الباب الرابع
قواعد البيانات

١٠. أساسيات قواعد البيانات.

١١. استخدام SQL.

١٢. استخدام كائنات ADO.

١٣. استخدام كائنات ADO.NET.

١٤. استخدام مجموعات البيانات DataSets.

١٥. العمل مع الطباعة.

الفصل العاشر
أساسيات قواعد البيانات

تعمل معظم التطبيقات التجارية من خلال مجموعة من البيانات المرتبة بصورة أو بأخرى. سنقوم فى هذا الفصل بإلقاء نظرة عامة على مفهوم قواعد البيانات وعناصرها المختلفة.

بانتهاء هذا الفصل ستتعرف على:

- *فكرة عامة عن قواعد البيانات.*
- *نظام قواعد البيانات العلائقية.*
- *الأنواع المختلفة للبيانات.*
- *تصميم نظام لإدارة قاعدة بيانات.*
- *تصميم جداول البيانات وإنشاء العلاقات بينها.*
- *الطرق المختلفة لمعالجة البيانات.*

أثناء تصميم قواعد البيانات يجب أن نهتـــم بعمل تصميم قوى يساعد على الوصول إلى المعلومة فى أقل وقتٍ ممكن، ويساعد على الاحتفاظ بحجم البيانات نفسها داخل قواعد البيانات بصورة منطقية توفر السهولة اللازمة لأعمال التعديل والتطوير المستقبلية، كما يوفر الكفاءة والمرونة اللازمة للتعامل مع البيانات.

مقدمة عن قواعد البيانات

قاعدة البيانات هى طريقة لتجميع البيانات والمعلومات بصورة منظمة وترتيب معين ومن الأمثلة المعروفة لقواعد البيانات دليل التليفونات الذى يحتوى على أسماء المواطنين وعناوينهم وأرقام تليفوناتهم. ويختلف ترتيب البيانات داخل قاعدة البيانات حسب حاجتك، فقد يكون حسب الترتيب الأبجدي لأسماء المشتركين أو حسب عناوينهم.

يمكن تغيير ترتيب البيانات داخل قاعدة البيانات كما ستعرف في هذا الفصل.

نظام قاعدة البيانات العلائقي

يطلق على نظام إدارة قواعد البيانات **Database Management System** وتختصر هكذا **DBMS** ، وعلى نظام قواعد البيانات العلائقي **Relational Database Management System** وتختصر هكذا: **RDBMS**.وهو نظام لإدارة قواعد البيانات يستخدم جدول أو أكثر من جدول بينها علاقة. وقد ذكرنا هنا كلمة علائقي للتفرقة بينه وبين نظم أخرى لإدارة قواعد البيانات منها مثلاً النظام الهرمي المستخدم فى حفظ المعلومات بأسلوب يوفر سهولة التعامل مع البيانات واسترجاعها بأكثر من طريقة، حيث تحتوى قاعدة البيانات على العناصر التالية:

جداول **Tables**، سجلات **Records** ، حقول **Fields** ، استعلامات **Queries** ، فهارس **Indexes** ، طرق عرض **Views**. يوضح جدول ١٠-١ التالى هذه العناصر ومفهوم كل منها.

جدول ١٠-١ عناصر قاعدة البيانات

العنصر	الوصف
قاعدة بيانات Database	هى مجموعة من جداول البيانات تحتوى على معلومات لها علاقة ببعضها ويمكن أن تحتوي قاعدة البيانات على جدول بيانات واحد فقط.
جدول Table	هى مجموعة من السجلات (كل سجل يعتبر سطر داخل الجدول) تحتوى هذه السجلات على معلومات من نفس النوع. ويمكن اعتبار دليل التليفونات جدول بيانات.
Record سجل	السجل عبارة عن سطر واحد فى الجدول ويحتوى على مجموعة من الحقول. ويمكن اعتبار أى سطر من أى صفحة من دليل التليفونات سجلاً.
Field حقل	هو عنصر محدد داخل السجل قد يكون (فى دليل الهاتف مثلا) اسم المواطن أو عنوانه أو رقم تليفونه ويتم تحديده من حيث النوع والحجم بواسطة مصمم البرنامج.
فهرس Index	هو نوع خاص من الجداول يسمى جدول الفهرسة ويحتوى على قيم لحقل أو أكثر، وتشير هذه القيم للأماكن الحقيقية لهذه الحقول وترتب هذه الأرقام طبقاً للترتيب الذى يحدده مصمم البرنامج (تنازلى أو تصاعدى) وفى مثالنا السابق يمكننا عمل جدول لفهرسة دليل التليفونات طبقاً لترتيب اسم المواطن الأبجدى وجدول آخر للفهرسة بترتيب أبجدى لعناوين المشتركين. كما يمكن عمل جدول فهرسة ثالث يقوم بترتيب البيانات والمعلومات داخل الجدول طبقاً لرقم التليفون. يستخدم الفهرس مفتاح أساسي (Key Field) يتم تحديده بواسطة مصمم البرنامج.

العنصر	الوصف
استعلام Query	يتم تصميم أوامر الاستعلام للحصول على معلومات مطلوبة من مجموعة من الجداول وإخراجها فى صورة مجموعة من السجلات وقد يتم إجراء بعض العمليات الحسابية طبقاً لشكل الجدول الجديد المطلوب. مثال على ذلك إذا أردنا الاستعلام عن أرقام تليفونات ساكنى حى مصر الجديدة وأسمائهم وعناوينهم، يتم فى هذه الحالة البحث فى جدول البيانات الذى يحتوى على البيانات الخاصة بدليل التليفونات واستخلاص البيانات المطلوبة منها عن طريق أمر استعلام يتم تصميمه.
تصفية Filter	هو شرط (ليس جزء من قاعدة البيانات) يوضع ليظهر البيانات المطلوبة أثناء تنفيذ البرامج.
عرض View	هى طريقة لعرض المعلومات (مجموعة من السجلات) بصورة يتم تحديدها بواسطة المبرمج في صورة نتيجة من أحد أمرين، إما أمر فهرسة Index أو أمر تصفية Filter فمثلاً يمكن عرض جميع المواطنين الذين تبدأ أسمائهم بحرف الألف، وتعتبر التقارير من أشهر الوسائل لعرض البيانات.

تصميم قاعدة البيانات

للوصول إلى تصميم جيد لقاعدة بيانات سليمة يجب اتباع الآتى:

١. تصميم خريطة النظام.
٢. تحديد أنواع البيانات المطلوبة للنظام.
٣. تنظيم البيانات داخل الجداول.
٤. إنشاء العلاقات بين الجداول.
٥. تصميم الاستعلامات اللازمة عن النظام.

٦. معالجة البيانات.

٧. مراجعة التصميم.

التعرف على وظائف النظام

يحتوى هيكل النظام على جميع الوظائف الموجودة بالنظام، وأسهل طريقة لتحديد أو توصيف وظائف النظام تكون بتحديد المطلوب من هذا النظام. أو بعبارة أخرى تحديد المخرجات المطلوب الحصول عليها من النظام، ومن معرفة المخرجات يمكننا تجميع البيانات واستخدام المعالجات المطلوبة لهذه البيانات للحصول على هذه المخرجات. وفى النهاية تكون قد حصلت على النظام الذي يحقق الوظائف المطلوبة.

سنوضح فيما يلي كيفية تصميم نظام لإدارة قواعد البيانات من خلال مثال عملي عبارة عن برنامج لمنشأة تجارية يقوم بالوظائف التالية:

- تقرير ببيانات عملاء المنشأة.
- تقرير ببيانات مندوبى المبيعات.
- تقرير ببيانات الموردين.
- تقرير بالأصناف الواردة .
- تقرير بجميع الطلبات الخاصة بعميل معين.
- تقرير بجميع الطلبات الخاصة بمندوب معين.
- تقرير بجميع الطلبات يومية / شهرية / سنوية.

ومن هذا السرد لوظائف النظام تلاحظ أن النظام يشتمل على خمسة برامج على النحو التالي:

- *في البرنامج الأول:* يقوم المستخدم بتسجيل بيانات العملاء بجدول "العملاء" ومن هذا الجدول يتم استخراج تقارير تحتوي على جميع بيانات عملاء المنشأة.
- *وفي البرنامج الثاني:* يقوم المستخدم بتسجيل بيانات المندوبين بجدول "المندوبين"ومن هذا الجدول يتم استخراج تقارير المندوبين.

- *وفي البرنامج الثالث:* يقوم المستخدم بتسجيل بيانات الموردين بجدول "الموردين" ومن هذا الجدول يتم استخراج تقارير الموردين
- *وفي البرنامج الرابع:* يقوم المستخدم بتسجيل بيانات الأصناف بجدول "الأصناف" بالإضافة إلى أسماء الموردين ومن هذا الجدول يتم استخراج تقارير الأصناف الواردة.
- *وفي البرنامج الخامس:* يتم تجميع بيانات العملاء والموردين والمندوبين والأصناف كل من الجدول الخاص به، ومن هذا البرنامج يتم الحصول على:
  - فاتورة بيع.
  - طلبات خاصة بعميل معين.
  - طلبات خاصة بمندوب معين.
  - الطلبات اليومية/الشهرية/السنوية.

تحديد أنواع البيانات المطلوبة

فى البداية لابد من أن تحدد ما هى البيانات التى يجب أن تحتفظ بها داخل قاعدة البيانات وما هى البيانات التى يمكن أن تتغاضى عنها حالياً.

باستخدام توصيف وظائف النظام يمكن للمبرمج أن يحدد البيانات التى تحتاجها كل وظيفة. فإذا نظرنا إلى وظيفة السائق الذى يقوم بتوصيل الطلبات إلى المنازل أو إلى الأماكن المخصصة لها فإننا سنجد أنه لابد من أن يكون معه يومياً تقرير بجميع الأماكن التى سيذهب إليها وكذلك تقرير آخر بالطلبات المطلوب إرسالها لكل عميل مرتبة بعنوان هذا العميل (كل منطقه على حده) ويجب أيضاً أن نكون على دراية بمكان تواجد هذا السائق فى أى وقت حتى يتم ترتيب عناوين العملاء طبقاً لقربهم من مكان السائق ويجب كذلك توزيع السائقين توزيعاً عادلاً على أماكن مختلفة مع مراعاة التوزيع الجغرافى لهذه الأماكن.

بنفس الطريقة يجب أن يتم تحديد البيانات التى يشتمل عليها النظام، بعبارة أخرى لابد من معرفة شكل البيانات المتداولة ونوعيتها. مثلاً هل يصلح أن يكون البيان نصى أو رقمى أو

تاريخ أو منطقى (نعم / لا).

يوضح الجدول التالي أنواع البيانات التى يمكن أن تتعامل معها داخل قاعدة البيانات

النوع	الوصف
Text نصى	يحتوى على عدد متغير من الحروف يصل طوله إلى ٢٥٥ حرفا ويستخدم مع البيانات التي لا تحتاج لإجراء عمليات حسابية عليها مثل الأسماء والعناوين والأكواد التى تشتمل على حروف وأرقام
Memo مذكرة	يحتوى على عدد كبير من الحروف (غير محدد بـ ٢٥٥ حرف) ومن الأمثلة على استخداماته استخدامه لحفظ مواصفات منتج
Integer عددى	يحتوى على أرقام صحيحة (بدون كسور عشرية) ومن أمثلة استخداماته حفظ كمية الأصناف من منتج معين أو عدد السكان فى حى معين
Decimal عشرى	يشتمل على أرقام تحتوى على كسور عشرية مثل تكلفة منتج ، ثمن المنتج ، ضريبة المنتج ، ....الخ
Date تاريخ	يحتوى على بيانات تاريخية ويظهر التاريخ مشتملاً على اليوم والشهر والسنة ويستخدم الأشكال الموجودة فى بيئة Windows ومن خلال أمر Format أيضاً يمكن إظهاره بأشكال أخرى
Boolean منطقى	يحدد بقيمتين فقط True / False أو Yes / No بمعنى صحيح / خطأ ومثال على ذلك الحالة الاجتماعية لشخص معين هل هو متزوج؟ فيكتب أمام هذا السؤال Yes بمعنى نعم ليدل على أنه متزوج أو يكتب No بمعنى أنه غير متزوج
Binary ثنائي	يمكن أن يشتمل على صور أو مقطوعات موسيقية أو كائنات OLE ومثال على ذلك يمكن الاحتفاظ بصور الأصناف التى تعرض للبيع أو غير ذلك

تحديد الجداول المطلوبة

بعد تحديد البيانات التى يحتاجها النظام يلزم تنظيم هذه البيانات داخل الجداول يتم حفظ البيانات داخل جدول أو أكثر ويتم عمل العلاقات **Relationships** اللازمة بين هذه الجداول.

يحتوى الجدول دائماً على مجموعة من البيانات بينها علاقة، فمثلاً جدول العملاء **Customers** يحتوى على بيانات منها اسم العميل – عنوان العميل – رقم تليفون العميل – ..... الخ وهى بيانات كثيرة ولكنها تختص بالعملاء فقط. أما إذا تناولنا جدول الطلبات فإننا نحتاج فيه إلى بيانات عن الطلبات مثل عدد الأصناف – كمية الأصناف – أسماء الأصناف – السعر – رقم الطلب – تاريخ الطلب وبيانات أخرى عن العملاء مثل اسم العميل وعنوانه ورقم تليفونه.

لاحظ معى حجم هذه البيانات وهل يصح جمعها فى جدول واحد أم لا ؟ فإذا تم جمع هذه البيانات كلها فى جدول واحد فإننا سنقع فى مشكلتين. المشكلة الأولى هى الزيادة الهائلة فى حجم البيانات نتيجة تكرارها. انظر شكل ١٠–١، تلاحظ أن كل بيانات العميل مثل الاسم والعنوان والتليفون تتكرر مع كل طلبية.

اسم العميل	العنوان	رقم الهاتف	تاريخ الطلب	قيمة الطلب
وليد عبدالرازق	عين شمس	106521042	12/12/2002	500
وليد عبدالرازق	عين شمس	106521042	01/11/2002	600
وليد عبدالرازق	عين شمس	106521042	12/01/2003	900
نجلة محمود	الزيتون	2405330	05/01/2002	300
نجلة محمود	الزيتون	2405330	12/10/2002	400
محمد نجيب	حلوان	6391295	15/11/2002	300

شكل ١٠–١ جدول بيانات يحتوى على بيانات مكررة

والمشكلة الثانية هي تكرار العمل ومثال ذلك إذا تغير رقم تليفون أحد العملاء، فإنه يلزم تعديل الرقم فى جميع السجلات التى تحتوى على هذا الرقم وقد يتم فى هذه الحالة تعديل معظم سجلات الجدول.

ولعلاج المشكلتين السابقتين ننصح بتقسيم هذا الجدول إلى جدولين: جدول لبيانات العملاء والآخر لأوامر الشراء كما فى شكل ١٠-٢ التالى.

رقم العميل	اسم العميل	العنوان	رقم الهاتف
1	وليد عبدالرازق	عين شمس	106521042
2	نجلاء محمود	الزيتون	2405330
3	محمد نجيب	حلوان	6391295

رقم الطلب	تاريخ الطلب	قيمة الطلب
1	12/12/2002	500
1	01/11/2002	600
1	12/01/2003	900
2	05/01/2002	300
2	12/10/2002	400
3	15/11/2002	300

شكل ١٠-٢ تقسيم الجدول إلى جدولين واحد للعملاء والآخر لطلبات الشراء

ويتم ربط الجدولين بناءً على بيانات حقل مشترك بينهما. مثلاً حقل رقم العميل، في هذا المثال يجب إضافة حقل جديد داخل جدول العملاء يحتوى على رقم العميل وبذلك يمكننا التعامل مع أى بيان يخص العميل عن طريق معرفة رقمه، ويتم أيضا إضافة حقل جديد داخل جدول الطلبات يحتوى على رقم العميل. وبذلك نكون قد وصلنا إلى نفس النتيجة المطلوبة من وضع البيانات داخل جدول واحد وبالتالي فإن تغيير رقم تليفون العميل يتم فى سجل واحد فقط في جدول العملاء.

حساب حجم البيانات

من الأمور الهامة والتى يجب أن تأخذ فى الاعتبار عند تصميم نظام لقاعدة البيانات حساب حجم البيانات والزيادة المتوقعة لها، فيتم حساب المساحة المتاحة للتخزين على وسيط التخزين (قرص صلب مثلاً) وليكن ١٠٠ ميجابايت، ويتم حساب حجم الزيادة الشهرية فى البيانات فإذا كانت نسبة الزيادة كبيرة بالنسبة لحجم المساحة المتاحة على القرص الصلب فيلزم إعادة تصميم قاعدة البيانات بأسلوب أفضل يوفر فى المساحة المطلوبة ولا يكون ذلك على حساب كفاءة النظام من حيث السرعة أو دقة البحث عن معلومة مطلوبة

تصميم جداول تابعة

من الحلول التى يمكن أن نستخدمها لتجنب البيانات المتكررة تصميم جداول متفرعة أو تابعة لجداول أخرى. فمثلاً فى جدول العملاء توجد عائلات كاملة بأفرادها تحمل نفس البيانات واللقب والعنوان ولا يوجد اختلاف إلا فى اسم العميل فقط، في مثل هذه الحالة يمكن عمل جدول بيانات للعائلات (جدول جديد) يحتوى على بيانات كل عائلة فى سجل واحد (بدلاً من استخدام سجل لكل فرد فى العائلة) ونضيف فيه حقل نسميه رقم العائلة حتى يسهل الوصول إلى بيانات العائلة بمجرد معرفة هذا الرقم ويتم استخدام نفس الرقم مع جدول العملاء الذى سوف تقتصر المعلومات بداخله على اسم العميل ورقم العائلة وتاريخ الميلاد فقط.

يطلق على جدول العائلة الجدول الأبوى (Parent Table) ويحتوى على معظم البيانات الأساسية للعائلة بينما يطلق على جدول العملاء الجدول الابن (Child Table) حيث أنه جدول منبثق من جدول العائلة.

فى أحوال أخرى نجد أن بيانات العملاء قد تشترك فى أسماء المدن والأحياء ويصبح من الضرورى أن نقوم بعمل جدول جديد (جدول الأحياء) ويحتوى على رقم الحى واسم الحى واسم المدينة وهذا قد يؤدى إلى اختصار شديد فى البيانات الموجودة داخل قاعدة البيانات.

إنشاء العلاقات بين الجداول

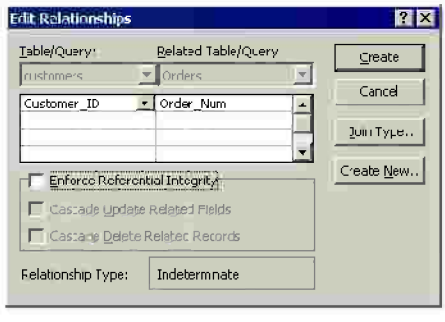
عند توزيع البيانات فى جدولين مثلاً فإنه يلزم وجود علاقة بينهما بحيث يمكنك الوصول إلى جزء من المعلومة واستكمالها من الآخر عن طريق هذه العلاقة . لإيجاد أو إنشاء علاقة بين جدولين، نستخدم مفتاحين هما المفتاح الأساسى Primary Key والمفتاح الأجنبى Foreign Key.

المفتاح الأساسى ***Primary Key***

هو الحقل الذى يحتوى على بيانات لا تتكرر داخل الجدول ومثال على ذلك يحتوى كل سجل من جدول العملاء على بيانات كاملة عن عميل معين، فمن المناسب أن نستخدم حقل "رقم العميل" لأنه لن يتكرر داخل الجدول ويكون هو المفتاح الأساسى أو **Primary Key** لهذا الجدول.

المفتاح الأجنبى ***Foreign Key***

هو الحقل الموجود بالجدول المرتبط وعادةً لا يكون الجدول المرتبط في حاجة إليه، ولكن وجوده ضرورى لربط هذا الجدول مع الجدول الأساسي. يشتمل شكل ١٠-٣ على مربع "علاقات" وفيه يظهر أن العلاقة بين جدول العملاء **Customers** وجدول الطلبات **Orders** تم تأسيسها بناءً على بيانات حقل رقم العميل **Customer_ID**، ويظهر بالشكل موقع المفتاح الأساسى وموقع المفتاح الأجنبي.



شكل ١٠-٣ إنشاء علاقة بين جدولين

ويعتبر رقم العميل **Customer_ID** داخل جدول العملاء **Customers** مفتاحاً أساسياً بينما يعتبر رقم العميل فى جدول الطلبات **Orders** مفتاحاً أجنبياً فهو لا يتعلق بموضوع الطلبات ولكن جدول الطلبات فى حاجه إليه لكى يتم ربطه مع جدول العملاء وبذلك يمكنك أن تفهم أن هذه الطلبات سوف يتم توصيلها إلى العميل رقم كذا وبياناته تقع فى جدول العملاء ومن ذلك يمكن تعريف المفتاح الأجنبى على أنه حقل مستخدم فى أحد

سجلات جدول (فى المثال السابق هو جدول الطلبات) (وهو قابل للتكرار) ليشير إلى سجل واحد فقط موجود فى جدول آخر (فى المثال السابق هو جدول العملاء).

أنواع علاقات الارتباط

يمكن ربط جدولين إذا كان كليهما يشتمل على حقل أو أكثر بهما نفس البيانات، وعادةً تسمى الحقول فى كلا الجدولين بنفس الاسم. مثل رقم العميل فى جدول بيانات العملاء ورقم العميل فى جدول الطلبات.

يمكن إنشاء ٣ أنواع من العلاقات: علاقة واحد مقابل واحد، وعلاقة واحد مقابل مجموعة وعلاقة مجموعة مقابل مجموعة، ونوضح فيما يلى الفرق بين هذه الأنواع من العلاقات.

علاقة واحد مقابل مجموعة One-to-many

تسمى هذه العلاقة أحياناً "علاقة ارتباط رأس بأطراف" ومعناها أيضا علاقة واحد مقابل مجموعة وهى الأكثر استخداما. وتعنى أن السجل الواحد فى جدول البيانات (يسمى الجدول الرئيسى أو **Primary Table**) يقابله أكثر من سجل فى جدول آخر (يسمى الجدول المرتبط أو **Related Table**). فمثلاً قاعدة البيانات **Sales** تسجل كل فاتورة فى سجل واحد فى جدول إجماليات الفواتير، وتسجل تفصيلات هذه الفاتورة فى سجل أو أكثر فى جدول "تفصيلات الفواتير"، ولذلك يقابل كل سجل (بيانات فاتورة) فى جدول إجماليات "الفواتير" سجلا أو أكثر (تفصيلات الفاتورة) فى جدول "تفصيلات الفواتير".

علاقة واحد مقابل واحد One-to-one

أحيانا تسمى هذه العلاقة "علاقة ارتباط رأس برأس" وهذا النوع من العلاقة أقل استخداماً من النوع السابق، وفيه كل سجل فى الجدول الرئيسى يقابله سجل واحد فى الجدول المرتبط به. ومن الأمثلة التى تستخدم فيها علاقة واحد مقابل واحد، عندما ترغب فى فصل معلومات العميل إلى بيانات عامة وبيانات خاصة، فمثلاً يمكن أن تضع معلومات عامة عن العميل مثل الاسم والعنوان فى الجدول الرئيسى وتضع معلومات خاصة عن العميل مثل الرصيد فى الجدول التابع.

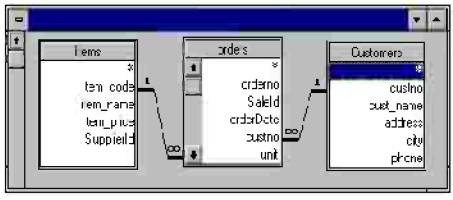
علاقة مجموعة مقابل مجموعة ***Many-to-many***

أحيانا تسمى هذه العلاقة "علاقة ارتباط أطراف بأطراف". وهذا النوع من العلاقة أيضا نادر الاستخدام وفيه يقابل كل سجل من الجدول الرئيسى عدة سجلات فى الجدول المرتبط، ويقابل السجل الواحد فى الجدول المرتبط عدة سجلات فى الجدول الرئيسى. ومن الأمثلة على ذلك فى قاعدة البيانات التى تشتمل على جدول للمنتجات وجدول لأوامر الشراء، يمكن أن يقابل السجل الواحد فى جدول أوامر الشراء أكثر من سجل فى جدول المنتجات، ومن الناحية الأخرى من الممكن أن يظهر المنتج الواحد فى عدة طلبيات وبالتالى يمكن أن تجد لكل سجل فى جدول المنتجات أكثر من سجل فى جدول أوامر الشراء.

هذا النوع من العلاقات معقد ويحتاج لدراية كافية بالبرمجة لأنه من الممكن أن يسبب مشكلة ما لم تتدخل لربط الجدولين بأسلوب غير مباشر يتلخص فى إنشاء جدول ثالث يعمل على تجزئة علاقة ارتباط مجموعة مقابل مجموعة إلى علاقتين من نوع واحد مقابل مجموعة، وفى هذه الحالة تضع المفتاحين الأساسيين لكلا الجدولين فى الجدول الثالث.

مثال: علاقة الارتباط بين جدول "العملاء"**Customers** وجدول "الأصناف" **Items** علاقة "مجموعة مقابل مجموعة" فأى عميل يمكن أن يشترى أى صنف والعكس صحيح فأى صنف يمكن أن يباع لأي عميل.

للتغلب على هذه المشكلة التى تسبب صعوبة بالغة عند التصميم، يتم تجزئة علاقة "مجموعة مقابل مجموعة" إلى علاقتين من نوع "واحد مقابل مجموعة" وذلك بإضافة جدول ثالث هو جدول "الطلبات" **Orders** وبذلك تصبح العلاقة بين كل جدول والجدول الذى يتعامل معه هى علاقة "واحد مقابل مجموعة". انظر شكل ١٠-٤ ومنه تستنتج أن جدول **Orders** يمكن أن يحتوى على أكثر من صنف ويمكن أيضا أن يحتوي على أكثر من عميل.



شكل ١٠-٤ تقسيم علاقة مجموعة مقابل مجموعة إلى علاقتين من نوع واحد مقابل مجموعة

معالجة البيانات

من أهم طرق المعالجة التى تتم على البيانات فى الجداول العمليات التالية:

- الفهرس **Index**
- التصفية **Filter**
- الترتيب أو الفرز **Sort**
- التحقق من البيانات **Validation**
- تكامل قاعدة البيانات **Integrity**

وفيما يلي نشرح كيفية استخدام كل طريقة من هذه الطرق في معالجة البيانات.

فهرسة الجداول

يتم إدخال المعلومات وترتيبها طبقاً لأسبقية الدخول ويسمى هذا الترتيب بالترتيب الطبيعى وعندما تريد أن ترى هذه البيانات بترتيب مخالف لترتيبها الطبيعى فإن استخدامك للفهرس **Index** يصبح ضرورة. والفهرس عبارة عن جدول من نوع خاص يحتوى على مفتاح الجدول الذى تريد ترتيبه طبقاً لهذا المفتاح (ترتيباً تنازلياً أو تصاعدياً) فيتيح لك الطريقة التى تساعدك على رؤية هذا الجدول طبقاً للترتيب المنطقى المطلوب. كما يحتوى هذا الفهرس على مجموعة من المؤشرات **Pointers** التى تتيح لك الوصول إلى المكان الحقيقى لأى سجل فى جدول معرف لديه. يتم استخدام الفهرس لزيادة سرعة البحث عن البيانات.

يمكن أن نستخدم أكثر من فهرس لجدول واحد للحصول على الجدول بترتيبات مختلفة.

تصفية البيانات Filter

تعتبر تصفية البيانات **Filtering** طريقة أخرى لإظهار المعلومات المطلوبة فقط من جدول أو أكثر فى قاعدة البيانات. يتم التحكم فى إظهار المعلومات المطلوبة فقط من خلال تحديد شروط معينة فتظهر المعلومات التى تحقق هذه الشروط دون الأخرى.

يتم استخدام المعاملات مثل أكبر من > وأصغر من < ويساوى = ولا يساوى <> ... الخ داخل الشرط. ويكون ناتج الشرط هو ناتج منطقى فكل سجل تتحقق فيه الشروط ويعطى النتيجة **True** تظهر بياناته وكل سجل لم يتحقق فيه الشرط يتم تجاهله.

ترتيب (فرز) البيانات

يتم استخدام الأمر **Sort** للحصول على بيانات مرتبه بترتيب معين (تنازلى أو تصاعدى) وتظهر أهمية استخدام هذا الأمر فى الأحوال التى لا يستخدم فيها الفهرس **Index** كما يحدث عند استخدام **Dynast** (وهى من الطرق التى تستخدم فى إظهار مجموعة من السجلات من جدول أو أكثر وسوف يتم مناقشتها بالتفصيل ضمن الطرق المختلفة للتعامل مع البيانات فى فصولٍ قادمة داخل هذا الكتاب).

كما يتم استخدام عملية الترتيب من خلال الأمر **ORDER BY** الخاص بالاستعلامات (سيتم مناقشة جميع الأوامر التى تستخدم فى الاستعلامات فيما بعد).

التحقق من البيانات Validation

المصمم الجيد للبرامج لا يعطى الفرصة للمستخدم أن يخطئ أخطاءً غير منطقية دون التنبيه عليه والتأكد التام من البيانات المحفوظة بالجداول.

أثناء تصميم قواعد البيانات يجب مراعاة إدخال الشروط اللازمة للتحقق من البيانات التى يتم إدخالها إلى الجداول. فمثلاً عند إدخال بيانات (فى حقل تاريخ **Date**) يجب اعتبار يوم ٣١ فبراير تاريخ خاطئ ، بل وأن يوم ٢٩ فبراير لا يعتبر تاريخ صحيح إلا مع بعض

السنوات الميلادية دون الأخرى... وهكذا. كما أنك إذا أردت إدخال عمر شخص فلا يعقل أن يساوى صفر أو أقل من صفر ... وهكذا لابد من مراعاة جميع الأخطاء الغير منطقية التى يمكن أن تتسرب كبيانات داخل الجداول من خلال المستخدم.

تتم عملية التحقق من البيانات إما أثناء كتابة تعليمات البرنامج أو عند تصميم الجداول وتوصيف الحقول بداخلها وهى إمكانية موجودة بالفعل داخل قاعدة البيانات Access وهى موجودة أيضاً فى معظم قواعد البيانات.

المحافظة على تكامل قاعدة البيانات Integrity

يقصد بالمحافظة على تكامل قاعدة البيانات أن تظل بيانات الجداول دائماً دقيقة وصحيحة تحت أى ظرف من الظروف المستقبلية. ومن مثالنا السابق تخيل أنك أردت أن تحذف سجل عائلة معينة من جدول العائلات، وفى هذه الحالة يلزمك حذف جميع أفراد العائلة التى تحمل نفس اللقب من جدول العملاء، وإن لم تفعل ذلك فإن هذه البيانات الموجودة فى جدول العملاء ستكون بدون مرجع لها (لقب العائلة) ومعنى هذا أنك ستجد بعض العملاء ليس لهم بيانات كافية رغم أنك تحققت تماماً من إدخال هذه البيانات داخل الجداول من قبل، وهنا تظهر المشكلة التى تزداد مع الوقت. وتقع هذه المسئولية على كل من المصمم والمبرمج فلابد من المحافظة الدائمة على تكامل البيانات فى قاعدة البيانات.

الاستعلامات Queries

إذا أردت أن تستعلم عن الطلبات فى تاريخ معين (بعد ٢٠٠٨/٧/١ مثلاً) من جدول الطلبات الموجود فى الجدول التالى:

رقم الطلب	التاريخ	إجمالى المبلغ	اسم العميل
1001	٢٠٠٨/٥/١	5.25	أحمد إبراهيم
1003	٢٠٠٨/٧/١	3.45	عصام سليمان
1004	٢٠٠٨/٧/١٥	1.33	عادل سليم
1002	٢٠٠٨/٦/١	10.75	محمود محمد

رقم الطلب	التاريخ	إجمالى المبلغ	اسم العميل
1005	٢٠٠٨/١٠/١	5.60	عمرو حسن

لكى تحصل على الاستعلام المطلوب، اكتب الأمر التالى (سنقوم بشرحه فيما بعد).

```
Select Orders.OrderDate , Orders.TotalCost , Orders. Cust_name
Where Orders.OrderDate > # 01 / 07 / 08 #
```

يتـــم ضبط شكل التاريخ فى بيئة **Windows** من خلال الإعدادات الإقليمية الموجودة داخل لوحة التحكم.

يقوم الأمر **Select** باختيار البيانات التالية من جدول **Orders** طبقاً للترتيب الموجود فى الأمر حيث تدل العبارة:

Orders.OrderDate

على تاريخ الطلب من الجدول **Orders** (جدول الطلبات) وتدل العبارة التى تليها **Orders.TotalCost** على القيمة المطلوب تحصيلها من العميل وهى موجودة أيضاً فى نفس الجدول ثم العبارة **Orders. Cust_name** على اسم العميل الموجود في جدول **Orders** وكلمة **Cust_name** تعنى اسم العميل.

يقوم الأمر **Select** بإظهار البيانات السابقة وبالترتيب المذكور داخل الأمر طبقاً للشرط المعلن عنه بعد كلمة **Where** وهو:

Orders. OrderDate > # 01/07/08#

ويعنى هذا الشرط أنه يجب أن يكون تاريخ الطلب أكبر من ٢٠٠٨/٧/١ أى بعده وتكون نتيجة هذا الأمر هو الاستعلام التالى:

رقم الطلب	التاريخ	إجمالى المبلغ	اسم العميل
1004	٢٠٠٨/٧/١٥	1.33	عادل سليم
1005	٢٠٠٨/١٠/١	5.60	عمرو حسن

مراجعة تصميم البرنامج

بعد عمل التصميم المبدئى لقاعدة البيانات لابد من عمل المراجعة اللازمة للبحث عن أخطاء موجودة فى التصميم وإمكانية حلها. ولعل التعامل مع بيانات شبه حقيقية وإدخالها إلى البرنامج وملاحظة عمل البرنامج من أفضل الطرق لعملية المراجعة.

السؤال الأول الذى ينبغى أن يكون على لسان مصمم البرنامج دائماً هو هل هذا التصميم يوفر كل مطالب العميل أم لا ؟ (أى هل حقق التصميم كل أهداف البرنامج المطلوبة من العميل) فإذا وجد أى خطأ فإن هذا يستدعى مراجعة كل وظيفة من الوظائف التى تكلمنا عنها سابقاً (والتى فى مجموعها تمثل النظام) باستخدام بيانات كأمثلة فإذا ما تبين مصدر الخطأ فإنه يلزم إعادة تصميم هذه الجزئية مرة أخرى بما لا يتعارض مع التصميم المنطقى للبرنامج ككل.

فى نهاية مراحل تصميم البرنامج يتم تجهيز قاعدة البيانات وتصميم جداولها وتوصيف جميع الحقول بسجلات الجداول.