كيفية إنشاء الإجرائيات المخزّنة
    - إنشاء الإجرائيات المخزّنة باستخدام T-SQL
    - إنشاء الإجرائيات المخزّنة باستخدام الأداةEnterprise Manager
    - بناء وإدارة القوادح في SQL Server
  - تعريف
  - استخدامات القوادح
  - كيفية إنشاء القوادح:
    - إنشاء القوادح باستخدام T-SQL
    - إنشاء القوادح باستخدام الأداةEnterprise Manager
    - إنشاء القوادح باستخدام الأداة Query Analyzer
  - أنواع القوادح:
    - قوادح AFTER
    - قوادح INSTEAD OF
- بناء وإدارة توابع المستخدم المعرّفة في SQL Server
  - تعريف
  - كيفية إنشاء توابع المستخدم المعرّفة:
    - إنشاء توابع المستخدم المعرّفة باستخدام T-SQL
    - إنشاء توابع المستخدم المعرّفة باستخدام الأداةEnterprise Manager
    - إنشاء توابع المستخدم المعرّفة باستخدام الأداةQuery Analyzer

- تعريف:

يُعرف المنظار ببساطة على أنه استعلام مُخزّن كغرض في قاعدة المعطيات؛
يمكن وصف المنظار بأنه جدول افتراضي، وذلك للتشابه الكبير بينه وبين الجداول من حيث الأداء، لذلك، غالباً ما يُشار للمناظير في عبارات T-SQL على أنها جداول مع بعض الاستثناءات؛
تُستخدم المناظير لعرض مجموعة جزئية من المعطيات، أو بشكل أدقّ، لعرض مجموعة من الأسطر أو الأعمدة سواء كانت من جدول معين أو من عدّة جداول ترتبط مع بعضها البعض من خلال عبارات JOIN أو UNION؛

- وصف:

باستثناء المناظير المفهرسة التي تحدّثنا عنها في شرائح سابقة، فإن المناظير عبارة عن تعليمات اختيار مخزّنة، وهي بالتالي لا تشكل –افتراضياً– عبئاً إضافياً على قاعدة المعطيات، وذلك لأنها لا تخزّن محتوياتها من المعطيات بشكل فعلي؛
يعتبر الخطأ الذي يفترض أن المناظير تحتاج لحيز تخزين للمعطيات التي تعرضها، من أكثر الأخطاء شيوعاً بين المطورين، وهو من الأسباب الأساسية التي تؤدي إلى ابتعاد أولئك المطورون عن استخدام المناظير في قواعد المعطيات؛

- استخدامات المناظير:

يختلف الغرض من استخدام المناظير، إلا أننا سنعدد فيما يلي أهم الاستخدامات الشائعة لهذا النوع من أغراض قاعدة المعطيات:

- تبسيط عمليات استرجاع المعطيات في الاستعلامات المعقّدة؛
- تجنب التعامل المباشر مع الجداول؛
- التحكم بعمليات الولوج إلى المعطيات على مستوى الأسطر أو الأعمدة.

- استخدام المناظير من أجل تبسيط عمليات استرجاع المعطيات في الاستعلامات المعقّدة:

كثيراً ما تُستخدم عمليات الدمج المعقدة أو التوابع التجميعية أو توابع SQL المُضمّنة أو غيرها، في عمليات استعلام المعطيات، وإن تكرار وتواتر هذه العمليات على المعطيات قد يؤدي إلى زيادة تعقيد الاستعلامات مع مرور الوقت؛
يمكننا من خلال المناظير أن نقوم بإخفاء التعقيد الناتج عن تلك الاستعلامات من خلال تخزينها في منظار مناسب ثم استعلام المعطيات من ذلك المنظار مباشرة؛

- استخدام المناظير لإخفاء بنية الجداول التي نتعامل معها:

يمكن استخدام المناظير لتكون عبارة عن طبقة وسيطة ما بين التطبيقات والجداول الفعلية، ما يسمح بالتحكم بالولوج إلى المعطيات كما يسمح بإخفاء التغيرات –التي يمكن تطبيقها على الجداول– عن المستخدمين؛

فعلى سبيل المثال، لنفترض أن أحد التطبيقات يقوم باستخدام معطيات جدول معين. إن أي تغيير مستقبلي على الجدول سيؤدي بالضرورة إلى تعديل التطبيق الذي يستعلم ذلك الجدول لكي يوافق التغيرات الجديدة. أما في حال استخدام منظار كطبقة وسيطة بين التطبيق والجدول، فإنه ليس من الضروري إجراء أي تعديل على التطبيق، إذ أنه يكفي تعديل المنظار ليجلب المعطيات التي يحتاجها ذلك التطبيق. فمادام المنظار يُعيد المعطيات المناسبة، بالتالي لا يوجد أي ارتباط مباشر بين التطبيق وجداول قاعدة المعطيات؛

تبرز كذلك الفائدة من استخدام المناظير في الجداول المجزأة، ففي بعض أنواع قواعد المعطيات يتم تجزئة الجداول التي يزداد حجمها بشكل كبير جداً إلى عدّة أقسام من أجل تحسين الأداء، بحيث يتم –أثناء بناء الاستعلام– تحديد أي جزء نريد اختيار المعلومات منه، إلا أنه وباستخدام المناظير يمكننا إجراء عملية الاستعلام تلك وكأنها من جدول وحيد، وذلك من خلال ما يُعرف باسم المنظار المُجزّئ، والذي سنناقشه بشكل أدقّ في الشرائح التالية؛

- استخدام المناظير للتحكم بأمن المعطيات:

تُستخدم المناظير أيضاً لحصر عمليات الولوج إلى المعطيات وتبسيط عملية إدارة السماحيات، إذ أنه ليس من الضروري توصيف سماحيات الولوج إلى الجداول إذا ما تمت تلك العملية على المنظار الذي يستخدم تلك الجداول، بحيث يمكن إسناد كافة أنواع السماحيات من مستوى المنظار؛

يمكن تصنيف سماحيات الولوج باستخدام المناظير إلى نوعين أساسيين، هما:

- الحماية على مستوى الأعمدة:

بحيث يمكن السماح أو عدم السماح لمستخدمي المنظار بالولوج إلى معطيات عمود بالكامل؛

- الحماية على مستوى الأسطر:

بحيث يمكن السماح أو عدم السماح لمستخدمي المنظار بالولوج إلى أسطر محددة من عمود معين، وذلك بتطبيق شرط فلترة مناسب باستخدام العبارة WHERE على تلك الأسطر أثناء تعريف المنظار؛

- إنشاء المناظير:

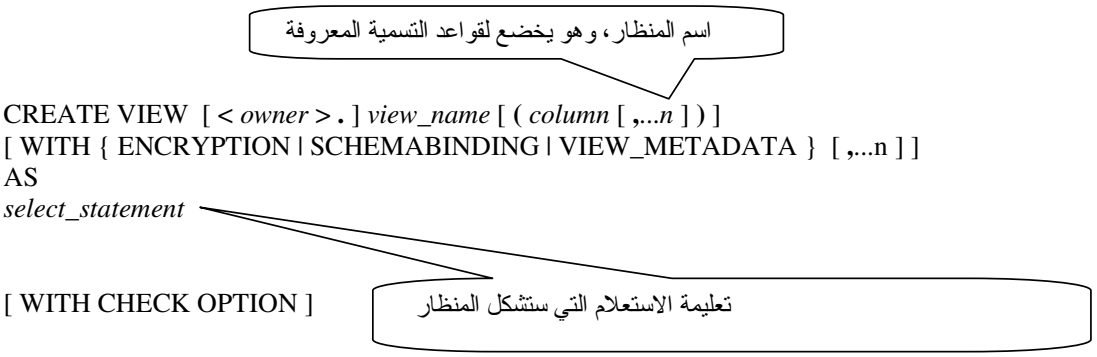
ينبغي قبل البدء بعملية إنشاء المنظار، الانتباه إلى عدّة نقاط، ومنها:

- ينبغي استخدام اسم معبّر للمنظار، وهو يخضع لقواعد التسمية المعروفة؛
- ينبغي تسمية أعمدة المنظار' وخاصة تلك التي تكون عبارة عن تابع تجميعي، أو الأعمدة ذات الأسماء المتشابهة بين الجداول التي يستخدمها المنظار؛
- يُفضل دائماً قبل إنشاء المنظار أن تتم عملية اختبار للاستعلام الذي يُكوّن ذلك المنظار؛
- لا يمكن أن تحتوي تعليمة CREATE VIEW على أي من الكلمات المفتاحية التالية: INTO و COMPUTE و COMPUTE BY؛
- لا يمكن استخدام العبارة ORDER BY في المنظار ما لم يتم استخدام TOP أيضاً؛
- لا يمكن أن يُشير المنظار لأكثر من 1024 عمود، كما لا يمكن أن يشير إلى جداول مؤقتة أو متحولات من نمط "جدول"؛

- لابد أن يتم إنشاء المنظار ضمن قاعدة المعطيات الحاليّة، مع العلم أنه يمكن للمنظار أن يُشير إلى عدّة جداول أو مناظير في قاعدة معطيات أخرى أو حتى على مخدِّم آخر (في حال استخدام استعلامات موزّعة).

- إنشاء المناظير باستخدام T-SQL:

فيما يلي القواعد التي تعبر عن المخطوط المعياري للتصريح عن المناظير:



- **WITH ENCRYPTION:**

يمكننا استخدام هذه الخاصة أثناء إنشاء المنظار وذلك بغرض تشفير محتوى المنظار من تعليمات T-SQL بحيث لا يمكن استعراض أو تعديل ذلك المنظار بعد تشفير محتوياته؛

تُستخدم هذه الخاصة لأغراض الإدارة أو لضمان الأمن بحيث لا يمكن نشر محتوى المنظار، مع العلم أنه من الضروري الحفاظ على نسخة من المخطوط الذي يعبر عن المنظار قبل تشفيره وذلك من أجل إعادة إنشائه أو تعديله مستقبلاً؛

- **WITH SCHEMABINDING:**

تتطلب هذه الخاصة التعبير عن كل من اسم الغرض واسم مالكه لكل من الأغراض المكوّنة للمنظار؛

يمنع هذا الخَيار كافة الأغراض ذات الارتباط بالمنظار من أن يتم حذفها أو تعديلها بطريقة تؤثر على المنظار، ما لم يتم حذف ذلك المنظار أو إزالة تفعيل هذه الخاصّة فيه.

- **WITH CHECK OPTION:**

على الرغم من أنه يمكننا حصر الأسطر التي يعيدها المنظار من خلال العبارة WHERE في عبارة الاختيار المكوّنة للمنظار، إلا أن ذلك لا يمنع المستخدم من تعديل أو إضافة أسطر جديدة إلى ذلك المنظار، كما في المثال التالي:

```
CREATE VIEW titles_view
AS
SELECT title, type, price, pubdate
FROM titles
WHERE TYPE IN 'business, IT'

GO

UPDATE titles_view SET price = 2000 WHERE title = 'SQL SERVER 2000'
```

GO

INSERT INTO titles_view(title, type, price, pubdate)
VALUES ('ABC' ,'IT' , 400, GETDATE())

يؤدي التصريح بعبارة WITH CHECK OPTION أثناء بناء المنظار إلى عدم السماح بإجراء عمليات الإضافة أو التعديل في المثال السابق:

```
CREATE VIEW titles_view
AS
SELECT title, type, price, pubdate
FROM titles
WHERE TYPE IN 'business, IT'
WITH CHECK OPTION
```

أمثلة:

مثال1:

إنشاء منظار على أعمدة جدول محدد:

```
CREATE VIEW titles_view
AS
SELECT title, type, price, pubdate

FROM titles
```

يستخدم هذا النوع من المناظير عندما نستعمل مجموعة من الأعمدة في جدول معين بشكل متكرر.

مثال2:

إنشاء منظار مشفر على أعمدة جدول محدد:

```
CREATE VIEW bublishers_view (fname, lname, pub_id)
WITH ENCRYPTION
AS
SELECT fname, lname, pub_id
FROM publishers
```

مثال3:

إنشاء منظار يحتوي على توابع مضمّنة:

```
CREATE VIEW categories (category, average_price)
AS
SELECT type, AVG(price)
FROM titles
GROUP BY type
GO
```

- المناظير المجزأة:

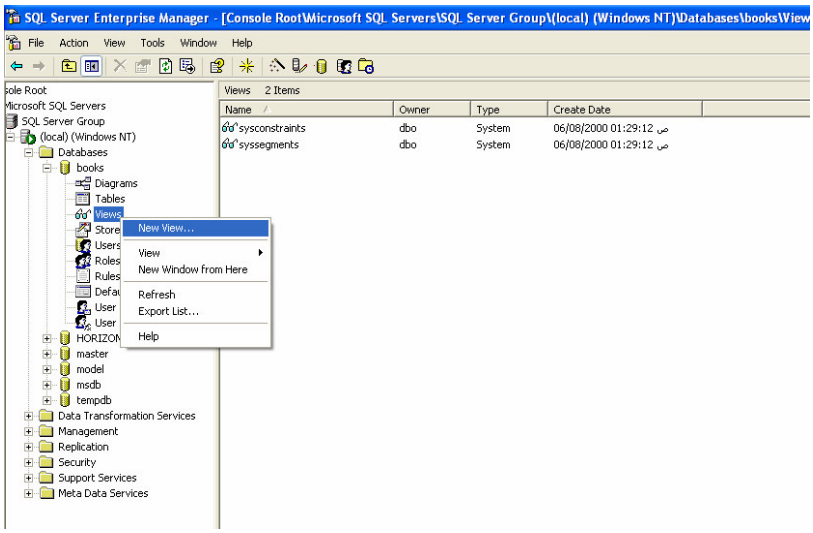
تستخدم المناظير المجزأة للولوج إلى المعطيات المقسمة أفقياً أو إلى المعطيات المجزأة على عدة أقسام في عدة جداول؛
يمكن أن تكون الجداول التي تحتوي على المعطيات المجزأة، على نفس قاعدة المعطيات أو في قاعدة معطيات أخرى أو حتى على مخدِّم آخر أو عدّة مخدمات؛
يتم إنشاء المناظير المجزأة التي تقوم بتجميع تلك المعطيات وعرضها مع بعضها البعض من خلال التعليمة UNION ALL وذلك بعد الإشارة إلى المخدم والجدول والجزء المناسب في كل استعلام من الاستعلامات المكوّنة للمنظار؛

مثال:

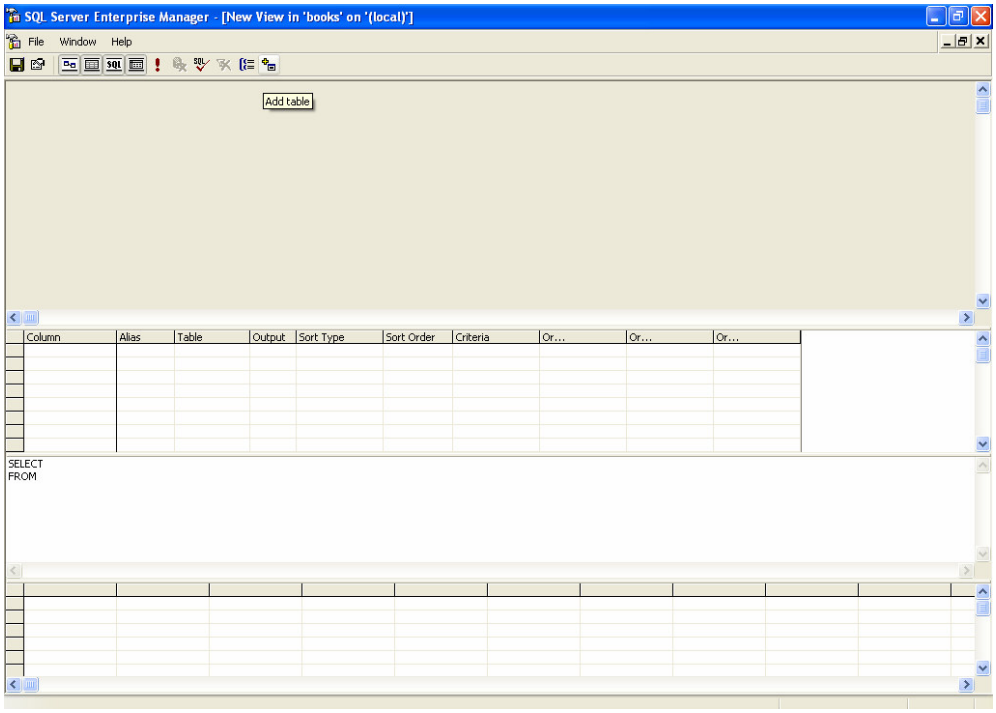
```
--create the tables and insert the values
CREATE TABLE SUPPLY1 (
   supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 1 and 150),
   supplier CHAR(50)
   )
CREATE TABLE SUPPLY2 (
   supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 151 and 300),
   supplier CHAR(50)
   )
CREATE TABLE SUPPLY3 (
   supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 301 and 450),
   supplier CHAR(50)
   )
CREATE TABLE SUPPLY4 (
   supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 451 and 600),
   supplier CHAR(50)
   )
INSERT SUPPLY1 VALUES ('1', 'CaliforniaCorp')
INSERT SUPPLY1 VALUES ('5', 'BraziliaLtd')
INSERT SUPPLY2 VALUES ('231', 'FarEast')
INSERT SUPPLY2 VALUES ('280', 'NZ')
INSERT SUPPLY3 VALUES ('321', 'EuroGroup')
INSERT SUPPLY3 VALUES ('442', 'UKArchip')
INSERT SUPPLY4 VALUES ('475', 'India')
INSERT SUPPLY4 VALUES ('521', 'Afrique')

--create the view that combines all supplier tables
CREATE VIEW all_supplier_view
AS
SELECT *
FROM SUPPLY1
   UNION ALL
SELECT *
FROM SUPPLY2
   UNION ALL
SELECT *
FROM SUPPLY3
   UNION ALL
SELECT *
FROM SUPPLY4
```

في بناء وإدارة المناظير Enterprise Managerاستخدام الأداة

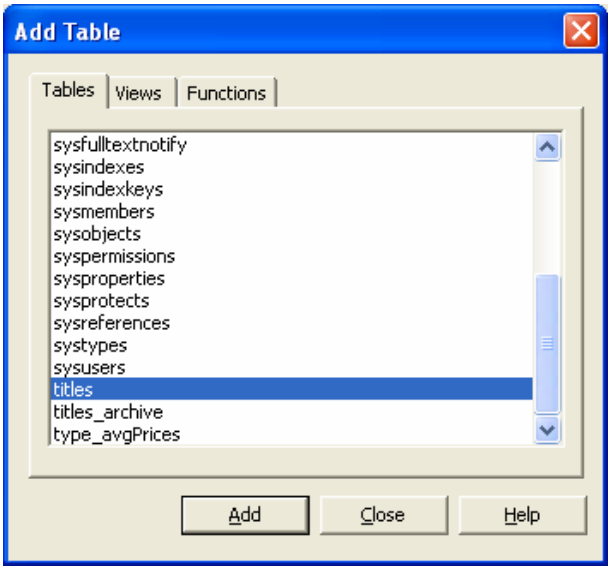
يمكننا إنشاء المناظير من خلال الأداة Enterprise Manager وذلك من خلال اختيار New View من قائمة المهمات الــسريعة لــ Views في قاعدة المعطيات المناسبة؛

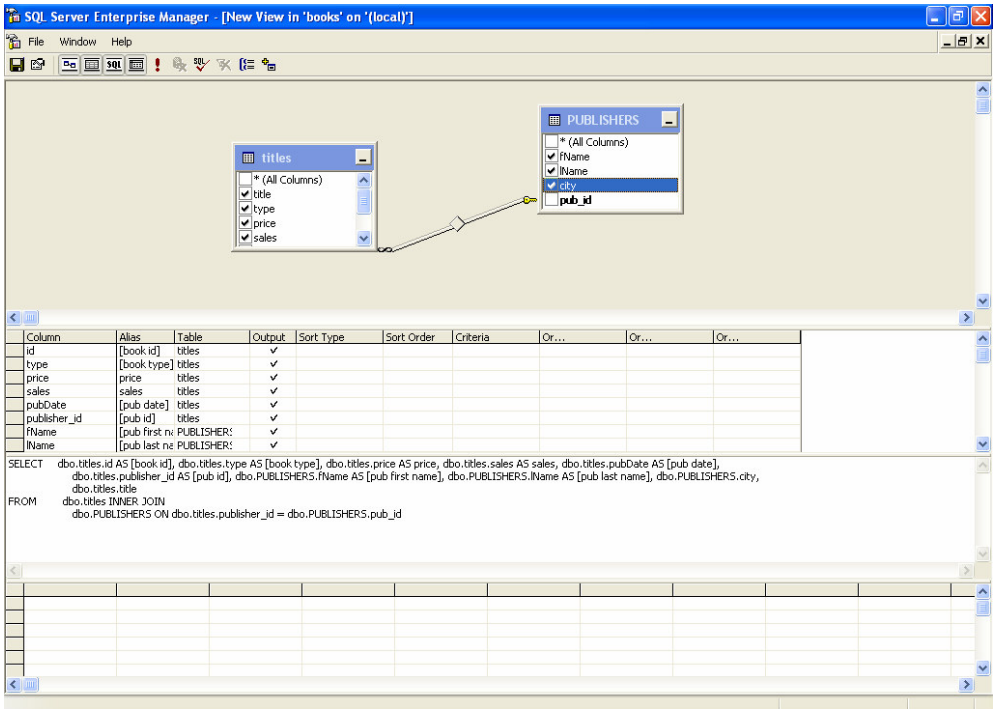


يعتمد إنشاء المناظير باستخدام الأداة Enterprise Manager بشكل أساسي على مصمم الاستعلامات (راجع جلسة أدوات SQL Server) بحيث يتم بناء المنظار بيانياً؛

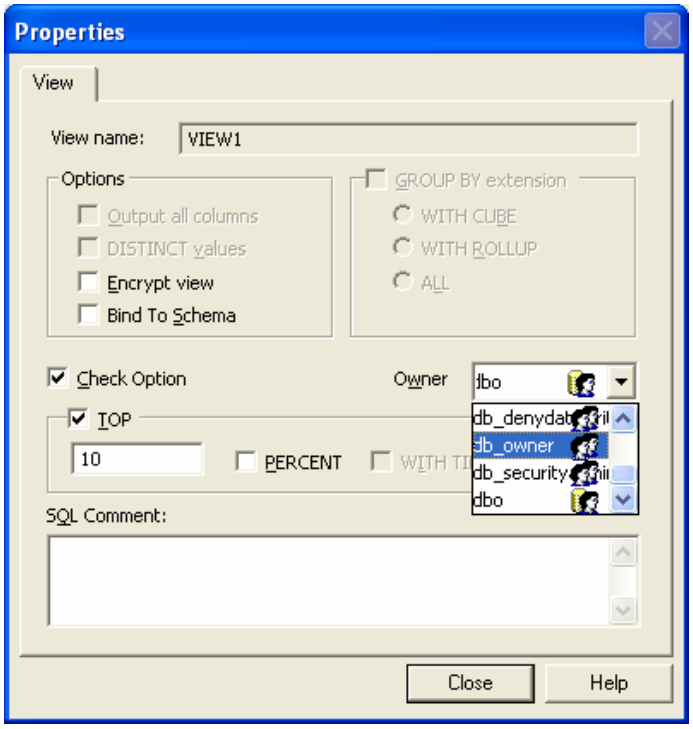


تعتبر الخطوة التالية الواجب تنفيذها لإنشاء المنظار هي إضافة الجداول التي سيعتمد عليها المنظار الجديد، ويتم ذلك مـــن خـــلال الضغط على أيقونة Add Table في مصمم الاستعلامات، بعد ذلك نقوم بتحديد الجداول التي نرغب باستخدامها، وأخيراً نقوم ببناء الاستعلام الخاص بالمنظار؛

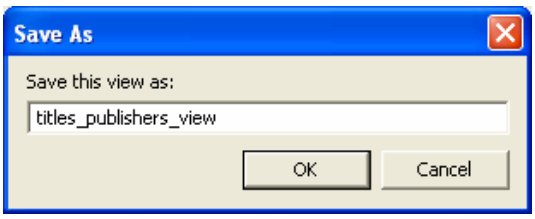




يمكننا بعد ذلك أن نقوم بتحديد خصائص المنظار الأخرى وذلك من خلال الواجهة properties:



بعد الانتهاء من بناء المنظار، يمكننا تخزين التعديلات و تسمية المنظار:



بناء وإدارة الإجرائيات المخزّنة في SQL Server

- تعريف:

تعبر الإجرائية المخزّنة عن تعليمة أو أكثر من تعليمات T-SQL مُخزّنة كغرض تنفيذي في قاعدة المعطيات؛
يمكن تمرير المعاملات من وإلى الإجرائيات المخزّنة، مما يزيد من فائدة ومرونة تلك الأغراض؛
تعيد الإجرائيات المخزّنة قيمة رقمية محددة أو مجموعة أسطر كنتيجة خرج.

- استخدامات وفوائد الإجرائيات المخزّنة:

يختلف الغرض من استخدام الإجرائيات المخزّنة، إلا أننا سنعدد فيما يلي أهم الاستخدامات الشائعة لهذا النوع من أغراض قاعدة المعطيات:

- يمكننا من خلال الإجرائيات المخزّنة أن نتّبع مفهوم البرمجة المُجتزأة، بحيث يتم تقسيم الرماز إلى عدة أجزاء تسهل عملية إدارتها؛
- حصر عمليات الولوج إلى الجداول والاعتماد على مفهوم الولوج إلى المعطيات اعتماداً على التوابع، ويُقصد بذلك إمكانية إتاحة سماحيات تنفيذ إجرائية ما والحصول على النتيجة دون امتلاك سماحيات الدخول إلى الجداول التي تستخدمها تلك الإجرائية؛
- تخفيض الضغط على الشبكة، خاصةً عندما تتكون الإجرائيات المخزّنة من مجموعة من عمليات T-SQL المختلفة بحيث يتم تنفيذها على المخدم مباشرةً دون الحاجة إلى إنشاء استدعاءات جديدة لكل منها بين المخدم والزبون؛
- تتمتع الإجرائيات المخزّنة بسرعة تنفيذ أكبر من سرعة تنفيذ مخطوطات T-SQL خاصةً وأن خطط الاستعلامات وعمليات مسح الاستعلام واختيار الطريق الأقل كلفةً للتنفيذ، يتم تخزينها في الذاكرة بعد تنفيذها لأول مرة، وليس من الضروري إعادة بناء تلك الخطط في كل مرّة يتم فيها تنفيذ تلك الإجرائية؛
- تخفيض الأخطاء البرمجية نسبياً، وذلك بحسب مبدأ فرق تسد؛
- إمكانيات ضمان تكامل الجداول، خاصةً إذا ما تم تنفيذ كافة التعديلات على تلك الجداول من خلال الإجرائيات المخزّنة.

- مساوئ الإجرائيات المخزّنة:

سنعدد فيما يلي بعض مساوئ استخدام الإجرائيات المخزّنة، والتي تعتمد بشكل رئيسي على بيئة التشغيل:

- لا تعتبر لغة T-SQL بلغة البرمجة القوية جداً مقارنةً مع لغات البرمجة الأخرى؛
- تتميز الإجرائيات المخزّنة بضعفها من ناحية التكامل مع البيئة البرمجية التي تعمل عليها، خاصةً وأن العديد من تطبيقات التطوير البرمجية تستخدم أدوات خاصة للتنقيح أو التعامل مع البيئة البرمجية، وأن العديد من تلك الأدوات المستخدمة لا يوفر الدعم الكامل للإجرائيات المخزّنة؛

- إنشاء وتنفيذ الإجرائيات المخزّنة باستخدام T-SQL:

فيما يلي القواعد التي تعبر عن المخطوط المعياري للتصريح عن الإجرائيات المخزّنة:

الوصف	الخاصة
عبارة إنشاء الإجرائية المخزّنة مع تحديد اسم المالك واسم الإجرائية الذي يخضع لشروط التسمية المعروفة؛	CREATE PROC[EDURE] [owner.] *procedure_name*
رقم يُستخدم لتجميع الإجرائيات التي تتشابه في الأسماء،	[**;** *number*]

	بحيث يمكن من خلال استخدام هذه الخاصة أن نقوم بحذف عدّة إجرائيات مخزّنة دفعة واحدة؛
[[() @*parameter data_type*[= *default_value*]	قائمة المعاملات، بحيث يتم تحديد اسم المعامل ونمطه، بالإضافة إلى قيمته الابتدائية؛
[OUTPUT] [,...*n*] [)]]	الكلمة المفتاحية التي تدل على أن المعامل المحدد هو معامل خرج؛ متتالية ببقية معاملات الإجرائية المخزنة؛
[WITH { RECOMPILE |	وهي الخاصة التي تحدد أنه لا ينبغي تخزين خطة الاختزال المطبقة من أجل عمليات التنفيذ المستقبلية للإجرائية المخزّنة، بحيث يتم إعادة بناء تلك الخطة في كل مرة وقت التنفيذ؛
ENCRYPTION |	خاصة تستخدم لتشفير محتوى الإجرائيات المخزّنة، (راجع خاصة التشفير في المناظير)؛
RECOMPILE , ENCRYPTION }]	ستخدام الخاصتين السابقتين معاً؛
AS *sql_statement* [...*n*]	مكوّنة للإجرائية؛
[return [*integer_status_value*]]	خرج –في حال وجودها– التي تعيدها الإجرائية المخزّنة، مع العلم أنه يفضل استخدام العبارة RETURN 0 في الحالات التي تم فيها تنفيذ الإجرائية المخزّنة بشكل سليم، وإعادة رقم آخر خـــلاف لك.

أمثلة:

مثال1:

إنشاء إجرائية بسيطة مع تعليمة اختيار معقدة:

```
IF EXISTS (SELECT name FROM sysobjects
        WHERE name = 'authors_info_all' AND type = 'P')
   DROP PROCEDURE authors_info_all

GO

CREATE PROCEDURE authors_info_all
AS
```

اختبار وجود غرض بهذا الاسم قبل استخدامه وإنشاء الإجرائية

```
SELECT au_lname, au_fname, title, pub_name
   FROM authors a INNER JOIN titleauthor ta
      ON a.au_id = ta.au_id INNER JOIN titles t
      ON t.title_id = ta.title_id INNER JOIN publishers p
      ON t.pub_id = p.pub_id
GO
```

مثال2:

إنشاء إجرائية مخزنة تأخذ معاملات دخل:

```
IF EXISTS (SELECT name FROM sysobjects
         WHERE name = 'au_info' AND type = 'P')
   DROP PROCEDURE au_info
GO
USE pubs
GO
CREATE PROCEDURE au_info
   @lastname varchar(40),
   @firstname varchar(20)
AS
SELECT au_lname, au_fname, title, pub_name
   FROM authors a INNER JOIN titleauthor ta
      ON a.au_id = ta.au_id INNER JOIN titles t
      ON t.title_id = ta.title_id INNER JOIN publishers p
      ON t.pub_id = p.pub_id
   WHERE  au_fname = @firstname
      AND au_lname = @lastname
GO
```

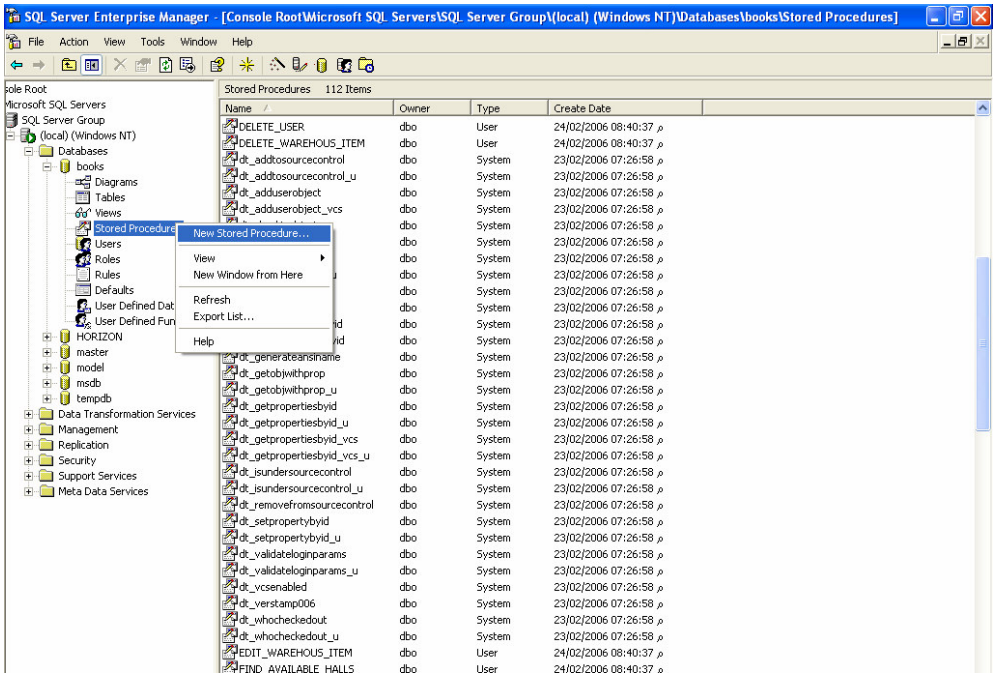
مثال3:

إنشاء إجرائية مخزنة تأخذ معاملات دخل وخرج:

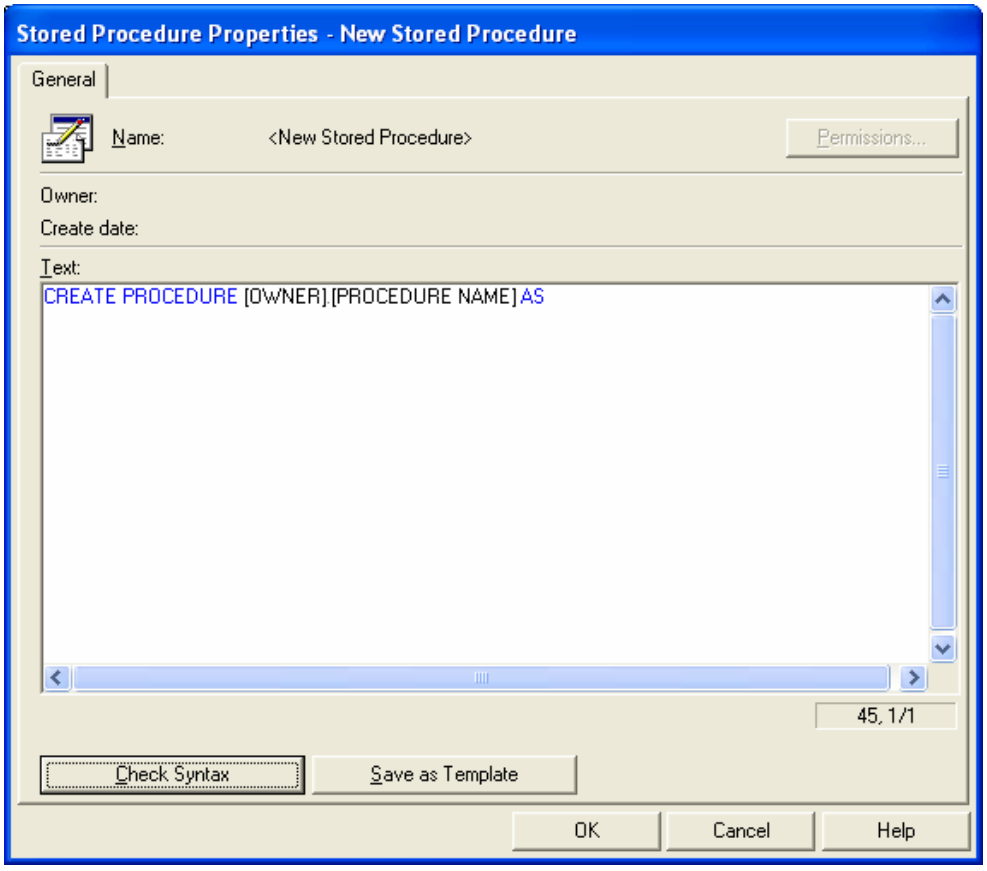
```
IF EXISTS(SELECT name FROM sysobjects
      WHERE name = 'titles_sum' AND type = 'P')
   DROP PROCEDURE titles_sum
GO

CREATE PROCEDURE titles_sum @@TITLE varchar(40) = '%', @@SUM money
OUTPUT
AS
SELECT 'Title Name' = title
FROM titles
WHERE title LIKE @@TITLE
SELECT @@SUM = SUM(price)
FROM titles
WHERE title LIKE @@TITLE
```

استخدام الأداة Enterprise Manager في بناء وإدارة الإجرائيات المخزّنة

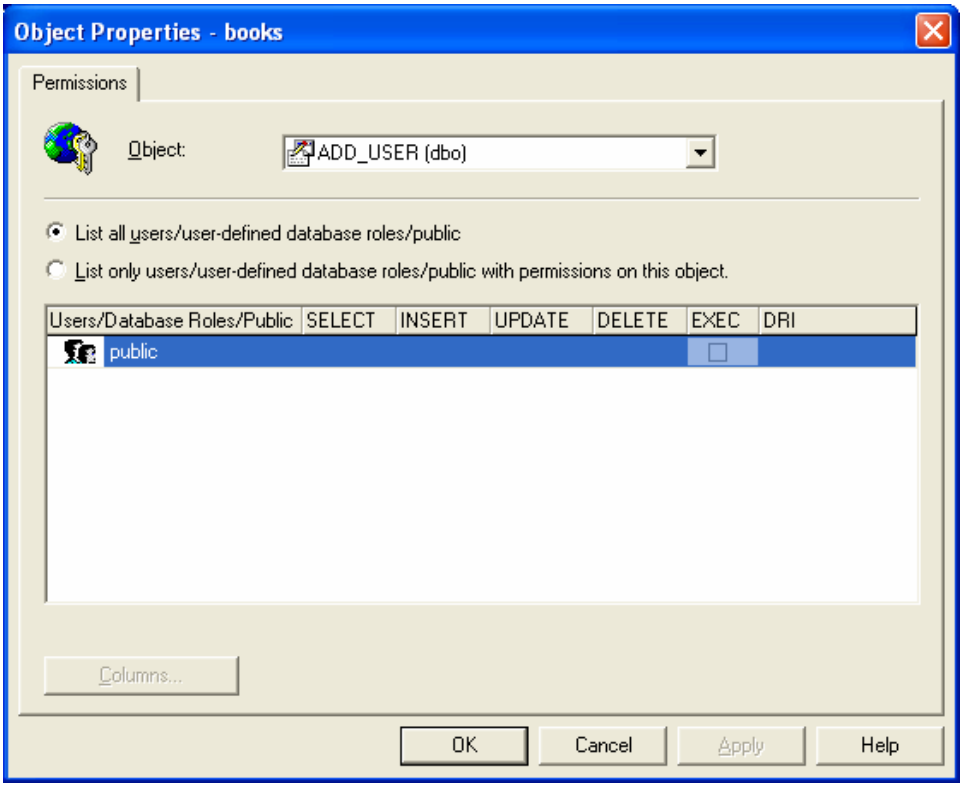
يمكننا إنشاء الإجرائيات المخزّنة من خلال الأداة Enterprise Manager وذلك من خلال اختيار New Stored Procedure من قائمة المهمات السريعة لـــ Stored Procedures في قاعدة المعطيات المناسبة؛



بحيث تظهر واجهة خاصة تسمح لنا بكتابة محتوى الإجرائية المخزّنة من استعلامات، كما يظهر فيها زر خاص باختبار صحة المخطوط المكتوب قواعدياً:

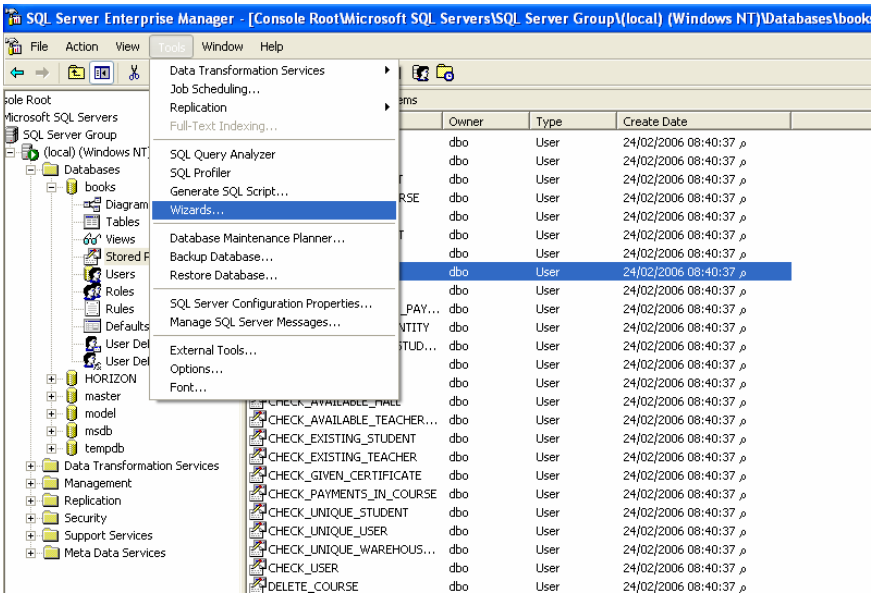


يمكننا من خلال زر Permissions في تلك الواجهة أن نستعرض السماحيات الممنوحة للمستخدمين أو لمجموعات المستخدمين للإجرائية المخزنة، بعد بنائها:

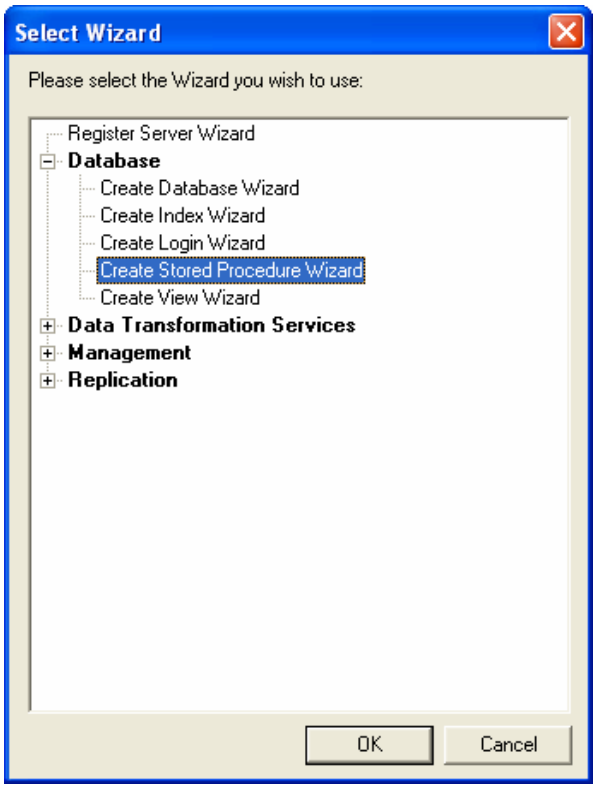


إنشاء الإجرائيات المخزّنة من خلال استخدام واجهات SQL Server Wizards:

نستطيع من خلال واجهات SQL Server Wizard الخاصة بإنشاء الإجرائيات المخزنة أن نقوم ببناء الإجرائية خطوة بخطوة:

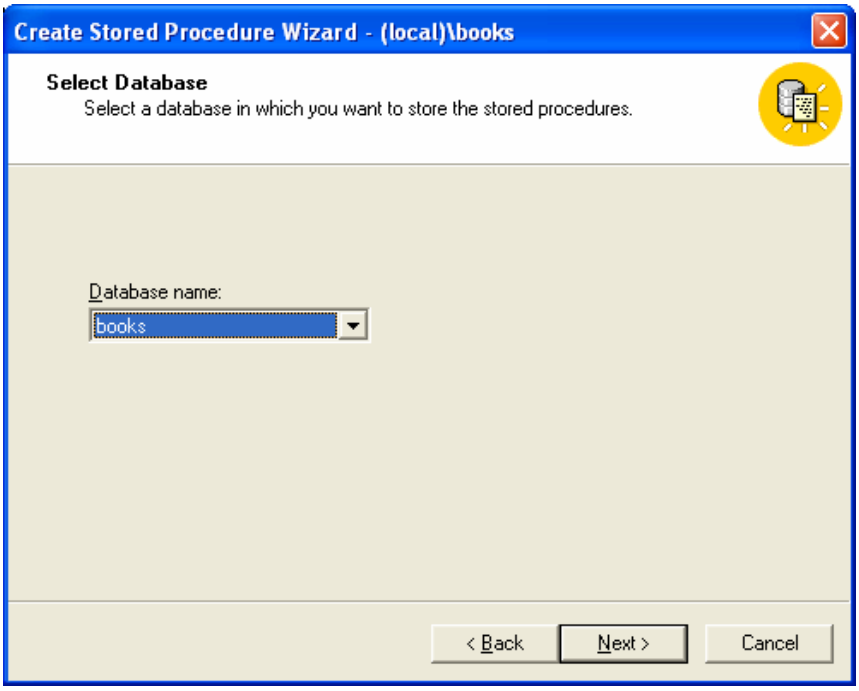


ثم نختار Create Stored Procedure Wizard من ضمن شجرة Database :

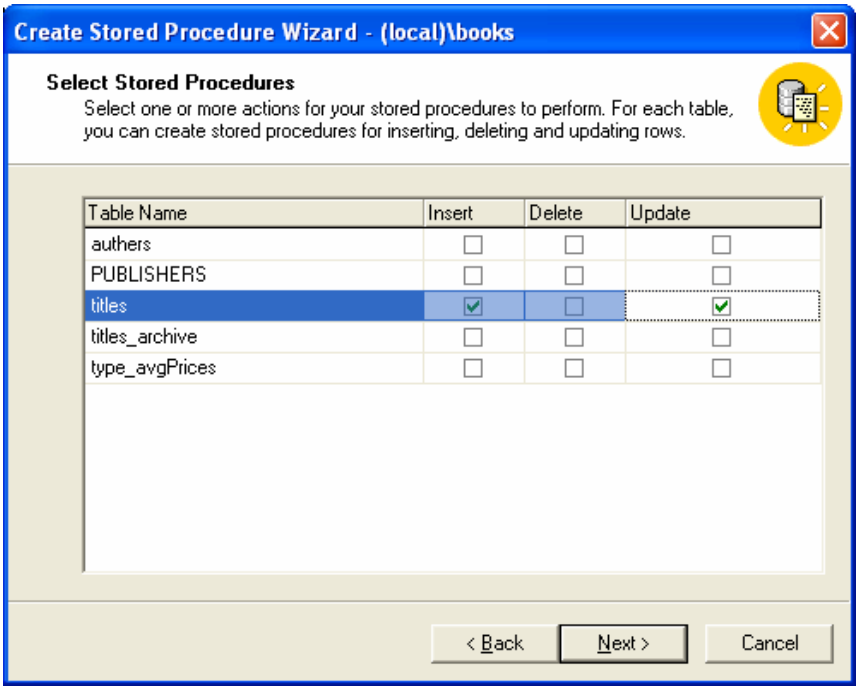


تظهر بعد ذلك الواجهة الرئيسية من واجهات الـــ Wizard الخاصة بإنشاء الإجرائيات المخزّنة:

نحدد بعد ذلك قاعدة المعطيات التي نرغب بإنشاء الإجرائية المخزّنة فيها:

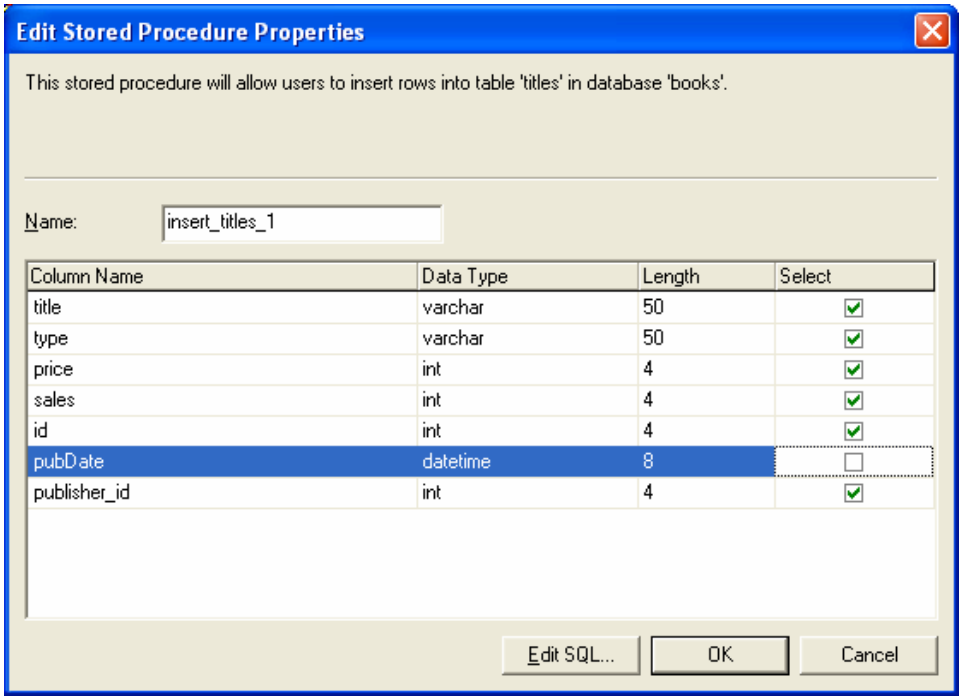


نحدد بعد ذلك الجدول أو الجداول التي نرغب بإجراء تعديلات عليها ضمن الإجرائية المخزنة التي نقوم بإنشائها، من نمط إضافة أو حذف أو تعديل، بحيث سنختار في المثال الذي نقوم بإنشائه، إجراء عملية إضافة وتعديل على جدول titles:

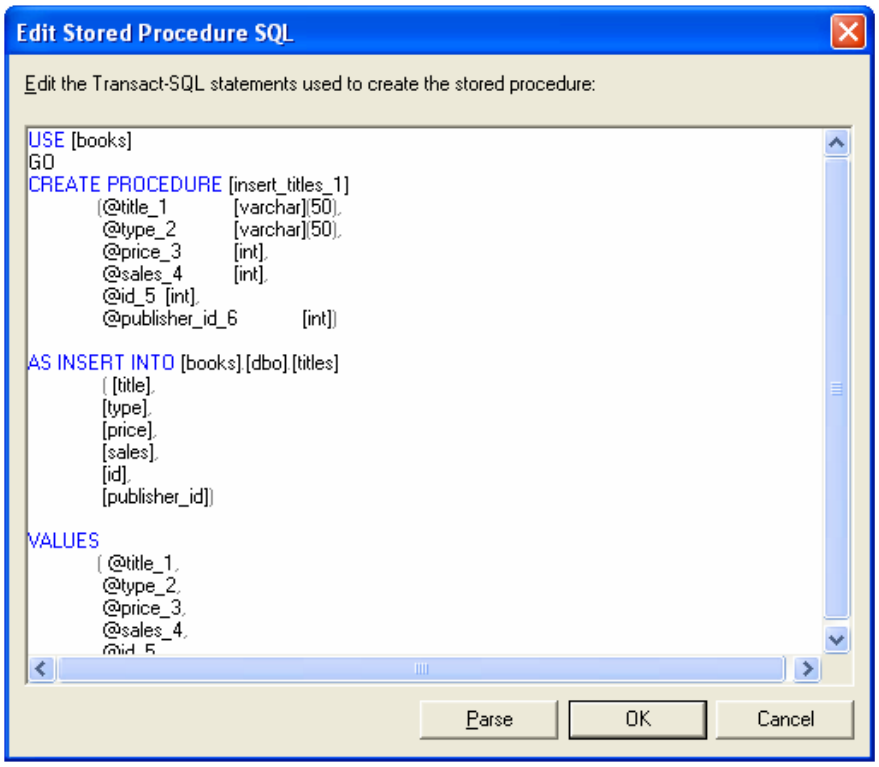


تظهر بعد ذلك واجهة خاصة تسمح لنا باستعراض أو تعديل مخطوط SQL المتكوّن عن الواجهات السابقة:

تظهر الواجهة التالية بعد الضغط على Edit لتعديل عملية الإضافة:

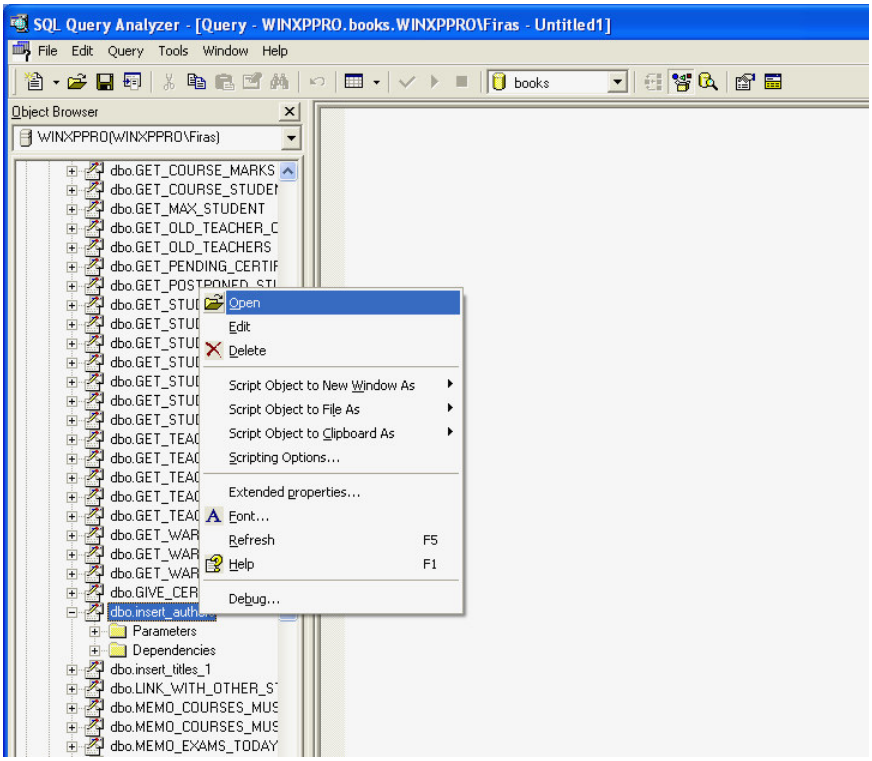


نستطيع استعراض التعديلات التي قمنا بإجرائها على هيئة مخطوط T-SQL بالضغط على زر Edit SQL...، كما يمكننا أيضاً إجراء تعديلات على المخطوط واختبار صحة التعديلات قواعدياً:

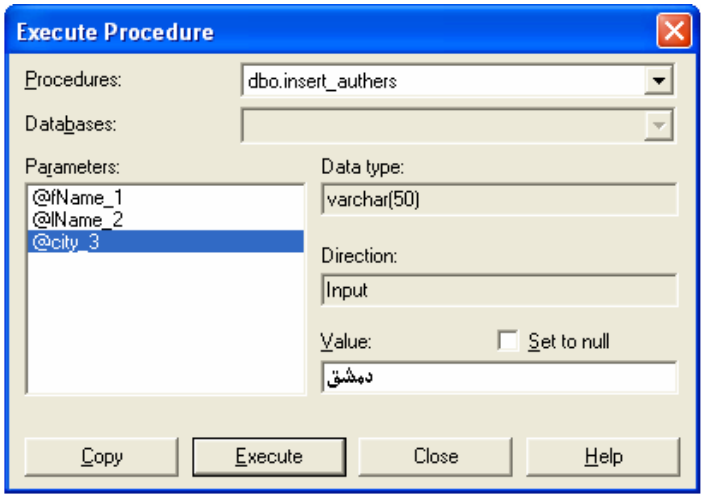


- سيتم في المثال السابق إنشاء إجرائيتين مخزّنتين وإضافتهما إلى قائمة الإجرائيات المخزّنة في قاعدة المعطيات المختارة، مع العلم أنه يمكننا تعديل تلك الإجرائيات بعد إنشائها؛
- يمكننا كذلك أن نقوم بإنشاء الإجرائيات المخزّنة من خلال الأداة SQL Query Analyzer.

- تنفيذ الإجرائيات المخزّنة:
- نستطيع من خلال الأداة SQL Query Analyzer أن نقوم وببساطة بتنفيذ الإجرائيات المخزّنة؛
- ينبغي أولاً الوصول إلى الإجرائية في شجرة SQL Query Analyzer، أي بعد تحديد المخدم المناسب ثم قاعدة المعطيات المناسبة ثم مجلد الإجرائيات المخزّنة في قاعدة المعطيات تلك وأخيراً نحدد الإجرائية التي نرغب بتنفيذها ونختار Open.



تظهر بعد ذلك الواجهة الخاصة بإدخال المعاملات –إن وجدت–:



بحيث نلاحظ في تلك الواجهة قائمة المعاملات مع وصف حول كل منها من حيث نمط المعطيات أو نوعه (أي معامل دخل أو خرج) بالإضافة إلى حيّز خاص بإضافة قيمة المعامل، مع العلم أن مربع الاختيار Set to null يسمح بالتعبير عن أن لمعامل الدخل القيمة الفارغة.

القوادح واستخداماتها في SQL Server

- تعريف:

تعبر القوادح حالات خاصة من الإجرائيات المخزّنة، بحيث يتم تنفيذها تلقائياً نتيجةً لحدوث تعديلات في المعطيات؛
يتم إنشاء القوادح على الجداول، كما يتم ربط تنفيذها بوقوع حدث أو أكثر بحيث يرتبط تنفيذها بحدوث تغيرات على المعطيات من إضافة أو حذف أو تعديل.

- استخدامات وفوائد القوادح:

تعتبر القوادح من الأدوات الهامة التي يتم استخدامها من أجل صيانة قاعدة المعطيات وضمان تكاملها، خاصةً وأنها تقوم بإجراء عمليات تقييم على المعطيات قبل أن يتم تأكيد التغيرات التي تحصل عليها؛

تستطيع القوادح –أثناء تنفيذها– أن تقوم بعدد كبير من الأعمال، منها:

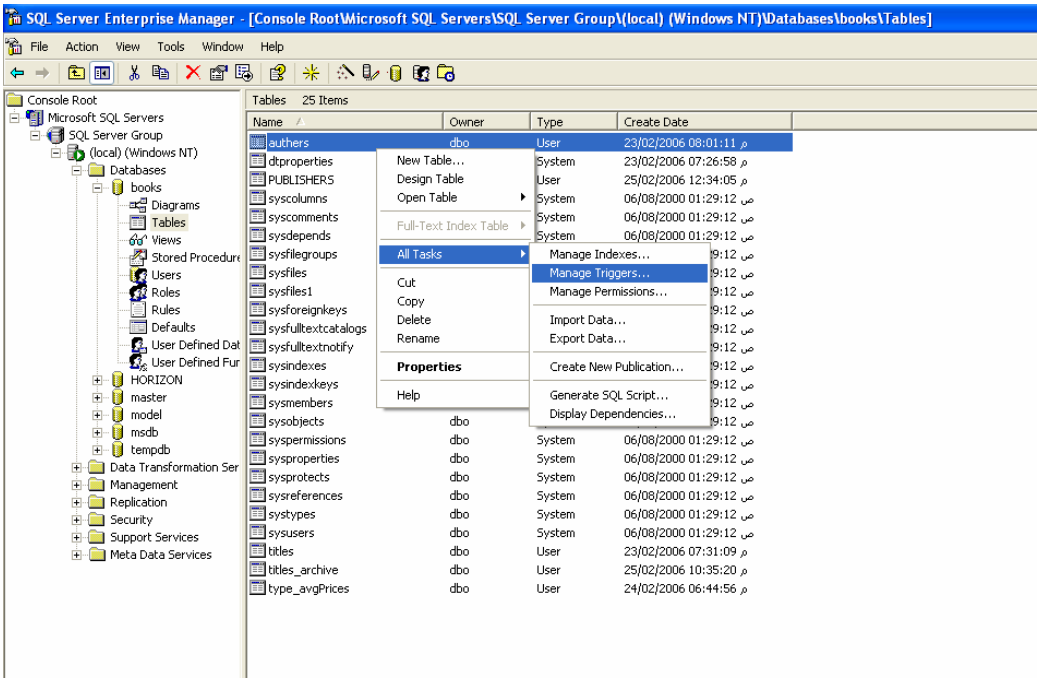
- القيام بعمليات مقارنة قبل وبعد تغير المعطيات؛
- التراجع عن تنفيذ التعديلات على المعطيات؛
- قراءة المعطيات من عدّة جداول من نفس قاعدة المعطيات أو من قواعد معطيات مختلفة؛
- تعديل الجداول من نفس قاعدة المعطيات أو من قواعد معطيات مختلفة؛
- تنفيذ الإجرائيات المخزّنة المحلية والبعيدة؛
- اختبار القيود المعقدة المفروضة على الأعمدة، وخاصة تلك التي تعتمد على بعض الأسطر الموجودة ضمن نفس الجدول أو جداول مختلفة؛
- ضمان التكامل التابعي الشلاّلي؛
- توليد القيم التلقائية المعقدة التي تعتمد على معطيات موجودة في عدة أسطر أو أعمدة أو جداول.

ملاحظة:

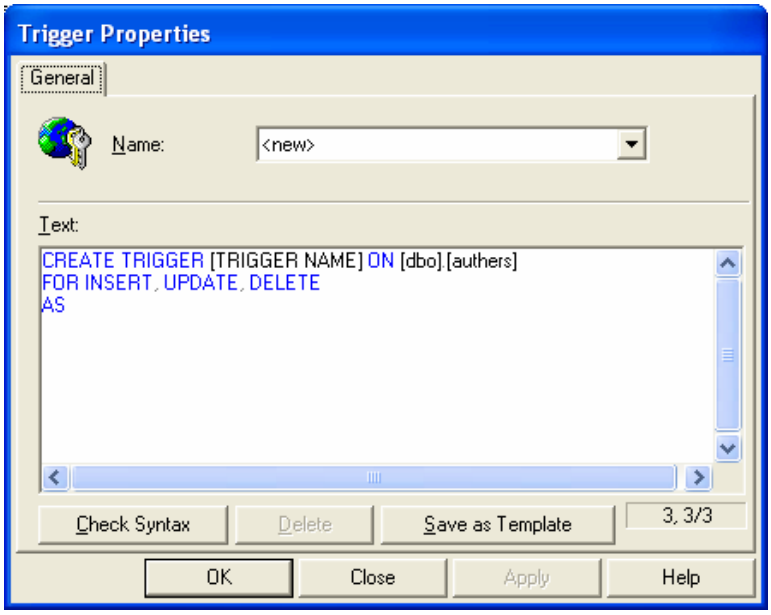
على الرغم من أنه يمكننا استخدام الإجرائيات المخزّنة لتنفيذ كافة المهام السابقة، إلا أن استخدام القوادح يتمتع بخصائص تميّزه عن الإجرائيات المخزّنة وذلك لإمكانية تنفيذ القوادح في أثناء أي نوع من أنواع التعديلات التي يمكن تطبيقها على المعطيات، في حين أن رماز الإجرائيات المخزّنة أو مخطوطات SQL المضمنة في رماز التطبيقات، لا يتم تنفيذها إلا عند إجراء التعديلات على المعطيات.

استخدام الأداة Enterprise Manager في بناء وإدارة القوادح

يمكننا إنشاء القوادح من خلال الأداة Enterprise Manager وذلك من خلال اختيار All Tasks من قائمة المهمات السريعة لـــ للجدول الذي بإدارة القوادح المعرّفة عليه، ثم نختار Manage Triggers؛

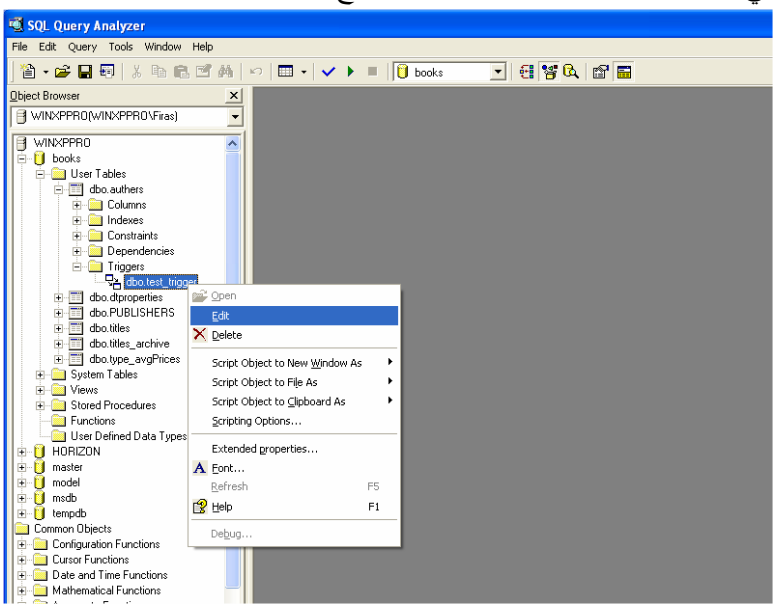


بحيث تظهر واجهة خاصة تحتوي على القالب الأولي الذي يعبر عن قادح جديد، كما تسمح لنا بكتابة محتـــوى القادح الجديد من استعلامات، كما يظهر فيها زر خاص باختبار صحة المخطوط المكتوب قواعدياً:



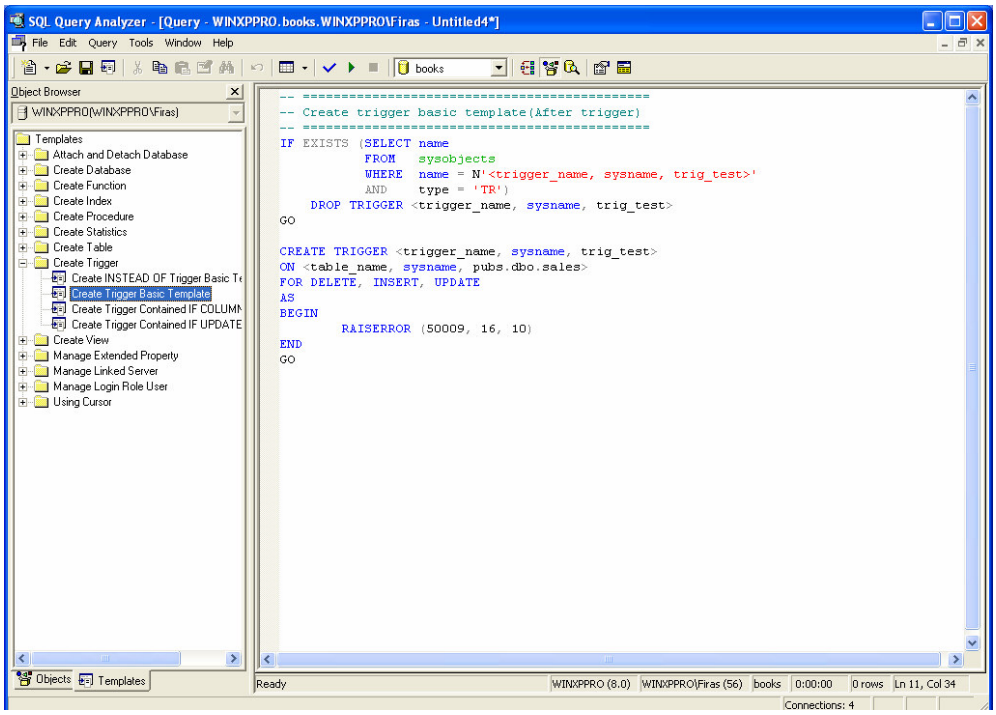
يدعم SQL Server 2000 من خلال الأداة SQL Query Analyzer إمكانية إنشاء وتعديل القوادح بطريقتين جديدتين، كما يلي:

- يمكننا من خلال شجرة الأغراض في SQL Query Analyzer أن نستعرض كافة الأغراض الموجـــودة في قاعدة المعطيات المحددة، ومنها القوادح؛

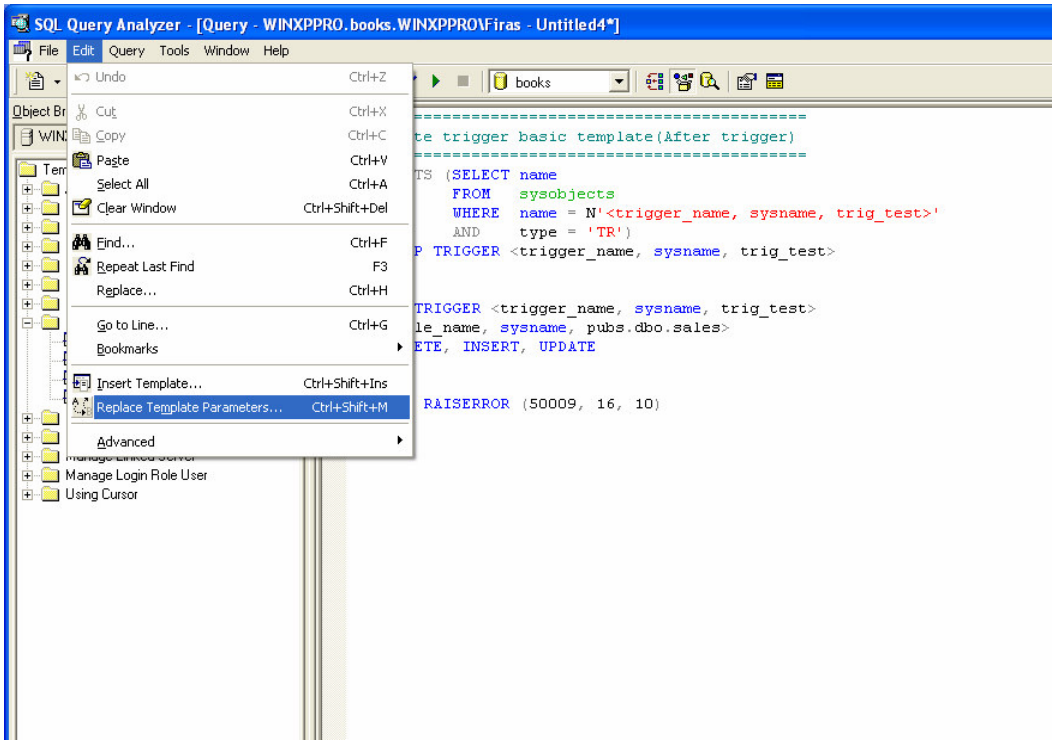


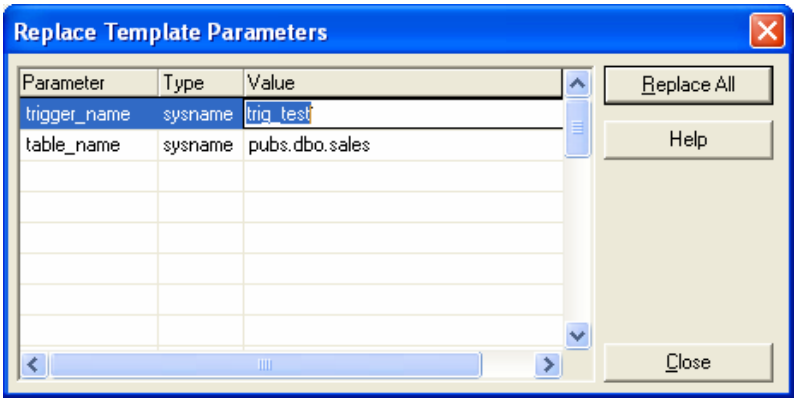
كما يمكننا أيضاً أن نحدد القادح المناسب ونختار Edit من قائمة المهمات السريعة الخاصة به لنقوم بتعديل محتوياته من استعلامات أو مخطوطات؛

- يمكننا من خلال شجرة القوالب الموجودة في الأداة Query Analyzer والتي تعرض كافة أنواع القوالب المطبقة على الأغراض في SQL Server، أن نستعرض القوالب الأولية المتاحة للقوادح، وذلك من خلال اختيار Create trigger basic template من ضمن المجلد Create Trigger في الشجرة تلك؛



يمكننا استبدال متحولات ذلك القالب إما يدوياً أو من خلال خَيار Replace Template Parameters الموجود في القائمة Edit في SQL Query Analyzer:


SQL Query Analyzer - [Query - WINXPPRO.books.WINXPPRO\Firas - Untitled4*]
File Edit Query Tools Window Help
Undo Ctrl+Z
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Clear Window Ctrl+Shift+Del
Find... Ctrl+F
Repeat Last Find F3
Replace... Ctrl+H
Go to Line... Ctrl+G
Bookmarks
Insert Template... Ctrl+Shift+Ins
Replace Template Parameters... Ctrl+Shift+M
Advanced
books
te trigger basic template(After trigger)
TS (SELECT name
FROM sysobjects
WHERE name = N'<trigger_name, sysname, trig_test>'
AND type = 'TR')
P TRIGGER <trigger_name, sysname, trig_test>
TRIGGER <trigger_name, sysname, trig_test>
le_name, sysname, pubs.dbo.sales>
ETE, INSERT, UPDATE
RAISERROR (50009, 16, 10)
Manage Login Role User
Using Cursor


Replace Template Parameters
Parameter
Type
Value
trigger_name
sysname
trig_test
table_name
sysname
pubs.dbo.sales
Replace All
Help
Close

أنواع وخصائص القوادح في SQL Server

قبل ظهور الإصدار SQL Server 2000، كانت After Triggers هي النوع الوحيد من القوادح المتاحة، بحيث لم يكن من الضروري استخدام العبارة "After" للدلالة على هذا النوع من القوادح؛
قدّمت الأداة SQL Server 2000 نوعاً جديداً من القوادح يطلق عليها اسم INSTEAD OF TRIGGERS (التي سنتحدث عنها لاحقاً) مما سمح باستخدام الكلمة المفتاحية AFTER للدلالة على الأنواع السابقة من القوادح؛

- AFTER Triggers:

يعتبر هذا النوع من القوادح هو الآلية الأصلية التي قدّمتها الأداة SQL Server لإدارة عمليات الاستجابة الناتجة عن التعديلات المطبقة على المعطيات؛
تعتبر طريقة عمل هذا النوع من القوادح بسيطة من حيث المفهوم، إلا أن هناك بعض النقاط الهامة التي ينبغي أخذها بعين الاعتبار عند استخدام AFTER Triggers؛
يتم إطلاق AFTER Triggers بعد تنفيذ التعديلات على المعطيات ولكن قبل تأكيد تلك التعديلات، بحيث يتم تسجيل كافة المناقلات –المسؤولة عن التعديلات التي يتم تطبيقها– في ملف سجلّ المناقلات، ولكن لا يتم تأكيدها حتى ينتهي تنفيذ القادح بالكامل؛
يمكن التراجع عن كافة التعديلات التي تم تطبيقها من خلال القادح أو من خلال العبارات التي قامت بإطلاقه؛

فيما يلي القواعد التي تعبر عن إنشاء القادح:

الوصف	الخاصة
جديد مع تحديد اسم ذلك القادح؛	CREATE TRIGGER *trigger_name*
ي سيتم إنشاء القادح عليه؛	**ON** *table_name*
تدل على نوع القادح المنشأ، مع العلم أنها العبارة AF عبارة تدل على العملية التي ستؤدي إلى إطلاق د إضافة سطر جديد، أو بعد تعديل سطر معين، أو بعد	AFTER {INSERT | UPDATE | DELETE}
ت التي سيتم تنفيذها من قبل القادح بعد إطلاقة.	**AS** *Sql_statements*

مثال:

سنقوم بإنشاء قادح على جدول AUTHORS يعيد رسالة بعدد الأسطر التي تم تعديلها، ثم سنقوم بإجراء عدّة عمليات تعديل على أسطر ذلك الجدول لمعاينة خرج القادح:

مخطوط القادح:

```
CREATE TRIGGER AUTHORS_UPDATE_TRIG
ON authors
AFTER UPDATE
AS
   PRINT 'TRIGGER OUTPUT:' + CONVERT(VARCHAR(5), @@ROWCOUNT) + ' rows
were updated '

GO
```

عملية التعديل الأولى:

```
UPDATE authors SET fname = fname WHERE ID=5
```

الخرج الناتج عن القادح (أي رسالة القادح الذي يتم إطلاقه بعد عملية التعديل):

```
TRIGGER OUTPUT: 1 rows were updated
(1 row(s) affected)
```

عملية التعديل الثانية:

```
UPDATE authors SET fname = fname
```

الخرج الناتج عن القادح:

```
TRIGGER OUTPUT: 11 rows were updated
(11 row(s) affected)
```

سؤال:

نعلم أن تنفيذ AFTER Triggers يتم بعد تطبيق التعديلات على المعطيات كعمليات الإضافة أو الحذف أو التعديل، ولكن ما علاقة ذلك مع بقية الأحداث التي يتم تنفيذها بما يتضمن تنفيذ القيود المفروضة على الأعمدة أو المعطيات؟

- فيما يلي عرض للأحداث التي يتم تنفيذها قبل تنفيذ AFTER Trigger:
  - معالجة القيود:

يتم قبل تنفيذ القادح معالجة القيود المفروضة، وذلك يشتمل على المفاتيح الأولية وقيود التفرُّد وقيود الاختبار؛

  - القيود التصريحية الارتباطية:

يتم قبل تنفيذ القادح معالجة القيود التي تعبر عن الارتباطات والعلاقات بين الجداول، أي القيود الناتجة عن المفاتيح الخارجية؛

- الحدث الذي أطلق القادح:

يتم أيضاً قبل تنفيذ القادح تطبيق كافة التعديلات التي أدّت إلى إطلاقه، بحيث يتم تطبيقها قبل تنفيذه ولكن لا يتم تأكيدها حتى الانتهاء من تنفيذه.

بالتالي ينبغي أخذ هذه النقاط بعين الاعتبار، وفهم آلية عملها تماماً قبل تصميم القادح.

ملاحظة:

- يسمح كل من SQL Server 7.0 و SQL Server 2000 ببناء أكثر من قادح من أجل كل عملية تغيير في المعطيات من إضافة أو تعديل أو حذف؛
- تعتبر هذه الخاصة مفيدة في بعض الحالات إلا أنها يمكن أن تؤدي إلى حدوث بعض الارتباكات خاصةً فيما يتعلق بترتيب عملية التنفيذ؛
- استطاع الإصدار SQL Server 2000 أن يزيل بعضاً من حالات الارتباك تلك من خلال السماح بتوصيف أول وآخر قادح يتم إطلاقهما نتيجةً لعمل معين، إلا أنه لم يحل تلك المشكلة بشكل جذري، ففي حال وجود أربعة قوادح يتم إطلاقها معاً، فسيبقى أسلوب تنفيذ القادحين الثاني والثالث غير معروف؛
- تُستخدم الإجرائية sp_settriggerorder لإعداد ترتيب تنفيذ القوادح، بحيث تأخذ ثلاثة معاملات، وهي:

```
Sp_settriggerorder (trigger_name,order_value,action)
```

- اسم القادح الذي نرغب بتحديد ترتيب تنفيذه
- نوع الترتيب لذلك القادح:) FIRST | LAST | NONE(
- الفعل الذي يؤدي لإطلاق هذا القادح:) INSERT | UPDATE | DELETE(

مثال:

```
Sp_settriggerorder authors_update_trig,FIRST,'UPDATE'
```

- INSTEAD OF Triggers:

يقدم الإصدار SQL Server 2000 نوعاً جديداً من القوادح يطلق عليها اسم INSTEAD OF Triggers، توسِّع هذه القوادح من الإمكانيات المتاحة للقوادح بشكل عام وتقدم بديلاً لقوادح الـ AFTER التي تم استخدامها في الإصدارات السابقة؛

يختلف هذا النوع من القوادح بشكل كبير في أسلوب عمله عن النوع السابق، بحيث يتم في INSTEAD OF Triggers – وكما يبدو من اسمها – باستبدال العبارة التي أدّت إلى إطلاق القادح بالعبارة الموصفة في القادح نفسه؛

فيما يلي القواعد التي تعبر عن إنشاء هذا النوع من القوادح:

الوصف	الخاصة
جديد مع تحديد اسم ذلك القادح؛	CREATE TRIGGE
ي سيتم إنشاء القادح عليه؛	**ON** *table_name*
INST تدل على نو ع القادح الذي نقوم ببنائه؛ INSTEAD عبارة تدل على العملية التي ستؤدي إلـــى إما بعد إضافة سطر جديد، أو بعد تعديل سطر معين، ، بحيث يتم استبدال تلك العملية بمحتوى القـــادح مـــن	INSTEAD OF {INSERT | UPDAT
ت التي سيتم تنفيذها من قبل القادح بعد إطلاقة.	**AS** *Sql_statements*

مثال:

سنقوم بإنشاء قادح على جدول AUTHORS يعيد رسالة بعدد الأسطر التي تم تعديلها، ثم سنقوم بإجراء عدّة عمليات تعديل على أسطر ذلك الجدول لمعاينة خرج القادح:

مخطوط القادح:

```
CREATE TRIGGER AUTHORS_UPDATE_insteadof_TRIG
ON authors
INSTEAD OF UPDATE
AS
   PRINT '--TRIGGER OUTPUT:' + CONVERT(VARCHAR(5), @@ROWCOUNT) + ' rows
were updated '

GO
```

سنستعرض أولا محتويات جدول Authors من مؤلفين من مدينة دمشق قبل إجراء أي تعديل:

```
SELECT * FROM AUTHORS WHERE city = 'دمشق'
```

الخرج الناتج:

fname	lname	city	id
سامر	حامد	دمشق	1
عمر	سعيد	دمشق	2
عامر	خليل	دمشق	4
سناء	حسن	دمشق	5
يامن	معين	دمشق	6
فراس	أحمد	دمشق	7
سامر	دياب	دمشق	8

عملية التعديل:

```
UPDATE authors SET fname = 'XXXXX'
where city ='دمشق'
```

الخرج الناتج:

```
--TRIGGER OUTPUT:8 rows were updated

(8 row(s) affected)
```

سنستعرض أولا محتويات جدول Authors من مؤلفين من مدينة دمشق بعد إجراء التعديل:

fname	lname	city	id
سامر	حامد	دمشق	1
عمر	سعيد	دمشق	2
عامر	خليل	دمشق	4
سناء	حسن	دمشق	5
يامن	معين	دمشق	6
فراس	أحمد	دمشق	7
سامر	دياب	دمشق	8

نلاحظ أن التغيرات التي يفترض أن تقوم بها عملية التعديل، لم تتم، وذلك لأنه قد تم استبدال عملية التعديل تلك بالرسالة التي تـــم إعادتها من قبل القادح؛

ملاحظة:

- فيما يلي عرض للأحداث التي يرتبط تنفيذها بتنفيذ INSTEAD OF Triggers:
  - الحدث الذي أطلق القادح:

يتم كخطوة أولى من خطوات تنفيذ INSTEAD OF Triggers استبدال العبارة التي أدت إلى إطلاق القادح بالعبارات التي يتكون منها ذلك القادح؛

  - معالجة القيود:

تتم بعد ذلك معالجة القيود المفروضة من المفاتيح أولية أو قيود تفرُّد أو قيود الاختبار، بعد تنفيذ القادح من نوع INSTEAD OF.

بالتالي ينبغي أخذ هذه النقاط بعين الاعتبار، وفهم آلية عملها تماماً قبل تصميم القادح.

توابع المستخدم المعرّفة

مقدمة:

يزودنا SQL Server بعدد من التوابع مسبقة التعريف مضمّنة في لغة Transact-SQL، والتي تختلف مهماتها من توابع لمعالجة سلاسل المحارف إلى توابع لإجراء حسابات رياضية، أو توابع لإجراء تحويلات في أنماط المعطيات وغيرها (راجع جلسة استخدام T-SQL في SQL Server)؛

على الرغم من أن SQL Server يزودنا بمجموعة كبيرة وواسعة من التوابع، إلا أننا يمكن أن نتعرّض في كثير من الأحيان إلى حالات نحتاج فيها لتابع ما غير معرّف في تلك القائمة؛

نستطيع من خلال الإجرائيات المخزّنة أن نقوم بإنشاء معالجة محددة تعيد نتيجة ما تناسب احتياجاتنا، إلا أن استخدام الإجرائيات المخزّنة يختلف عن أسلوب التعامل مع التوابع، فعلى سبيل المثال، لا يمكننا أن نستخدم الإجرائيات لإعادة قيمة ثابتة؛

نستطيع من خلال SQL Server 2000 أن نقوم ببناء توابع مستخدم معرّفة يمكنها أن تعيد قيمة ثابتة، تماما كما تعمل توابع SQL Server المضمّنة. هذا بالإضافة إلى أن توابع المستخدم المعرّفة تلك يمكنها إعادة مجموعة أسطر ضمن متحول من نمط table؛

- إنشاء توابع المستخدم المعرّفة باستخدام T-SQL:

فيما يلي القواعد التي تعبر عن إنشاء تابع مستخدم معرّف:

الخاصة	الوصف
CREATE FUNCTION [*owner.*] *function_name*	تعليمة إنشاء تابع جديد مع تحديد اسم ذلك التابع واسم مالكه؛
([{ *@parameter_name scalar_datatype* **[= default] } [, … n]])**	تحديد المعاملات التي يمكن تمريرها للتابع، مع نمط كل منها؛
RETURNS *scalar_datatype*	التصريح عن نمط القيمة التي يُعيدها التابع؛
[WITH {ENCRYPTION, SCHEMABINDING}	لتشفير محتوى التابع أو لربطه مع أغراض أخرى ضمن الـ Schema؛
[AS]	فصل بين التصريح عن التابع وعن محتوياته، وهذه العبارة ليست إجبارية؛
BEGIN	للتعبير عن بداية جزء العبارات المكوّنة للتابع؛
Sql_statements	تعليمات T-SQL التي يتكون منها التابع
RETURN *scalar_datatype*	قيمة الخرج التي يعيدها التابع؛
	للتعبير عن انتهاء التابع؛

ملاحظة:

يمكن أن تشتمل عبارات T-SQL التي تستخدم ضمن توابع المستخدم المعرفة على ما يلي:

- يمكن استخدام عبارات تصريحيه للتعبير عن المتحولات أو الـ Cursors المحلية ضمن التابع من خلال التعليمة DECLARE؛
- استخدام عبارات إسناد القيم إلى المتحولات من خلال التعليمة SET؛
- يمكن استخدام كافة تعليمات الـ Cursors من فتح أو إغلاق أو التفاف أو جلب أو غيرها؛
- يمكن استخدام كافة أنواع تعليمات البنى البرمجية من IF أو ELSE أو WHILE أو غيرها…
- يمكن استخدام تعليمات تعديل المعطيات، من UPDATE أو INSERT أو DELETE؛
- يمكن استدعاء وتنفيذ إجرائيات مخزّنة من ضمن التوابع، وذلك باستخدام التعليمة EXECUTE.

مثال:

سنقوم في المثال التالي بإنشاء تابع يقوم بإجراء عملية معينة، كما سنعمل على توصيف الخاصة WITH SCHEMABINDING أثناء إنشاء ذلك التابع؛

يؤدي استخدام هذه الخاصة إلى ربط التابع المُنشأ مع أغراض قاعدة المعطيات التي يتعامل معها، بحيث لا يمكن تعديل أو حـــذف تلك الأغراض قبل حذف التابع أو إيقاف خاصة SCHEMABINDING فيه:

مخطوط التابع:

```
CREATE FUNCTION func_AVG_price (@price int= 0)
RETURNS @table table (type varchar(12) null, avgPrice int null )
WITH SCHEMABINDING
AS
BEGIN
        INSERT @table
            SELECT type, avg(price) as avgPrice
            FROM dbo.titles
            GROUP BY type
            HAVING avg(price) > @price
        RETURN
END

GO
```

سنحاول الآن أن نقوم بتعديل أحد أعمدة الجدول titles:

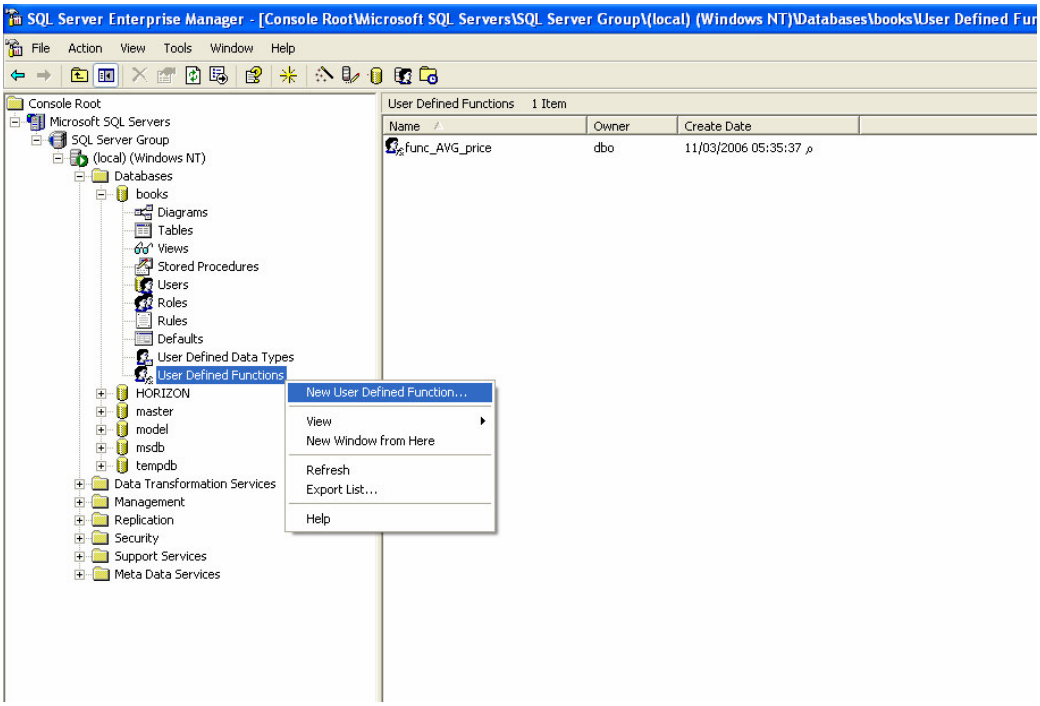
```
ALTER TABLE titles ALTER COLUMN price money null
GO
```

سيؤدي ذلك إلى حدوث خطأ، وستظهر الرسالة التالية للتعبير عن السبب:

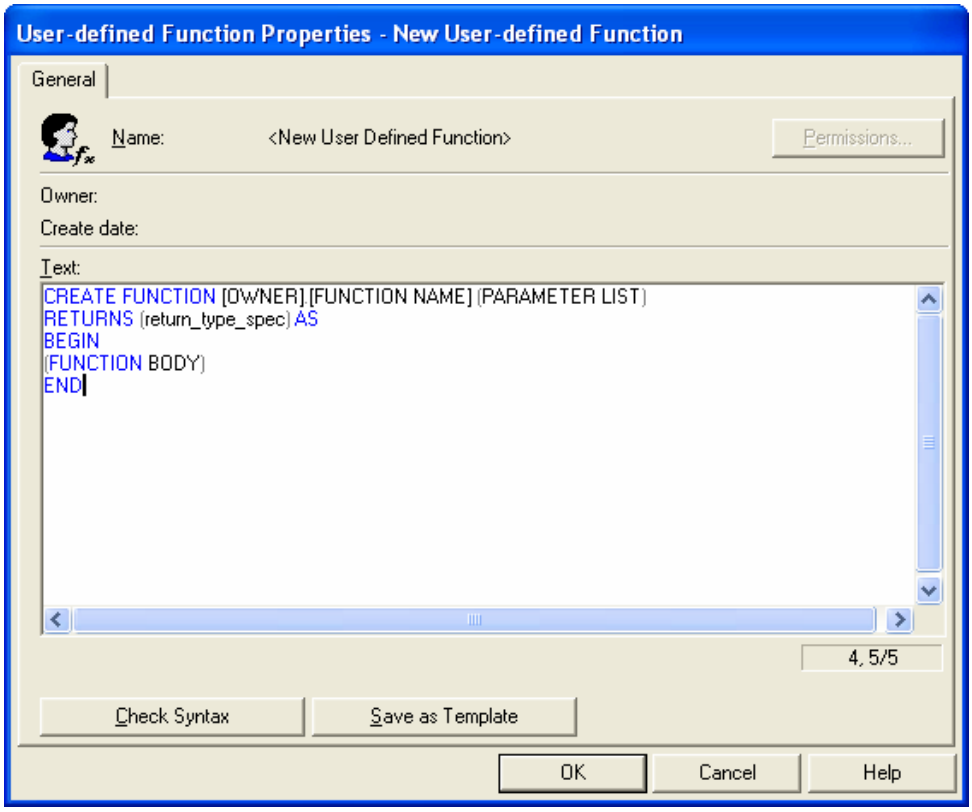
```
Server: Msg 5074, Level 16, State 3, Line 1
The object 'func_AVG_price' is dependent on column 'price'.
Server: Msg 4922, Level 16, State 1, Line 1
ALTER TABLE ALTER COLUMN price failed because one or more objects access
this column.
```

استخدام الأداة Enterprise Manager في بناء وإدارة التوابع

يمكننا إنشاء التوابع من خلال الأداة Enterprise Manager وذلك من خلال اختيار New User Defined Function من قائمة المهمات السريعة لمجلد توابع المستخدم المعرّفة الموجود ضمن قاعدة المعطيات التي نعمل عليها؛

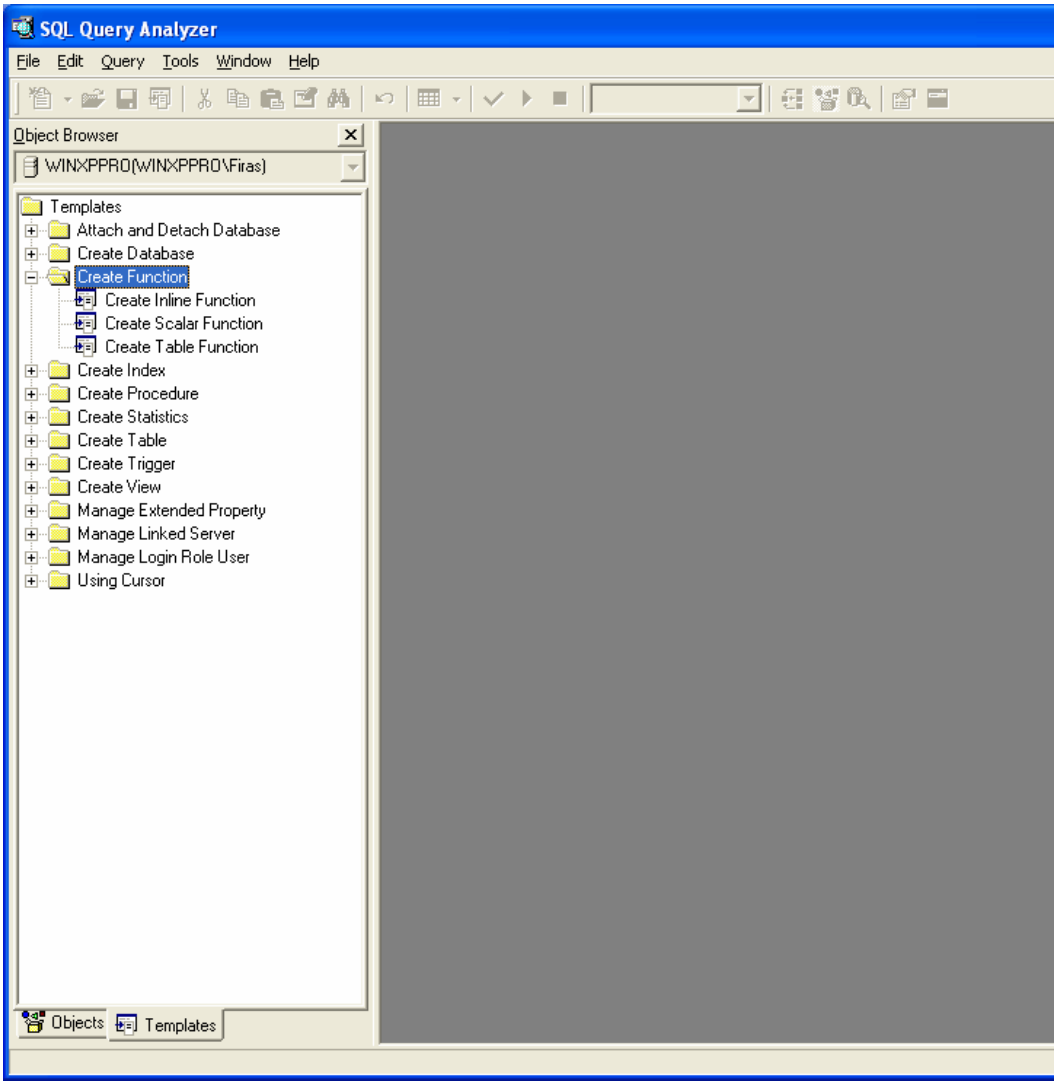


تظهر واجهة خاصة تحتوي على القالب الأولي الذي يعبر عن تابع مستخدِم جديد، كما يمكننا أن نقوم فيها بكتابة محتوى التابع الجديد من استعلامات، كما يظهر زر خاص باختبار صحة المخطوط المكتوب قواعدياً:

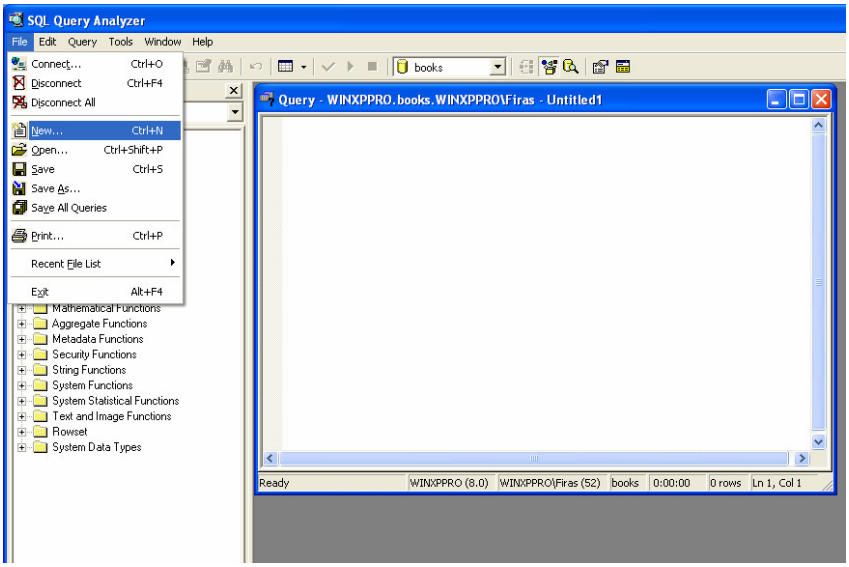


استخدام الأداة SQL Query Analyzer في بناء وإدارة التوابع

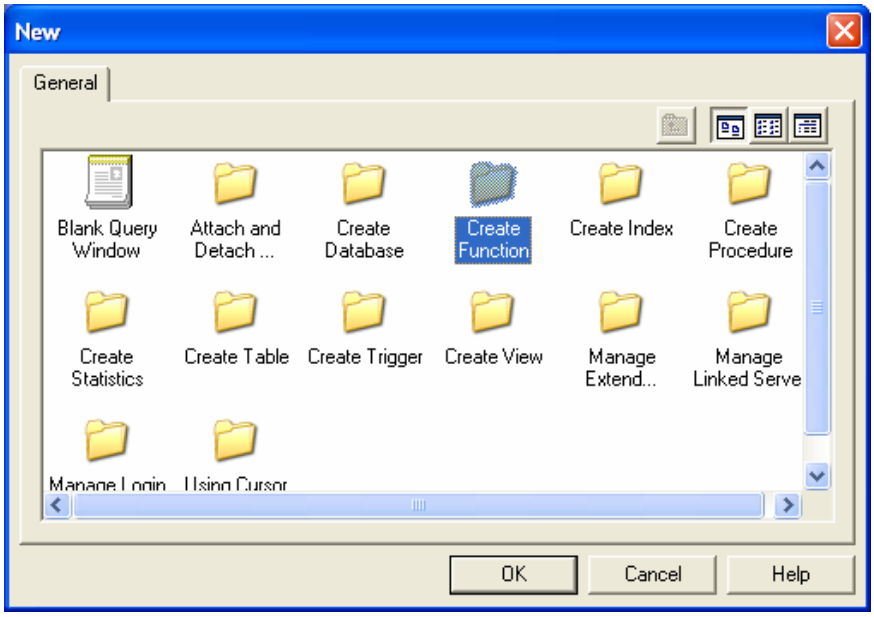
يدعم SQL Server 2000 من خلال الأداة SQL Query Analyzer إمكانية إنشاء وتعديل التوابع من خلال عدّة قوالب يمكن الوصول إليها من خلال قائمة Templates في Query Analyzer ثم باختيار CREATE FUNCTION ثم اختيار القالب المناسب؛

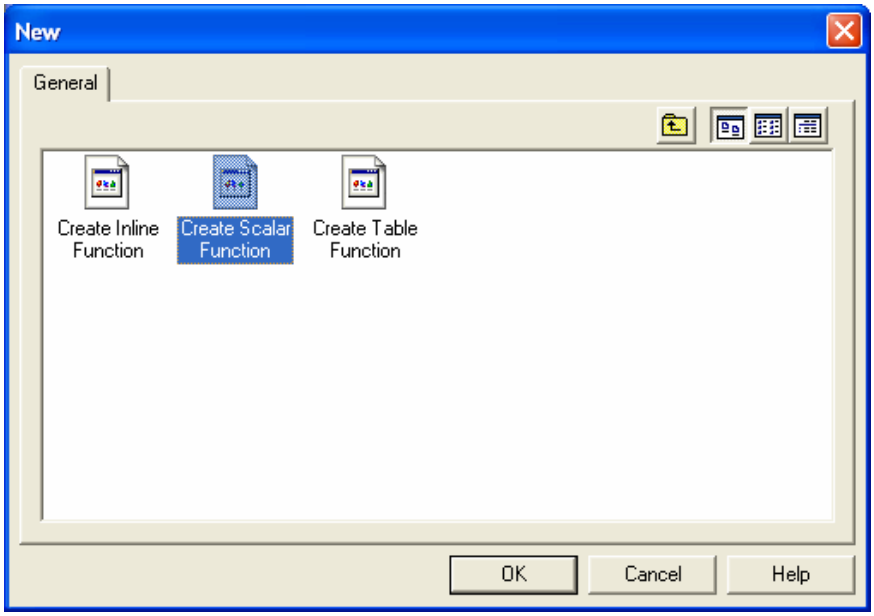


يمكننا كذلك أن نصل إلى تلك القوالب من القائمة File ثم New



لتظهر بعد ذلك واجهة جديدة، نختار من بين المجلدات التي تعرضها المجلد Create Function، ثم نحدد القالب المناسب:

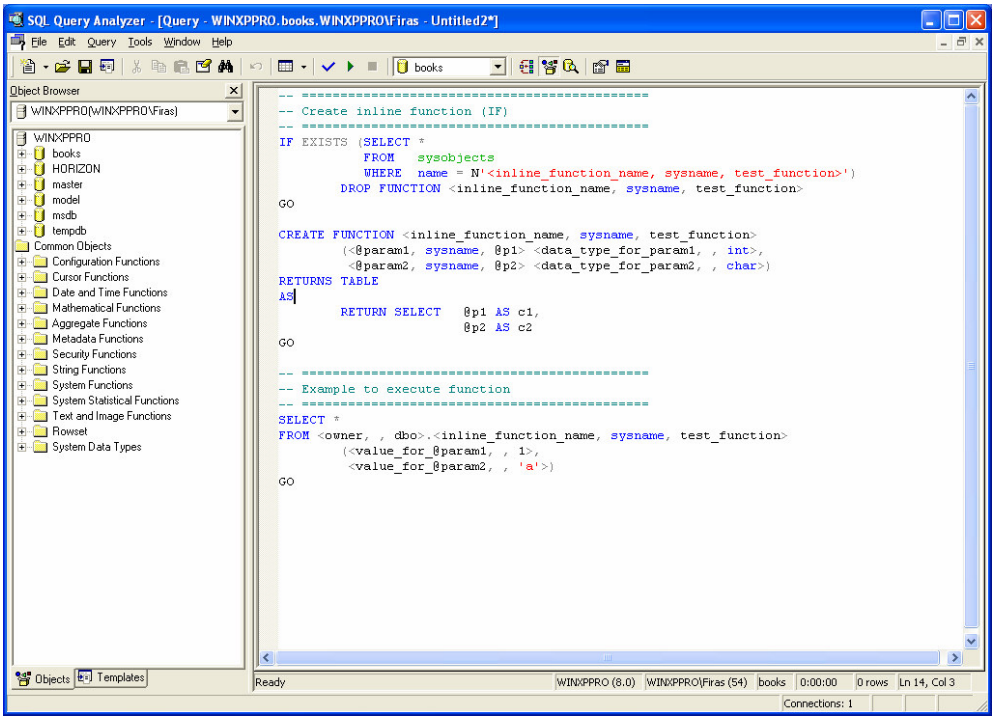




يحتوي SQL Query Analyzer على ثلاثة قوالب لتوابع افلمستخدم المعرّفة، وهي:

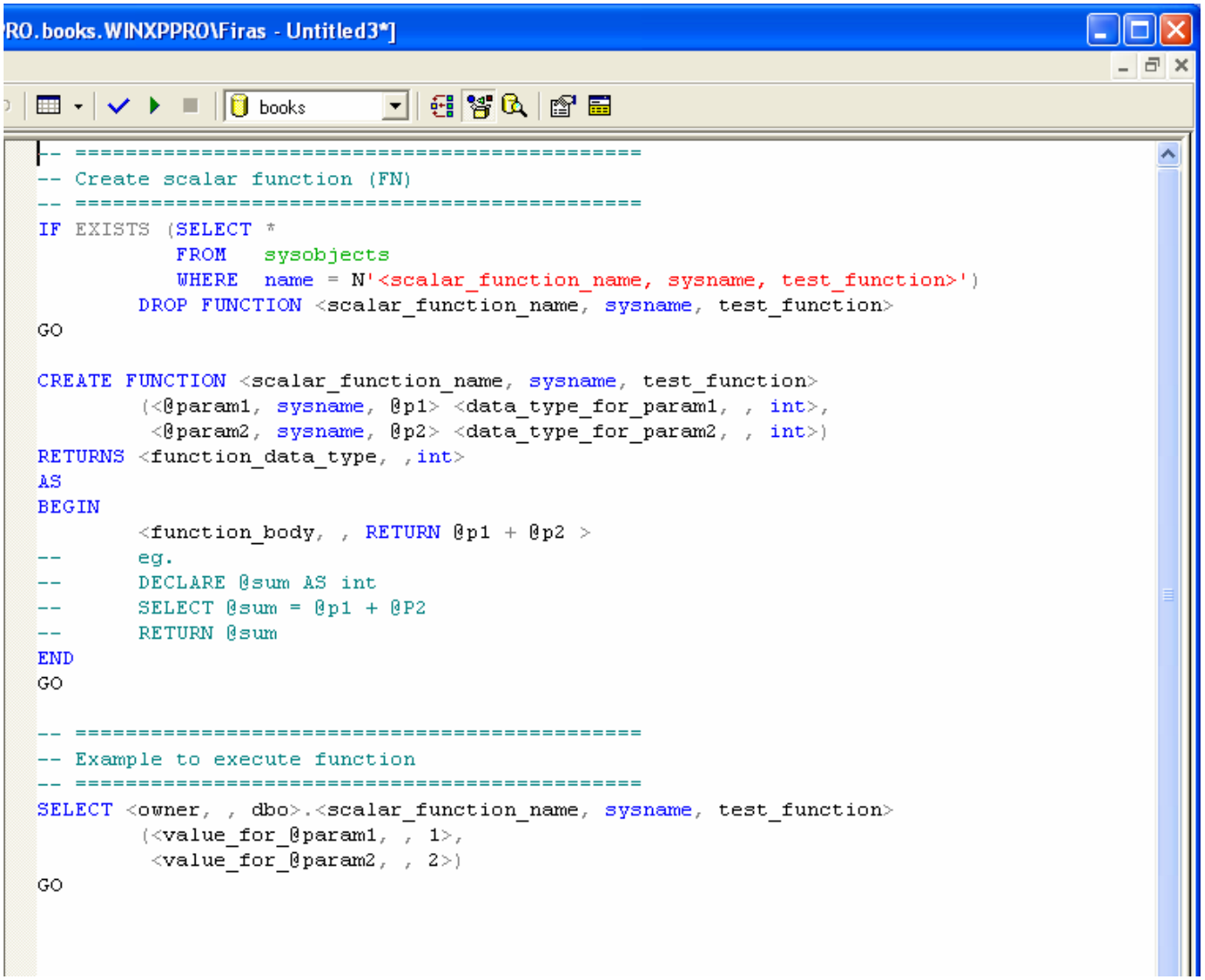
- Inline Fuction:

وهو القالب الذي يمثل تابع المستخدم المعرّف الذي يعيد نتيجة استعلام معين:



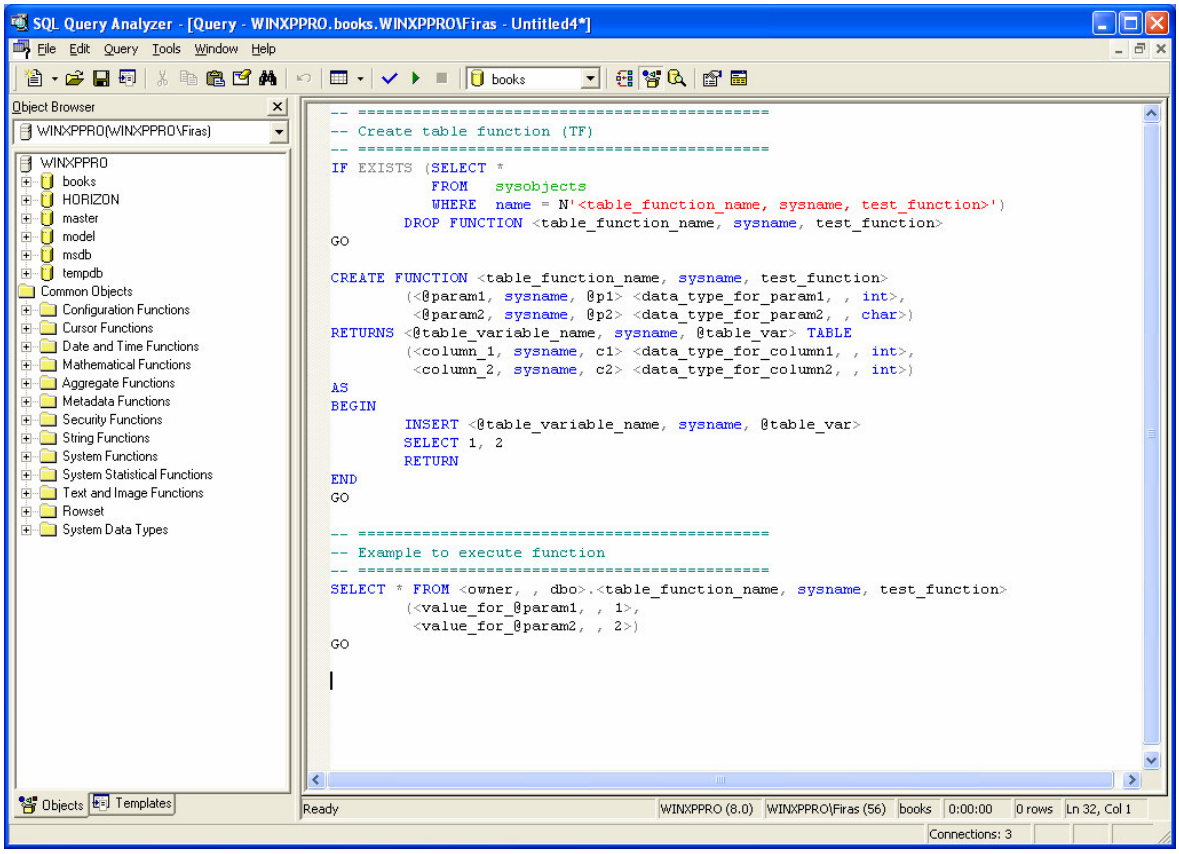
- Scalar Function:

وهو القالب الذي يمثل تابع مستخدم معرّف يعيد قيمة ثابتة:

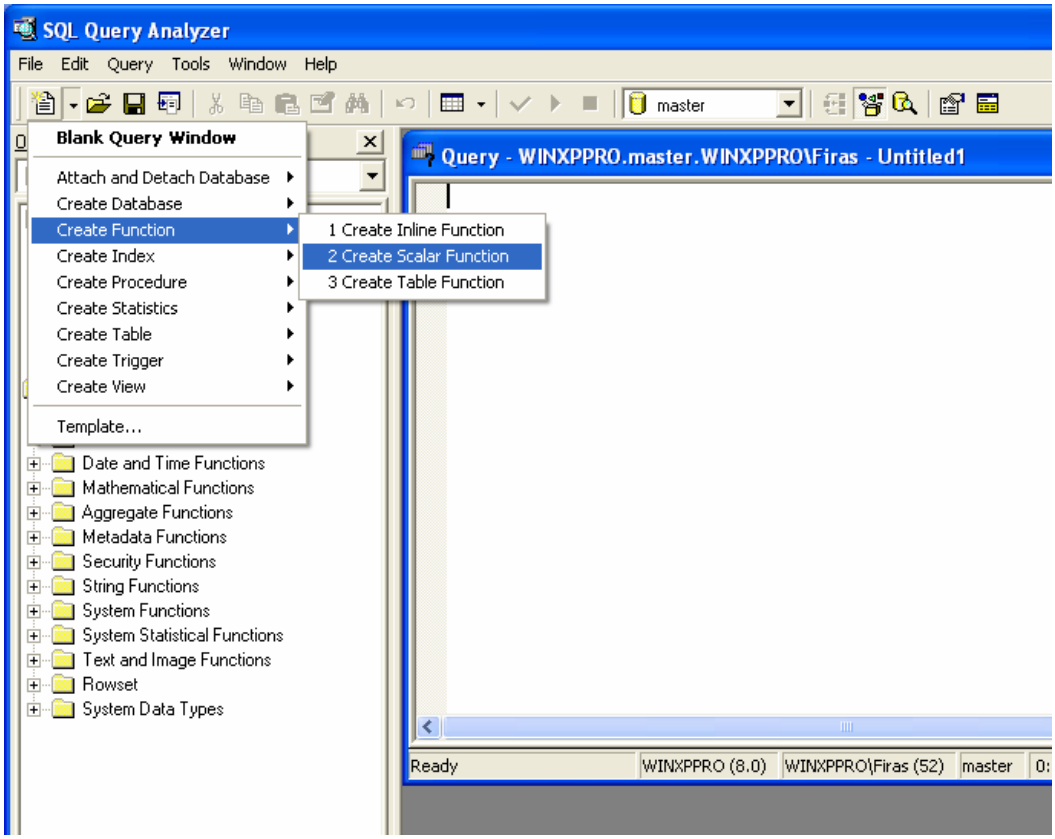


- Table Function:

وهو القالب الذي يمثل تابع المستخدم المعرّف الذي يعيد متحول من نمط جدول:



يمكننا كذلك الوصول إلى تلك القوالب من خلال الضغط على رمز "السهم" الموجود إلى جانب اختصار "ملف جديد" في شريط الأدوات الخاص بالأداة SQL Query Analyzer، ثم اختيار Create Function ثم تحديد القالب الذي نرغب باستخدامه:


SQL Query Analyzer
File Edit Query Tools Window Help
master
Blank Query Window
Attach and Detach Database
Create Database
Create Function
Create Index
Create Procedure
Create Statistics
Create Table
Create Trigger
Create View
Template...
1 Create Inline Function
2 Create Scalar Function
3 Create Table Function
Query - WINXPPRO.master.WINXPPRO\Firas - Untitled1
Date and Time Functions
Mathematical Functions
Aggregate Functions
Metadata Functions
Security Functions
String Functions
System Functions
System Statistical Functions
Text and Image Functions
Rowset
System Data Types
Ready
WINXPPRO (8.0)
WINXPPRO\Firas (52)
master

الفصل السالع عشر

عنوان الموضوع:

أمن SQL Server وإدارة المستخدمين.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

سنناقش في هذه الجلسة مسألة أمن الأداة SQL Server من خلال دراسة أساليب تعريف حسابات الدخول والفرق بينها وبين حسابات المستخدمين بالإضافة إلى دراسة مفهوم سماحيات الولوج وكيفية إسنادها إلى المستخدمين؛

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- عملية إدارة أمن SQL Server
- طرائق التحقق من الهوية في SQL Server
  - نموذج نظام التشغيل Windows في التحقق من الهوية
  - النموذج المختلط في التحقق من الهوية
- كيفية إعداد نموذج التعرف على الهوية
- حسابات الدخول وإدارتها
- حسابات المستخدمين وإدارتها
- مفهوم سماحيات الولوج وإسنادها
- الأدوار:
  - أدوار المخدم الثابتة وإدارتها
  - أدوار قاعدة المعطيات الثابتة وإدارتها
  - أدوار المستخدم المعرَّفة وإدارتها
  - أدوار التطبيقات وإدارتها

أمن SQL Server وإدارة المستخدمين

مقدمة

- تشتمل عملية إدارة أمن SQL Server على إدارة عملية الولوج إلى SQL Server وقواعد معطياته وكافة الأغراض التي تحتويها، بالإضافة إلى تحديد سماحيات المستخدمين المسؤولين عن عمليات الولوج تلك؛

- سنستعرض في هذه الجلسة كافة الطرائق التي تسمح بإدارة عملية الأمن تلك وكيفية منح السماحيات المختلفة؛

- ينبغي قبل أن نبدأ بدراسة عملية إدارة أمن SQL Server أن ندرك أن تلك العملية مبنية على أساس نموذج من مستويين، بحيث يتحكم المستوى الأول بعمليات الولوج إلى الأداة SQL Server نفسها، بينما يدير المستوى الثاني عمليات الولوج إلى قواعد المعطيات المختلفة.

- تشتمل عملية إدارة أمن SQL Server على إدارة عملية الولوج إلى SQL Server وقواعد معطياته وكافة الأغراض التي تحتويها، بالإضافة إلى تحديد سماحيات المستخدمين المسؤولين عن عمليات الولوج تلك؛

- سنستعرض في هذه الجلسة كافة الطرائق التي تسمح بإدارة عملية الأمن تلك وكيفية منح السماحيات المختلفة؛

- ينبغي قبل أن نبدأ بدراسة عملية إدارة أمن SQL Server أن ندرك أن تلك العملية مبنية على أساس نموذج من مستويين، بحيث يتحكم المستوى الأول بعمليات الولوج إلى الأداة SQL Server نفسها، بينما يدير المستوى الثاني عمليات الولوج إلى قواعد المعطيات المختلفة؛

- يتم في المستوى الأول من نموذج إدارة أمن SQL Server، التحقق من هوية المستخدم الذي يحاول الولوج إلى الأداة SQL Server بحيث يتم التأكد من أنه يملك حساب دخول أو ما يُعرف باسم login؛

- يتم في المستوى الثاني من نموذج إدارة أمن SQL Server، إدارة عملية منح سماحيات المستخدمين للولوج إلى قواعد المعطيات، بحيث يتم ربط أولئك المستخدمين بحسابات SQL Server التي تم التحقق منها في المستوى الأول السابق، مع العلم أنه يمكن تحديد سماحيات الولوج تلك بقاعدة معطيات أو أكثر؛

- لنفترض أن شخصاً ما يرغب بدخول مكتب معين في بناء، بحيث يتأكد حارس المبنى من أن ذلك الشخص مخول بالدخول إلى المبنى بغرض إجراء عمل ما فيه ويقوم بمرافقته إلى المصعد، يمكن تشبيه ذلك بالمستوى الأول من نموذج أمن SQL Server، في حين يمكننا تشبيه المستوى الثاني بالحالة التي يستخدم فيها ذلك الشخص بطاقة دخول تسمح له باستخدام المصعد للوصول إلى الطابق الذي يريده في ذلك المبنى.

طرائق التحقق من الهوية في SQL Server

يُطلق على الإجراء الذي يتحكم بعملية الولوج إلى الأداة SQL Server ويتحقق من صحّتها اسم عملية التحقق من الهوية، وتتم تلك العملية بطريقتين مختلفتين، بحيث تعتمد الأولى على نظام التشغيل بحد ذاته في حين تستخدم الثانية الأداة SQL Server للتحقق من هوية الاتصال الذي يتم إنشاءه؛

- نموذج نظام التشغيل Windows في التحقق من الهوية:

يتم في هذا النموذج، التحقق من الاتصالات التي يتم إنشاؤها على SQL Server اعتماداً على حساب Windows نفسه الذي يقوم بطلب الاتصال.

- النموذج المختلط في التحقق من الهوية:

يتم في هذا النموذج –وكما يبدو من اسمه– استخدام كلاً من نظام التشغيل Windows والأداة SQL Server للتحقق من الهوية.

- يُطلق على الإجراء الذي يتحكم بعملية الولوج إلى الأداة SQL Server ويتحقق من صحّتها اسم عملية التحقق من الهوية، وتتم تلك العملية بطريقتين مختلفتين، بحيث تعتمد الأولى على نظام التشغيل بحد ذاته في حين تستخدم الثانية الأداة SQL Server للتحقق من هوية الاتصال الذي يتم إنشاءه؛

- نموذج نظام التشغيل Windows في التحقق من الهوية:

يستخدم هذا النموذج لتسهيل عملية الاتصال بـ SQL Server في بيئة نظام التشغيل Windows 2000 أو Windows NT 4.0؛

يتم في هذا النموذج، التحقق من الاتصالات التي يتم إنشاؤها على SQL Server اعتماداً على حساب Windows نفسه الذي يقوم بطلب الاتصال، بحيث يختبر SQL Server وجود login مترابط معه في جدول sysxlogins ويتم منح الاتصال بناءً على ذلك؛

يُطلق على الاتصال المُنشأ من خلال هذا النموذج اسم الاتصال الموثوق، بحيث يثق SQL Server بمتحكم المجال بحد ذاته للتحقق من هوية المتّصل؛

تكمن المنافع التي يمكن تحقيقها من استخدام هذا النموذج في أنه يزودنا بحساب دخول وحيد، كما أنه يسمح لنا بدعم عملية الأمن من خلال استخدام خصائص أمن المجال نفسه كالتحكم بطول كلمة المرور ومدّة صلاحيتها بالإضافة إلى التحكم بعمليات قفل الحساب أو التشفير أو الإشراف على الحسابات ومتابعتها.

- النموذج المختلط في التحقق من الهوية:

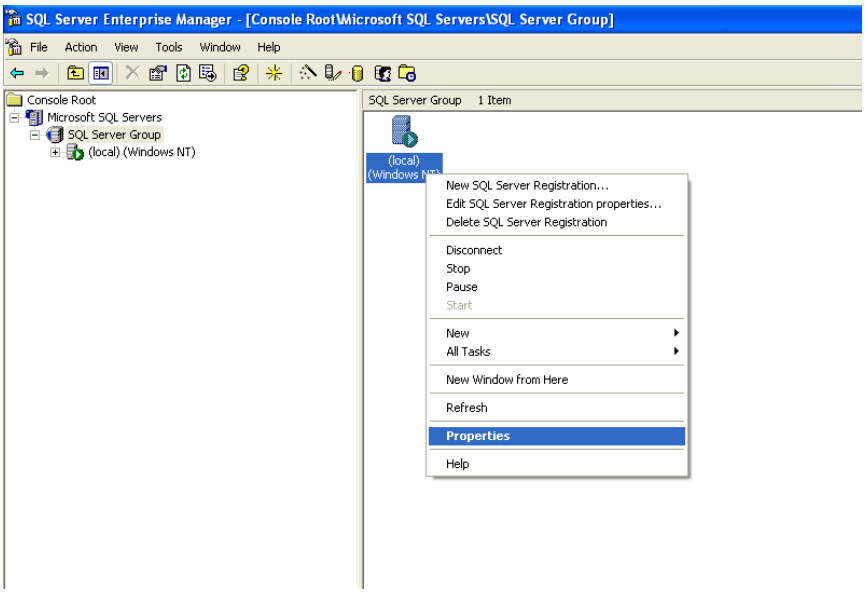
يتم في هذا النموذج –وكما يبدو من اسمه– استخدام كلاً من نظام التشغيل Windows والأداة SQL Server للتحقق من الهوية؛

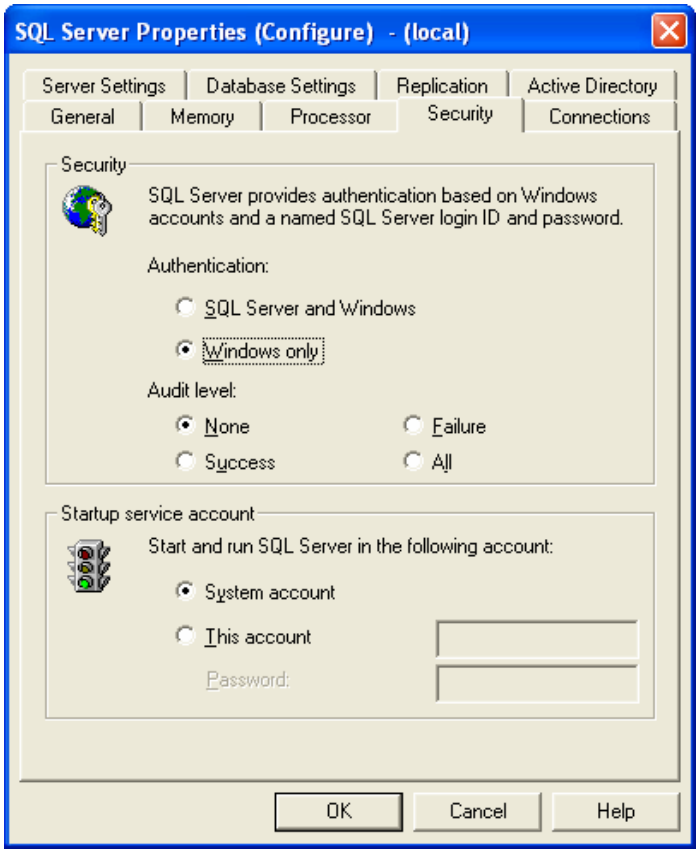
يتم عند محاولة إنشاء اتصال إلى SQL Server يستخدم النموذج المختلط في التحقق من الهوية، عرض واجهة دخول إضافية تسمح للمستخدم بإدخال اسم حساب الدخول وكلمة المرور؛

يفيد هذا النموذج في دعم التطبيقات التي تحاول إنشاء اتصال مع SQL Server وفي بيئة لا يستطيع فيها مُتحكم مجال Windows من يسيطر على عمليات الولوج الشبكية. كما في شبكة NetWare على سبيل المثال.

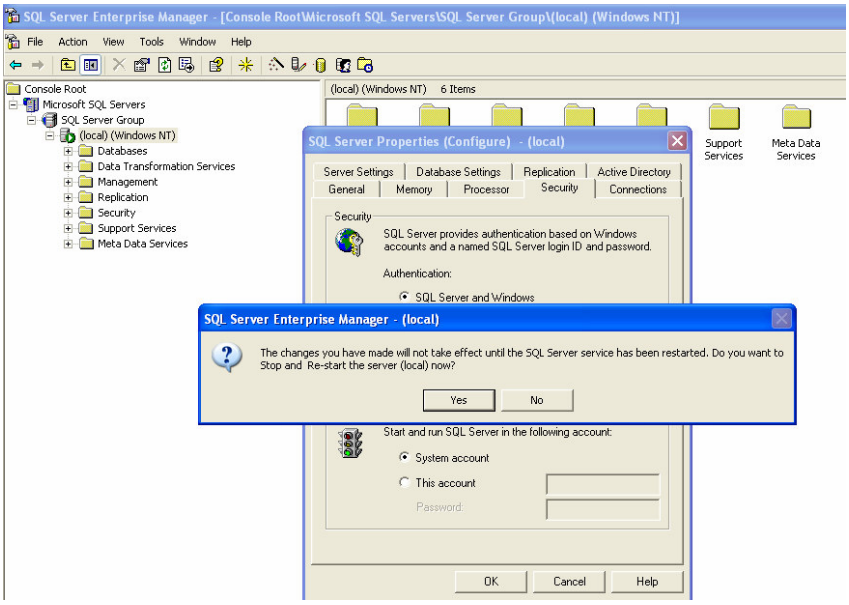
إعداد نموذج التعرف على الهوية

يمكننا تحديد نموذج التعرف على الهوية الذي نرغب باستخدامه إما أثناء عملية التنصيب (كما مر معنا في جلسة تنصيب الأداة SQL Server) أو بعد عملية التنصيب، من خلال اختيار Properties بعد الضغط بالزر اليميني على المخدم، ثم اختيار الواجهة الفرعية Security؛





يمكننا من خلال هذه الواجهة، اختيار نموذج التعرف على الهوية الذي نرغب باستخدامه، أي إما نمـوذج Windows فقـط، أو النموذج المختلط في التعرف على الهوية، مع العلم أن تطبيق تلك التغيرات سيؤدي إلى ضرورة إعادة تشغيل SQL Server؛



يمكننا كذلك أن نحدد في هذه الواجهة مستوى الإشراف على المستخدمين الذي نرغب بتطبيقه؛

السماحيات

تتوقف العمليات التي يمكن القيام بها على المخدم بعد الانتهاء من إنشاء الاتصال إلى SQL Server، على السماحيات المختلفة التي يتم إسنادَها إلى المستخدمين أو مجموعات المستخدمين، أو على الأدوار المختلفة التي تعبر عن مجموعة من السماحيات المترابطة؛

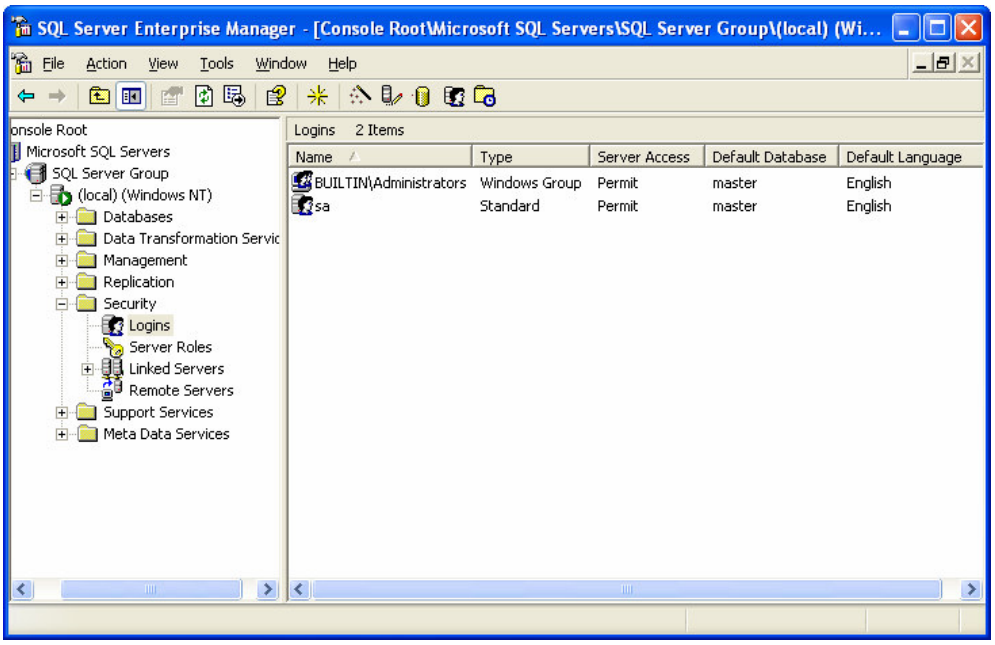
- يطلق على السماحيات التي تسمح بإنشاء وتعديل الأغراض اسم Statement Permissions، كتعليمات إنشاء قاعدة المعطيات أو الجداول أو تعديلها، أي ما يُعرف باسم لغة تعريف المعطيات DDL. تنحصر إمكانية منح هذا النوع من السماحيات فقط بأعضاء كل من الأدوار التالية: sysadmin, db_owner, db_securityadmin؛
- يطلق على السماحيات التي تسمح بالولوج إلى الأغراض اسم Object Permissions، كتعليمات اختيار أو إضافة أو حذف أو تعديل المعطيات، أي ما يُعرف باسم لغة معالجة المعطيات DML. ينحصر هذا النوع من السماحيات بمستخدمي قاعدة المعطيات؛
- يطلق على السماحيات الأخرى التي لا يمكن منحها من خلال المستويين السابقين، اسم السماحيات مسبقة التعريف Predefined Permissions، والتي يمكن الحصول عليها من خلال العضوية في بعض الأدوار الخاصة أو عندما يكون المستخدم هو مالك قاعدة المعطيات؛

أمن SQL Server

حسابات الدخول

- تزودنا حسابات الدخول –أو ما يُعرف باسم Logins– بإمكانيات الولوج إلى الأداة SQL Server؛
- تمثل حسابات الدخول أغراضاً خاصة يتم تخزينها في جدول النظام sysxlogins في قاعدة المعطيات Master؛
- يمكن أن تكون حسابات الدخول عبارة عن حسابات Windows أو مجموعات Windows، أو حساب دخول SQL Server في حال استخدام النموذج المختلط في التحقق من الهوية وهو ما يُعرف كذلك باسم حساب الدخول المعياري Standard Login؛
- ما أن يتم تنصيب الأداة SQL Server حتى يتم إنشاء حسابين هما:
  - مجموعة Windows يطلق عليها اسم builtin/Administrators، والتي تسمح لكافة أعضاء المجموعة Windows Local Administrators بالولوج إلى SQL Server؛
  - حساب دخول معياري يطلق عليه اسم sa.

يعتبر كلا الحسابين السابقين أعضاءً في الدور sysadmin في SQL Server، ما يسمح بإعطائهم امتيازات مطلقة على SQL Server وعلى كافة قواعد المعطيات؛



ملاحظة:
ينبغي الحذر عند التعامل مع الحسابين السابقين، خاصة مع الامتيازات الكبيرة التي يتمتعان بها، وبما أن حساب BUILTIN/Administrators يتضمن كافة المدراء المحللين في Windows، وَجب الحذر في التعامل معه، أما إذا لم نرغب بالسماح بذلك فيمكننا حذف ذلك الحساب وإضافة المستخدمين المخولين فقط، مع العلم أن الحساب sa لا يمكن حذفه، إذ ينبغي في ذلك الحين إسناد كلمة مرور مناسبة له تسمح بالتحكم بعمليات الولوج إلى SQL Server.

- يمكننا بعد الانتهاء من عملية تنصيب الأداة SQL Server، أن نستخدم لحسابين السابقين للولوج إلى تلك الأداة وتعريف المستخدمين الجدد.

أمن SQL Server

المستخدمين

- كما أشرنا سابقاً، تستخدم حسابات الدخول لتأمين عملية الولوج إلى الأداة SQL Server، في حين يتم تأمين عملية الولوج إلى قواعد المعطيات من خلال المستخدمين؛

- بما أن المستخدمين يرتبطون بقاعدة المعطيات، بالتالي يتم إنشاؤهم ضمن قاعدة معطيات ذلك المستخدم ويتم تخزينهم في الجدول sysusers؛

- لابد عند إنشاء مستخدم جديد من أن يتم ربطه مع حساب دخول معين من أجل تأمين الولوج لذلك المستخدم؛

- يتمتع SQL Server بمستخدمَين من نوع خاص، هما dbo و guest:
  - dbo:

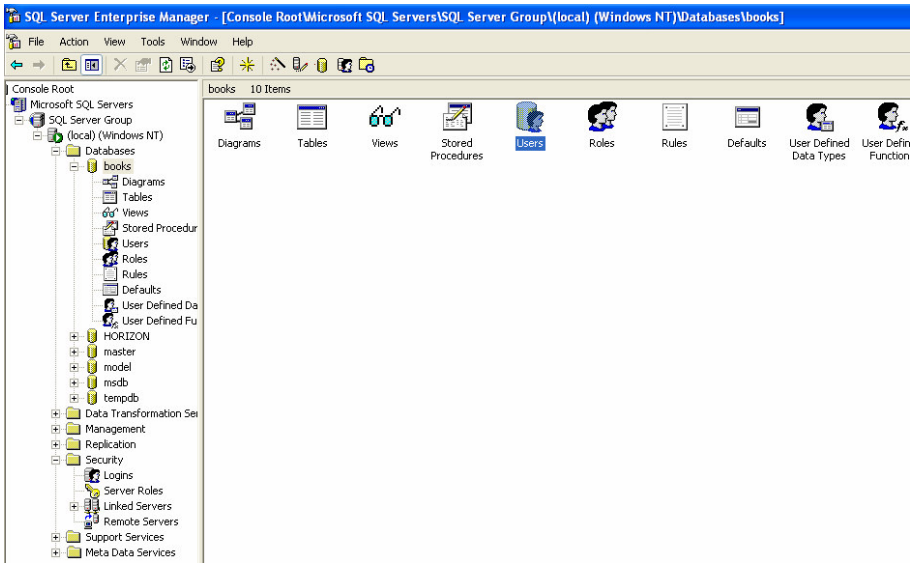
يعتبر حساب dbo هو مالك قاعدة المعطيات، بالتالي لا يمكن حذفه؛
يتم إنشاء هذا الحساب تلقائياً عند كل إنشاء لقاعدة معطيات جديدة؛

  - guest:

يستخدم الحساب guest لتزويد إمكانية ولوج قاعدة معطيات لا تمتلك حساب مستخدم يرتبط بحساب دخول؛
لا يتم إنشاء هذا الحساب تلقائياً، ولكن إنشاءه في قاعدة معطيات ما، يؤدي إلى السماح لكافة حسابات الدخول –التي لا ترتبط بحسابات مستخدمين– من الولوج إلى قاعدة المعطيات تلك واستخدامها؛
ينبغي استخدام هذا الحساب فقط في قواعد المعطيات ذات المستوى المتدني من الأمن.

- كما أشرنا سابقاً، تستخدم حسابات الدخول لتأمين عملية الولوج إلى الأداة SQL Server، في حين يتم تأمين عملية الولوج إلى قواعد المعطيات من خلال المستخدمين؛

- بما أن المستخدمين يرتبطون بقاعدة المعطيات، بالتالي يتم إنشاؤهم ضمن قاعدة معطيات ذلك المستخدم ويتم تخـزينهم فـي الجدول sysusers؛

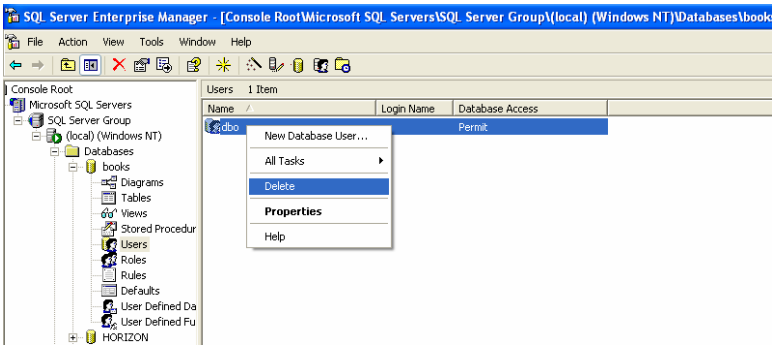


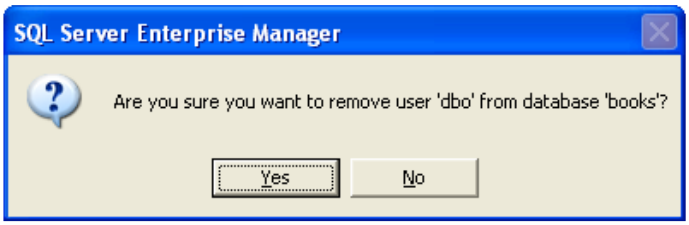
- لابد عند إنشاء مستخدم جديد من أن يتم ربطه مع حساب دخول معين من أجل تأمين الولوج لذلك المستخدم؛

- غالباً ما يتم إنشاء حساب مستخدم وحساب دخول بنفس الاسم وذلك للدلالة على أن المستخدم المخوّل بـالولوج إلـى قاعـدة المعطيات هو نفسه المستخدم المخوّل بالولوج إلى SQL Server، على الرغم من أنهما غرضين مختلفين تمامـاً، إلاّ أن ذلـك لا يعني أن مستخدم قاعدة المعطيات يرتبط بحساب دخول وحيد، إذ أنه من الممكن ربط حساب قاعدة المعطيات بحساب دخول يمثل مجموعة كبيرة من حسابات Windows، بالتالي يمكن إدارة آلاف المستخدمين من خلال حساب دخول وحيـد وحـساب قاعـدة معطيات وحيد؛

- يتمتع SQL Server بمستخدمَين من نوع خاص، هما dbo و guest:
  - dbo:

يعتبر حساب dbo هو مالك قاعدة المعطيات، بالتالي لا يمكن حذفه؛
يتم إنشاء هذا الحساب تلقائياً عند كل إنشاء لقاعدة معطيات جديدة؛





يتم ربط كافة أعضاء الدور sysadmin بحساب المستخدم dbo في كافة قواعد المعطيات، مما يسمح لؤلئك الأعضاء بإدارة قواعد المعطيات تلك؛

بما أن dbo هو الحساب المالك التلقائي لكافة الأغراض في كافة قواعد معطيات المخدم، بالتالي يمكننا استخدام اسم الغرض مباشرةً دون الحاجة إلى توصيف مالك ذلك الغرض، إلا في الحالات التي يتم فيها تعريف مالك آخر لذلك الغرض، مثلاً، يمكننا الدلالة على جدول معين يملكه dbo إما بالشكل dbo.table1 أو بالشكل table1 مباشرةً؛

- guest:

يستخدم الحساب guest لتزويد إمكانية ولوج قاعدة معطيات لا تمتلك حساب مستخدم يرتبط بحساب دخول؛

لا يتم إنشاء هذا الحساب تلقائياً، ولكن إنشاءه في قاعدة معطيات ما، يؤدي إلى السماح لكافة حسابات الدخول –التي لا ترتبط بحسابات مستخدمين– من الولوج إلى قاعدة المعطيات تلك واستخدامها؛

ينبغي استخدام هذا الحساب فقط في قواعد المعطيات ذات المستوى المتدني من الأمن.

أمن SQL Server

الأدوار

- تؤمن الأدوار نموذجاً مرناً ومتسقاً في عملية إدارة الأمن؛

- يمكننا تشبيه الأدوار بالمجموعات المستخدمة في إدارة الشبكات، بحيث يتم تحديد سماحيات الدور ومن ثم تتم إضافة الأعضاء إليه، وبالتالي يتمتع أي عضو بكافة السماحيات التي يمتلكها ذلك الدور؛

- يمكن أن يكون المستخدم عضواً في دور أو أكثر، كما يمكن إسناد مختلف أنواع السماحيات للأدوار –باستثناء application roles التي سنتحدث عنها لاحقاً– مما يسمح لنا بإنشاء واستخدام نموذجاً لإدارة السماحيات؛

- مثال:

يمكننا إنشاء دور خاص يطلق عليه اسم "SAELS" مثلاً، ويحتوي على كافة السماحيات التي تتعلق بعملية البيع؛
كما يمكننا إنشاء دور آخر يطلق عليه اسم "Managers" مثلاً، ويحتوي على كافة السماحيات التي تتعلق بالإدارة؛
بعد ذلك يمكننا ربط مستخدمي المبيعات بالدور الأول والمدراء بالدور الثاني، كما يمكن ربط مدير المبيعات بالدورين السابقين معاً؛

- يصنف SQL Server الأدوار بثلاثة أنواع رئيسية وهي:
  - أدوار المخدم الثابتة، وأدوار قاعدة المعطيات الثابتة. وكلاهما تمتلك سماحيات مسبقة التعريف؛
  - أدوار المستخدم المعرَّفة. وهي الأدوار التي يتم إنشاؤها ضمن قاعدة المعطيات، لإدارة المستخدمين؛
  - أدوار التطبيقات. وهي تتمتع بخصائص تتعلق بضمان الأمن في تطبيقات المستخدم.

سنناقش فيما يلي من شرائح، لمحة سريعة عن كل من تلك الأدوار.

أمن SQL Server

الأدوار

- أدوار المخدم الثابتة:

تُستخدم هذه الأدوار –وكما يبدو من اسمها– لإدارة السماحيات على مستوى المخدم، ولا يمكن تعديلها أو حذفها؛
تستخدم أدوار المخدم الثابتة لإدارة عدة مستويات من السماحيات الإدارية للمستخدمين، فعلى سبيل المثال يمكن من خلالها إعطـــاء سماحيات لمستخدم معين على الشبكة بحيث يمكنه الولوج إلى SQL Server وتعريف حسابات دخول جديدة؛
يعرض الجدول التالي قائمة بأدوار المخدم الثابتة:

الدور	السماحيات
sysadmin	تنفيذ أية عمل
dbcreator	إنشاء وتعديل وحذف قاعدة المعطيات
diskadmin	إنشاء وإدارة أقراص الملفات
processadmin	إدارة إجراءات SQL Server
serveradmin	تغيير إعدادات المخدم أو إغلاقه
setupadmin	تنصيب عملية تكرار قواعد المعطيات Replication وتنصيب مخدمات مترابطة linked Servers
securityadmin	إدارة حسابات دخول SQL Server وكلمات مرورها، وقراءة سجلّ الأخطاء وإدارة سماحيات إنشاء قاعدة المعطيات
bulkadmin	تنفيذ تعليمة إدخال معطيات ضخمة

يمكننا من خلال إجرائيتي النظام ()sp_addsrvrolemember و ()sp_dropsrvrolemember إدارة عملية إضافة أو حذف عضو –على الترتيب– من أدوار المخدم الثابتة؛

مثال:

إضافة المستخدم Maher كعضو من أعضاء الدور sysadmin:

```
Exec sp_ addsrvrolemember  'Maher', 'sysadmin'
GO
```

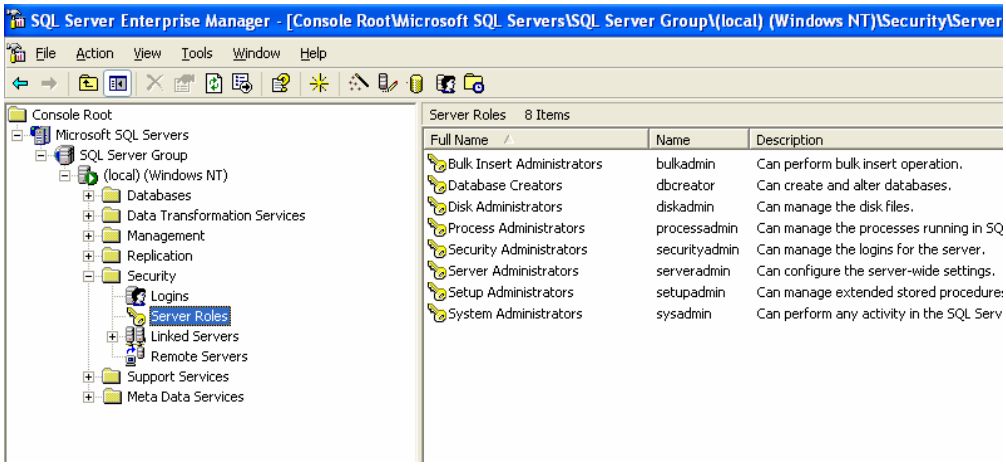
مثال:

حذف المستخدم Maher من بين أعضاء الدور sysadmin:

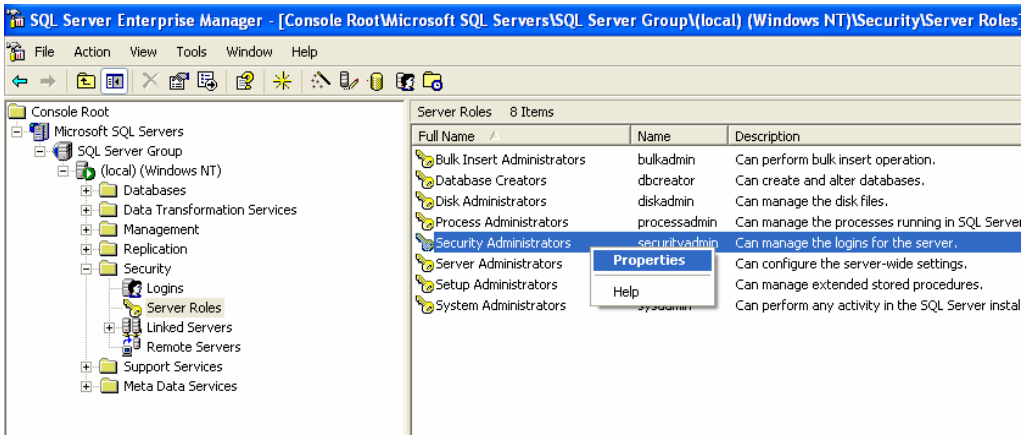
```
Exec sp_dropsrvrolemember  'Maher', 'sysadmin'
GO
```

يمكننا استخدام الأداة Enterprise Manager لإجراء العملية السابقة كما يلي:

من مجلد Security نختار Server Roles لتظهر بعد ذلك قائمة بكافة أدوار المخدم الثابتة؛

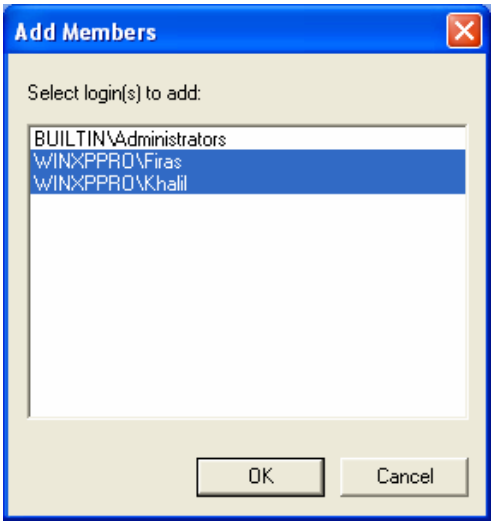


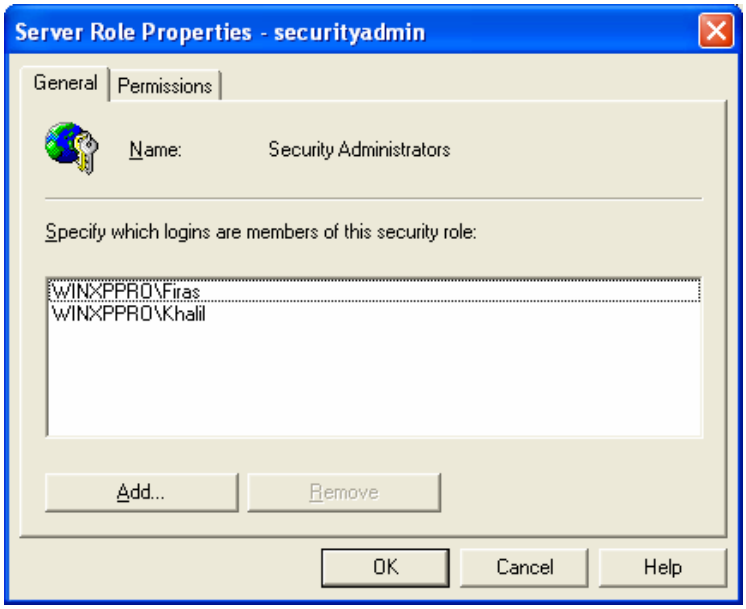
نختار بعد ذلك الدور الذي نرغب بإضافة عضو جديد إليه، ثم نختار Properties من قائمة المهمات السريعة الخاصة به:



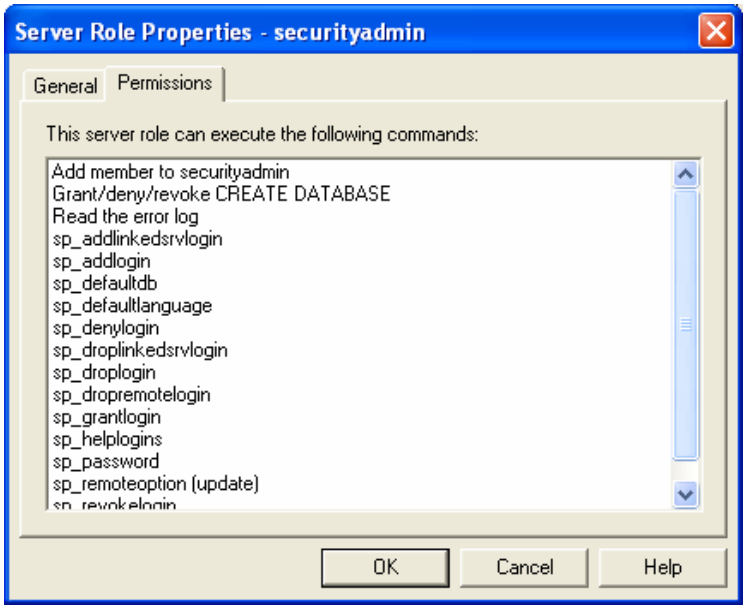
نضغط بعد ذلك على الزر ADD لإضافة عضو جديد إلى الدور المحدد:

ثم نختار العضو أو الأعضاء الجدد الذين نرغب بإضافتهم:





نلاحظ في الواجهة الفرعية Permissions قائمة بكافة السماحيات التي يمكن للدور المحدد أن يقوم بها:



الأدوار

- أدوار قاعدة المعطيات الثابتة:

تمتلك كل قاعدة معطيات مجموعة مسبقة التعريف من الأدوارالتي تُستخدم لإدارة السماحيات على مستوى قاعدة المعطيـــات، ولا يمكن تعديلها أو حذفها فيما عدا Public Role الذي لا يمكن حذفه في حين يمكن تعديل السماحيات التي يتكون منها؛

يعرض الجدول التالي قائمة بأدوار قاعدة المعطيات الثابتة:

السماحيات	الدور
وهو الدور التلقائي الذي يشتمل على كافة السماحيات؛	public
تنفيذ أي نشاط على قاعدة المعطيات؛	db_owner
إضافة حذف أو تعديل أي غرض من أغراض قاعدة المعطيات؛	db_ddladmin
إضافة أو حذف مستخدمين أو أدوار في قاعدة المعطيات؛	db_accessadmin
ربط وإسناد السماحيات وإدارة عضوية الأدوار؛	db_securityadmin
لإنشاء نسخة احتياطية من قاعدة المعطيات؛	db_backupoperator
لقراءة (أو اختيار) المعطيات من أي جدول من جداول المستخدم؛	db_datareader
لإضافة أو تعديل أو حذف المعطيات في أي جدول من جداول المستخدم؛	db_datawriter
لإلغاء إمكانية قراءة المعطيات من أي جدول من جداول المستخدم؛	db_denydatareader
إضافة أو تعديل أو حذف المعطيات من أي جدول من جداول المستخدم.	db_denydatawriter

يمكننا من خلال إجرائيتي النظام sp_addrolemember() و sp_droprolemember () إدارة عملية إضافة أو حذف عضو –على الترتيب– من أدوار قاعدة المعطيات الثابتة؛

مثال:

إضافة المستخدم Firas كعضو من أعضاء الدور db_owner:

```
Exec sp_ addrolemember  'db_owner' , 'Firas'
GO
```

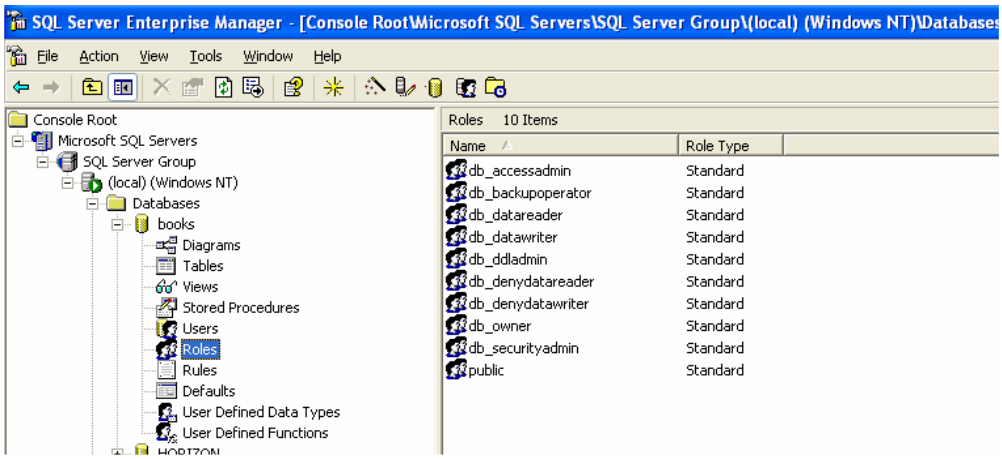
مثال:

حذف المستخدم Firas من بين أعضاء الدور db_owner :

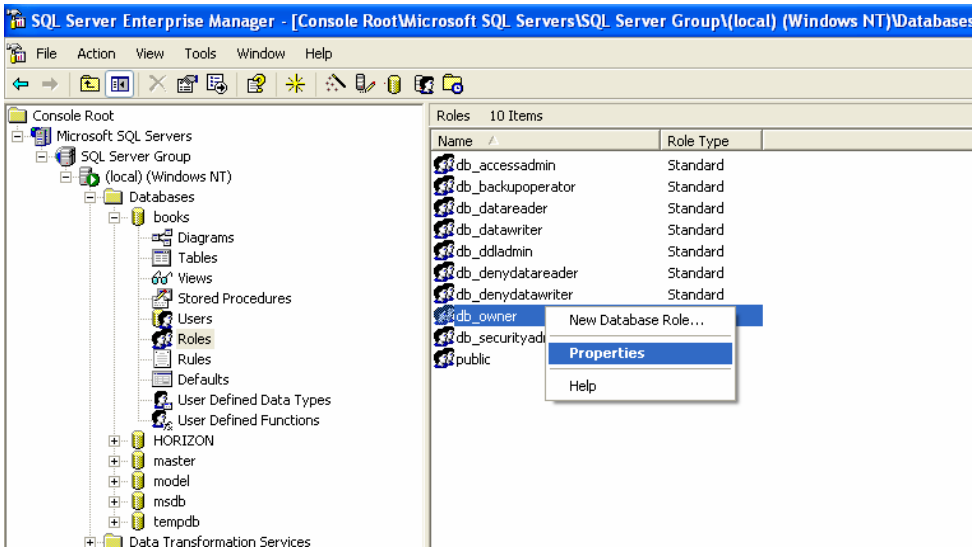
```
Exec sp_droprolemember  'db_owner' , 'Firas'
GO
```

يمكننا استخدام الأداة Enterprise Manager لإجراء العملية السابقة كما يلي:

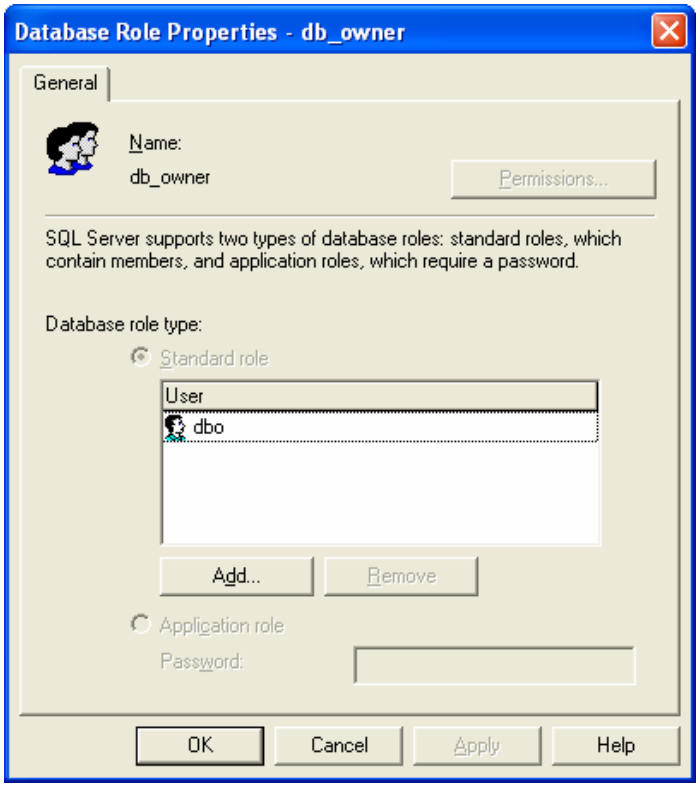
من قاعدة المعطيات التي نعمل عليها نختار Roles لتظهر بعد ذلك قائمة بكافة أدوار قاعدة المعطيات الثابتة؛



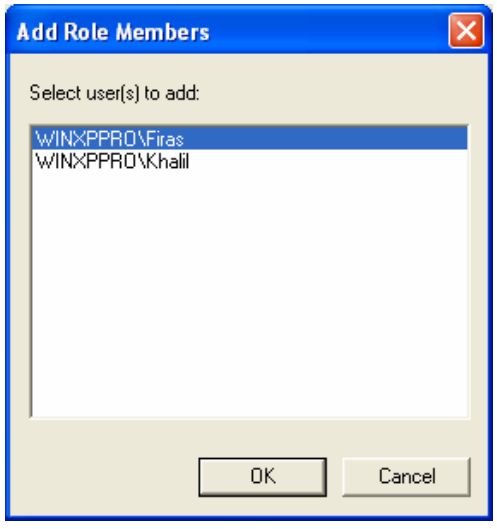
نختار بعد ذلك الدور الذي نرغب بإضافة عضو جديد إليه، ثم نختار Properties من قائمة المهمات السريعة الخاصة به:

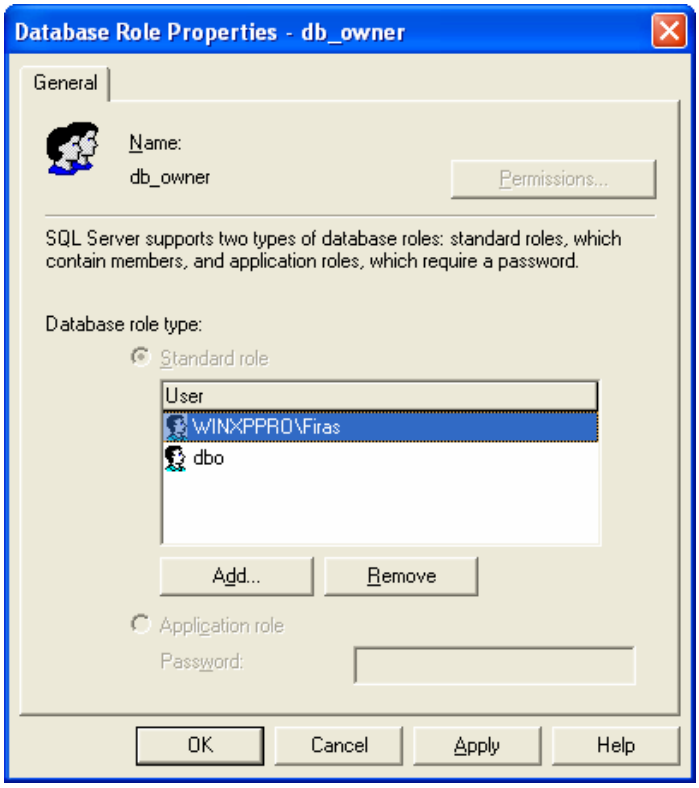


نضغط بعد ذلك على الزر ADD لإضافة عضو جديد إلى الدور المحدد:



ثم نختار العضو أو الأعضاء الجدد الذين نرغب بإضافتهم (مع العلم أنه ينبغي أن يتم تعريف مستخدمي قاعدة المعطيات تلك مسبقاً <كما سيمر معنا لاحقاً>):

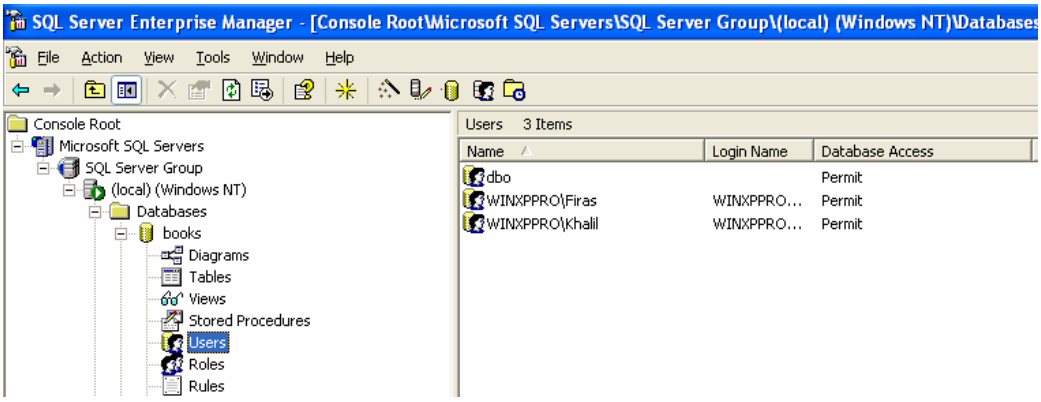




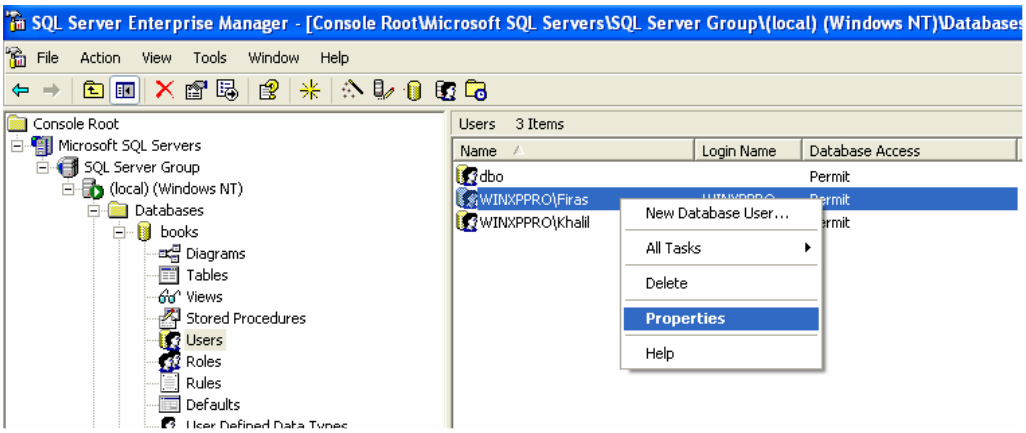
ملاحظة:

نستطيع القيام بالعملية السابقة بأسلوب آخر في الأداة Enterprise Manager، وذلك كما يلي:

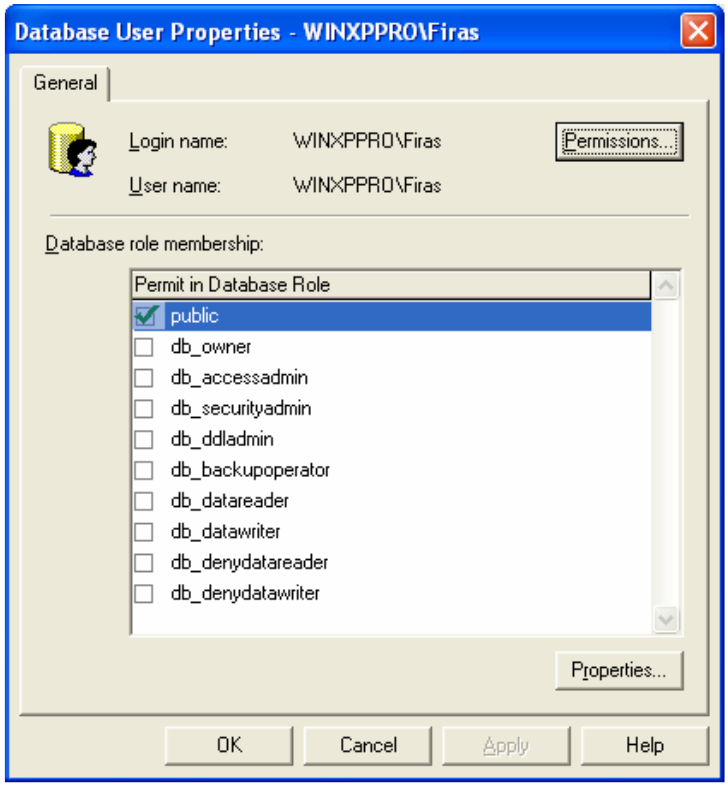
من قاعدة المعطيات التي نعمل عليها نختار Users لتظهر بعد ذلك قائمة بكافة مستخدمي قاعدة المعطيـــات، أي dbo المـــستخدم التلقائي، وغيره من مستخدمين تمت إضافتهم (سنناقش كيفية إضافة المستخدمين إلى قاعدة المعطيات لاحقاً)؛

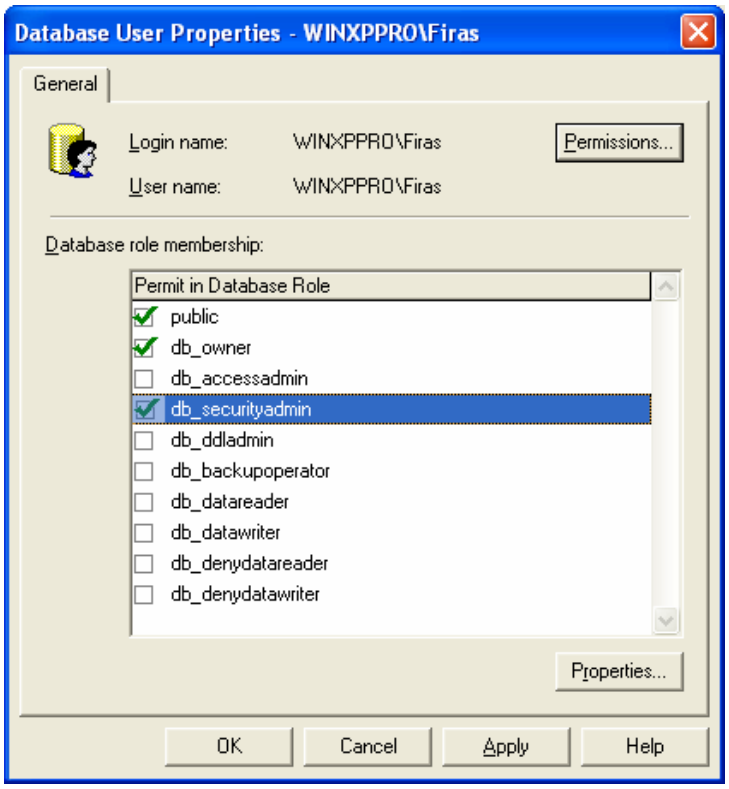


نختار بعد ذلك المستخدم الذي نرغب بإضافته إلى دور معين، ثم نختار Properties من قائمة المهمات السريعة الخاصة به:



تظهر بعد ذلك واجهة تعرض كافة سماحيات قاعدة المعطيات الثابتة التي يتمتع بها أو لا يمتلكها ذلك المستخدم، بحيث يمكننا اختيار السماحيات التي نرغب بها:





أمن SQL Server

الأدوار

- أدوار المستخدم المعرَّفة:

تُستخدم أدوار المُستخدم المعرَّفة –أو ما يُعرف أيضاً باسم أدوار قاعدة المعطيات– لتزويد مجموعة مخصصة من السماحيات الشائعة التي يمكن إسنادها لمجموعة من المستخدمين؛

يمكن أن تتضمن أدوار المستخدم المعرَّفة، المستخدمين أو الأدوار على حد سواء، مما يمكن أن يفيد في بناء نموذج أمن على مستوى نظام إدارة قواعد المعطيات؛

مثال:

لنفترض أننا نقوم بإنشاء دور X معين مكون من مجموعة من السماحيات التي تسمح بتشغيل تطبيق معين, بعد ذلك نقوم إنشاء الأدوار Y و Z بسماحيات أخرى ونُضيف المستخدمين المناسبين إليها؛

يمكننا بإضافة الأدوار Y و Z إلى الدور X أن نسمح لمستخدمي الدورين الأولين أن يقوموا بتشغيل التطبيق الذي يمكن لمستخدمي الدور الأخير X أن يقوموا بتشغيله؛

يمكننا من خلال إجرائيتي النظام sp_addrole() و sp_droprole() إدارة عملية إضافة أو حذف دور مستخدم مُعرف –على الترتيب– ؛

مثال:

فيما يلي عرض للمخطوط المستخدم لإضافة الدور X الذي يملكه المستخدم dbo:

```
Exec sp_ addrole 'X' , 'dbo'
GO
```

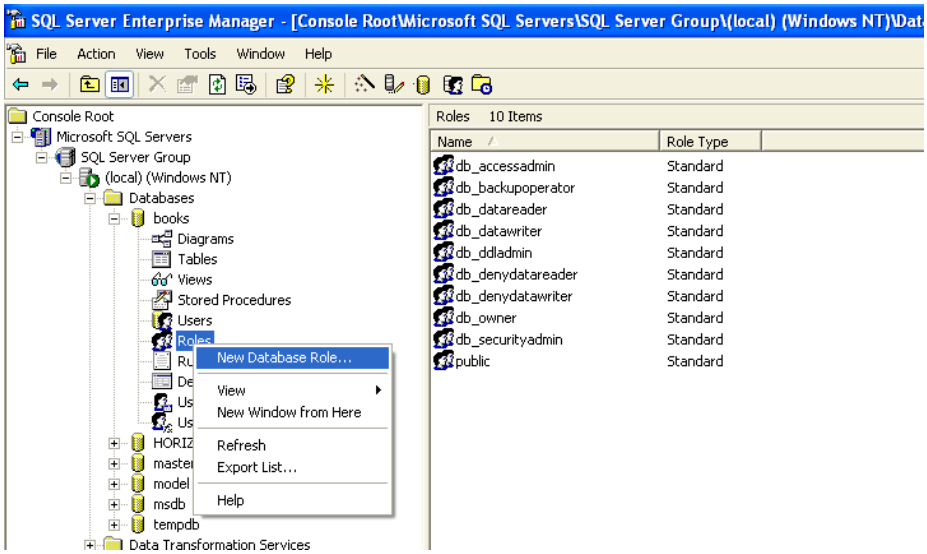
مثال:

فيما يلي عرض للمخطوط المستخدم لحذف الدور X ، مع العلم أنه لا يمكننا حذف دور معين قبل حذف كافة مالكيه أولا، وبالتالي سنقوم بحذف المالك dbo من الدور X قبل حذف ذلك الدور:

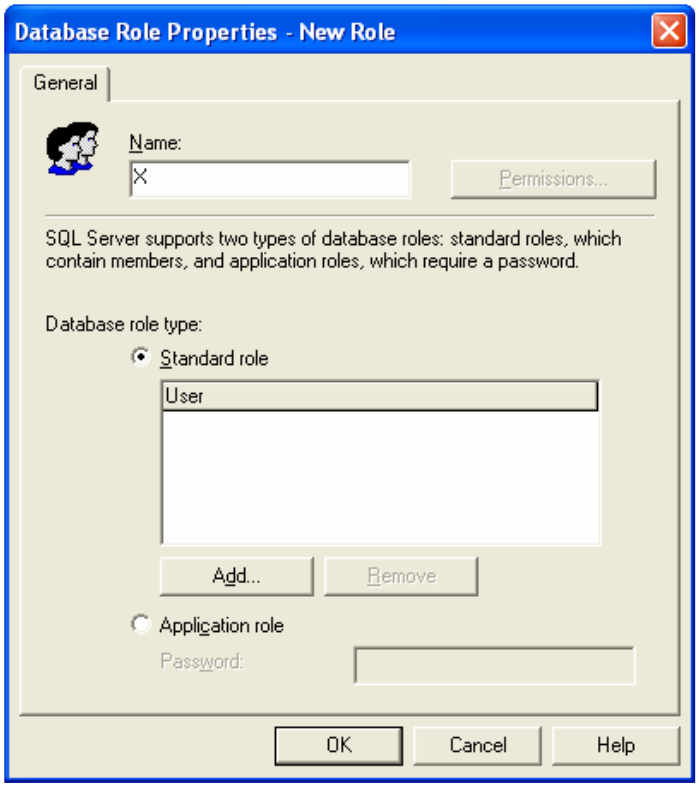
```
Exec sp_droprolemember 'X' , 'dbo'
GO

Exec sp_droprole 'X'
GO
```

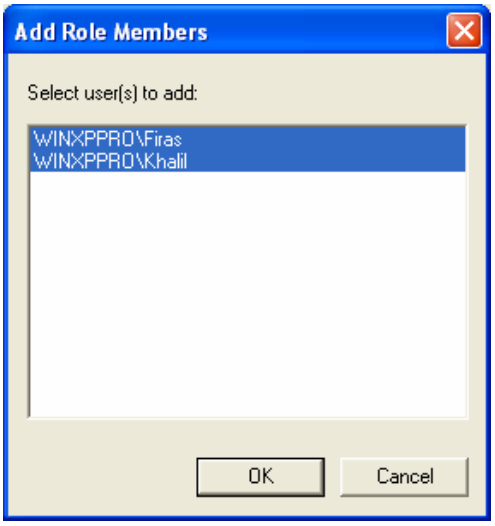
يمكننا استخدام الأداة Enterprise Manager لإجراء العمليات السابقة كما يلي:

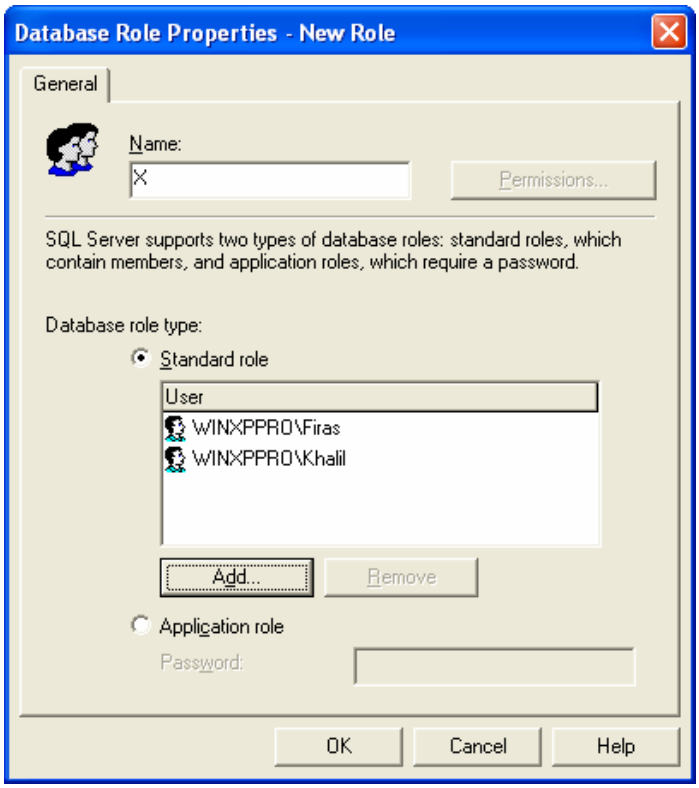
من قاعدة المعطيات التي نعمل عليها نختار Roles لتظهر بعد ذلك قائمة بكافة أدوار قاعدة المعطيات الثابتة، ثـم نختـار New Database Role… من قائمة المهمات السريعة لـ Roles؛



نحدد بعد ذلك اسم الدور الجديد ونضغط على الزر ADD لإضافة عضو جديد إلى الدور المحدد:

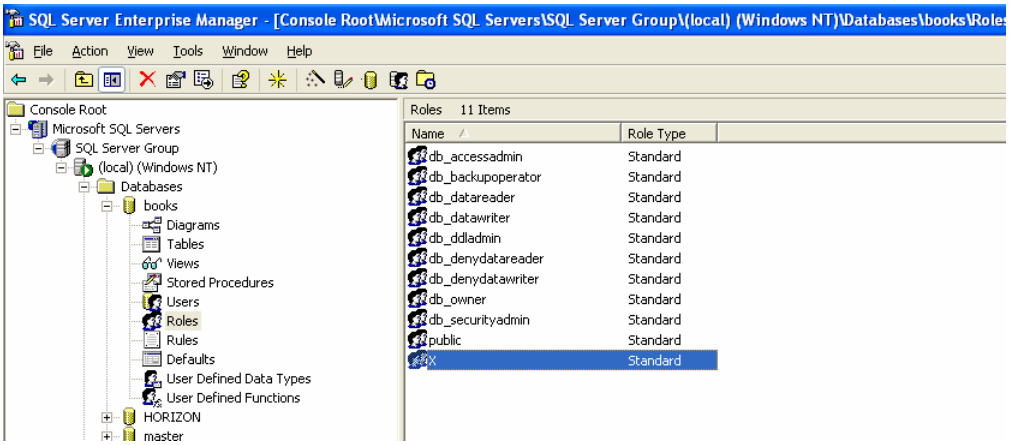


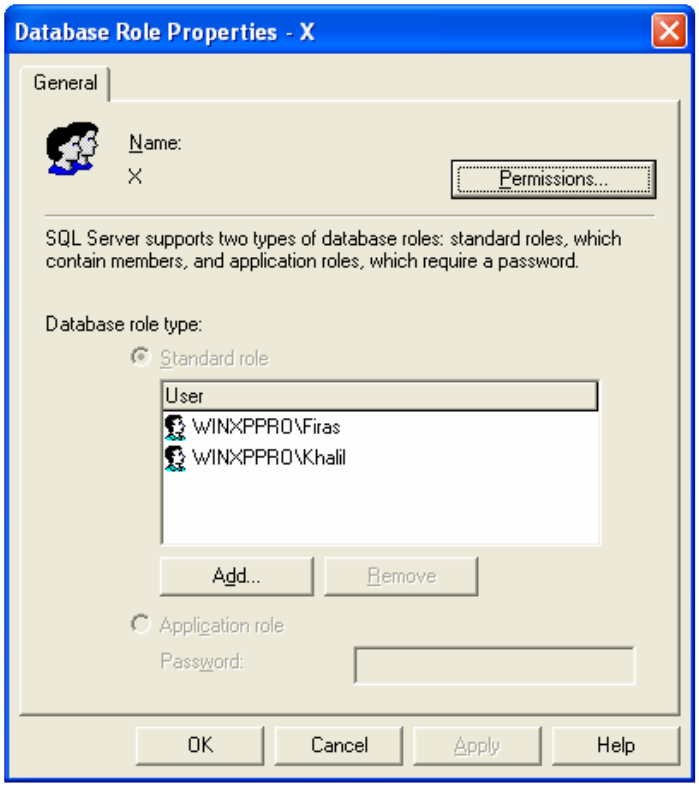


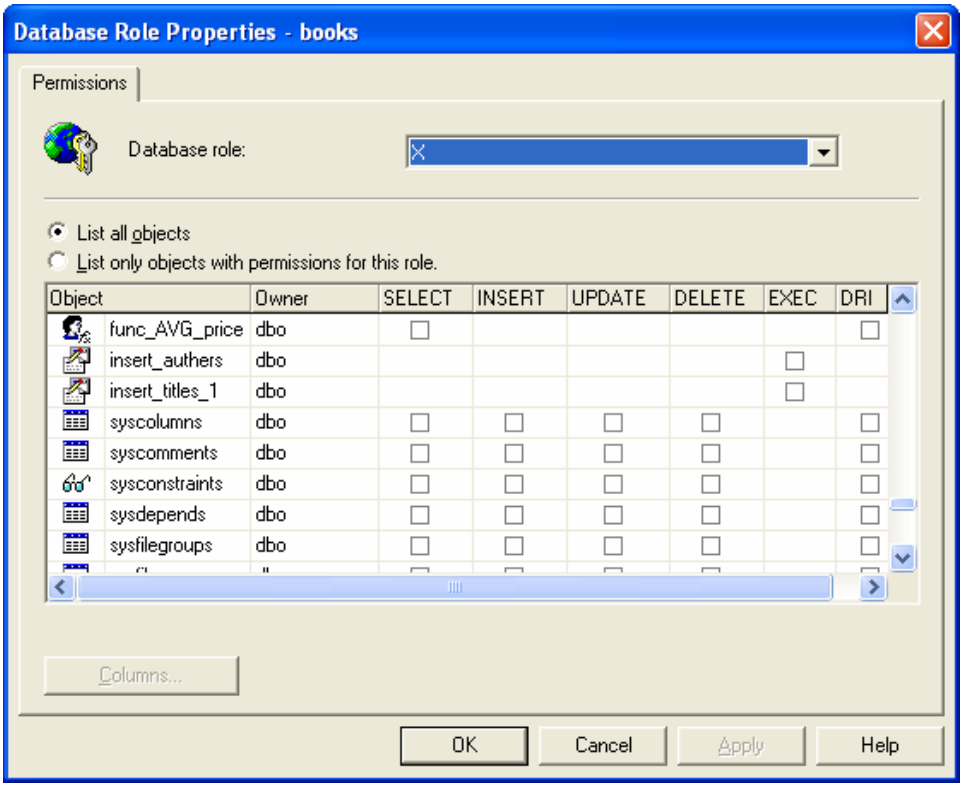


لا يمكننا تحديد سماحيات المستخدمين قبل أن ننتهي من إنشاء الدور؛
(اقتراح)

(هنا ينبغي متابعة عرض كيفية الانتهاء من إنشاء الدور ثم العودة إلى قائمة الأدوار لاستعراض الدور الجديد ثم استعراض سماحياته كما في الواجهات التالية)







أمن SQL Server

الأدوار

- أدوار التطبيقات:

تختلف أدوار التطبيقات عن غيرها من الأدوار في أنها لا تمتلك مستخدمي قاعدة المعطيات؛

لا نقوم عند إنشاء دور من أدوار التطبيقات بربطه مع مستخدمي قاعدة المعطيات في حين نقوم بإسناد كلمة مرور مناسبة إليه؛

تُستخدم أدوار التطبيقات لتأمين السماحيات للأغراض من خلال التطبيقات **فقط وبالتالي لا يمكن للمستخدم أن يحصل على السماحيات ما لم يقوم بتشغيل التطبيق المناسب والذي ينقل من خلال عملية اتّصاله مع قاعدة المعطيات كلمة المرور المناسبة للوصول إلى سماحيات التشغيل المطلوبة؛**

يمكننا من خلال الإجرائية sp_setapprole() أن نقوم بإضافة دور تطبيق جديد مع كلمة مرور مرافقة؛

مثال:

فيما يلي عرض للمخطوط المستخدم لإضافة الدور myAppRole مع كلمة المرور abcdef:

```
Exec sp_ setapprole  'myAppRole' , 'abcdef'
GO
```

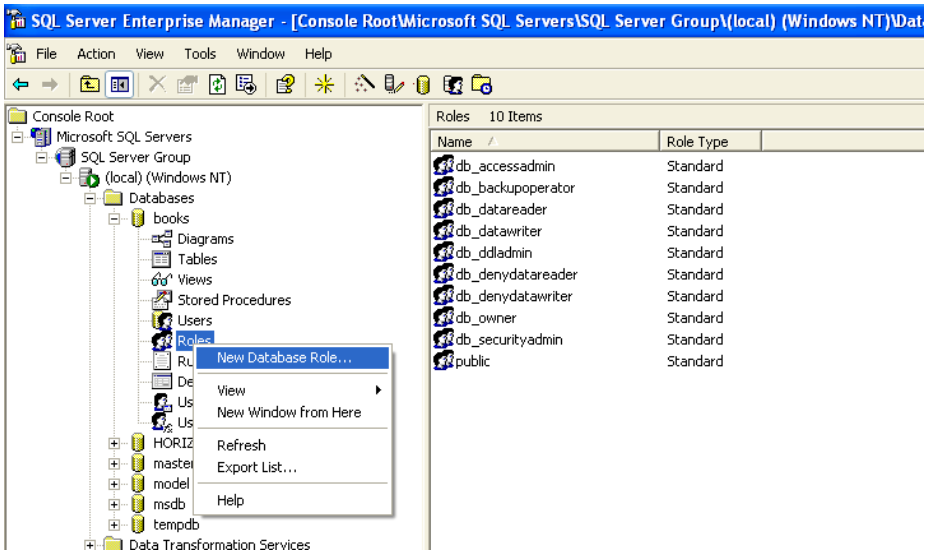
مثال:

فيما يلي عرض للمخطوط المستخدم لإضافة الدور myAppRole مع كلمة مرور مشفرة abcdef:

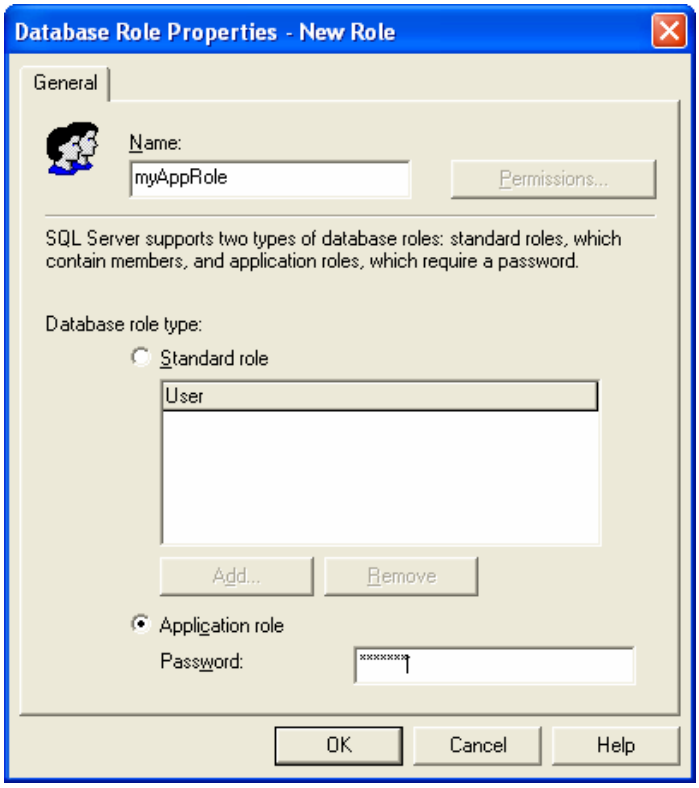
```
Exec sp_ setapprole  'myAppRole' ,{ENCRYPT N 'abcdef'}, 'odbc'
GO
```

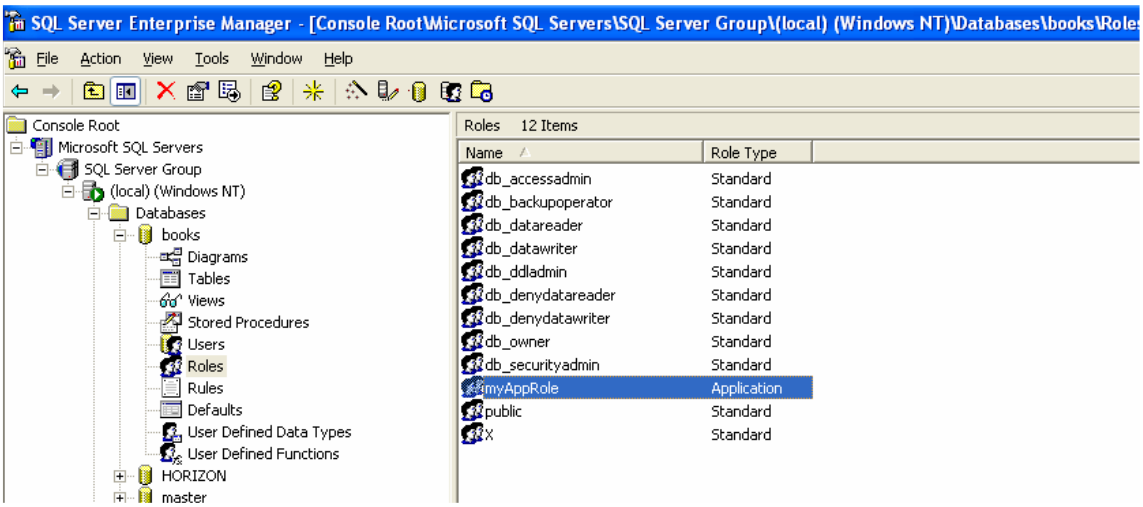
يمكننا استخدام الأداة Enterprise Manager لإجراء العمليات السابقة كما يلي:

من قاعدة المعطيات التي نعمل عليها نختار Roles لتظهر بعد ذلك قائمة بكافة أدوار قاعدة المعطيات الثابتة، ثـم نختـار New Database Role… من قائمة المهمات السريعة لـ Roles؛



نحدد بعد ذلك اسم الدور الجديد وثم نختار Application role ونقوم بإدخال كلمة المرور المناسبة:


Database Role Properties - New Role
General
Name:
myAppRole
Permissions...
SQL Server supports two types of database roles: standard roles, which contain members, and application roles, which require a password.
Database role type:
Standard role
User
Add...
Remove
Application role
Password:
OK
Cancel
Help


SQL Server Enterprise Manager - [Console Root\Microsoft SQL Servers\SQL Server Group\(local) (Windows NT)\Databases\books\Role
File Action View Tools Window Help
Console Root
Microsoft SQL Servers
SQL Server Group
(local) (Windows NT)
Databases
books
Diagrams
Tables
Views
Stored Procedures
Users
Roles
Rules
Defaults
User Defined Data Types
User Defined Functions
HORIZON
master
Roles 12 Items
Name
Role Type
db_accessadmin Standard
db_backupoperator Standard
db_datareader Standard
db_datawriter Standard
db_ddladmin Standard
db_denydatareader Standard
db_denydatawriter Standard
db_owner Standard
db_securityadmin Standard
myAppRole Application
public Standard
x Standard

إدارة حسابات دخول SQL Server

- إدارة حسابات دخول SQL Server باستخدام T-SQL:

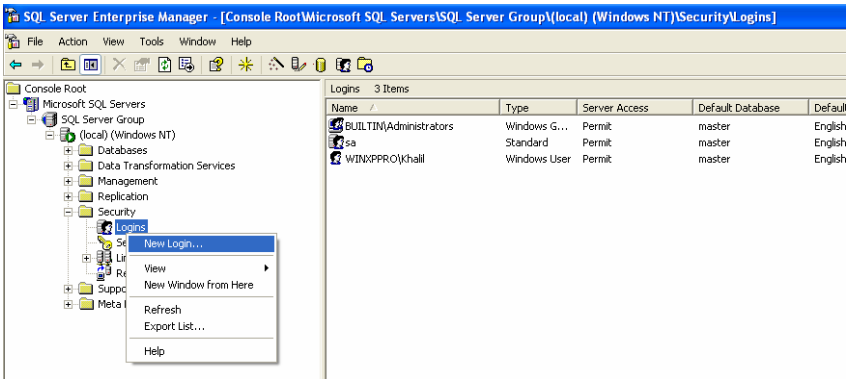
فيما يلي جدول بقائمة إجرائيات مخزّنة تستخدم من أجل إدارة حسابات الدخول:

الوصف	الإجرائية
لإضافة حساب دخول جديد لـــ SQL Server مثال: **EXEC sp_addlogin** (*login_name, password, database_name*)	sp_addlogin
لإضافة حساب دخول جديد لـــ Windows مثال: Exec sp_grantlogin 'Domain1\Feras'	sp_grantlogin
لحذف حساب دخول SQL Server	sp_droplogin
لحذف حساب دخول Windows	sp_revokelogin
رفض ولوج حساب Windows	sp_denylogin
تغيير كلمة مرور حساب الدخول	sp_password
تغيير قاعدة المعطيات التلقائية	sp_defaultdb
تغيير اللغة التلقائية المستخدمة	sp_defaultlanguage

- إدارة حسابات دخول SQL Server باستخدام الأداة Enterprise Manager:

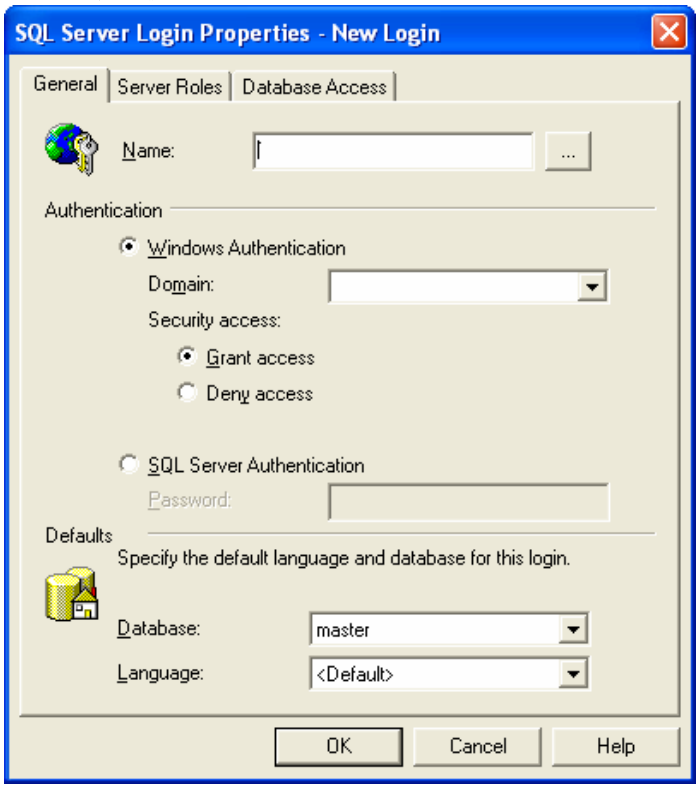
يمكننا استخدام الأداة Enterprise Manager لإجراء العمليات السابقة كما يلي:

نختار الرمز Logins من مجلد Security في الأداة Enterprise Manager، ثم نختار New Login من قائمة المهمات السريعة لذلك الرمز

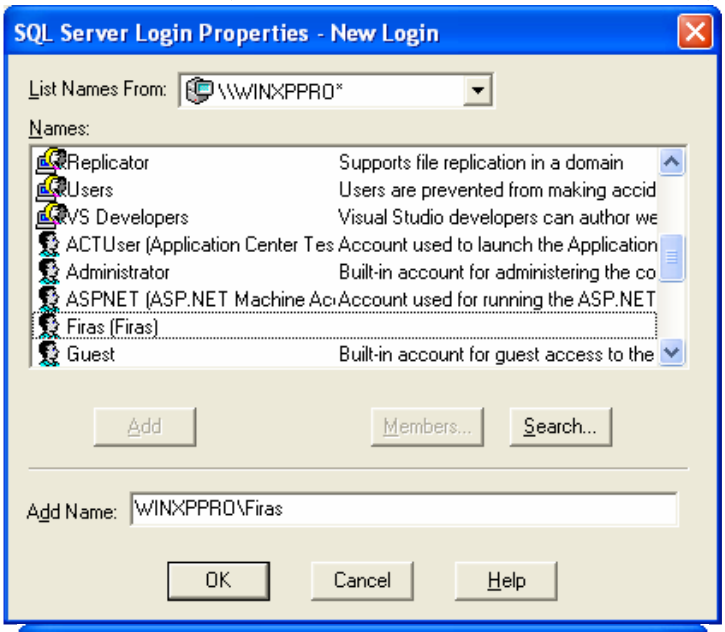


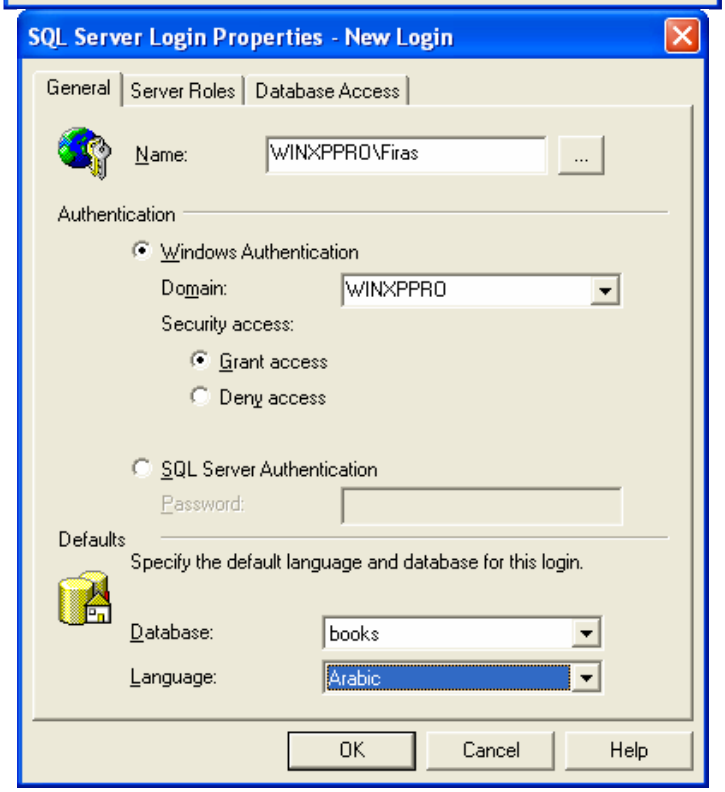
لتظهر بعد ذلك الواجهة الخاصة بتعريف حساب دخول جديد بحيث تتكون من ثلاث واجهات فرعية؛

يمكننا من خلال الواجهة الفرعية General أن نقوم بتحديد اسم حساب الدخول الجديد، كما يمكننا تحديد نموذج التعرف على الهوية الذي نرغب باستخدامه مع ذلك الحساب الجديد، كما يمكننا فيها منح أو عدم منح إمكانيات الولوج لذلك الحساب المحدد، بالإضافة إلى تحديد قاعدة المعطيات التلقائية واللغة التلقائية التي سيتم استخدامها؛

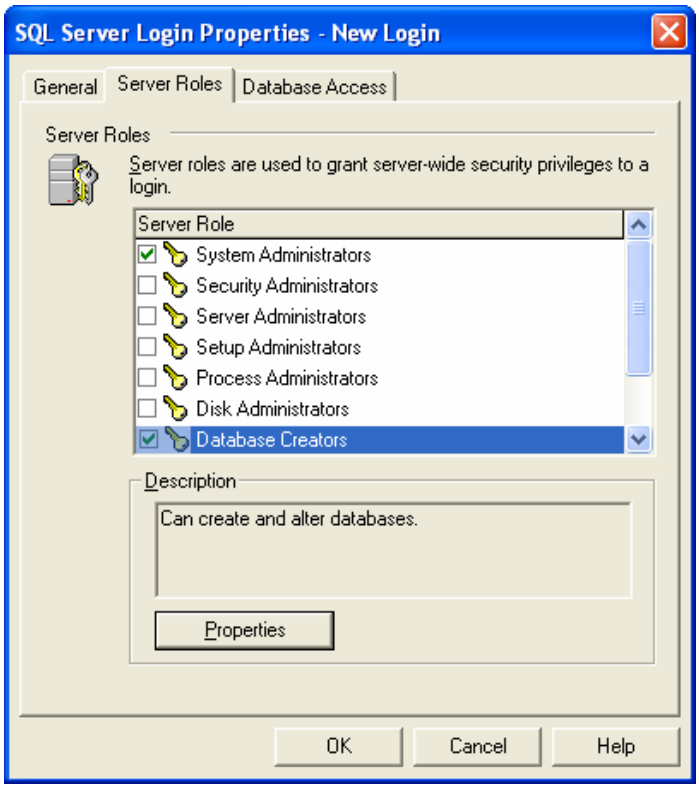


يمكننا كذلك إما تحديد اسم حساب الدخول أو اختيار انتقاءه من بين حسابات نظام التشغيل:

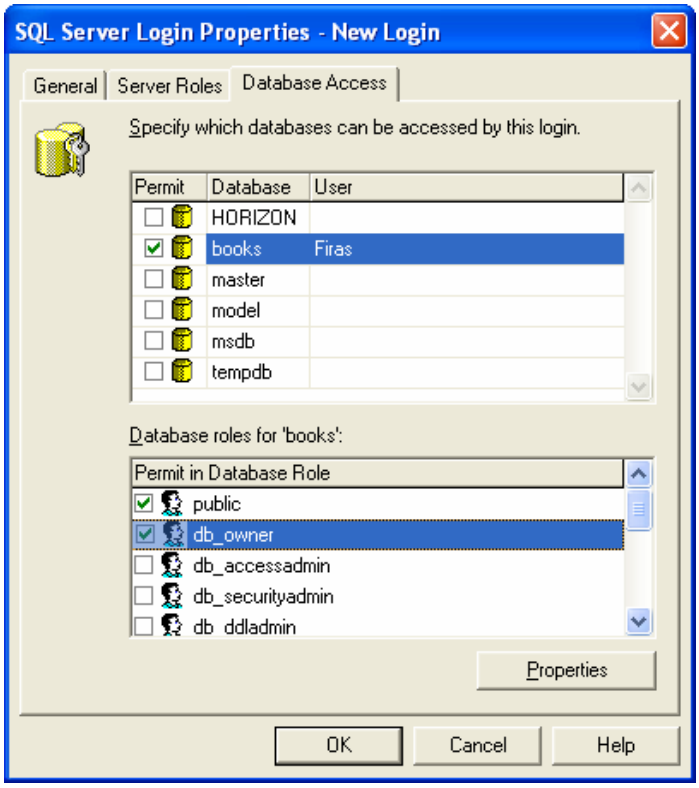


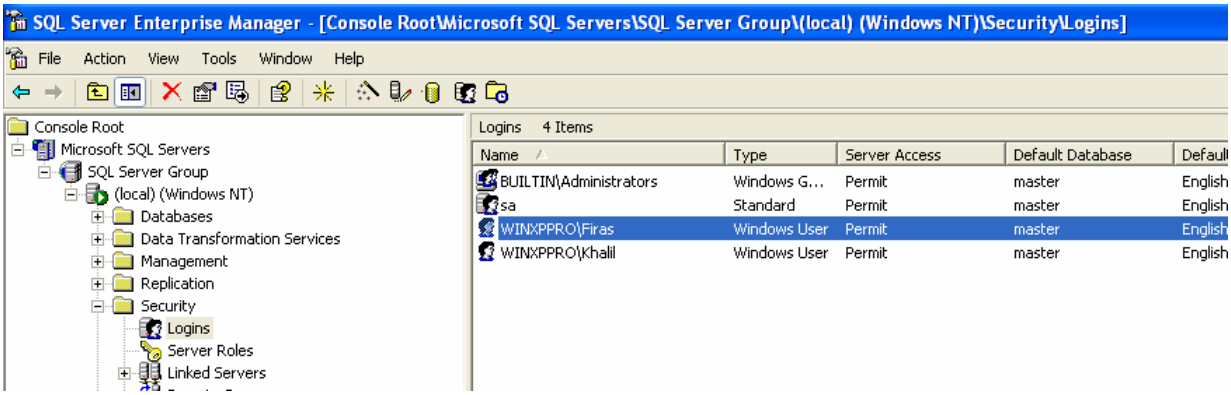


نستطيع من خلال الواجهة الفرعية الثانية Server Roles أن نحدد كافة أدوار المخدم التي يمكن لحساب الدخول الجديد أن ينتمي إليها:



يمكننا كذلك من خلال الواجهة الفرعية الثالثة Database Access أن نحدد كافة أدوار قاعدة المعطيات التي يمكن لحساب الدخول الجديد أن ينتمي إليها:

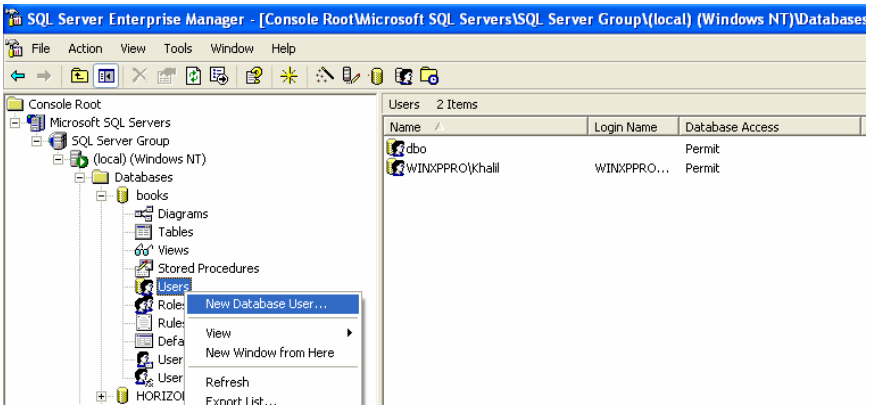

SQL Server Login Properties - New Login
General
Server Roles
Database Access
Specify which databases can be accessed by this login.
Permit
Database
User
HORIZON
books
Firas
master
model
msdb
tempdb
Database roles for 'books':
Permit in Database Role
public
db_owner
db_accessadmin
db_securityadmin
db_ddladmin
Properties
OK
Cancel
Help


SQL Server Enterprise Manager - [Console Root\Microsoft SQL Servers\SQL Server Group\(local) (Windows NT)\Security\Logins]
File
Action
View
Tools
Window
Help
Console Root
Microsoft SQL Servers
SQL Server Group
(local) (Windows NT)
Databases
Data Transformation Services
Management
Replication
Security
Logins
Server Roles
Linked Servers
Logins 4 Items
Name
Type
Server Access
Default Database
BUILTIN\Administrators
Windows G...
Permit
master
English
sa
Standard
Permit
master
English
WINXPPRO\Firas
Windows User
Permit
master
English
WINXPPRO\Khalil
Windows User
Permit
master
English

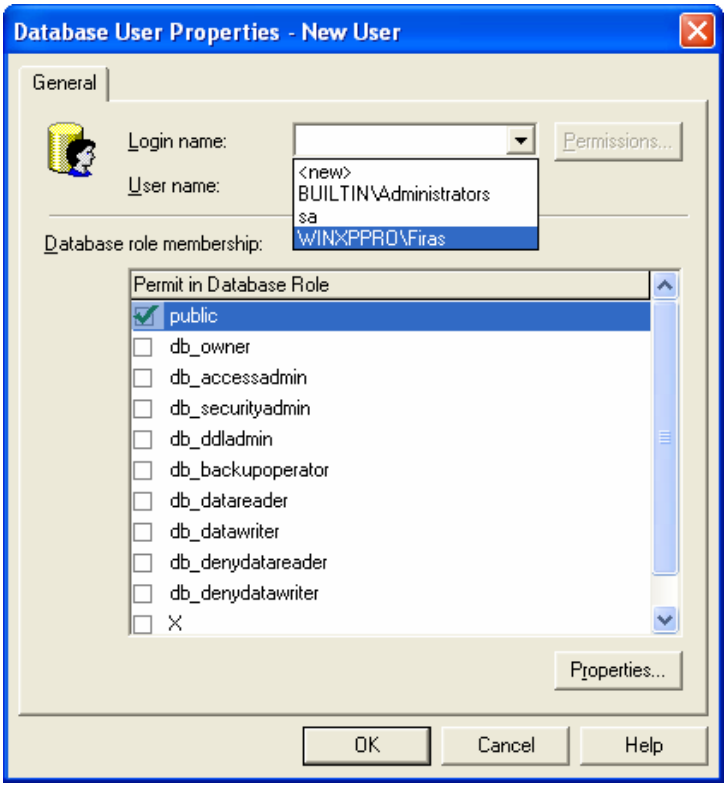
- إدارة حسابات دخول SQL Server باستخدام Enterprise Manager:

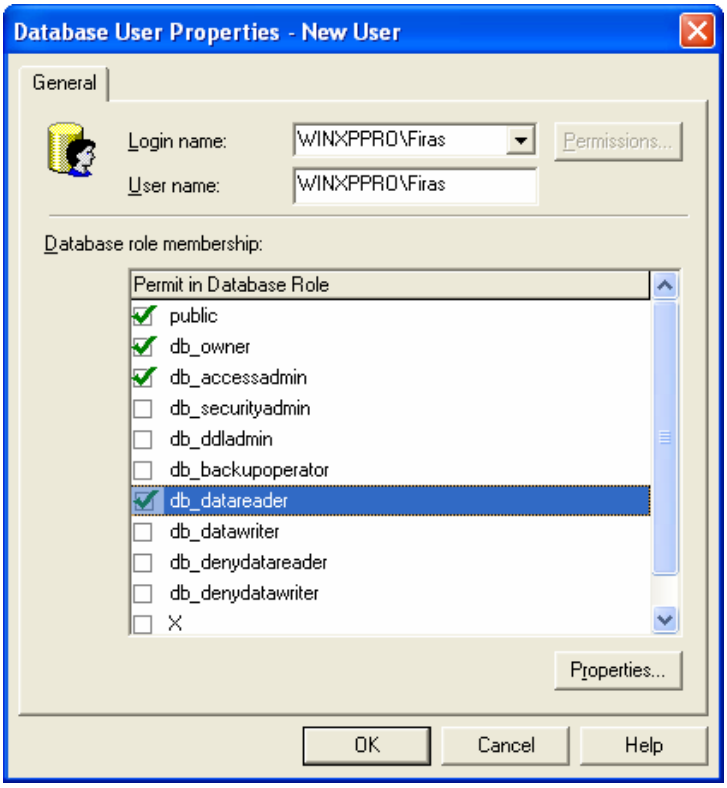
يمكننا استخدام الأداة Enterprise Manager لإضافة مستخدم جديد إلى قاعدة معطيات معينة كما يلي:

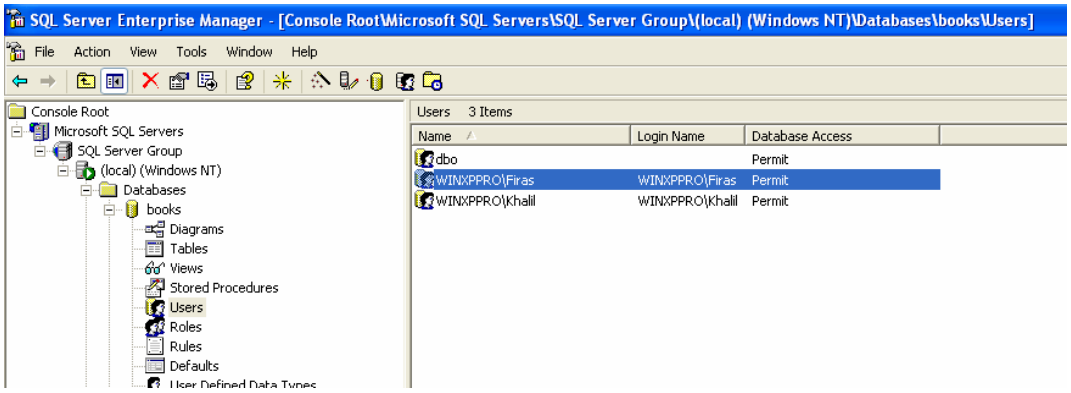
نختار الرمز Users من مجلد قاعدة المعطيات المناسبة في الأداة Enterprise Manager، ثم نختار New Database User من قائمة المهمات السريعة لذلك الرمز:



لتظهر بعد ذلك الواجهة الخاصة بتعريف حساب مستخدم جديد بحيث نقوم بتحديد اسم حساب المستخدم الجديد، بالإضافة إلى تحديد سماحيات قاعدة المعطيات لذلك الحساب؛

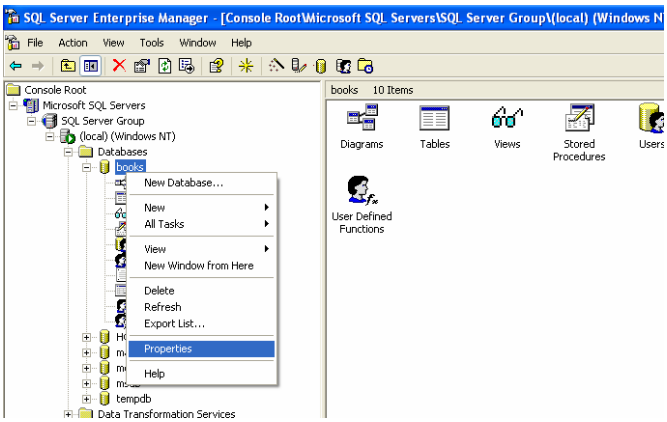

Database User Properties - New User
General
Login name:
User name:
Permissions...
<new>
BUILTIN\Administrators
sa
WINXPPRO\Firas
Database role membership:
Permit in Database Role
public
db_owner
db_accessadmin
db_securityadmin
db_ddladmin
db_backupoperator
db_datareader
db_datawriter
db_denydatareader
db_denydatawriter
Properties...
OK
Cancel
Help

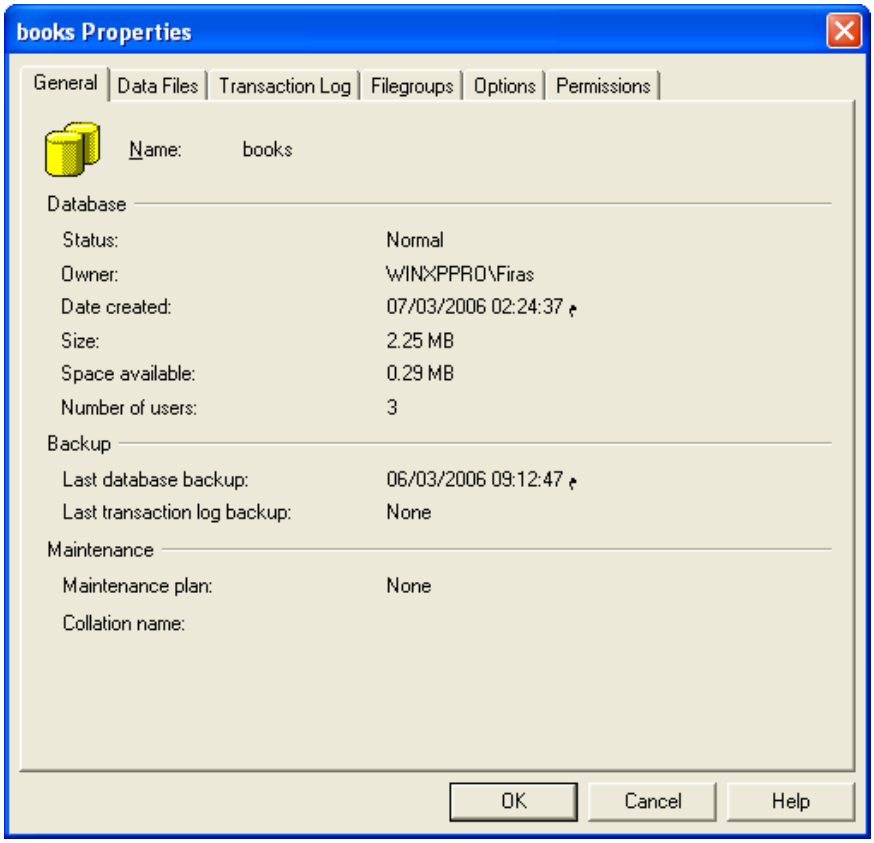

Database User Properties - New User
General
Login name:
WINXPPRO\Firas
Permissions...
User name:
WINXPPRO\Firas
Database role membership:
Permit in Database Role
public
db_owner
db_accessadmin
db_securityadmin
db_ddladmin
db_backupoperator
db_datareader
db_datawriter
db_denydatareader
db_denydatawriter
Properties...
OK
Cancel
Help

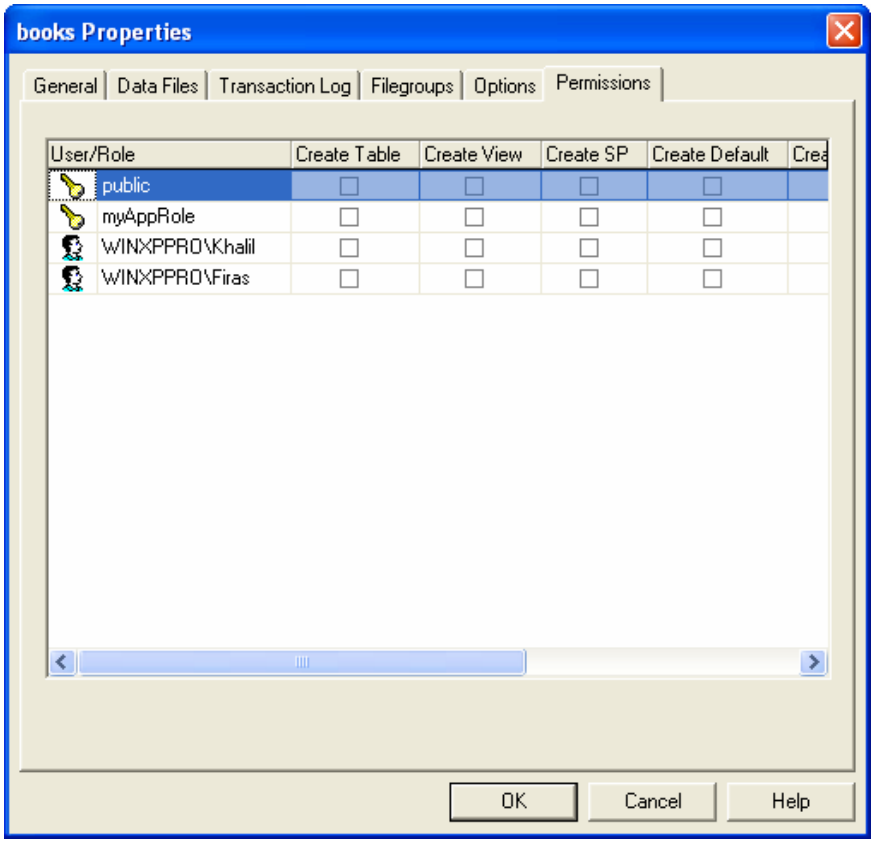

SQL Server Enterprise Manager - [Console Root\Microsoft SQL Servers\SQL Server Group\(local) (Windows NT)\Databases\books\Users]
File Action View Tools Window Help
Console Root
Microsoft SQL Servers
SQL Server Group
(local) (Windows NT)
Databases
books
Diagrams
Tables
Views
Stored Procedures
Users
Roles
Rules
Defaults
Users 3 Items
Name
Login Name
Database Access
dbo
Permit
WINXPPRO\Firas
WINXPPRO\Firas
Permit
WINXPPRO\Khalil
WINXPPRO\Khalil
Permit

- إدارة سماحيات SQL Server باستخدام Enterprise Manager:

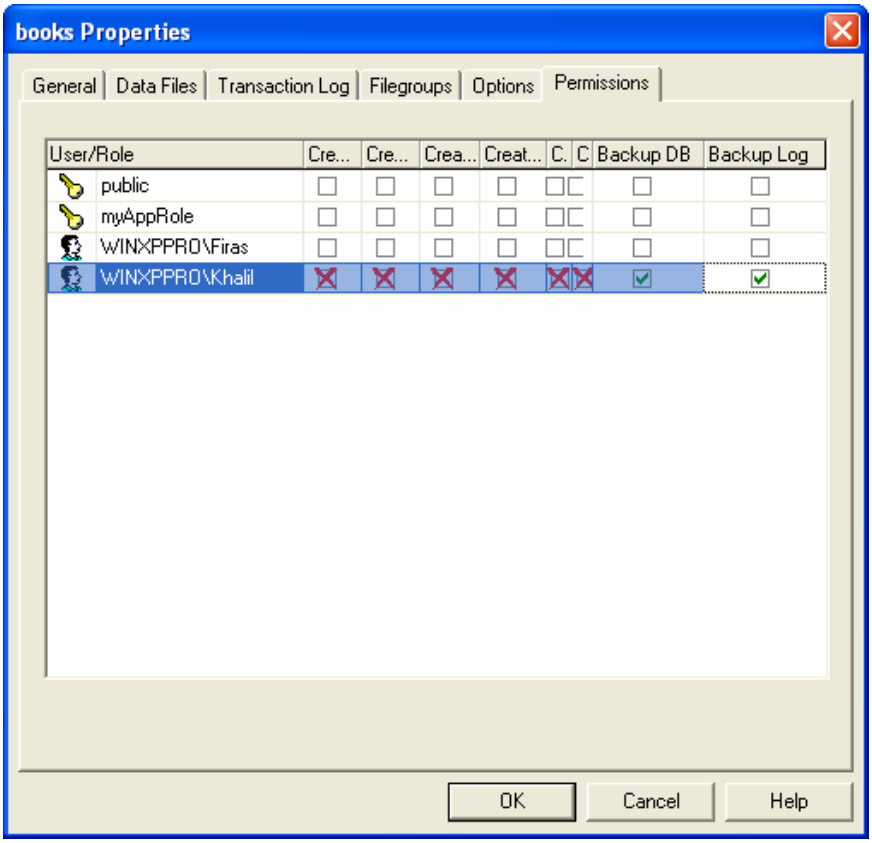
يمكننا استخدام الأداة Enterprise Manager لإدارة السماحيات الممنوحة لقاعدة معطيات معينة في الأداة SQL Server وذلك من خلال واجهة خصائص قاعدة المعطيات تلك، بحيث نجد فيها واجهة فرعية خاصة وهي Permissions:




books Properties
General
Data Files
Transaction Log
Filegroups
Options
Permissions
Name:
books
Database
Status:
Normal
Owner:
WINXPPRO\Firas
Date created:
07/03/2006 02:24:37
Size:
2.25 MB
Space available:
0.29 MB
Number of users:
3
Backup
Last database backup:
06/03/2006 09:12:47
Last transaction log backup:
None
Maintenance
Maintenance plan:
None
Collation name:
OK
Cancel
Help

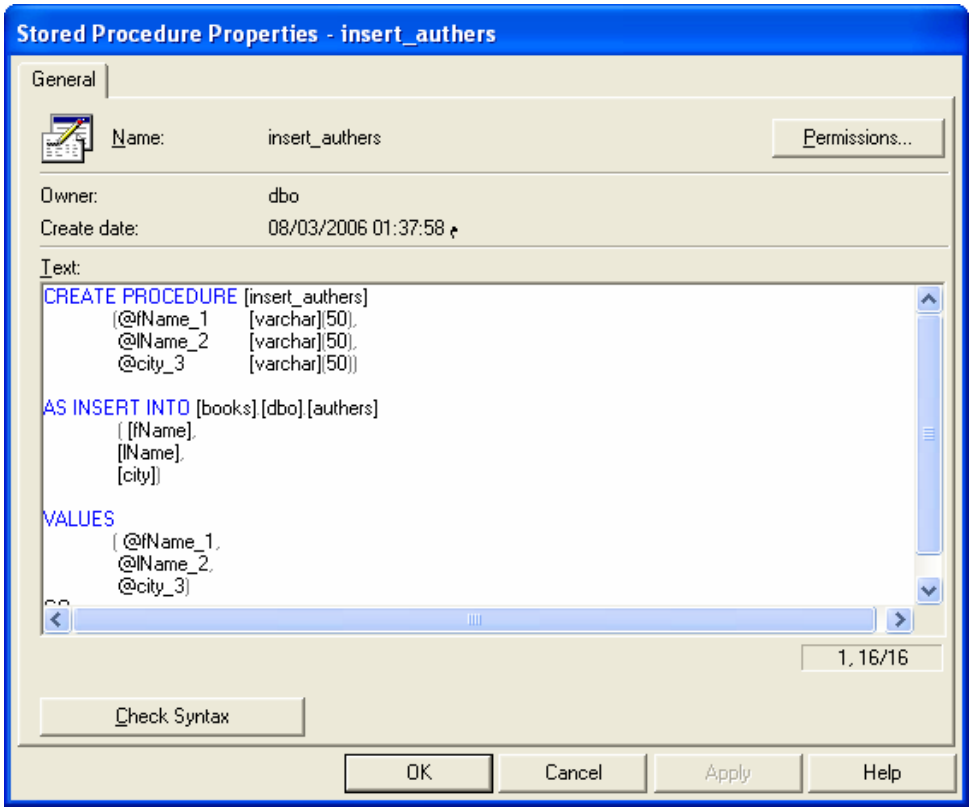


تعرض لنا الواجهة permissions قائمة بكافة الأدوار وحسابات المستخدمين المُنشأة على قاعدة المعطيات المحددة، بالإضافة إلى قائمة بمجموعة السماحيات التي يمكن التحكم بإسنادها أو عدم إسنادها إلى المستخدمين أو الأدوار بحيث يمكننا أن نمنح سـماحية التنفيذ أو أن نحرم المستخدم أن الدور منها؛

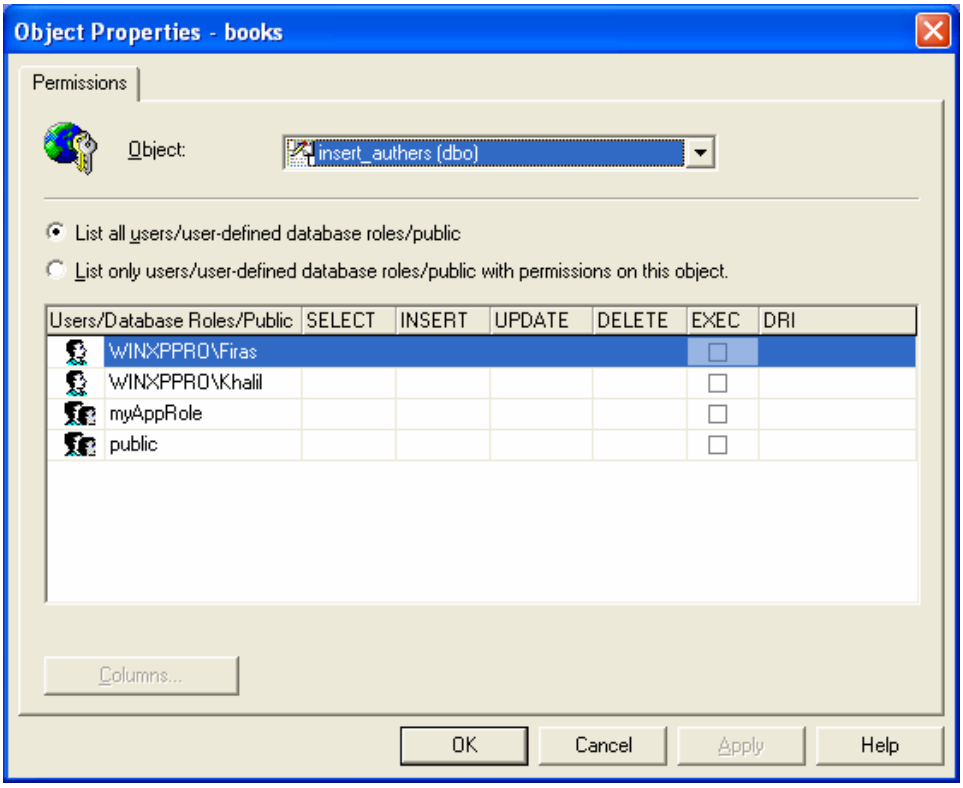


يعرض المثال، أن المستخدم Khalil غير مخوّل بإنشاء الجداول أو المناظير أو الإجرائيات المخزّنة أو... في حـــين أن مهماتـــه تنحصر بإجراء نسخة احتياطية من قاعدة المعطيات ومن ملف سجلّ المناقلات؛

- يمكننا كذلك أن نجد زر أو واجهة فرعية يُطلق عليها اسم Permissions في عدة أغراض في قاعدة المعطيات، فعلى سبيل المثال يمكننا اختيار Properties من قائمة المهمات السريعة لأحد الإجرائيات المخزنة:

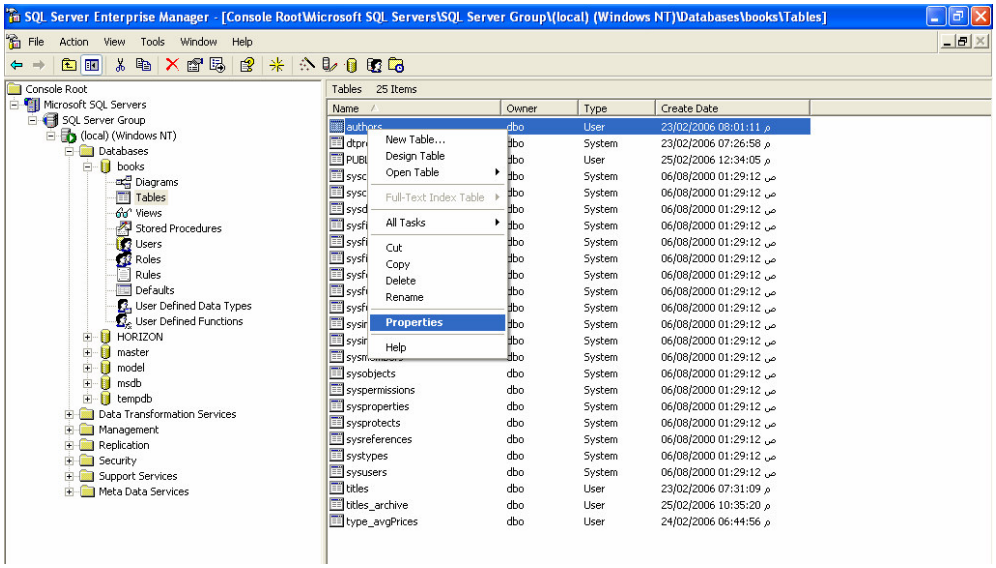


نستطيع من خلال الزر Permissions الذي يظهر في تلك الواجهة أن نستعرض (كما مرّ معنا) قائمة المستخدمين والأدوار التي ترتبط بهذا الغرض، إلا أننا نلاحظ أنّ السماحيات التي يتم عرضها في هذه الواجهة تتوافق مع الغرض المحدد؛ فهنا على سبيل المثال نجد أن السماحيات المتاحة هي تنفيذ أو عدم تنفيذ الإجرائية المخزّنة؛



مثال 2:

سنستعرض سماحيات جدول ما:



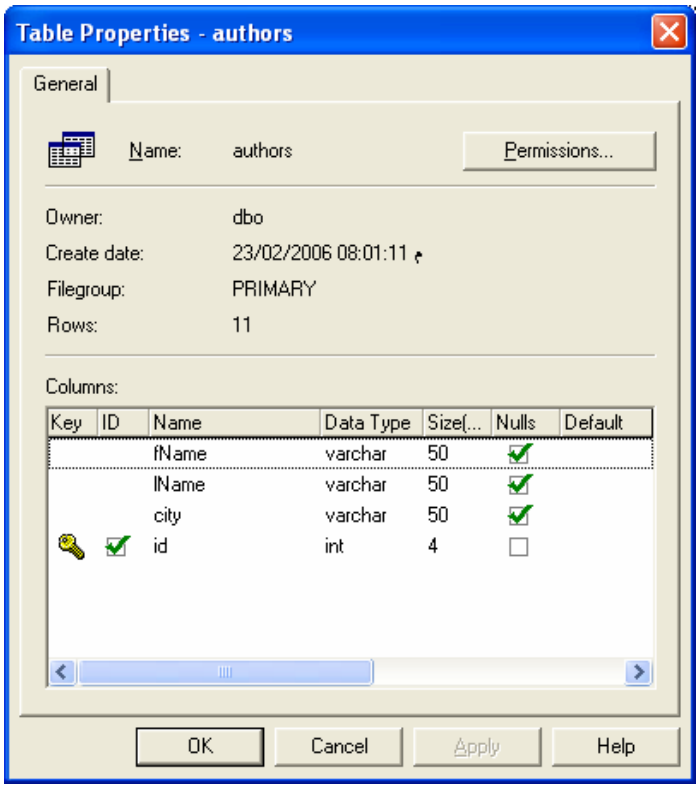
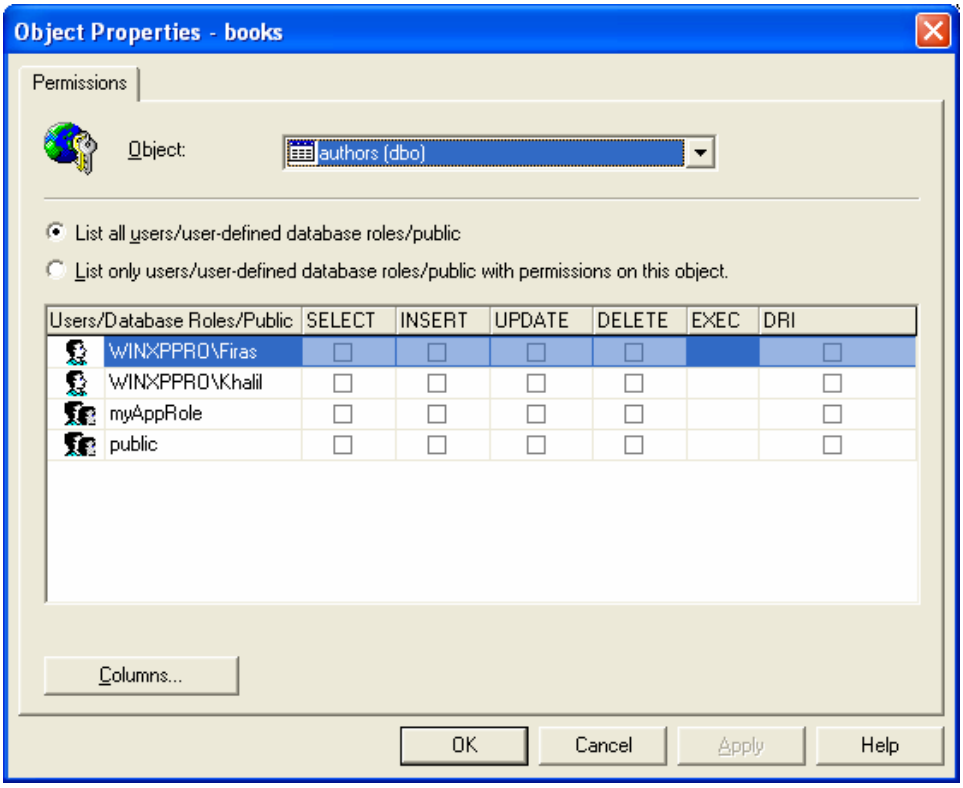


Table Properties - authors
General
Name: authors
Permissions...
Owner: dbo
Create date: 23/02/2006 08:01:11
Filegroup: PRIMARY
Rows: 11
Columns:
Key ID Name Data Type Size(... Nulls Default
fName varchar 50
lName varchar 50
city varchar 50
id int 4
OK
Cancel
Apply
Help


Object Properties - books
Permissions
Object:
authors (dbo)
List all users/user-defined database roles/public
List only users/user-defined database roles/public with permissions on this object.
Users/Database Roles/Public
SELECT
INSERT
UPDATE
DELETE
EXEC
DRI
WINXPPRO\Firas
WINXPPRO\Khalil
myAppRole
public
Columns...
OK
Cancel
Apply
Help

عنوان الموضوع:

أمثَلة الاستعلامات في SQL Server.

الكلمات المفتاحية:

انظر ملف Glossary المرفق.

ملخص:

سنناقش في هذه الجلسة كيف تُعالج الأداة SQL Server الاستعلامات وتنفذها بالطريقة الأفضل، وسنقوم باستعراض الحالات والمراحل التي تمر بها عملية أمثَلة الاستعلامات، بالإضافة إلى كيفية حساب تكلفة الاستعلامات وإنشاء خطط التنفيذ؛

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على ما يلي:

- عملية أمثَلة الاستعلامات
- المعايير التي يتم استخدامها في عملية أمثَلة الاستعلامات
- ترجمة الاستعلام وأمثَلته:
  - ترجمة عبارات DML
- خطوات عملية الأمثَلة:
  - عملية تحليل الاستعلام:
    - تحديد معاملات البحث
    - تحديد عبارات OR المنطقية
    - تحديد عبارات الربط
  - عملية اختيار الفهرس:
    - حساب تكلفة استخدام الفهارس العنقودية
    - حساب تكلفة استخدام الفهارس غير العنقودية
    - حساب تكلفة استخدام عملية مسح الجدول بالكامل
  - عملية اختيار الربط المناسب:
    - استراتيجيات معالجة عملية ربط الجداول:
- خوارزمية nested loop join
- خوارزمية merge join

- خوارزمية Hash join
  - عملية اتقاء خطة التنفيذ

أمثَلة الاستعلامات في SQL Server

مقدمة

- تعريف؛

عملية أمثَلة الاستعلامات هي الإجراء الذي تستخدمه الأداة SQL Server من أجل تحليل الاستعلامات الفردية وتحديد الطريقة الأمثل لمعالجتها وتنفيذها؛

- المعايير التي يتم استخدامها في عملية أمثَلة الاستعلامات؛

- إن فهم كيفية عمل أداة أمثَلة الاستعلامات وكيف يتم اختيار خطط التنفيذ، يمكن أن يكون له دور كبير في تحسين أسلوب كتابة الاستعلامات أو اختيار الفهارس أو التنبؤ المسبق بمشاكل الأداء.

- تعريف؛

عملية أمثَلة الاستعلامات هي الإجراء الذي تستخدمه الأداة SQL Server من أجل تحليل الاستعلامات الفردية وتحديد الطريقة الأمثل لمعالجتها وتنفيذها؛

- المعايير التي يتم استخدامها في عملية أمثَلة الاستعلامات:

يستخدم SQL Server منهجية تعتمد على التكلفة من أجل اختيار الطريقة الأمثل لتنفيذ الاستعلامات، إذ تقوم أداة أمثلة الاستعلامات برسم خطة تنفيذ يتم من خلالها الولوج إلى المعطيات بأسرع ما يمكن، مع العلم أن مفهوم التكلفة يتضمن التكلفة المنطقية والفيزيائية على حد سواء؛

تقوم أداة الأمثلة بفحص استعلامات SQL أولاً، ثم –وبالاعتماد على المعلومات التي تتضمنها أغراض تلك الاستعلامات، سواء كانت عدد صفحات الجدول أو نوع الفهارس المستخدمة أو الإحصائيات المتوافرة حول تلك الفهارس أو غيرها...– يتم إنشاء خطة تنفيذ ملائمة تتكون عادة من عدّة خطوات سنقوم بالتحدث عنها لاحقاً؛

- إن فهم كيفية عمل أداة أمثَلة الاستعلامات وكيف يتم اختيار خطط التنفيذ، يمكن أن يكون له دور كبير في تحسين أسلوب كتابة الاستعلامات أو اختيار الفهارس أو التنبؤ المسبق بمشاكل الأداء؛

- تعتبر عملية أمثَلة الاستعلامات في الإصدار SQL Server 7.0 والإصدار SQL Server 2000 معقدة نوعا ما بما تحتويه من نماذج حساب تكلفة أو خوارزميات ولوج معطيات، إلا أننا سنحاول من خلال الجلسة الحالية أن نسعى لتكوين صورة بسيطة عن أسلوب عمل تلك الأداة وكيفية اختيارها لخطة التنفيذ.

ترجمة الاستعلام وأمثَلَته

تعبِّر عملية ترجمة الاستعلام عن الإجراء الكامل اعتباراً من إطلاق الاستعلام وحتى نهاية تنفيذه، بحيث تمر تلك العمليـــة بعـــدة خطوات تعتبر عملية الأمثَلة إحداها؛

- ترجمة عبارات DML:

عندما يقوم SQL Server بترجمة خطة تنفيذ لعبارات DML فإنه يقوم فعلياً بتنفيذ الخطوات الأساسية التالية:

- اختبار صحة عبارات T-SQL قواعدياً و تشكيل ما يُعرف باسم شجرة القواعد
- يتم بعد ذلك تنظيم شجرة القواعد
- توليد مخطط الاستعلام
- توليد خطة التنفيذ
- تنفيذ الخطة السابقة وإعادة النتائج

- خطوات عملية الأمثَلة:

يتم –بعد إنشاء مخطط الاستعلام وتمريره إلى أداة الأمثلة– تنفيذ عدة خطوات متسلسلة لتجزئة ذلك الاستعلام إلى مجموعـــة مـــن المكوّنات بغرض تحليله وإنشاء خطة التنفيذ الأفضل؛

1- عملية تحليل الاستعلام
2- عملية اختيار الفهرس
3- عملية اختيار الربط المناسب
4- عملية اختيار خطة التنفيذ

تعبِّر عملية ترجمة الاستعلام عن الإجراء الكامل اعتباراً من إطلاق الاستعلام وحتى نهاية تنفيذه، بحيث تمر تلك العمليـــة بعـــدة خطوات تعتبر عملية الأمثَلة إحداها؛

تتم ترجمة كامل عبارة T-SQL في حين أنه لا يتم أمثَلة كامل تلك العبارة، فالتعليمات التي تتعرض للأمثَلة هي فقط تعليمات لغة معالجة المعطيات DML، أي عمليات Select و Insert و Update و Delete ، أما بقية العبارات التي تتضمن بنـــى إجرائيـــة كتعليمات IF أو WHILE أو المتحولات المحلية أو غيرها فيتم ترجمتها كإجراءات منطقية ولكنها لا تتطلب أمثَلة؛

- ترجمة عبارات DML:

عندما يقوم SQL Server بترجمة خطة تنفيذ لعبارات DML فإنه يقوم فعلياً بتنفيذ الخطوات الأساسية التالية:

- يتم أولاً اختبار صحة عبارات T-SQL قواعدياً، بحيث يتم مسحها واستخراج الكلمات المفتاحية والتعابير والعمليات والمعرّفات التي يتكون منها ذلك الاستعلام، وذلك بهدف تشكيل ما يُعرف باسم "شجرة القواعد" التي يستخدمها SQL Server للتعبير عن الاستعلام؛
- يتم بعد ذلك تنظيم شجرة القواعد، ويتم خلال تلك العملية تحديد الجداول والأعمدة والمعطيات السامية المضمَّنة في الشجرة تلك من أنماط معطيات أو قيم فارغة أو إحصائيات فهارس أو غيرها؛
- يختبر SQL Server فيما إذا كان ذلك الاستعلام مكوّن من عبارات DML، ثم يولّد ما يُعرف باسم "مخطط استعلام" اعتمادا على الاستعلام المنظّم السابق؛
- يتم بعد ذلك أمثَلة مخطط الاستعلام ويتم توليد "خطة تنفيذ" مناسبة؛
- أخيراً يقوم SQL Server بتنفيذ الخطة السابقة وإعادة النتائج.

- خطوات عملية الأمثَلة:

يتم –بعد إنشاء مخطط الاستعلام وتمريره إلى أداة الأمثَلة– تنفيذ عدة خطوات متسلسلة لتجزئة ذلك الاستعلام إلى مجموعة من المكوّنات بغرض تحليله وإنشاء خطة التنفيذ الأفضل؛

يمكن حصر الخطوات التي تقوم أداة الأمثَلة بتنفيذها من خلال النقاط الأربع التالية:

1- عملية تحليل الاستعلام:

يتم في هذه المرحلة تحليل الاستعلام واستخراج معاملات البحث وعبارات الربط الموجودة؛

يتم التصريح عن معاملات البحث من خلال العبارة WHERE عندما تتم مقارنة عمود ما بقيمة ثابتة؛

يتم التصريح عن عبارات الربط من خلال العبارة WHERE عندما تتم مقارنة عمود ما من جدول معين بعمود آخر؛

2- عملية اختيار الفهرس:

يتم اختيار الفهرس المناسب اعتماداً على معاملات البحث وعبارات الربط (إن وُجدت)، كما يتم إعطاء الفهارس قيماً معينة تعبر عن تكلفة استخدام كلاً منها، وذلك اعتماداً على القيم الإحصائية لتلك الفهارس؛

3- عملية اختيار الربط المناسب:

يتم في هذه المرحلة تحديد الترتيب الأنسب لربط الجداول مع بعضها البعض بغرض الولوج إلى معطياتها، ويتم ذلك من خلال خوارزميات ربط خاصة تستخدمها أداة الأمثَلة؛

4- عملية اختيار خطة التنفيذ:

يتم في الخطوة الأخيرة اختيار خطة التنفيذ الأقل كلفةً من بين الخطط التي قامت أداة الأمثَلة برسمها.

سنقوم في الشرائح التالية باستعراض كل من تلك الخطوات الأربع بالتفصيل.

الخطوة الأولى من عملية الأمثَلة

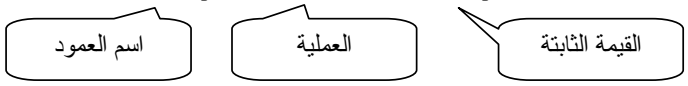
تحليل الاستعلام

تعتبر الخطوة الأولى في عملية أمثَلة الاستعلام هي مرحلة تحليل الجداول واستخراج معاملات البحث أو عبارات الربط الموجودة أو عبارات (OR) المنطقية التي سيتم استخدامها في الخطوة الثانية، أي عملية اختيار الفهرس المناسب لعملية الاستعلام؛

- تحديد معاملات البحث:

مرّ معنا أن التصريح عن معاملات البحث يتم من خلال العبارة WHERE عندما تتم مقارنة عمود ما بقيمة ثابتة؛
تسمح معاملات البحث لأداة الأمثَلة بتحديد الأسطر التي سيتم البحث عنها لتنفيذ الاستعلام؛
يعتبر الهدف الرئيسي من تحديد معاملات البحث هو من أجل مقارنتها مع الفهارس المتاحة لاختيار الفهرس الأنسب؛
فيما يلي عرض للقواعد التي تعبر عن معاملات البحث التي نتحدث عنها:

Column *operator* *constant_expression* [**and** …]



يمكن أن تكون العمليات المتاحة في معاملات البحث أحد العمليات التالية:
= (مساواة)، أكبر، أكبر أو يساوي، أصغر، أصغر أو يساوي، BETWEEN أو حتى LIKE (في بعض الأحيان)؛

مع العلم أنه يمكن التعبير عن أكثر من معامل بحث من خلال العملية AND المنطقية؛

أمثِلة:

```
Num = 5
Salary > 10000
City = 'Damascus'
Price BETWEEN 50 and 100
F_name LIKE 'sa%'
```

يمكن أن تكون القيمة الثابتة في معامل البحث عبارة رياضية أو متحول محلي أو تابع مضمَّن أو مجموعة محارف متّصلة أو استعلام جزئي؛

ملاحظة:
يتم اعتبار العبارة LIKE على أنها معامل بحث فقط عندما يكون المحرف الأول فيها ثابتاً، فمثلاً لا تعتبر العبارة التالية معامل بحث:

```
F_name LIKE '%sa'
```

وبشكل عام، لا تعتبر العبارات التي لا تحدُّ من أسطر البحث معاملات بحث، فمثلاً عملية عدم المساواة (<> أو =!) تخضع لنفس التعليق السابق، فالفهارس تساعد على إيجاد قيم معينة في حين أنه لا يمكن الاستفادة منها في الحالات التي تتطلب مقارنات على عدم التطابق مثلاً؛

يمكننا استبدال الاستعلام التالي الذي لا يحتوي على معامل بحث، باستعلام آخر يعيد نفس النتيجة ولكن مع استخدام معامل بحث يمكن من خلاله السماح لأداة أمثَلة الاستعلامات بتحسين الأداء وتسريع الاستجابة؛

مثال1:

الاستعلام الأول (بدون معامل البحث):

```
SELECT title FROM titles WHERE price != 0
```

الاستعلام الثاني (مع معامل البحث):

```
SELECT title FROM titles WHERE price > 0
```

وذلك على اعتبار أنه لا يمكن استخدام قيمة سالبه للتعبير عن السعر؛

على الرغم من أن SQL Server يقوم في كلتا الحالتين بإجراء مسح لكامل الجدول للحصول على النتيجة، إلاّ أن أداة الأمثَلة تستطيع من خلال معامل البحث المصرّح في الاستعلام الثاني من أن تحسن الأداء، على الأقل من خلال استخدام أحد الفهارس للوصول إلى النتيجة.

مثال2:

لا تعتبر العبارات المطبقة على أعمدة تحتوي على توابع، بأنها معاملات بحث:

الاستعلام الأول (بدون معامل البحث):

```
WHERE substring(name, 1, 1) = 'A'
```

الاستعلام الثاني (مع معامل البحث):

```
WHERE name LIKE 'A%'
```

مثال3:

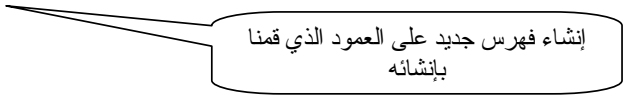
هناك حالات خاصة لا يمكننا فيها أن نتجنب وجود تابع ما على عمود في عبارة البحث، كما في المثال التالي:

```
SELECT count(*) FROM titles
WHERE datepart(month, pub_date) = 5
```

يمنعنا التابع datepart –في هذا المثال– من تطبيق أي فهرس على عملية البحث تلك، إلا أننا نستطيع تحسين الأداء فيما إذا تكرر استخدام هذا النوع من الاستعلامات كثيراً، من خلال إنشاء عمود جديد مفهرس إضافي كما يلي:

```
Alter table titles add pub_date_month as datepart(month, pub_date)

Create Index date_part_month_idx  on titles(pub_date_month)
```

إضافة عمود جديد يمثل الشهر



يمكننا الآن إعادة كتابة الاستعلام السابق كما يلي:

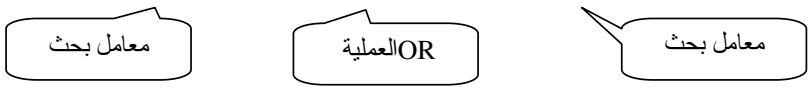
SELECT count(*) **FROM** *titles*
WHERE pub_date_month = 5

ستقوم هنا أداة الأمثلة باستخدام الفهرس المنشأ للوصول إلى النتيجة المطلوبة مما يؤدي إلى تحسين الأداء على الرغم من أن الاستعلام بحد ذاته يعتبر من الاستعلامات التي تقوم بمسح الجدول للحصول على النتيجة،

- تحديد عبارات OR المنطقية:

بعد الانتهاء من تحديد معاملات البحث في الاستعلام، تنتقل أداة الأمثلة إلى الخطوة التالية بحيث يتم فيها البحث عن عبارات OR المنطقية؛
يُطلق اسم عبارات OR المنطقية على الحالات التي تتوافر فيها عدة معاملات بحث ترتبط بعضها ببعض من خلال التعليمة OR؛

فيما يلي عرض للقواعد التي تعبر عن عبارات OR المنطقية التي نتحدث عنها:



يمكن أيضا التعبير عن عبارات OR المنطقية كما يلي:

Column = constant1 **or** *Column* = constant2 **or …**

بالإضافة إلى أن عبارات IN كذلك تخضع لنفس التعليق:

Column **IN** (constant1, constant2 **,…**)

أمثِلة:

Where F_name= 'سامر' or L_name = 'السيد'
Where (type = 'IT' and price < 10000) or id = 120
Where f_name IN ('سامر' , 'أيمن' , 'فادي' , 'نزار')

- تعيد العبارة OR كافة الأسطر التي توافق أحد المعيارين الذين تصل بينهما، مع العلم أنه ينبغي إعادة الأسطر التي تحقق المعيارين معاً مرة واحدة؛

- لا يمكن عادةً أن يتم تحقيق العبارة OR من خلال فهرس وحيد، فبالعودة إلى المثال السابق، نجد:

Where F_name= 'سامر' or L_name = 'السيد'

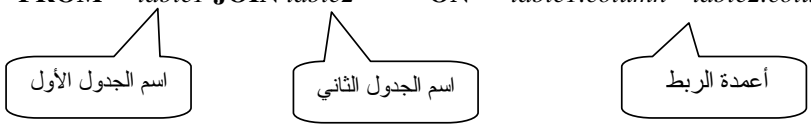
إن وجود فهرس على الاسم الأول ثم الاسم الأخير يساعدنا كثيراً عندما نرغب بالبحث عن الأسطر التي تحقق أن الاسم الأول هو "سامر" والاسم الأخير هو "السيد"، إلا أن ذلك الفهرس لن يفيدنا بشيء عندما نرغب بالبحث عن كافة الأشخاص الذين كنيتهم "السيد"، وبالتالي سنحتاج لإجراء عملية مسح أخرى للحصول على النتيجة المطلوبة؛

- تعتبر عملية مسح الجدول طريقة المكلفة لمعالجة الاستعلام، بالتالي تبحث أداة الأمثَلة عن حلّ بديل لمسألة عبارات OR المنطقية، والذي يمكن أن يكون من خلال استخدام فهارس على كل جزء من أجزاء عبارة الـ OR، أي معاملات البحث المتاحة، وهذا ما يُعرف باسم إستراتيجية الفهارس المتقاطعة؛

- تحديد عبارات الربط:

تعتبر الخطوة الأخيرة من خطوات عملية تحليل الاستعلام والتي تسعى أداة الأمثَلة إلى البحث عنها، هي عبارات الربط؛
يتم التعبير عن شرط الربط من خلال العبارة FROM وذلك باستخدام الكلمة المفتاحية Join كما يلي:

FROM *table1* **JOIN** *table2* **ON** *table1.column= table2.column*



يمكن أيضا التعبير عن شرط الربط بأسلوب آخر، من خلال العبارة Where كما يلي:

Table1.Column operator *table2.Column*

عادةً ما تتم عملية الربط بين جدولين مختلفين، إلا أنه يمكن إجراء عملية ربط لجدول مع نفسه، بحيث يعامل SQL Server ذلك الجدول على أنه جدولين منفصلين؛

الخطوة الثانية من عملية الأمثَلة

اختيار الفهرس

ما أن تنتهي مرحلة تحليل الاستعلام ويتم تحديد كافة معاملات البحث وعبارات OR المنطقية وعبارات الربط حتى تبدأ الخطوة التالية من عملية الأمثَلة والتي يتم فيها اختيار العبارات التي تدل على الأسطر المطلوبة في الاستعلام بالطريقة المثلى، وتحديد تكلفة الفهارس التي يمكن استخدامها لإيجاد تلك الأسطر؛

يتم قياس تكاليف الفهارس المتاحة من خلال عدد عمليات الدخل/خرج المنطقية المستخدمة في كل منها، ثم تتم مقارنة تلك النتائج مع بعضها البعض ومع تكلفة عملية المسح الكامل، لتحديد الطريقة الأسرع والأقل تكلفة للوصول إلى النتيجة المطلوبة؛

يتم احتساب عملية الدخل/خرج المنطقية في كل مرة يتم فيها الولوج إلى صفحة ما. إذا لم تكن تلك الصفحة في الذاكرة المخبئية، بالتالي لا بد أولاً من إجراء عملية دخل/خرج فيزيائية يتم من خلالها جَلب تلك الصفحة إلى الذاكرة المخبئية ومن ثم يتم إجراء عملية الدخل/خرج المنطقية من جديد؛

لا تستطيع أداة الأمثَلة أن تتنبّأ بوجود أو عدم وجود الصفحة في الذاكرة عندما يتم تنفيذ الاستعلام، كما أنها لا تأخذ تكلفة عملية الدخل/خرج الفيزيائية كمعامل من معاملات عملية الأمثَلة؛

تعتبر الفهارس مفيدة عادةً لتعبير معين، عندما يتم استخدام العمود الأول من أعمدة الفهرس في ذلك التعبير، وعندما يؤمن معامل البحث في ذلك التعبير –بطريقة أو بأخرى– أسلوباً لحدّ عملية البحث تلك؛

عندما لا تتوافر فهارس مفيدة لتعبير ما، تقوم أداة الأمثَلة بإجراء عملية مسح للجدول للحصول على النتيجة المطلوبة، بحيث تعتبر هذه الطريقة هي الخَيار الأخير الذي يمكن استخدامه؛

- حساب تكلفة استخدام الفهارس العنقودية:

تعتبر الفهارس العنقودية ملائمة لعمليات البحث، لأنه قد تم فيها تجميع الأسطر التي تطابق معاملات البحث ضمن نفس الصفحة أو ضمن صفحات متجاورة، بحيث يحتاج SQL Server لأن يجد الطريق إلى الصفحة الأولى ثم يقوم بقراءة كافة الأسطر التي توافق المعايير التي يتم البحث عنها في تلك الصفحة أو في سلسلة الصفحات التي تليها، ويتوقف فقط عندما يصل إلى أسطر لا توافق معايير البحث؛

يتم حساب التكلفة التقديرية لعمليات الدخل/خرج المنطقية المتوقعة باستخدام الفهارس العنقودية كما يلي:

عدد مستويات الفهرس في الفهرس العنقودي + عدد الصفحات التي سيتم مسحها ضمن مجال القيم؛

يتم تقدير عدد الصفحات التي سيتم مسحها على أساس عدد الأسطر التي يُتوقع أن توافق نتيجة البحث مقسومة على عدد الأسطر في الصفحة؛

مثال:

إذا كان SQL Server يخزن 250 سطر في كل صفحة وذلك من أجل جدول معين، وكان عدد الأسطر التي تحقق مجال القيم الذي يتم البحث عنه هو 600 سطر، بالتالي يكون عدد الصفحات التي سيتم مسحها ضمن مجال القيم مساوياً لـ 600 ÷ 250 = 3.

وبالتالي ستكون تكلفة عمليات الدخل/خرج المنطقية المتوقّعة –على اعتبار وجود ثلاثة مستويات للفهرس العنقودي– تساوي:

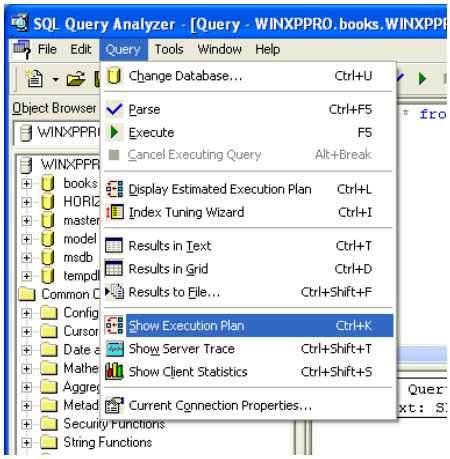
3 (عدد المستويات) + (600 ÷ 250) = 3 + 3 = 6 عمليات دخل/خرج منطقية؛

ملاحظة:

باستخدام فهرس عنقودي فريد وعملية مساواة كمعامل بحث، تُحسب تكلفة عمليات الدخل/خرج المنطقية كما يلي:

صفحة معطيات وحيدة + عدد مستويات الفهرس التي ينبغي تجاوزها للولوج إلى صفحة المعطيات تلك؛

نستطيع من خلال الأداة SQL Query Analyzer أن نستعرض إحصائيات ومعلومات عن خطة التنفيذ التي تم استخدامها للاستعلامات التي يتم تنفيذها، ويمكننا عرض خطة التنفيذ للاستعلامات من خلال القائمة "Query" واختيار Show Execution Plan أو من خلال الضغط على الاختصار Ctrl + K، بحيث سيتم بعد ذلك عرض واجهة توضح خطة التنفيذ لكل استعلام يتم كتابته؛



مثال عملي:

سنقوم فيما يلي بتنفيذ الاستعلام التالي:

```
select * from titles where id =4
```

يتم تنفيذ هذا الاستعلام على جدول ذو فهرس عنقودي، وكما نلاحظ، فإن الاستعلام يستخدم معامل بحث مع عملية مساواة؛

نستطيع أن نستعرض في واجهة خطة التنفيذ، المعلومات المتعلقة بأسلوب تنفيذ الاستعلام السابق،