الفصل الثانى
التخطيط الجيد لقواعد البيانات

في هذا الفصل سوف نتناول قواعد البيانات بصفه عامة ولغة SQL كلغة برمجة تستخدمها العديد من نظم إدارة قواعد البيانات العلائقية الموجودة بالساحة الآن وتشمل هذه المقدمة الموضوعات التالية :

- *نظرة تاريخية على قواعد البيانات*
- *تطور قواعد البيانات.*
- *تنظيم وترتيب قواعد البيانات.*
- *التخطيط لقواعد البيانات .*
- *تجنب تكرار البيانات .*
- *تجنب الحشو والزيادة .*
- *أنواع علاقات الارتباط .*
- *تصميم قواعد البيانات .*
- *دليل قواعد البيانات .*
- *نظرة عامة على لغة SQL.*

نظرة تاريخية على قواعد البيانات

لا تستغرب عزيزي القارئ في أننا نسهب في التمهيد للموضوع ، فالواقع أن الحديث حول قواعد البيانات يكتسب أهمية بالغة ، اذ أنه يشرح لنا ما هية قواعد البيانات ونشأتها وتطورها الأمر الذي يوضح الحاجة الى لغة قواعد البيانات الهيكلية SQL وكيف أننا نحتاج هذه اللغة لاستخدامها في الاستعلام فهي بحق Structured Query Language أي أنها تستخدم قواعد البيانات الهيكلية للتحكم في بناء البيانات واستخراجها أو الاستعلام عن معلومة معينة .

تخيل أنك تعمل في شركة بترول أو بنك أو مسئول عن ادارة سوبر ماركت أو صيدلية أو غير ذلك من المجالات ، أو لنقل كل المجالات ، ما هو حجم البيانات المطلوب تسجيله؟ ولو كانت البيانات هذه في هيئة حكومية مثل مصلحة المعاشات أو السجل المدني، كم يستغرق البحث عن معلومة؟ يوم أو شهر، في بعض الأحيان وبعض الجهات كان البحث في الماضي يستغرق عدة شهور أو سنة ومن هنا فان المثل الدارج يوم الحكومة بسنة لم يأت من فراغ فبعض المعلومات كان يستغرق الوصول إليها عام وسط آلاف عدة من الملفات !

في بداية العمل الالكتروني كانت البيانات تسجل بالتتابع وبالطبع كان البحث أسرع من الطريقة اليدوية ، ولكن الأمر كان يستغرق بعض الوقت ، فمثلا لو كان لديك بيانات لمليون صنف وكان الصنف الذي تبحث عنه في النصف الأخير من القائمة المسجلة فسوف تمر على النصف مليون الأول لتستعلم عن الصنف الى أن تجده في النهاية .

هنا ظهرت الحاجة لقواعد البيانات وفيها يمكن تسجيل البيانات عن طريق ايجاد علاقات معينة بين الحقول بحيث يستغرق الأمر عدة ثوان لتصل الى المعلومة وهذا يتوقف على كيفية بناء البيانات داخل قاعدة البيانات .

معظم قواعد البيانات تعتمد على وجود علاقات بين البيانات أو حقول البيانات.

ويعتمد هذا الأسلوب على تصنيف البيانات في صورة مجموعات متجانسة كل

مجموعة تشكل في داخلها علاقات. وحول هذا المضمون يتم تكوين الجداول ، وشكل الجدول عبارة عن مجموعة من الصفوف ذات علاقة مترابطة يسمى سجل **Record** بينما يشكل كل عمود في الجدول حقل واحد متكرر في كل صفوف الجدول ، ولنأخذ مثالا لأحد الجداول كما في الجدول ٢-١ التالي :

تاريخ الميلاد	اسم العائلة	الاسم الأول	رقم الموظف
10-11-1960	يعقوب	محمد	1
4-10-1977	يعقوب	أحمد	2
2-3-1980	علي	حسن	3
1-9-1958	سعد	فؤاد	4
5-5-1984	فهمي	سيد	5
3-6-1981	أبو العطا	سارة	6
21-1-1979	بدر	جمال	7
23-12-1983	كمال	كامل	8
27-9-1969	محمد	وحيد	9

في الجدول السابق نلاحظ أن كل موظف تم تسجيل بياناته في صف **Row** هذا الصف يسمى **Record** أو سجل . بينما كل عمود يسمى حقل **Field** وعندما نقوم بالبحث داخل هذا الجدول باستخدام لغة الاستعلام **SQL** فاننا نستخدم اسم الحقل ونحن هنا سنجد عناوين الحقول في الصف الأول . نفترض أننا سنبحث عن شخص أسمه الأخير أبو العطا واسمه الأول سارة فإننا سنطلب الاستعلام عن اسم العائلة إذا كان يساوي "أبو العطا" ثم نستعلم عن الاسم الأول إذا كان يساوي "سارة" وهكذا .

تطور قواعد البيانات

منذ فترة طويلة تربو على الربع قرن فقدت مني بطاقتي الشخصية وبدأت في إجراءات استخراج بطاقة شخصية (بدل فاقد) ، اذكر ان المنظر الذي رأيته هو أول تعامل

حقيقي لي مع قواعد البيانات. لا تضحك عزيزي القارئ ، فقد شاهدت قواعد البيانات الحقيقية في مكتب السجل المدني .

غرفة واسعة بها عدد كبير من الدواليب المعدنية (هكذا يطلق عليها في مصر ويقصد بها الخزائن المعدنية) كل دولاب (خزانة) مقسم الى عدد من الأرفف ، وكل رف محفوظ فيه عدد من الملفات ، كل ملف خاص بمواطن ويوجد بداخل هذا الملف كافة الأوراق الخاصة بالمواطن منذ استخراج بطاقته الشخصية الأولى ، وستجد في هذا الملف على سبيل المثال شهادة الميلاد وسجل العائلة ، وفصيلة الدم والمؤهل الدراسي والمهنة وهكذا العديد من الأوراق والمستندات .

وعلى غير العادة، فقد تجاوب معي الموظف وشرح لي كيف يمكنه أن يصل الى أي ملف حيث أن كل رف عليه أرقام البطاقات الشخصية الخاصة به وبالتالي فان هذه الخزانة أو الدولاب مكتوب عليه أرقام الأرفف وبالتالي أرقام البطاقات ، ولدى الموظف دفتر مسجل فيه أسماء الدواليب أو الخزائن وأمام كل خزانة أرقام الأرفف وأرقام البطاقات .

كنت في بداية حياتي مع الكمبيوتر (Mainframe) وقد استغرق مني الأمر جهدا كبيرا لاستثارة الموظف لكي يشرح لي نظام قواعد البيانات الذي يستخدمه .

كنت أقارن بين السرعة التي أقوم بها كمبرمج على الكمبيوتر الكبير للحصول على معلومة وتلك التي يصل بها الموظف إلى المعلومة ، ولولا أنني استثرت الموظف واستثرت ملكاته لقال لي كما يقول لغيري "فوت علينا بعد أسبوعين" ، على كل حال فان الموظف كان يستخدم نظام لقواعد البيانات وان كان يدويا .

هذه الأيام يحمل كل مواطن رقم قومي عند بلوغه السن القانونية أو يتمتع بالرقم القومي للعائلة إذا كان دون السن ، على حد فهمي لهذا النظام بطبيعة الحال حيث يصعب استنتاج نظام قواعد البيانات المستخدم والعلاقات المستخدمة في نظام الرقم القومي.

بين قاعدة البيانات التي شاهدتها في بداية عملي كمبرمج كمبيوتر، والرقم القومي مرت العديد من السنوات، فماذا حدث في تلك السنوات ؟؟؟

في هذه الفترة حدثت طفرات تكنولوجية هامة ، بدءا من إنتاج الحاسب الشخصي

Personal Computer بإمكانيات تقارب تلك المستخدمة في الحاسب الكبير من حيث سرعة المعالجة وحجم التخزين ، وحدثت تطورات عديدة وسريعة على الحاسب الشخصي. حدث أيضا تطور في نظام الشبكات المحلية Local Area Network LAN بحيث يتم ربط عدد من الحاسبات الشخصية في منظومة واحدة وأصبح بالإمكان تداول البيانات وتبادل المعلومات بين أجهزة الحاسبات الشخصية ثم ظهرت تكنولوجيا الخادم / العميل Client/Server والتي تميزت بعدد من المميزات التي ساهمت في معالجة البيانات واستخدام قواعد البيانات بكفاءة ، ففي هذه التكنولوجيا نجد أن قواعد البيانات يمكن وضعها على خادم Server ويتم التعامل بها من خلال تطبيقات موجودة على الحاسب الشخصي (العميل) أو على الخادم بحيث يمكن تطوير التطبيقات على الحاسب الشخصي لتتعامل مع قواعد البيانات الموجودة على الخادم وهذا تطلب مهارات خاصة للمبرمجين للتعامل مع هذه التقنية. أيضا أتاحت هذه التقنية اتصال خادم في شبكة بخادم آخر في شبكة أخرى وهكذا لنصل في النهاية الى أن عدد من الشبكات المحلية تتعامل مع قواعد بيانات واحدة موجودة على خادم في شبكة من الشبكات ويتم التعامل معها من خلال عميل الى خادم في هذه الشبكة ثم من الخادم الى خادم في شبكة أخرى وهذا هو الفكر المستخدم في الشبكة الدولية أو الانترنت .

وفي هذه التقنية وباستخدام بروتوكول تناقل بيانات أصبح بالإمكان التعامل مع عدد من أنظمة التشغيل المختلفة مثل Windows, IBM OS/2, Macintosh, and/or UNIX/Linux باستخدام SQL بإمكانياتها المتعددة في الاتصال بقواعد البيانات الموجودة تحت أنظمة التشغيل المختلفة .

وباستخدام تقنية الخادم العميل Client/Server إضافة إلى مميزات لغة SQL العديدة أصبح بالإمكان التعامل مع قواعد نظم إدارة البيانات العلائقية RDBMS .

تنظيم وترتيب قواعد البيانات

باستثناء قواعد البيانات الصغيرة يندر أن تشتمل قاعدة البيانات على جدول واحد، وإنما الواقع أنك تنشئ أكثر من جدول وتستخدم الحقول المشتركة لربط هذه الجداول. ولذلك

فان التصميم الجيد لقاعدة البيانات يتطلب منك أن تنشئ جداول صغيرة يشتمل كل منها على بيانات ذات طبيعة واحدة وتقوم بربط هذا الجداول باستخدام حقل مشترك بينها أو أكثر. وبهذا يمكنك التعامل مع بيانات أكثر من جدول بسهولة.

التخطيط لقاعدة البيانات Database Normalization

قبل أن تنشئ قاعدة البيانات ، اجلس مع نفسك وأحضر ورقة وقلما وحاول أن تضع خطة لمحتويات كل جدول وأن تحدد كيف ستبنى العلاقة بين الجداول قبل أن تشرع فى تصميم الجداول . التخطيط الجيد لقاعدة البيانات يوفر عليك وقت إعادة تصميم قاعدة البيانات ويوفر أيضا المساحة التى تستخدمها على القرص المغناطيسى.

استعن بالإرشادات التالية عند التخطيط لقاعدة البيانات :

- حدد بالضبط المطلوب من قاعدة البيانات.
- حاول أن تكون الجداول صغيرة وضع فى اعتبارك أن تكون سهلة ويمكن تعديلها فى المستقبل.
- تجنب بقدر استطاعتك تكرار البيانات فى الجداول الموجودة بقاعدة بيانات واحدة.
- حاول أن تضع فى كل جدول البيانات التى تنتمى إلى مجموعة واحدة مثلا فى قاعدة بيانات العملاء جدول لبيانات العميل الشخصية وجدول لأوامر الشراء وجدول ثالث للأصناف ... وهكذا .
- ضع فى اعتبارك أن معظم الناس تستخدم التقارير لاستخراج البيانات وتجميعها ولذلك من المناسب أن تتجنب الحقول التى تشتمل على ناتج عمليات حسابية أو إجماليات.

ونوضح فيما يلي أمثلة لقواعد بيانات تتوفر فيها الإرشادات السابقة، وفي نفس الوقت تلبي حاجة النظام بدون تكرار للمعلومات، وتتجنب الحشو والزيادة في البيانات .

تجنب تكرار البيانات

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 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 **Orders** 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 معي حجم هذه البيانات وهل يصح جمعها فى جدول واحد أم لا ؟.

الإجابة : إذا تم جمع هذه البيانات كلها فى جدول واحد فإننا سنقع فى مشكلتين ،

المشكلة الأولى: هى الزيادة الهائلة فى حجم البيانات نتيجة تكرارها. انظر الشكل٢-٢، تلاحظ أن كل بيانات العميل مثل الاسم والعنوان والتليفون تتكرر مع كل طلبية.

اسم العميل	العنوان	رقم الهاتف	تاريخ الطلب	قيمة الطلب
وليد عبدالرازق	منوف - المنوفية	436139	08/07/99	430
وليد عبدالرازق	منوف - المنوفية	436139	13/09/99	300
وليد عبدالرازق	منوف - المنوفية	436139	12/10/99	270
حسام البسانى	بلبيس - الشرقية	5433456	15/07/99	470
حسام البسانى	بلبيس - الشرقية	5433456	12/11/99	600
سعيد حمودة	طنطا - الغربية	668400	11/11/99	220

شكل ٢-٢ جدول بيانات يحتوى على بيانات مكررة.

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

ولعلاج المشكلتان السابقتان ننصح بتقسيم هذا الجدول إلى جدولين: جدول لبيانات العملاء والآخر لأوامر الشراء كما فى الشكل ٢-٣.

ويتم ربط الجدولين بناء على بيانات حقل مشترك بينهما. 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 أيضا 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 .

رقم العميل	اسم العميل	العنوان	رقم الهاتف
1	وليد عبدالرازق	منوف - المنوفية	436139
2	حسام اليماني	بلبيس - الشرقية	5433456
3	سعيد حمودة	طنطا - الغربية	668400

جدول العملاء

رقم العميل	تاريخ الطلب	قيمة الطلب
1	08/07/99	430
1	13/09/99	300
1	12/10/99	270
2	12/11/99	600
2	15/07/99	470
3	11/11/99	220

شكل ٢-٣ تقسيم الجدول إلى جدولين: واحد للعملاء والآخر لطلبات الشراء .

تجنب الحشو والزيادة

بفرض أنك تريد إنشاء قاعدة بيانات للمتدربين، وتريد أن تعرف من قاعدة البيانات هذه الدورات التدريبية التى حصل عليها أى موظف. من الطبيعى أن ، يكون في شركتك العديد من الموظفين، وأن يحصل الموظف الواحد على أكثر من دورة. يمكن إنشاء جدول واحد في قاعدة البيانات يشتمل على بيانات المتدربين والدورات، في هذه الحالة يمكن أن يكون الجدول مثل شكل ٢-٤ .

اسم الموظف	الوظيفة	هاتف الموظف	تاريخ الدورة	اسم الدورة	عدد الساعات	أكمل؟
وليد عبدالرازق	مبرمج	436139	01/10/99	تخطيط المشروعات	5	☑
صبحي عمران	مدير مبيعات	436806	11/09/99	البرج الاحترافي	7	☑
شريف عرفة	مدير حسابات	5466647	02/10/99	مهارات التفاعل	9	☑
سمير البشبيشي	مدير تنفيذي	2405330	18/10/99	مهارات القيادة	5	☐

شكل ٢-٤ جدول بيانات يحتوى على بيانات أكثر من اللازم.

لكن يعاب على هذا الجدول صعوبة استخدامه عمليا، افرض مثلا أن أحد المتدربين

حصل على أكثر من دورة تدريبية، في هذه الحالة سيشتمل الجدول على أكثر من سطر لهذا العميل، سطر واحد لكل دورة. وهي نفس المشكلة التى واجهتنا في المثال السابق، وهي أكثر من سجل لنفس المتدرب.

افرض أن موظفا حصل على دورة أو أكثر ترك العمل بالشركة، عندما تقوم بحذف بيانات هذا الموظف ستحذف أيضا الدورات التي حصل عليها، وبالتالي لن تكون بيانات الدورات عندك سليمة.

الحل في مثل هذه الحالات تقسيم جدول البيانات الموجود بشكل ٢-٤ السابق إلى ٣ جداول. واحد للبيانات الأساسية للموظف، والثاني للدورات التدريبية، والثالث للدورات. يوضح شكل ٢-٥ شكل الجداول الثلاثة والحقول التى يشتمل عليها كل جدول .

شكل ٢-٥ تقسيم جدول البيانات إلى ٣ جداول لتجنب التكرار والحشو .

أنواع علاقات الارتباط

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

توجد عادة في قواعد البيانات ٣ أنواع من العلاقات: علاقة ارتباط "رأس برأس"، علاقة ارتباط "رأس بأطراف"، وعلاقة ارتباط "أطراف بأطراف"، سنشرح فيما يلى المقصود بكل

نوع من هذه الأنواع من علاقات الارتباط .

علاقة ارتباط رأس بأطراف One-To-Many

هذا النوع من علاقات الارتباط هو الأكثر استخداما. وتعنى أن السجل الواحد فى جدول البيانات (يسمى الجدول الرئيسي أو **Primary Table**) يقابله أكثر من سجل فى جدول آخر(يسمى الجدول المرتبط أو **Related Table**). فمثلا قاعدة البيانات **Sales** التي نستخدمها فى شرح هذا الكتاب تسجل كل فاتورة فى سجل واحد فى جدول الفواتير **Invoices**، وتسجل تفصيلات هذه الفاتورة فى سجل أو أكثر فى جدول تفصيلات الفواتير **Invoice_details** ، ولذلك يقابل كل سجل (بيانات فاتورة) فى جدول الفواتير سجلاً أو أكثر (تفصيلات الفاتورة) فى جدول تفصيلات الفواتير.

علاقة ارتباط رأس برأس One-To-One

هذا النوع من العلاقة أقل استخداما من النوع السابق، وفيه كل سجل فى الجدول الرئيسي يقابله سجل واحد فى الجدول المرتبط به. ومن الأمثلة التي تستخدم فيها علاقة "رأس برأس"، عندما ترغب في فصل معلومات العميل إلى بيانات عامة وبيانات خاصة، فمثلا يمكن أن تضع معلومات عامة عن العميل مثل الاسم والعنوان فى الجدول الرئيسي وتضع معلومات خاصة عن العميل مثل الرصيد فى الجدول التابع.

علاقة ارتباط أطراف بأطراف Many-To-Many

هذا النوع من العلاقة أيضاً نادر الاستخدام وفيه يقابل كل سجل من الجدول الرئيسي عدة سجلات فى الجدول المرتبط، ويقابل السجل الواحد فى الجدول المرتبط عدة سجلات فى الجدول الرئيسي. 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 وبالتالي يمكن أن تجد لكل سجل فى جدول "المنتجات" أكثر من سجل فى جدول "أوامر الشراء" .

هذا النوع من العلاقات معقد ويحتاج لدراية كافية بقواعد البيانات والتعامل معها لأنه من الممكن أن يسبب مشكلة ما لم تتدخل لربط الجدولين بأسلوب غير مباشر يتلخص فى إنشاء جدول ثالث يعمل على تجزئة علاقة ارتباط أطراف بأطراف إلى علاقتين من نوع رأس بأطراف، وفى هذه الحالة تضع الـــمفتاحين الأساسيين لكلا الجدولين فى الجدول الثالث.

تصميم قواعد البيانات

تحدثنا في الفصل الأول نظرة عامة على قواعد البيانات عن التخطيط الجيد لقواعد البيانات **Normalization** وكيف تم تصميم قاعدة البيانات بطريقة تضمن عدم تكرار الحقول بين الجداول وكيف نقوم بتقسيم البيانات المتجانسة في جداول صغيرة يسهل التعامل معها لكي نضمن أن نتعامل مع جداول بعينها حسب الحاجة ونترك باقي الجداول لمعالجات أخرى .

يجب أن نضع العوامل التالية أثناء تصميم قاعدة البيانات وقبل الإنشاء الفعلي لها :

- تأمين وحماية قاعدة البيانات
- سرعة التعامل مع قاعدة البيانات ومعالجتها
- المساحات المتاحة لديك من الأقراص الصلبة
- سرعة تعديل البيانات وقواعد البيانات
- السرعة أثناء التعامل مع أكثر من جدول في نفس الوقت
- مساندة نظام إدارة قواعد البيانات المستخدم " **RDBMS** " للجداول المؤقتة .

تعتبر المساحة المتاحة على الأقراص الصلبة من العوامل المهمة التي نضعها في اعتبارنا أثناء تصميم قاعدة البيانات ، رغم أنكم عزيزي القارئ تتحدثون الآن بوحدة قياس كبيرة **Giga Byte** ولكن مساحة قواعد البيانات لا زالت مؤثرة سواء في تصميم قاعدة البيانات من حيث حجم البيانات وحجم التطبيقات المستخدمة ، كما أن المساحة مؤثرة من حيث اختيار نظام إدارة قواعد البيانات " **RDBMS** " الذي يساند حجم البيانات المتوقع .

دليل قواعد البيانات Data Dictionary (System Catalog)

يعتبر فهرس البيانات أو كتالوج نظام قواعد البيانات بما يحتويه من بيانات قد تضطر لتعديلها أو إنشاؤها ، من الموضوعات الهامة التي يجب أن يلم بها مصمم قواعد البيانات **Database Designer** وأيضا مدير قواعد البيانات **Database Administrator DBA** والأخير هذا قد يشارك في تصميم قواعد البيانات خاصة إذا كانت كبيرة ، ويساعد بصفة عامة في عمليات التخطيط الجيد **Normalization** وفي حجز المساحات ، على أنه من أهم المعلومات التي يجب أن يشملها الكتالوج ما يلي:

- وصف قاعدة البيانات والغرض منها ومن سوف يستخدمها
- توثيق خصائص قاعدة البيانات نفسها وأين تم تخزينها والحجم الأساسي لقاعدة البيانات وحجم ملف التسجيل **Log File** وهو الملف الذي يتم فيه تسجيل حركة تشغيل قواعد البيانات
- تخزين أي تعليمات بلغة المصدر **SQL Scripts Source Code**
- وصف تفصيلي لكل جدول من حيث عدد الحقول وأسماؤها وغرض كل جدول
- الوصف الداخلي لبناء كل جدول **Internal Table Structure** وسوف نتحدث عند ذلك بالتفصيل في فصول لاحقة
- وصف للضوابط الموضوعة لقاعدة البيانات

هناك العديد من الأدوات التي تأتي مع نظم قواعد البيانات المختلفة إما أن تكون جاهزة للعمل ضمن نظام قواعد البيانات أو تأتي مصاحبة للنظام وتقوم أنت بتثبيتها ، هذه الأدوات تساعد مصمم قواعد البيانات على بناء الدليل **Dictionary (System Catalog)** وسوف نتعرف على بعض هذه الأدوات لاحقا في هذا الكتاب .

عموما فانك حين تقوم بإنشاء قاعدة البيانات ، فان هذا الكتالوج سوف يتم بناؤه تلقائيا بناء على المعطيات الثابتة **Defaults** وستجد في هذا الكتالوج معلومات مفيدة جدا خاصة بكل قاعدة بيانات يتم إنشاؤها .

نستخدم كثيرا اللفظ قاعدة بيانات واللفظ نظام إدارة قواعد البيانات ، في الواقع أنت تنشئ قاعدة بيانات واحدة أو أكثر داخل نظام إدارة قواعد البيانات والمقصود بها نظام إدارة قواعد البيانات مثل **MS SQL Server** أو نظام **MySQL** وقواعد البيانات **Oracle** وغيرها ، فقد جرت العادة إطلاق اسم نظام إدارة قواعد البيانات مقابل الاختصار **RDBMS** ومعناه **Relational Database Management System** أما قاعدة بيانات فهي من تصميمك وتكوينها من مسئوليتك داخل **RDBMS** .

نظرة عامة على لغة SQL

تستخدم لغة **SQL** كلغة قياسية لمعالجة البيانات داخل نظم إدارة قواعد البيانات العلائقية **RDBMS**، حيث يمكن لأي من مبرمجي **SQL** القيام بالمهام التالية :

- تعديل هيكل قاعدة البيانات
- تغيير اعدادات نظام الحماية في قواعد البيانات
- اضافة صلاحيات لمستخدم قواعد البيانات أو جداولها
- الاستعلام عن معلومات من داخل قاعدة البيانات
- تعديل محتويات قواعد البيانات

ولنتأمل الوظائف أو المهام التي تقوم بها اللغة ونقارنها بما تحمله اللغة من معنى **SQL** وهى عبارة عن كلمة **Structured** أي هيكلي أو بنائي وكلمة **Language** ومعناها لغة وباستخدام هاتين الكلمتين فان الأمر مفهوما ، ولكن كلمة **Query** هي هنا غير مفهومة ولا ندري من الذي أدخل هذه الكلمة ، فهي تعني استعلام ، ولكن اللغة تقوم بالحذف والاضافة والتعديل وما إلى ذلك ، على أي الأحوال فلا أملك تعديل هذا الاسم .

في تعليق آخر، يعمد العديد من المبرمجين الى نطق اللغة هكذا (سيكويل) أو **Seekwell** ولو دققت النظر في هذا النطق سوف تجده معبرا ولو قليلا **seek well** هل هذا يعني شيئا ؟ مش موضوعنا .

ولأن هذه اللغة تعتبر قاسما مشتركا في العديد من نظم إدارة قواعد البيانـــات العلائقيـــة RDBMS فاننا سنتناول بعضا من هذه الأنظمة فيما يلي:

أشهر نظم إدارة قواعد البيانات التي تستخدم مع SQL

أود الإشارة إلى أنني استخدم المصطلح نظام إدارة قواعد البيانات مقابل المصطلح الإنجليزي **Relational Database Management System** وتختصر هكذا **RDBMS** . أما المصطلح قواعد البيانات فإنني استخدمه مقابل المصـــطلح **Database** وتعـــني قواعـــد البيانات التي تنشئها أنت بنفسك تحت النظام أو تتعامل معها .

- *نظام قواعد البيانات* ***Microsoft Access***
- *نظام قواعد البيانات* ***MySQL***
- *نظام قواعد البيانات أوراكل* ***ORACLE***