يعتبر هذا الفصل امتداداً للفصل السابق الذي تناولنا فيه مفهوم SQL والقواعد التي تحكم استخدامها داخل Access2007 وكيفية ترجمة عبارات SQL داخل نافذة تصميم الاستعلام وفي هذا الفصل نشرح مفاهيم إضافية عن استخدام T-SQL مع الاستعلامات الإجرائية والجداول والنماذج والتقارير .

بانتهاء هذا الفصل ستتعرف على:

- كتابة الاستعلامات الإجرائية والإجراءات المخزنة
- العمل مع الجداول الموجودة بقاعدة بيانات أخرى
- العمل مع الجداول من خلال الكود
- استخدام عبارات SQL داخل النماذج والتقارير وأدوات التحكم

كتابة الاستعلامات الإجرائية والإجراءات المخزنة

يتم تنفيذ أوامر لغة معالجة البيانات أو Data Manipulation Language (DML) من خلال الاستعلامات الإجرائية Action Queries مثل استعلام الإضافة Append واستعلام التحديث Update واستعلام الحذف Delete واستعلام إنشاء جدول جديد Make-table. ويمكنك داخل Access إنشاء الاستعلامات الإجرائية بسهولة شديدة من خلال نافذة تصميم الاستعلام، أما داخل مشروعات SQL Server فيتم استخدام الإجراءات المخزنة Stored Procedures أو الدوال للحصول على الاستعلامات الإجرائية المختلفة التى ذكرناها في الفصل السابق. سنوضح فيما يلى الصيغة العامة لهذه الاستعلامات سواءً داخل SQL أو داخل T-SQL.

يحتوى استعلام الإضافة Append Query على الصيغة العامة التالية:

```
INSERT INTO dest_table
    SELECT [ALL|DISTINCT] select_list
    FROM source_table
  [WHERE append_criteria]
```

فإذا قمت بإهمال الجزء WHERE يتم إضافة جميع سجلات الجدول source_table إلى الجدول dest_table.

أما استعلام التحديث Update Query، فيحتوى على الصيغة العامة التالية:

```
UPDATE table_name
    SET column_name = value [,column_name = value[,...]]
   [WHERE update_criteria]
```

وفيه يتم استخدام SET فى تعيين القيم الجديدة للحقول مع فصل الحقول وقيمها باستخدام علامة الفاصلة فى حالة الرغبة فى تحديث أكثر من حقل من خلال العبارة. وإذا قمت بحذف الجزء WHERE، يتم تحديث جميع السجلات الموجودة بالجدول.

ويحتوى استعلام الحذف Delete Query على الصيغة العامة التالية:

```
DELETE FROM table_name
[WHERE delete_criteria]
```

فإذا قمت بتجاهل الجزء WHERE، سيتم حذف جميع سجلات الجدول بالكامل.

أما استعلام إنشاء جدول جديد **Make-table Query** فيحتوى على الصيغة العامة التالية:

```
SELECT [ALL|DISTINCT] select_list
    INTO new_table
    FROM source_table
    [WHERE append_criteria]
```

فإذا أردت نسخ الجدول الأصلى، استبدل **select_list** بالعلامة * مع عدم استبدال الجزء **WHERE**. كما يجب أن تتطابق أنواع البيانات وأحجام الحقول بالجدول **new_table** مع تلك الموجودة بالجدول **source_table**.

استخدام المعاملات فى المعايير وقيم التحديث

عند إنشاء الاستعلامات أو الإجراءات المخزنة فى تحديث بيانات الجداول (من خلال استعلام التحديث **UPDATE** أو استعلام الإضافة **INSERT**)، يتم استخدام معاملات فى إمداد القيم إلى جزء المعايير الموجود بالشرط **WHERE** وقيم الحقول، فإذا قمت بتعيين نوع بيانات لمعاملات الإدخال، يقوم **Access** بإضافة الإعلان **PARAMETERS** قبل عبارة **SQL** للإعلان عن هذه المعاملات، حيث يحدد ترتيب المعاملات داخل الإعلان ترتيب ظهور مربعات الإدخال المختلفة التى تطالبك بإدخال قيم هذه المعاملات عند تشغيل الاستعلام أو الإجراء المخزَّن.

يوضح كود **SQL** التالى استعلام تحديث يحتوى على معاملات لتعديل كمية أحد الأصناف داخل الجدول **Order Details**:

```
PARAMETERS [Type Order Number] Long,
           [Type Product Code] Long,
           [Type New Quantity] Short;
Update [Order Details]
    SET [Order Details].Quantity = [Enter New Quantity]
    WHERE [Order Details].Order_no = [Type Order Number]
AND
          [Order Details].Product_no = [Type Product Code];
```

وتستخدم الفاصلة الموجودة بعد الإعلان عن المعاملات للإشعار ببدء عبارة **SQL**.

أما T-SQL فيستخدم المتغيرات فى تمثيل المعاملات، حيث يتم تعريف المتغيرات داخل SQL Server باستخدام البادئة @. كما يدعم T-SQL معاملات الإدخال والإخراج وكذلك إرجاع القيم. يوضح الكود التالى كود T-SQL المكافئ لعبارة SQL السابقة:

```
CREATE PARAMETER tsql_edititems
      (@Order_no int,
       @Product_no,
       @Quantity smallint)
  AS UPDATE [Order Details]
  SET Quantity = @Quantity
WHERE Order_no = @Order_no AND
    Product_no = @Product_no
```

والأقواس الموجودة حول المعاملات اختيارية.

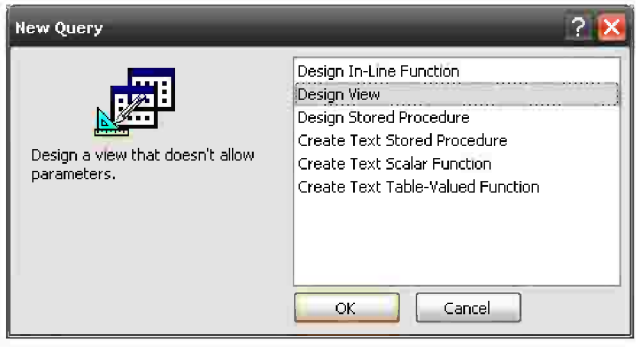
استخدام التعاملات داخل الإجراءات المخزَّنة

Using Transactions in stored procedures

يقوم Access تلقائياً باستخدام التعاملات Transactions عند تحديث البيانات فى أكثر من جدول باستخدام عبارة SQL واحدة. أما SQL Server فلا يقوم بأداء ذلك من تلقاء نفسه، وإنما يجب عليك تعريف بداية التعامل Transaction ونهايته. فحينما تقوم بكتابة عبارة T-SQL لتحديث أكثر من جدول، قم بوضع الكود بين عبارتى BEGIN TRAN وCOMMIT TRAN. يستخدم الإجراء المخزَّن التالى عبارة T-SQL للتأكد من عدم وجود سجلات داخل الجدول Order Details بعد حذف سجلها المصاحب بالجدول Orders:

```
CREATE PROCEDURE tsql_delorder
    @Order_no int
AS
BEGIN TRAN
     DELETE FROM Order_details
         WHERE Order_no = @Order_no
     DELETE FROM Orders
         WHERE Order_no = @Order_no
COMMIT TRAN
```

لإنشاء الإجراء المخزن السابق، اختر **Create Text Stored Procedure** (وليس **Design Stored Procedure**) من المربع الحوارى **New Query** (انظر شكل ١-١٨).



شكل ١-١٨ المربع الحوارى **New Query**.

يوضح كود **T-SQL** التالى إجراءاً مخزناً يقوم بإضافة طلب صرف جديد إلى الجدول **Orders** (على فرض عدم احتواء الجدول على حقل من النوع **identity**) ثم إضافة السطر الأول من طلب الصرف إلى الجدول **Order Details**:

```
CREATE PROCEDURE tsql_addorder
   @CustNo varchar (5),
   @OrdDate datetime,
   @ChrgDate datetime,
  @Freight real,
  @ShipName varchar(40),
   @ShipAddr varchar(60),
   @ProdNo int,
  @Price money,
  @Quan int,
  @Disc real

AS DECLARE @Order_no int

SET NOCOUNT ON
BEGIN TRAN
   SELECT @Order_no = MAX(Order_no) FROM Orders
```

```
  SELECT @Order_no = @Order_no + 1
  INSERT Orders
    VALUES(@Order_no, @CustNo, @OrdDate,
           @ChrgDate,@Freight, @ShipName,
           @ShipAddr)
  INSERT Order_details
     (Order_no, Product_no, Unit_price, Quentity, Discount)
      VALUES(@Order_no, @ProdNo , @Price, @Quan, @Disc)
COMMIT TRAN
SET NOCOUNT OFF
IF @@error = 0
  RETURN @Order_no
ELSE
  RETURN 0
```

وعن هذا الكود، نوضح ما يلى:

- يستخدم الجزء DECLARE @Order_no int فى إنشاء متغير داخلى باسم @Order_no يقوم بدوره بإرجاع رقم طلب الصرف الجديد إلى إجراء VBA الذى يقوم باستدعاء الإجراء المخزَّن وإمداده بقيم المعاملات المختلفة.
- تقوم عبارتا SELECT بالحصول على آخر قيمة بالحقل Order_no ثم زيادتها بمقدار ١ وتعيين قيمة طلب الصرف الجديد.
- يستخدم الجزء SET NOCOUNT ON حتى لا يتم الرجوع إلى الخادم مرةً أخرى لإخباره بعدد السجلات التى تأثرت بالإجراء مما يعمل على سرعة تنفيذ الإجراء.
- توضح العبارتان INSERT طريقتين مختلفتين لاستخدام الدالة VALUES، حيث لا تحتوى الأولى على قائمة حقول، لذا يجب أن تتطابق القيم المعرفة داخل قائمة VALUES مع ترتيب الحقول داخل الجدول. أما العبارة الثانية فتحتوى على قائمة حقول تحدد ترتيب القيم المعرفة داخل قائمة Values.
- تستخدم العبارة الشرطية IF @@error فى تخصيص القيمة الجديدة للمتغير @Order_no إلى القيمة RETURN فى حالة نجاح التعامل (أى @@error =

0). و@@error عبارة عن أحد متغيرات النظام داخل SQL Server والتى تقوم بإرجاع قيمة غير الصفر فى حالة حدوث أى خطأ أثناء التنفيذ وفى هذه الحالة يتم إرجاع القيمة 0 من العبارة الشرطية.

العمل مع الجداول الموجودة بقاعدة بيانات أخرى

يمكنك من خلال Access فتح قاعدة بيانات واحدة فقط فى الوقت الواحد، إلا إذا قمت بكتابة الكود الذى يقوم بفتح جدول داخل قاعدة بيانات أخرى. ويمكنك استخدام كلمة IN مع أحد الاستعلامات الإجرائية كاستعلام التحديث أو الحذف أو الإضافة أو إنشاء جدول، فى تحديث أو إنشاء جداول بقاعدة بيانات أخرى.

يوضح كود SQL التالى إنشاء نسخة من الجدول Customers بقاعدة البيانات Sales.accdb إلى قاعدة بيانات أخرى:

```
SELECT *
    INTO Customers
    IN 'd:\NewDB.accdb'
FROM Customers
```

ولكى يتم تنفيذ هذا الكود بشكل صحيح، يجب عدم وجود جدول باسم Customers داخل قاعدة البيانات NewDB.accdb كما يجب أن يكون مسار قاعدة البيانات صحيحاً.

أما داخل SQL Server، فيتم استخدام الأسماء المكونة من ثلاثة أجزاء فى تعريف الجداول الموجودة بقاعدة بيانات أخرى، حيث يحتوى الاسم على الصيغة *Database.Schema.Table* والتى غالباً ما يتم فيها استبدال Schema بمالك قاعدة البيانات وهو dbo داخل SQL Server. فعلى سبيل المثال، للعمل مع الجدول Customers بقاعدة البيانات SalesProjectCS يتم استخدام الصيغة SalesProjectCS.dbo.Customers.

يوضح الكود التالى إجراءً مخزناً يقوم بنسخ محتويات الجدول Customers بقاعدة البيانات الحالية (ولتكن SalesProjectCS) إلى قاعدة البيانات master وهى قاعدة

البيانات الافتراضية داخل SQL Server والتى يتم إنشاؤها تلقائياً أثناء عملية الإعداد، وذلك باستخدام التركيب SELECT ... INTO داخل T-SQL:

```
SELECT dbo.Customers.*
      INTO master.dbo.Customers
FROM dbo.Customers
```

وبمجرد إنشاء الجدول الجديد داخل قاعدة بيانات أخرى، يمكنك إنشاء أحد العروض أو إحدى الدوال أو الإجراءات المخزَّنة باستخدام الجداول الموجودة فى قاعدتى البيانات. فعلى سبيل المثال، لإنشاء عرض جديد مكون من الجدول Customers داخل قاعدة البيانات NewDB.accdb والجدول Orders بمشروع البيانات SalesProjectCS، تابع معنا الخطوات الآتية:

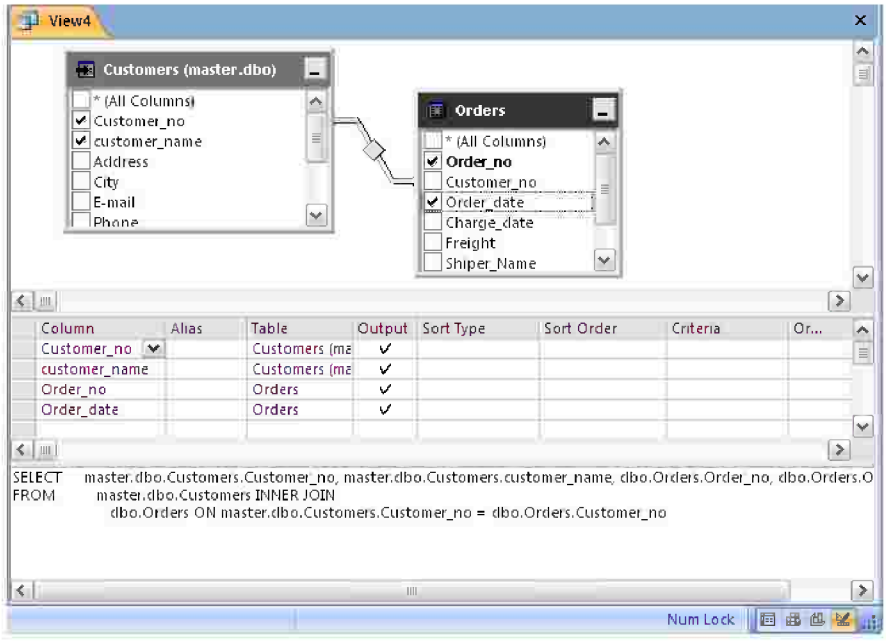
١. افتح مشروع البيانات SalesProjectCS ثم قم بإنشاء عرض جديد داخل مصمم المشروعات وقم بإضافة الجدول Customers والجدول Orders إلى نافذة المصمم ولاحظ وجود رمز المفتاح الأساسى والرمز بخط الصلة الموجود بين الجدولين.

٢. قم بتنشيط مربعات الاختيار المصاحبة للعمودين Customer_no وCustomer_name بالجدول Customers والعمودين Order_no وOrder_date بالجدول Orders.

٣. انقر الزر لفتح لوحة SQL ثم قم بإضافة البادئة master. إلى كل وجود للجدول dbo.Customers وهذا يعنى استخدام الجدول Customers الموجود بقاعدة البيانات master لا الموجود بقاعدة البيانات الحالية SalesProjectCS.

٤. انقر الزر للتأكد من صحة كود T-SQL بعد تعديله، وحينئذٍ تلاحظ إضافة مؤشر صغير إلى رأس قائمة حقول الجدول Customers وإضافة الجزء master.dbo إلى شريط عنوانه، كما يختفى الرمزان ، من خط الصلة

وذلك لأن الارتباط إنما يكون بين جداول قاعدة البيانات الواحدة فقط (انظر شكل ٢-١٨).



شكل ٢-١٨ إنشاء عرض من قاعدتى بيانات.

العمل مع الجداول من خلال الكود

يمكنك من خلال كود SQL إنشاء جداول جديدة وإضافتها إلى قاعدة البيانات الحالية أو حتى تعديلها باستخدام كلمات DDL المحجوزة والتى تتشابه داخل كل من SQL و-T SQL. وفيما يلى نوضح صيغ DDL المختلفة المستخدمة فى إنشاء وتعديل وحذف الجداول داخل قاعدة البيانات من خلال كود SQL.

إنشاء جدول جديد

يمكنك إنشاء جدول جديد بقاعدة البيانات الحالية من خلال العبارة CREATE TABLE التى تحتوى على الصيغة العامة التالية:

```
CREATE TABLE table_name (field_name data_type [(field-size)]
[,field_name data_tpe ...])
```

تقوم العبارة بإنشاء جدول جديد باسم table_name داخل قاعدة البيانات الحالية يحتوى على الحقول المحددة والمفصولة بعلامة الفاصلة. ويجب وضع أسماء الحقول التى تحتوى على مسافات بين القوسين [ ]، كما يمكنك استخدام أى من أنواع البيانات سواءً المعرفة داخل Access أو داخل SQL Server. وفى حالة تجاهل نوع البيانات المصاحب لأحد الحقول، يتم استخدام نوع البيانات Text فى حالة Access أو نوع البيانات char فى حالة SQL Server. كما يتم استخدام الحجم ٥٠ حرف فى حالة تجاهل حجم أحد الحقول سواءً داخل Access أو داخل SQL Server.

يمكنك تغيير نوع البيانات والحجم الافتراضى المستخدم داخل Access أو SQL Server من خلال التبويب Tables/Queries بالمربع الحوارى Options.

تعريف التكامل المرجعى للبيانات

يمكنك تعريف التكامل المرجعى للبيانات DRI باستخدام العبارة CONSTRAINT والتى تحتوى على الصيغة العامة التالية:

```
CONSTRAINT constraint_name {PRIMARY KEY|UNIQUE|REFERENCES foreign_table [(foreign_field)] }
```

فى تعريف التكامل المرجعى للبيانات DRI الخاصة بالجدول، حيث يقوم كل من Access وSQL Server بإنشاء فهرس Index لاسم الحقل الذى يسبق هذا التعبير مباشرةً، حيث يمكنك تعريف الفهرس كحقل مفتاح أساسى PRIMARY KEY أو كحقل فريد UNIQUE، كما يمكنك أيضاً إنشاء علاقة بين الحقل وحقل آخر موجود بجدول أجنبى باستخدام الجزء REFERENCES foreign_table [(foreign_field)]، حيث يجب استخدام الحقل foreign_field إذا لم يكن هذا الحقل مفتاحاً أساسياً.

إنشاء القيود

يمكنك إنشاء القيود داخل الجدول باستخدام العبارة CHECK التى تحتوى على الصيغة العامة التالية:

```
CHECK (expression)
```

تقوم هذه العبارة بإنشاء قيد إضافى مشابه إلى حدٍ كبير لقاعدة التحقق من الصحة المستخدمة داخل Access إلا أنه أكثر مرونة. ويمكنك استخدام المعامل expression فى مقارنة القيم الناتجة من الجداول الأخرى من خلال عبارة SELECT.

إنشاء الفهارس

يمكنك إنشاء الفهارس داخل الجداول باستخدام العبارة CREATE INDEX التى تحتوى على الصيغة العامة التالية:

```
CREATE [UNIQUE] INDEX index_name ON table_name
(field_name [ASC|DESC] [,field_name [ASC|DESC], ...]) [WITH
{PRIMARY|DISALLOW NULL|IGNORE NULL}]
```

تستخدم العبارة فى إنشاء فهرس فى حقل أو أكثر من حقول الجدول. فإذا قمت باستخدام المعرف WITH PRIMARY، يتم ضمنياً استخدام UNIQUE دون الحاجة إلى كتابتها. كما تستخدم DISALLOW NULL فى منع إضافة السجلات التى تحتوى على القيمة NULL (لا يوجد بها بيانات) إلى حقل الفهرس. أما IGNORE NULL فتستخدم فى حالة عدم الرغبة فى فهرسة السجلات التى تحتوى على القيم NULL بالحقل field_name.

التحكم فى حقول الجدول

يمكنك استخدام العبارة ALTER TABLE فى إضافة حقول إلى الجدول أو حذفها، حيث تحتوى العبارة على الصيغة العامة التالية:

```
ALTER TABLE
```

ADD COLUMN فى إضافة حقول جديدة إلى الجدول ويتم استخدام التعبير field_name فى إضافة حقول جديدة إلى الجدول، بينما يتم استخدام التعبير:

```
DROP COLUMN column_name ON table_name
```

فى حذف حقول موجودة بالفعل داخل الجدول، حيث يتم فى هذه العبارة حذف الحقل column_name من الجدول table_name.

تغيير خصائص حقول الجدول

يمكنك استخدام العبارة **ALTER COLUMN** فى تغيير خصائص حقول الجدول وتحتوى على الصيغة العامة التالية:

```
ALTER COLUMN table_name (field_name data_type[field_size])
```

حذف الفهارس من الجدول

يمكنك استخدام العبارة **DROP INDEX** فى حذف الفهارس من الجدول، وتحتوى على الصيغة العامة التالية:

```
DROP INDEX index_name ON table_name
```

وفيها يتم حذف الفهرس **index_name** من الجدول **table_name**.

حذف الجداول

يمكنك حذف أحد الجداول الموجودة بقاعدة البيانات الحالية من خلال التعبير **DROP TABLE** الذى يحتوى على الصيغة العامة التالية:

```
DROP TABLE table_name
```

وفيها يتم حذف الجدول **table_name** من قاعدة البيانات.

استخدام عبارات SQL داخل النماذج والتقارير وأدوات التحكم

فى حالة إنشاء الكثير من النماذج والتقارير المبنية على الاستعلامات أو العروض أو الإجراءات المخزنة، فسوف تحتوى قائمة الاستعلامات الموجودة بقاعدة بياناتك على كمية هائلة من الاستعلامات وخاصةً مع إضافة الاستعلامات اللازمة لتعبئة القوائم ومربعات السرد والتحرير. والأفضل من ذلك هو كتابة كود **SQL** المصاحب للاستعلام أو نسخه من لوحة **SQL** داخل الاستعلام ثم استخدامه كمصدر لبيانات النموذج أو التقرير أو غيره، ومن ثم يمكنك الاستغناء عن الاستعلام المصاحب وحذفه نهائياً من قاعدة البيانات.

وعامةً يمكنك استخدام عبارات **Access SQL** أو **T-SQL** فى الحالات التالية:

- بالخاصية **Record Source** بمربع خصائص النماذج والتقارير، وذلك بكتابة عبارة **SQL** بدلاً من اسم الاستعلام مصدر البيانات.

- بالخاصية **Row Source** بمربع خصائص القوائم والقوائم المنسدلة ومربعات السرد والتحرير الموجودة بأحد النماذج. ومن خلال عبارة **SQL** يمكنك التحكم فى ترتيب الأعمدة داخل أداة التحكم.
- بالخاصية **SQL** المصاحبة للاستعلام أو بدلاً من المعامل **strSource** بالوظيفة **OpenRecordset** داخل كود **VBA**.
- بالخاصية **Source** بالكائن **DAO.Recordset** المحدد داخل الخاصية **Recordset** المصاحبة للنموذج أو التقرير أو أداة التحكم.
- بالخاصية **Source** بالكائن **ADODB.Recordset** المحدد داخل الخاصية **Recordset** المصاحبة للنموذج أو التقرير أو أداة التحكم.