يحسب التابع SUM مجموع قيم حقل معين:

- يُستخدم الخيار All عندما نريد الحصول على مجموع قيم حقل معين بما فيها القيم المكررة. يُعتبر هذا الخيار هو الخيار التلقائي في حال عدم تحديد أي من الخيارين Distinct أو All.

- يستخدم الخيار Distinct عندما نريد الحصول على مجموع القيم حقل معين مع استبعاد أي تكرار في القيم.

تجميع البيانات 1

عندما نتكلم عن التوابع التجميعية فلابد لنا أن نتساءل: هل نستطيع أن نطبق التوابع التجميعية على مجموعات جزئية من السجلات بدلاً من تطبيقها على كامل السجلات؟

فإذا كان لدينا جدول منتجات (Products) وأردنا حساب مجموع كلف المنتجات (unitCost) التي نحصل عليها من المورد الأول، وحساب مجموع كلف المنتجات التي نحصل عليها من المورد الثاني، وحساب مجموع كلف المنتجات التي نحصل عليها من المورد الثالث، فإننا سنحتاج لكتابة ثلاث تعليمات منفصلة تعتمد على التابع التجميعي SUM كما يلي:

```
Select sum(unitCost) from products where supplier ID= 1;

Select sum(unitCost) from products where supplier ID= 2;

Select sum(unitCost) from products where supplier ID= 3;
```

لكن هل يمكننا أن نصل إلى تركيب شبيه بالتركيب الظاهر في المثال السابق باستخدام تعليمة واحدة؟

تغطي SQL هذا المتطلب بآلية تساعد على تقسيم السجلات إلى مجموعات جزئية وتساعد على تطبيق التوابع التجميعية على كل مجموعة جزئية على حدى.

عندما نتكلم عن التوابع التجميعية فلابد لنا أن نتساءل: هل نستطيع أن نطبق التوابع التجميعية على مجموعات جزئية من السجلات بدلاً من تطبيقها على كامل السجلات؟
فإذا كان لدينا جدول منتجات، وأردنا حساب مجموع كلف المنتجات التي نحصل عليها من المورد الأول، وحساب مجموع كلف المنتجات التي نحصل عليها من المورد الثاني، وحساب مجموع كلف المنتجات التي نحصل عليها من المورد الثالث، فإننا سنحتاج لكتابة ثلاث تعليمات منفصلة تعتمد على التابع التجميعي SUM، لكن هل يمكننا أن نصل إلى نفس النتيجة بتعليمة واحدة فقط؟

تغطي SQL هذا المتطلب بآلية تساعد على تقسيم السجلات إلى مجموعات جزئية وتساعد على تطبيق التوابع التجميعية على كل مجموعة جزئية على حدى.

تجميع البيانات 2

لتجميع البيانات في SQL نستخدم تعليمة Group by والتي تأخذ الصيغة التالية:

```
Select columnA, aggFunc (aggFuncSpec) from table
where whereSpec
Group by columnA
```

مثال:

إذا كان لدينا جدول Sales يحتوي أسماء المنتجات ProductName وكمية البيع في مراكز مختلفة Quantity، يمكننا كتابة التعليمة التي تعطي الكمية المباعة من كل منتج في جميع المراكز بعد تاريخ معين.

```
Select productName, sum (quantity) from sales
where saleDate > #20/5/2013#
Group by productName
```

لتجميع البيانات في SQL نستخدم تعليمة Group by.

الكلمة المفتاحية Having

- عندما نفكر في وضع شرط ما على استعلام معين، يرد إلى ذهننا استخدام الكلمة المفتاحية where مع شرط مناسب.

```
Select field_name from table_name where condition
```

ولكن هذا الحل البديهي يكون قاصراً في بعض الحالات مثل الحالة التي يكون فيها أحد عناصر الشرط تابعاً تجميعياً.

- نستخدم الكلمة المفتاحية Having في حال أردنا أن نضع شرطاً على استعلام ما، بحيث يكون أحد عناصر الشرط تابعاً تجميعياً.

- نستعمل Having ضمن الصيغة التالية:

```
Select columnA, aggFunc (aggFuncSpec) from table
where whereSpec
Group by columnA
Having filterCondition
```

انتبه: لا تلغي الكلمة المفتاحية Having دور where بل يمكن استخدامهما معاً في صيغة واحدة كما يظهر في الصيغة السابقة

مثال:

إذا كان لدينا الجدول StudentsGrade الحاوي على أرقام الطلاب (StudentNumber) وعلى علاماتهم (grade) في مختلف المواد، وأردنا معرفة أي من الطلاب قد حصل على معدل جيد جداً (أكبر من 70) أو سيئ (أصغر من 50)، سيخطر لنا استخدام التعبير التالي:

```
Select studentNumber, Avg(grade) as averageMark from studentGrades
Where avg(grade)>70 or avg(grade)<50
Group by studentNumber
```

إن الحل السابق خاطئ، لأن where تقوم بعملية الترشيح قبل تنفيذ التابع التجميعي (avg) وليس بعده، لذا نحن بحاجة لاستخدام Having لحلّ هذه المسألة. يكون الحل كما يلي:

```
Select studentNumber, Avg(grade) as averageMark from studentGrades
Group by studentNumber
Having avg(grade)>70 or avg(grade)<50
```

عندما نفكر في وضع شرط ما على استعلام معين، يرد إلى ذهننا استخدام الكلمة المفتاحية where مع شرط مناسب.

ولكن هذا الحل البديهي يكون قاصراً في بعض الحالات مثل الحالة التي يكون فيها أحد عناصر الشرط تابعاً تجميعياً

عندها يأتي دور الكلمة المفتاحية having في حال أردنا أن نضع شرطاً على استعلام ما، بحيث يكون أحد عناصرالشرط تابعاً تجميعياً.

عموماً، لا بد من الإشارة إلى أن الكلمة المفتاحية Having لا تلغي دور الكلمة المفتاحية where بل يمكن استخدامهما معاً في صيغة واحدة.

التعبير Top N

يُستخدم التعبير (Top N) كمرافق للتوابع التجميعية ولكن استخدامه لا يقتصر عليها فقط.

يُعيد هذا التعبير أول N سجل من نتيجة الاستعلام.

يأخذ هذا التعبير الصيغة:

```
Select top N field1, field2 … from table_name
```

مثال:

لنفرض أن لدينا جدول StudentsGrade التالي الذي يحتوي على أسماء الطلاب (StudentName) وعلاماتهم (StudentMark) في جميع المواد، ولنفرض أننا أردنا معرفة أسماء الطلاب الخمسة الأوائل مُرتبة، بحسب معدلاتهم، ترتيباً تنازلياً (معدل الطالب هوالمتوسط الحسابي لجميع علاماته).

subject	studentMark	studentName
math	15	ahmad
math	22	adel
history	26	ahmad
...	...	...

نستخدم الصيغة:

```
Select top 5 studentName, avg(studentMark)
From studentsGrades
Group by studentName
order by avg(studentMark) DESC
```

يُستخدم التعبير (Top N) كمرافق للتوابع التجميعية ولكن استخدامه لا يقتصر عليها فقط. ويُعيد هذا التعبير أول N سجل من نتيجة الاستعلام.

استخدام التعبير Top N في مختلف أنظمة إدارة قواعد المعطيات

تختلف الصيغة الخاصة بالتعبير (Top N)، من نظام إدارة قواعد بيانات إلى آخر.

- ففي Mysql تكون الصيغة على الشكل التالي:

```
Select field1, field2 from table_name
Limit 0,N
```

حيث تشير الكلمة المفتاحية limit إلى مجال أرقام السجلات التي نريد الحصول عليها (سجل البداية 0 وسجل النهاية N) من مجموعة السجلات التي تعيدها تعليمة Select.

- أما في قواعد بيانات DB2 فتكون الصيغة على الشكل التالي:

```
Select field1, field2 from table_name
Fetch first N rows only
```

حيث يحدد المعامل N في التعبير "fetch first N rows only" عدد السجلات التي نريد الحصول عليها وذلك ابتداءاً من السجل الأول.

- وتعتمد الصيغة في قواعد بيانات Oracle على الكلمة المفتاحية rowNum وهي عبارة عن قيمة تلقائية يولدها نظام Oracle وتحدد ترتيب السجلات التي نريد الحصول عليها ضمن مجموعة السجلات التي تعيدها تعليمة Select. فتكون الصيغة على الشكل:

```
Select field1, field2 where rowNum<= N
```

مثال:

في حال أردنا الحصول على الأرقام الثلاثة الأولى التي سجلت أكبر عدد من الاتصالات لدى شركة هواتف محمولة وذلك اعتباراً من سجل اتصالات الزبائن (callLog) المؤلف من حقول: إسم المستخدم (userName)، ورقم هاتف المستخدم (phoneNumber). وبفرض أن نظام إدارة قواعد المعطيات المُستخدم هو نظام Oracle، نستخدم الصيغة:

```
Select phoneNumber, count(userName) from callLog
Group by phoneNumber
Order by count(userName)
Where rowNum<= 3
```

تختلف الصيغة الخاصة بالتعبير (Top N)، من نظام إدارة قواعد بيانات إلى آخر.

ففي Mysql نستخدم الكلمة المفتاحية limit التي تشير إلى مجال أرقام السجلات التي نريد الحصول عليهاK من مجموعة السجلات التي تعيدها تعليمة Select.

أما في قواعد بيانات DB2 فنستخدم التعبير "fetch first N rows only" الذي يشير إلى عدد السجلات التي نريد الحصول عليها وذلك ابتداءاً من السجل الأول.

وتعتمد الصيغة في قواعد بيانات Oracle على الكلمة المفتاحية rowNum وهي عبارة عن قيمة تلقائية يولدها نظام Oracle وتحدد ترتيب السجلات التي نريد الحصول عليها ضمن مجموعة السجلات التي تعيدها تعليمة Select.

مسألة:

تحتفظ شركة متعددة الفروع بسجل خاص بالمهام والأعمال التي نفذها موظفو الشركة والدخل الناتج عن كل مهمة من هذه المهام.
بفرض أن جدول المعلومات المتوفرة هو جدول المهام tasks التالي:

taskID	taskDate	taskIncom	taskHandler
1	27/1/2003	10000	adel
...	...	...	...

حيث يعبر الحقل taskID عن رقم المهمة
ويعبر الحقل taskDate عن تاريخ المهمة
ويعبر الحقل taskIncom عن الدخل الناتج عن المهمة
ويعبر الحقل taskHandler عن اسم الموظف الذي نفذ المهمة

والمطلوب الحصول على جدول يحدد:

- عدد الأعمال التي قام بها كل موظف.
- الربح الكلي الذي حصلت عليه الشركة من عمل هذا الموظف وذلك خلال العام 2004.

مع ترتيب الجدول على نحو يظهر فيه الموظف الذي نفذ المهمة التي حققت أعلى دخل للمؤسسة، في أول سجل في الجدول.

الحل:

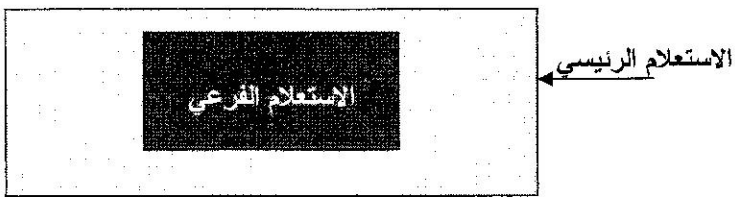
سيظهر في جدول النتيجة معلومات حول اسم الموظف، وعدد المهمات التي أوكلت له، ومجموع الدخول التي حصلت عليها الشركة من عمل الموظف، وقيمة الدخل الأعظمي الذي حصل عليه الموظف من مهامه، مع ترتيب قيم الدخول الأعظمية ترتيباً تنازلياً، فيكون أول اسم في الجدول هو اسم الموظف الذي نفذ المهمة التي حققت أعلى دخل للمؤسسة.

يمكن الحصول على الحل المطلوب بالصيغة التالية:

```
Select taskHandler, count(taskID), sum(taskIncom), max(taskIncom)
From tasks
Group by taskHandler
Where taskDate between#01/01/2004# and#31/12/2004 #
Order by max(taskIncom) DESC
```

الاستعلامات الفرعية

الاستعلام الفرعي هو أي استعلام يتم تضمينه في استعلام آخر.



هناك نوعان أساسيان من الاستعلامات الفرعية

انتبه:

- يمكن للاستعلام الفرعي أن يحتوي استعلاماً فرعياً آخر.
- ليس من الضرورة أن تَستخدم الاستعلامات الفرعية نفس الجداول التي تستخدمها الاستعلامات الرئيسية.
- لا تدعم قواعد بيانات MySQL الاستعلامات الفرعية، بل تستعيض عنها بـ . Join التي سنأتي على ذكرها لاحقاً.

الاستعلام الفرعي هو أي استعلام يتم تضمينه في استعلام آخر.

هناك نوعان أساسيان من الاستعلامات الفرعية:

1- الاستعلامات الفرعية المرتبطة
2- الاستعلامات الفرعية غير المرتبطة

الاستعلامات الفرعية

1- **الاستعلامات الفرعية المرتبطة باستعلام رئيسي:** هي الاستعلامات الفرعية التي تعتمد في عملها على بيانات من استعلامات رئيسية حاوية لها.

في هذا النوع من الاستعلامات الفرعية، يتم تكرار عملية تنفيذ الاستعلام بعدد مرات مساوٍ لعدد السجلات التي يُعيدها الاستعلام الرئيسي.

مثال:

إذا كان لدينا الجدول Customers الحاوي على معلومات الزبائن، والجدول Orders الحاوي على معلومات الطلبيات. لإظهار قائمة باسم كل زبون (customerName) وعدد الطلبيات لكل زبون نكتب الاستعلام:

```
Select customerName,(select count(*) from Orders
where Orders.customerID=Customers.customerID) from Customers;
```

نلاحظ في المثال السابق بأنه سيتم تكرار تنفيذ الاستعلام الفرعي (الظاهر بين القوسين) بعدد سجلات الزبائن وأن الاستعلام سيعيد عدد الطلبيات لكل زبون.

2- **الاستعلامات الفرعية المستقلة:** وهي الاستعلامات الفرعية التي تكون مستقلة تماماً عن الاستعلامات الرئيسية الحاوية لها، أي أن الاستعلام الفرعي سيُنَفَّذ بشكل كامل ويُمرر القيمة أو مجموعة القيم الناتجة إلى الاستعلام الرئيسي.

مثال:

إذا كان لدينا جدول Students يحتوي أسماء الطلاب (studentName) وأرقامهم التسلسلية (studentID). وجدول آخر Grades يحتوي علامات الطلاب (grade) وأرقامهم التسلسلية (studentID). لإعادة لائحة بأسماء الطلاب الناجحين فقط نكتب الاستعلام:

```
Select studentName from Students where Students.studentID in (select
Grades.studentID from Grades where Grades.grade>=50);
```

(اعتُبرت علامة النجاح 50).

نلاحظ في الاستعلام السابق أنه سيتم تنفيذ الاستعلام الفرعي (بين القوسين) مرة واحدة فقط، وبصورة مستقلة عن عدد السجلات في الجدول Students.

انتبه:

- تحتاج الاستعلامات الفرعية المرتبطة باستعلام رئيسي للكثير من الوقت والمعالجة بالمقارنة مع الاستعلامات غير المرتبطة باستعلامها الرئيسي.
- يتطلب العمل بالاستعلامات الفرعية أحياناً استخدام حقول من أكثر من جدول، فاحرص على عدم استخدام أسماء حقول طموحة بل استخدم الصيغة Table_Name.Field_Name.

1- **الاستعلامات الفرعية المرتبطة باستعلام رئيسي:** هي الاستعلامات الفرعية التي تعتمد في عملها على بيانات من استعلامات رئيسية حاوية لها.

في هذا النوع من الاستعلامات الفرعية، يتم تكرار عملية تنفيذ الاستعلام بعدد مرات مساوٍ لعدد السجلات التي يُعيدها الاستعلام الرئيسي.

2- **الاستعلامات الفرعية المستقلة:** وهي الاستعلامات الفرعية التي تكون مستقلة تماماً عن الاستعلامات الرئيسية الحاوية لها، أي أن الاستعلام الفرعي سيُنَفَّذ بشكل كامل ويُمرر القيمة أو مجموعة القيم الناتجة إلى الاستعلام الرئيسي.

استعمال الاستعلام الفرعي كعمود من أعمدة الاستعلام الرئيسي

يأخذ هذا الاستعلام الصيغة:

```
Select columnA, (subquery) as columnB from Table_Name;
```

في الصيغة السابقة سوف يتم تنفيذ الاستعلام Subquery على كل سجل يعيده الاستعلام الرئيسي. يفيد هذا النوع من الاستعلامات الفرعية في توليد علاقة بين جدول وآخر.

وفقاً لهذه الصيغة يجب ألا يعيد الاستعلام الفرعي أكثر من قيمة واحدة لكل سجل في سجلات الاستعلام الرئيسي.

مثال:

نفرض أن لدينا الجدول Accounts الحاوي على الأرقام التسلسلية للحسابات المصرفية accountID وقيم أرصدة هذه الحسابات accountBalance. ولنفرض أن لدينا الجدول Clients الحاوي على أسماء أصحاب الحسابات clientName، ورقم الحساب لكل زبون accountID. لإظهار قائمة بأرقام الحسابات وأرصدتها وأسماء أصحابها نستخدم الصيغة:

```
Select Accounts.accountID, (select clientName from Clients where
Clients.accountID = Accounts.accountID)as myClientName,
Accounts.accountBalance
from Accounts;
```

يُعتبر المثال السابق عيّنة من الاستعلامات الفرعية المرتبطة باستعلام رئيسي نظراً لكون الاستعلام الفرعي (الظاهر بين القوسين) لن يعمل وحيداً، بل يحتاج إلى معلومات من الاستعلام الرئيسي وهي قيم Accounts.accountID.

انتبه:

- لاحظ أننا استخدمنا في بعض من الصيغ السابقة أسماء الجداول مع أسماء الحقول مفصولة بنقاط '.' وذلك لمنع حدوث خلط في تحديد تبعية أحد الحقول لأي من الجداول.
- لكي تعمل الصيغة في المثال السابق دون أخطاء، يُشترط أن يعيد الاستعلام الفرعي قيمة واحدة لكل سجل من سجلات الاستعلام الرئيسي.

يُعتبر استعمال الاستعلام الفرعي كعمود من أعمدة الاستعلام الرئيسي، أحد التركيبات المُستخدمة بكثرة في الاستعلامات الفرعية.

استعمال الاستعلامات الفرعية ضمن شرط Where

يأخذ هذا الاستعلام الصيغة:

```
Select columnA, columnB from Table_Name where columnB=(Subquery);
```

في هذه الصيغة أيضاً يجب أن يعيد Subquery قيمة وحيدة.

نلاحظ أن نتيجة الاستعلام الفرعي دخلت كجزء من الشرط في تعبير Where.

مثال:

لدينا الجدول Tickets الحاوي على المخالفات المرورية للعربات والجدول Owners الحاوي على أرقام لوحات العربات carNumber وأسماء أصحابها ownerName. لإظهار اسم صاحب العربة ownerName التي ارتكبت المخالفة رقم 1234، نكتب الصيغة:

```
Select ownerName, Owners.carNumber from Owners
where Owners.carNumber=(select Tickets.carNumber from tickets
where ticketNumber=1234);
```

تدخل نتيجة الاستعلام الفرعي ضمن الشرط في تعبير Where شرط أن يعيد الاستعلام الفرعي قيمة وحيدة.

الاستعلامات الفرعية التي تعيد مجموعة من القيم

كي لا يفشل الاستعلام، اشترطنا في دراستنا للاستعلامات الفرعية حتى الآن، أن يعيد الاستعلام الفرعي قيمة وحيدة.

لكي نتمكن من استخدام الاستعلامات الفرعية التي تعيد أكثر من قيمة نستخدم الصيغة التالية:

```
Select columnA, columnB from Table_Name where columnC IN(Subquery);
```

نلاحظ أننا استخدمنا هنا التعبير IN.

لكن يُشترط في الاستعلام الفرعي من هذا النوع (أي الاستعلام المُمثل بـ . Subquery)، أن يعيد قيم عمود واحد فقط أي يكون من الشكل:

```
Select column1 from Table1;
```

مثال:

لنعد إلى مثال المخالفات المرورية، للحصول على قائمة بأسماء الأشخاص الذين لديهم مخالفات يكون شكل الاستعلام:

```
Select ownerName from Owners where Owners.carNumber IN
(select Distinct Tickets.carNumber from Tickets);
```

استخدمنا هنا الاستعلام الفرعي لإعادة أرقام السيارات التي لها مخالفات في سجل المخالفات مع التخلص من التكرار باستخدام Distinct.

اشترطنا حتى الآن في الاستعلامات الفرعية، سواء تلك المستخدمة كأعمدة في الاستعلام الرئيسي أو تلك المستخدمة ضمن شرط التعبير Where، أن يعيد الاستعلام الفرعي قيمة وحيدة كي لا يفشل الاستعلام.

لكي نتمكن من استخدام الاستعلامات الفرعية التي تعيد أكثر من قيمة نستخدم التعبير IN، لكن يجب أن نراعي أن يعيد الاستعلام الفرعي المستخدم مع التعبير IN حقلاً واحداً فقط (نقصد هنا حقلاً وحداً وليس قيمة واحدة).

استخدام تعبيرات Exists وAll وAny مع الاستعلامات الفرعية

التعبير Exists:

يُستخدم التعبير Exists للتحقق من إعادة الاستعلام الفرعي الذي يليه لأي سجل. ويأخذ التعبير كاملاً القيمة True في حال أرجع الاستعلام الفرعي سجلاً أو أكثر، والقيمة False إذا لم يُرجع الاستعلام الفرعي أي سجل.

يُكتب التعبير Exists وفق الصيغة:

```
Select columnA, columnB from Table_Name where Exists (Subquery);
```

مثال:

إذا كان لدينا جدول Orders بجميع الطلبيات الخاصة بشركة ، والذي يحتوي رقم الطلبية orderID ونوعها orderType ورقم الزبون clientID الذي طلبها. وجدول Clients الحاوي على أسماء الزبائن وأرقامهم clientID. لإظهار قائمة بأسماء الزبائن الذين قاموا سابقاً بإرسال طلبية من نوع "تجهيزات كهربائية"، نستخدم الصيغة:

```
Select Clients.clientName from Clients where Exists

( select * from Orders
where Orders.clientID = Clients.clientID
and
orderType='تجهيزات كهربائية');
```

استخدام تعبيرات Exists وAll وAny مع الاستعلامات الفرعية

التعبير All:

يُستخدم التعبير All للتحقق من كون جميع القيم المعادة من استعلام فرعي، تحقق شرطاً ما في تعبير Where التابع للاستعلام الرئيسي.

يُمكن أن يُكتب التعبير All وفق الصيغة:

```
Select columnA from TableA
where columnA > All from (select columnB from TableB);
```

حيث يتم هنا اختيار السجلات من الجدول A، بحيث تكون كل قيمة من قيم الحقل columnA أكبر من جميع القيم التي يعيدها الاستعلام الفرعي، أي من جميع قيم columnB في الجدول TableB.

مثال:

تريد الجهات المعنية مقارنة الأرقام Time التي سجلها العداؤون في بطولة معينة والمحفوظة مع أسمائهم Name في جدول currentRecords، بالأرقام المسجلة في جدول أرقامهم القديمة oldRecords وذلك لاستخلاص قائمة بأسماء العدائين الذين حطمت أرقامهم جميع الأرقام القديمة oldTime.

للحصول على هذه القائمة، نكتب الصيغة:

```
Select Name from currentRecords
where time < All (select oldTime from oldRecords);
```

استخدام تعبيرات Exists و All و Any مع الاستعلامات الفرعية

التعبير ANY:

يُستخدم التعبير ANY للتحقق من كون قيمة أو أكثر من القيم المعادة من استعلام فرعي، تحقق شرطاً واحداً على الأقل من شروط تعبير Where الخاص بالإستعلام الرئيسي.

يُمكن أن يُكتب التعبير Any وفق الصيغة:

```
Select columnA from TableA
where columnA > ANY from (select columnB from TableB);
```

سيتم هنا اختيار السجلات من الجدول A، حيث تكون كل قيمة من قيم الحقل columnA أكبر من قيمة واحدة على الأقل من القيم المعادة من الاستعلام الفرعي، أي من قيمة واحدة على الأقل من قيم columnB في الجدول TableB.

مثال:

تريد الجهات المعنية مقارنة الأرقام Time التي سجلها العداؤون في بطولة معينة والمحفوظة مع أسمائهم Name في جدول currentRecords، بالأرقام القياسية العالمية المسجلة في جدول bestRecords وذلك لاستخلاص قائمة بأسماء العدائين الذين حطمت أرقامهم أحد الأرقام القياسية bestTime.

للحصول على هذه القائمة نكتب الصيغة:

```
Select Name from currentRecords
where time < ANY (select bestTime from bestRecords);
```

دمج البيانات من استعلامين باستخدام التعابير Union و Intersect و Except

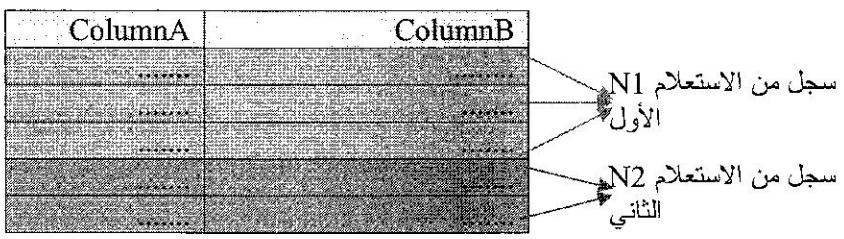
تُستخدم تعابير Union و Intersect و Except(Minus) في دمج استعلامين. وتختلف نتائج هذا الدمج بحسب التعبير المستخدم.

تُستخدم التعابير السابقة وفق الصيغة:

```
select columnA,columnB from tableA
Operator
Select columnC,columnD from tableB;
```

حيث يعبر Operator عن أحد هذه التعابير.

تكون هنا قائمة القيم المعادة على الشكل:



حيث N1 و N2 أكبر أو تساوي الصفر.

الاستعلام عن أكثر من جدول واستخدام الربط الداخلي والخارجي

سبق ورأينا في الجلسات السابقة، أن الاستعلام عن أكثر من جدول باستخدام الاستعلامات الفرعية قد وفر قدرات جيدة على معالجة البيانات. ولكن هذه القدرات قد لا تمكننا دائماً من الحصول على كل النتائج التي نحتاجها. كما أن الصيغة قد تصبح صعبة الفهم بعض الشيء وقد تؤدي إلى انخفاض مستوى الأداء أحياناً.

توفر SQL إمكانية الاستعلام عن جداول متعددة في وقت واحد باستخدام صيغة أبسط ندعوها الربط.

لا تستطيع الصيغة الجديدة استبدال كل التقنيات التي تؤمنها الاستعلامات الفرعية، ولكنها تمثل الحل الأمثل في بعض الحالات، وخاصةً في الاستعلامات التي تربط بين سجلات من جداول مختلفة.

نبدأ فيما يلي بعرض حالات الربط البسيطة لنصل بعد ذلك إلى الحالات الأكثر تعقيداً:

- الربط البسيط
- الربط بالتساوي
- الربط بعدم المساواة
- الربط الخارجي

الربط البسيط

- الربط البسيط هو استعلام عن أكثر من جدول في صيغة واحدة دون استعمال أي شرط.
- إن أبسط صيغة للتعبير عن الربط البسيط هي:

```
Select Table1.Column1, Table2.Column2 from Table1, Table2;
```

إذا فكرنا في استثمار هذه الصيغة لإعادة إسم، وصف كل طالب من الجدولين Names, Classes أول ما سيحضر إلى أذهاننا هو كتابة الصيغة:

```
Select class, Name from Classes, Names;
```

يستخدم هذا الاستعلام الربط البسيط ولكنه للأسف لن يعيد النتيجة التي نحتاجها.

- إن آلية عمل الربط البسيط في حال عدم وجود أي شرط مرافق هي عبارة عن عملية جداء ديكارتي لقيم الحقول المحددة من الجدولين. فإذا كانت قيم الحقل المطلوب من الجدول الأول هي {A, B, C} وقيم الحقل المطلوب من الجدول الثاني هي {D, E, F} فسيكون عدد القيم المعادة 9 قيم هي التالية:
{(A,D), (A,E), (A,F), (B,D), (B,E), (B,F), (C,D), (C,E), (C,F)}

بالنتيجة، إذا كان لدينا 100 سجل في كل من الجدولين المربوطين ربطاً بسيطاً سنحصل على 10000 سجل يعيدها استعلام الربط البسيط.

- يسمى استعلام الربط البسيط أيضاً بالربط المتصالب. ويمكن التعبير عن نفس صيغة الربط السابقة، بالصيغة:

```
Select Table1.Column1, Table2.Column2 from Table1 Cross Join Table2;
```

مثال:

يريد كيميائي اختبار تأثير مجموعة من المواد الكيميائية على مجموعة من المنتجات التي تصنعها الشركة التي يعمل لديها. فإذا كانت أسماء المنتجات (productName) مدرجة في جدول Products وأسماء المواد المراد اختبار تأثيرها (materialName) مدرجة في جدول ChemicalEffects، المطلوب مساعدة هذا الكيميائي في تجهيز لائحة الاختبارات.

الحل:

```
Select productName, materialName from Products, ChemicalEffects;
```

أو

```
Select productName, materialName from Products Cross Join
ChemicalEffects;
```

انتبه:

لا تدعم قواعد بيانات DB2 التعبير Cross Join

الربط البسيط

الربط البسيط هو استعلام عن أكثر من جدول في صيغة واحدة دون استعمال أي شرط.

إن آلية عمل الربط البسيط في حال عدم وجود أي شرط مرافق هي عبارة عن عملية جداء ديكارتي لقيم الحقول المحددة من الجداول المرتبطة. فإذا كانت قيم الحقل المطلوب من الجدول الأول هي {A, B, C} وقيم الحقل المطلوب من الجدول الثاني هي {D, E, F} فسيكون عدد القيم المعادة 9 قيم وهي التالية:
{(A,D), (A,E), (A,F), (B,D), (B,E), (B,F), (C,D), (C,E), (C,F)}

بالنتيجة، إذا كان لدينا 100 سجل في كل من الجدولين المربوطين ربطاً بسيطاً سنحصل على 10000 سجل يعيدها استعلام الربط البسيط.

يسمى استعلام الربط البسيط أيضاً بالربط المتصالب.

الربط بالتساوي

- يُعرّف الربط بالتساوي على أنه الربط البسيط بين سجلات جدول أول، وسجلات جدول ثان اعتماداً على مساواة بين قيمة حقل في سجل من الجدول الأول وقيمة حقل في سجل من الجدول الثاني.

- يُعبّر عن الربط بالتساوي بالصيغة:

```
Select Table1.Column1, Table1.Column2, Table2.Column3
From Table1, Table2 where Table1.Column1 = Table2.Column2;
```

لاحظ أننا قد استخدمنا مساواة بين الحقل الأول من الجدول الأول Table1.Column1 والحقل الثاني من الجدول الثاني Table2.Column2 مما يعني أن عملية الربط لاتستخدم بالضرورة نفس الحقول التي يجب أن يعيدها الاستعلام، أي:
Table1.Column1, Table1.Column2, Table2.Column3

- يمكن الوصول إلى نفس النتيجة السابقة باستخدام الصيغة:

```
Select Table1.Column1, Table1.Column2, Table2.Column3
From Table1
Join Table2
ON Table1.Column1 = Table2.Column2;
```

مثال:
إذا كان لدينا جدول Names يحوي أسماء جميع الأشخاص (name) المقيمين في قرية مع أرقامهم التأمينية(INumber) والجدول Addresses الذي يتضمن عناوين المنازل (address) في تلك القرية مع الرقم التأميني (INumber) للمقيم في المنزل، وأردنا كتابة الاستعلام الذي يعيد قائمة بأسماء الأشخاص وعناوينهم، يمكننا استخدام الصيغة:

```
Select Names.name, Addresses.address from Names, Addresses
Where Names.INumber = Addresses.INumber;
```

أو

```
Select Names.name, Addresses.address
From Names
Join Addresses
ON Names.INumber = Addresses.INumber;
```

استخدام الربط بالتساوي مع عدة جداول

لا تتوقف إمكانيات الربط بالتساوي عند حدود الربط بين جدولين فقط بل يمكن أن تتعداها إلى مجموعة من الجداول.
نستخدم لإتمام هذه العملية الصيغة:

```
Select Table1.Column1, Table2.Column2, Table3.Column4
From Table1 Join Table2
ON Table1.Column1 = Table2.Column2
Join Table3
ON Table1.Column3 = Table3.Column4;
```

هنا يتم الربط بين الجداول اعتماداً على أسماء الحقول المتساوية المحددة بعد التعبير ON. ففي حالة الصيغة السابقة ربطنا الجدول Table1 مع Table2 معتمدين على تساوي قيم العمود Column1 من الجدول Table1 مع قيم العمود Column2 من الجدول Table2، وربطنا الجدول Table3 بالجدول Table1 معتمدين على تساوي قيم العمود Column3 من الجدول Table1 مع قيم العمود Column4 من الجدول Table3.

مثال:

لدينا الجدول Customers الحاوي على رمز الزبون customerID، اسمه customerName والجدول CreditCards الحاوي على رقم بطاقة الزبون الائتمانية cardNumber وعلى رقمه customerID، والجدول Addresses الحاوي على عناوين الزبائن وكان المطلوب إعادة قائمة برقم الزبون وأسمه، ورقم بطاقته الإئتمانية، والبلد الذي يقيم فيه country. للحصول على المطلوب، نكتب الاستعلام:

```
Select Customers.customerID, Customers.customerName,
CreditCards.cardNumber, Addresses.country
From Customers
Join CreditCards
ON Customers.customerID = CreditCards.customerID
Join Addresses
ON Customers.customerID = Addresses.customerID;
```

استخدام الربط بالتساوي مع عدة جداول (تتمة)

قد لا تعمل الصيغة السابقة على نظام إدارة قواعد البيانات MS Access لذلك قد يفضل البعض استخدام تقنية الربط المتساوي المتداخل أي استخدام الصيغة:

```
Select Table1.Column1, Table2.Column2, Table3.Column4
From Table2
Inner Join
(Table3 Inner Join Table1 ON Table3.Column4 = Table1.Column1)
ON Table2.Column2 = Table1.Column1;
```

تشبه هذه الصيغة تلك التي استخدمناها في الشريحة السابقة ولكن MS Access يواجه صعوبة في ترجمة عمليات الربط بصورة مباشرة، لذلك لجأنا إلى توليد بنى ربط متداخلة مجمعة بأقواس. فكما نلاحظ قمنا بربط الجدول Table3 مع الجدول Table1 بتساوي الحقلين Column4 من الجدول Table3 مع الحقل Column1 من الجدول Table1، ثم تعاملنا مع التعبير كاملاً كطرف في عملية الربط مع الجدول Table2 بتساوي الحقل Column2 من الجدول Table2 مع الحقل Column1 من الجدول Table1.

مثال:

إذا كان لدينا مخزن ألبسة يعتمد قاعدة بيانات تحتوي ثلاثة جداول: الأول جدول الأقسام Sectors (ولادي- نسائي- سن محير...)، الذي يحتوي على رمز القسم sectorID وإسم القسم sectorName. والثاني جدول الفصول Seasons الذي يحتوي رمز الفصول والسنوات (خريف 2004 – شتاء2003...) seasonID، وأسمائها seasonInfo. والثالث جدول المنتجات Products الذي يحتوي الأرقام التسلسلية للقطع productID، وتوصيفها productDescription، وأسعارها productPrice، ورمز القسم seasonID، ورمز القسم sectorID الذي تتبع له.

والمطلوب إظهار قائمة بتوصيف القطع وأسعارها، والأقسام التي تتبع لها والقسم الخاص بها.

الحل:

```
Select productDescription, productPrice, seasonName, sectorName
From Sectors
Inner Join
(Seasons Inner Join Products ON Seasons.seasonID = Products.seasonID)
ON Sectors.sectorID = Products.sectorID;
```

قد لا تعمل الصيغة السابقة على نظام إدارة قواعد البيانات MS Access لذلك قد يفضل البعض استخدام تقنية الربط المتساوي المتداخل. إذ يواجه MS Access صعوبة في ترجمة عمليات الربط بصورة مباشرة، لذلك نلجيء عادةً إلى توليد بنى ربط متداخلة مجمعة بأقواس.

الربط باللامساواة

يعتمد الربط بالمساواة على استخدام المساواة في شرط التعبير Where ولكن هذا لا يعني أننا لا نستطيع استخدام عمليات المقارنة الأخرى (أكبر، أصغر، وغيرها) كما في الصيغة التالية:

```
Select Table1.Column1, Table2.Column2
From Table1, Table2
Where Table1.Column1 < Table2.Column2;
```

مثال:

لدينا جدول Stores الخاص بمعمل أقمشة، والذي يحتوي يحتوي أسماء المخازن storeName وأرقامها storeID وجدول آخر Occupation يحتوي معلومات المشغولية للمخازن التي تتضمن رقم المخزن stroreID وكمية البضاعة في المخزن quantity ونوعها Type.

لإعادة قائمة بالمخازن الفارغة غير المشغولة نكتب الاستعلام:

```
Select  Stores.storeID, Stores.storeName from Stores, Occupation
Where Stores.storeID <> Occupation.storeID
```

الربط الخارجي

لنتمكن من فهم الربط الخارجي يجب أن نعود إلى فكرة الربط بالتساوي حيث استخدمنا التعبير Inner Join.

في حالة Inner Join، كانت السجلات التي أرجعها الاستعلام، هي السجلات التي تحقق شرط الربط الذي يظهر بعد تعبير ON، حيث تم اسقاط السجلات غير المتطابقة من جدول النتائج. أما في حالة الربط الخارجي Outer Join فلا يتم إسقاط السجلات غير المتطابقة.

للربط الخارجي ثلاثة أنواع: Left, Right, Full.

- لأخذ جميع السجلات من الجدول الأول Table1 وفقط السجلات من الجدول الثاني Table2 التي تتطابق فيها قيمة الحقل Column1 من الجدول Table1 مع قيمة الحقل Column2 من الجدول الثاني Table2، نكتب الصيغة:

```
Select * from Table1 LEFT OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

- لأخذ جميع السجلات من الجدول الثاني Table2 وفقط السجلات من الجدول الأول Table1 التي تتطابق فيها قيمة الحقل Column1 من الجدول Table1 مع قيمة الحقل Column2 من الجدول الثاني Table2، نكتب الصيغة:

```
Select * from Table1 RIGHT OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

- لأخذ جميع السجلات من الجدول الثاني Table2 وجميع السجلات من الجدول الأول Table1 بحيث تتوضع السجلات التي تتطابق فيها قيمة الحقل Column1 من الجدول Table1 مع قيمة الحقل Column2 من الجدول الثاني Table2 في نفس السجل من جدول القيم المعادة، نكتب الصيغة:

```
Select * from Table1 FULL OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

ينتج عن عمليات الربط الخارجي، في الحالة العامة، سجلات تحتوي في حقول معينة القيمة NULL بسبب اختلاف عدد السجلات التي نريد ربطها، وهذا ما سنوضحه بالتفصيل لاحقاً مع مثال مناسب لكل نوع من أنواع الربط الخارجي.

الربط الخارجي

لنتمكن من فهم الربط الخارجي يجب أن نعود إلى فكرة الربط بالتساوي حيث استخدمنا التعبير Inner Join.

في حالة Inner Join، كانت السجلات التي أرجعها الاستعلام، هي السجلات التي تحقق شرط الربط الذي يظهر بعد تعبير ON، حيث تم اسقاط السجلات غير المتطابقة من جدول النتائج. أما في حالة الربط الخارجي Outer Join فلا يتم إسقاط السجلات غير المتطابقة.

للربط الخارجي ثلاثة أنواع: Left وRight وFull. ينتج عن عمليات الربط الخارجي، في الحالة العامة، سجلات تحتوي في حقول

معينة القيمة NULL بسبب اختلاف عدد السجلات التي نريد ربطها، وهذا ما سنوضحه بالتفصيل لاحقاً مع مثال مناسب لكل نوع من أنواع الربط الخارجي.

Left Join

لنستخدم صيغة الربط الخارجي التالية على الحقلين Column1 من الجدول الأول Table1، وColumn2 من الجدول الثاني Table2:

```
Select * from Table1 LEFT OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

يُرجع هذا النوع من أنواع الربط الخارجي:

- جميع القيم الخاصة بالحقل التابع للجدول الأول،
- جميع القيم المطابقة لها والخاصة بالحقل التابع للجدول الثاني، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الثاني وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الأول مطابقة لقيمة حقل الجدول الثاني.

كما يكون عدد السجلات المعادة مطابقاً لعدد السجلات التي يعيدها الاستعلام عن قيم حقل الجدول الأول.

مثال:

لنستخدم الصيغة:

```
Select * from Table1 LEFT OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

إذا كان لدينا جدول Table1 يحتوي في الحقل Column1 القيم {1, 5, 8, 3} والجدول Table2 الذي يحتوي في الحقل Column2 القيم {6, 5, 7, 9} فإن تطبيقها سيولد النتائج التالية:

Column1	Column2
1	Null
5	5
8	Null
3	Null

نلاحظ أنه تم إدراج القيمة Null لقيم الحقل Column2 في السجلات التي لم تكن فيها قيمة الحقل Column2 مطابقة لقيمة الحقل Column1.

كما يكون عدد السجلات المعادة مطابقاً لعدد السجلات التي يعيدها الاستعلام عن قيم الحقل Column1 من الجدول Table1.

عندما نطبق هذا النوع من أنواع الربط الخارجي على حقلين من جدولين مختلفين، نحصل على:

- جميع القيم الخاصة بالحقل التابع للجدول الأول،
- جميع القيم المطابقة لها والخاصة بالحقل التابع للجدول الثاني، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الثاني وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الأول مطابقة لقيمة حقل الجدول الثاني.

كما يكون عدد السجلات المعادة مطابقاً لعدد السجلات التي يعيدها الاستعلام عن قيم حقل الجدول الأول.

Right Join

لنستخدم صيغة الربط الخارجي التالية على الحقلين Column1 من الجدول الأول Table1، وColumn2 من الجدول الثاني Table2:

```
Select * from Table1 RIGHT OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

يُرجع هذا النوع من أنواع الربط الخارجي:

- جميع القيم الخاصة بالحقل التابع للجدول الثاني،
- جميع القيم المطابقة لها والخاصة بالحقل التابع للجدول الأول، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الأول وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الثاني مطابقة لقيمة حقل الجدول الأول.

كما يكون عدد السجلات المعادة مطابقاً لعدد السجلات التي يعيدها الاستعلام عن قيم حقل الجدول الثاني.

مثال:

لنستخدم الصيغة:

```
Select * from Table1 RIGHT OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

إذا كان لدينا جدول Table1 يحتوي في الحقل Column1 القيم {1, 5, 8, 3} والجدول Table2 الذي يحتوي في الحقل Column2 القيم {6, 5, 7, 9} فإن تطبيقها سيولد النتائج التالية:

Column1	Column2
Null	6
5	5
Null	7
Null	9

نلاحظ أنه تم إدراج القيمة Null لقيم الحقل Column2 في السجلات التي لم تكن فيها قيمة الحقل Column2 مطابقة لقيمة الحقل Column1.

كما يكون عدد السجلات المعادة مطابقاً لعدد السجلات التي يعيدها الاستعلام عن قيم الحقل Column2 من الجدول Table2.

عندما نطبق هذا النوع من أنواع الربط الخارجي على حقلين من جدولين مختلفين، نحصل على:

- جميع القيم الخاصة بالحقل التابع للجدول الثاني،
- جميع القيم المطابقة لها والخاصة بالحقل التابع للجدول الأول، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الأول وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الثاني مطابقة لقيمة حقل الجدول الأول.

كما يكون عدد السجلات المعادة مطابقاً لعدد السجلات التي يعيدها الاستعلام عن قيم حقل الجدول الثاني.

Full Join

لنستخدم صيغة الربط الخارجي التالية على الحقلين Column1 من الجدول الأول Table1، و Column2 من الجدول الثاني Table2:

```
Select * from Table1 FULL OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

يُرجع هذا النوع من أنواع الربط الخارجي:

- جميع القيم الخاصة بالحقل التابع للجدول الأول، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الأول وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الثاني مطابقة لقيمة حقل الجدول الأول.
- جميع القيم المطابقة لها والخاصة بالحقل التابع للجدول الثاني، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الثاني وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الأول مطابقة لقيمة حقل الجدول الثاني.

مثال:

لنستخدم الصيغة:

```
Select * from Table1 FULL OUTER JOIN Table2
ON Table1.Column1 = Table2.Column2;
```

إذا كان لدينا جدول Table1 يحتوي في الحقل Column1 القيم {1, 5, 8, 3} والجدول Table2 الذي يحتوي في الحقل Column2 القيم {6, 5, 7, 9} فإن تطبيقها سيولد النتائج التالية:

Column1	Column2
1	Null
5	5
8	Null
3	Null
Null	6
Null	7
Null	9

نلاحظ أنه تم إدراج القيمة Null لقيم الحقل Column1 في السجلات التي لم تكن فيها قيمة الحقل Column1 مطابقة لقيمة الحقل Column2 وإدراج القيمة Null لقيم الحقل Column2 التي لم تتطابق فيها قيمة Column1 مع Column2.

Full Join

عندما نطبق هذا النوع من أنواع الربط الخارجي على حقلين من جدولين مختلفين، نحصل على:

- جميع القيم الخاصة بالحقل التابع للجدول الأول، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الأول وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الثاني مطابقة لقيمة حقل الجدول الأول.
- جميع القيم المطابقة لها والخاصة بالحقل التابع للجدول الثاني، حيث يتم إدراج القيمة Null في قيم الحقل التابع للجدول الثاني وذلك في السجلات التي لا تكون فيها قيمة حقل الجدول الأول مطابقة لقيمة حقل الجدول الثاني.

استخدام Natural Join

يقوم التعبير Natural Join بعملية ربط اعتماداً على الحقول ذات الأسماء المُشتركة بين الجدولين والتي تحتوي على قيم متطابقة. يأخذ التعبير الصيغة:

```
Select Table1.Column1, Table2.Column1 from Table1 Natural Join
Table2;
```

نلاحظ هنا أننا لم نستخدم التعبير ON وشرط التساوي لأنهما مُستخدمان ضمنياً في التعبير Natural Join حيث يتم تحقيق الربط عن طريق استخدام الحقل الذي يشترك الجدولان بإسمه.

مثال:

ليكن لدينا الجدول Pictures الحاوي على الأرقام التسلسلية للصور (pictureID) وعام تصويرها بالإضافة إلى شرح عن الصورة (pictureDescription). وليكن لدينا الجدول Names الحاوي على أسماء أصحاب الصور (clientName) والأرقام التسلسلية للصور (pictureID). فإذا كان المطلوب إعادة قائمة باسم كل شخص ووصف الصورة الخاصة به يمكننا كتابة الصيغة:

```
Select clientName, pictureDescription from Names Natural Join
Pictures;
```

نلاحظ في الصيغة السابقة أن الربط قد تم اعتماداً على الحقل الذي يشترك الجدولان بإسمه وهو pictureID.

استخدام Natural Join

يقوم التعبير Natural Join بعملية ربط لجميع الحقول ذات الأسماء المُشتركة والتي تحتوي على قيم متطابقة من الجدولين.
في هذا الاستعلام لا نستخدم التعبير ON وشرط التساوي لأنهما مُستخدمان ضمنياً في التعبير
Natural Join حيث يتم تحقيق الربط عن طريق استخدام الحقل الذي يشترك الجدولان بإسمه.

الربط باستخدام التعبير Using

يستخدم التعبير Using في حال كان اسم الحقل نعتمد على قيمه في عملية الربط متشابهاً في الجدولين.

يأخذ التعبير Using الصيغة:

```
Select Table1.Column2, Table2.Column3
From Table1 Join Table2 Using (Column1)
```

اعتبرنا هنا أن الربط يتم اعتماداً على قيم الحقل Column1 من الجدول الأول و Column1 من الجدول الثاني.

انتبه:

إن التعبيرين Using و Natural Join مدعومان من قواعد بيانات Oracle في نسختها 9I

يستخدم التعبير Using في حال كان اسم الحقل نعتمد على قيمه في عملية الربط متشابهاً في الجدولين.

التعامل مع أغراض قواعد البيانات

ركّزنا حتى الآن، وفي كل المواضيع التي تطرقنا إليها، على الاستعلام عن جداول في قاعدة بيانات معتبرين أن تلك قواعد البيانات منشأة مسبقاً والجداول موجودة. إلا أننا لم نتطرق إلى كيفية إنشاء قواعد البيانات: الجداول والفهارس والقيود عليها. وهذا ما يندرج تحت عنوان التعامل مع أغراض قواعد البيانات.

سنتعرف من خلال هذه الجلسة والجلسة القادمة على:

- كيفية إنشاء وحذف قاعدة بيانات.
- كيفية إنشاء وحذف وتعديل بنية الجداول.
- كيفية إنشاء وحذف وتعديل القيود على الحقول.
- كيفية إنشاء وحذف وتعديل الجداول المؤقتة وكيفية التعامل معها.
- كيفية إنشاء وحذف وتعديل الفهارس.
- كيفية إنشاء وحذف وتعديل المعاينات.

توليد وحذف قاعدة بيانات

تختلف طريقة تخزين وإدارة البيانات بين أنظمة إدارة قواعد البيانات. إلا أننا سنركز على التقنيات الأكثر شيوعاً.

* توليد قاعدة بيانات:

في أنظمة إدارة قواعد البيانات Oracle وSQL Server وMySQL وDB2 يمكنك ببساطة استخدام الصيغة:

```
CREATE DATABASE database_name;
```

* حذف قاعدة بيانات:

يمكننا حذف قاعدة بيانات ما باستخدام الصيغة:

```
DROP DATABASE database_name;
```

إلا أن التعبير (DROP DATABASE) مدعوم فقط من قواعد بيانات SQL Server وMy SQL وDB2، أما في Oracle فيمكننا استخدام نفس التوليد (CREATE DATABASE) لحذف قاعدة بيانات، كما يمكننا أن نستخدم تطبيق Oracle Database Assistant لحذف قاعدة بيانات.

انتبه:

- إن استخدام (CREATE DATABASE) مع إسم قاعدة بيانات موجودة في Oracle يؤدي إلى إلغاء واستبدال تلك القاعدة فتوخ الحذر عند استخدام هذا التعبير.
- تمتلك أنظمة إدارة قواعد البيانات آليات خاصة لتوليد قاعدة بيانات دون الحاجة إلى كتابة الصيغة الآنفة الذكر. فمثلاً يستخدم SQL Server تطبيق Enterprise Manager لتنفيذ عملية التوليد، وتستخدم قواعد بيانات DB2 تطبيق Control Center لنفس الغرض.
- لايدعم MS Access أياً من تعابير إنشاء وحذف قواعد البيانات لذلك نلجأ لاستخدام الخيار New من القائمة File في نافذة التطبيق Access. ولحذف قاعدة بيانات Access يكفي حذف ملف قاعدة البيانات تلك والذي يحمل اللاحقة (.mdb).

تختلف طريقة تخزين وإدارة البيانات بين أنظمة إدارة قواعد البيانات. إلا أننا سنركز على التقنيات الأكثر شيوعاً.

لتوليد قاعدة بيانات في أنظمة إدارة قواعد البيانات Oracle وSQL Server وMySQL وDB2 يمكنك ببساطة استخدام التعبير (CREATE DATABASE)

أما حذف قاعدة بيانات قيتم باستخدام التعبير (DROP DATABASE)

إلا أن التعبير (DROP DATABASE) مدعوم فقط من قواعد بيانات SQL Server وMy SQL وDB2، أما في Oracle فيمكننا استخدام نفس التوليد (CREATE DATABASE) لحذف قاعدة بيانات، كما يمكننا أن نستخدم تطبيق Oracle Database Assistant لحذف قاعدة بيانات.

إنشاء وحذف الجداول

لإنشاء جدول نستخدم الصيغة:

```
CREATE TABLE table_name
(column1_name column1_data_type column1_constraints,
column2_name column2_data_type column2_constraints,...);
```

لحذف جدول نستخدم الصيغة:

```
DROP TABLE table_name;
```

لتفريغ جميع السجلات من جدول ما نستخدم الصيغة:

```
TRUNCATE TABLE table_name;
```

مثال:

إذا أردنا إنشاء قاعدة بيانات Store وأردنا إنشاء جدولين ضمنها أحدهما باسم Customers والآخر باسم Products.

الجدول Customers يحوي الرقم التسلسلي للزبون (ID) واسم الزبون (name) ورقم هاتفه (phone). والجدول Products يحتوي الرقم التسلسلي (ID) لمنتج ووصف له (description).

لإنشاء هذه البنية نكتب الصيغة:

```
CREATE DATABASE Store;

CREATE TABLE Customers
(ID Int, name varchar(50), phone varchar(15);

CREATE TABLE Products
(ID Int, description varchar(75));
```

يمكننا بعد إنشاء الجداول البدء بإدراج سجلات ضمنها بالصيغة التي تعرفنا عليها في جلسات سابقة:

```
Insert into products (ID,description) values (1,'HPComputer');
```

نلاحظ هنا استخدامنا لأنواع البيانات Int وvarchar.

انتبه:

- لا يمكن عكس عملية حذف جدول من قاعدة البيانات في MySQL أي لا يمكن استرجاع الجدول بعد حذفه بعكس SQL Server وOracle وDB2 حيث يمكن التراجع عن الحذف واستعادة البيانات في حال تم إجراء تلك العملية كجزء من مناقلة.
- يمكننا في SQL Server, Oracle, MySQL إخلاء الجدول عوضاً عن حذفه أي تفريغ جميع السجلات، وهي أيضاً عملية غير عكوسة ولكنها أسرع من استخدام الصيغة:

```
DELETE from table_name
```

تستخدم التعابير (CREATE TABLE) و(DROP TABLE) و(TRUNCATE TABLE) لتوليد وحذف وإفراغ الجداول. وتوضح الصيغ الظاهرة كيفية استخدام كل منها.

نسخ الجداول

لنسخ جدول ما أثناء توليده يمكننا استخدام الصيغة:

```
CREATE TABLE table_name_copy AS Select* from table_name;
```

تتغير الصيغة بشكل طفيف في SQL Server وتصبح على الشكل:

```
Select * Into table_name_copy from table_name;
```

وفي MySQL تصبح الصيغة:

```
CREATE TABLE table_name_copy Select* from table_name;
```

في جميع الصيغ السابقة تمثل table_name إسم الجدول اذي نود نسخ بنيته مع البيانات الموجودة فيه و table_name_copy الجدول المنسوخ عن الجدول table_name.

في حال أردنا نسخ بنية الجدول فقط بدون نسخ السجلات في الجدول يمكننا وضع شرط في تعبير Where له نتيجة False دائماً أي تصبح الصيغة على الشكل:

```
CREATE TABLE table_name_copy AS Select* from table_name
Where 1 = 0;
```

نلاحظ أن الشرط 1 = 0 لن يتحقق أبداً لذا سيتم نسخ بنية الجدول فقط ولن يوجد أي سجل سيحقق الشرط 1= 0.

أما في قواعد بيانات DB2 فسنضطر إلى إضافة التعبير DEFINITION ONLY فتصبح الصيغة من الشكل:

```
CREATE TABLE table_name_copy AS (Select* from table_name) DEFINITION
ONLY;
```

مثال:

لنسخ الجدول Logs إلى جدول آخر يسمى OldLogs نستخدم الصيغة:

```
CREATE TABLE OldLogs AS Select * from Logs;
```

لنسخ جدول ما أثناء توليده يمكننا استخدام تعبير (CREATE TABLE) أيضاً مع تغييرات طفيفة على صيغة الإستعلام بين Oracle و MySQL و SQLServer.

أما في حال أردنا نسخ بنية الجدول فقط بدون نسخ السجلات في الجدول فيمكننا وضع شرط في تعبير Where له نتيجة False دائماً، كوضع 0=1 على سبيل المثال.

وتشذ قواعد بيانات DB2 على ماسبق وتضطرنا لإضافة التعبير DEFINITION ONLY من أجل سخ بنية الجدول فقط دون سجلاتها.

تعديل بنية الجداول

لتعديل بنية جدول نستخدم الصيغة:

```
ALTER TABLE table_name [ADD | DROP COLUMN] (column_name [data_type] );
```

تساعد ADD على إضافة حقول جديدة، في حين تؤدي DROP إلى حذف حقول موجودة.

مثال:

لدينا الجدول Members الذي يحتوي الأرقام التسلسلية للأعضاء ID وأسماء الأعضاء Name.

يراد تعديل بنية الجدول بإضافة حقل جديد لإدخال نمط العضوية Type. لذا نكتب الصيغة:

```
ALTER TABLE Members ADD  (Type varchar(15) );
```

لنفرض أننا نريد إلغاء الحقل ID من بنية الجدول Members. لتنفيذ ذلك نكتب الصيغة:

```
ALTER TABLE Members DROP COLUMN ID;
```

انتبه:

يساعد التعبير ALTER TABLE على إضافة أو إزالة قيود أو فهارس في بنية الجدول وهو ما سنعالجه لاحقاً في هذا الدرس.

لتعديل بنية جدول نستخدم التعبير (ALTER TABLE) الذي يمكننا من إضافة حقول أو حذف حقول من الجدول.

القيود على الحقول

لنعد مجدداً إلى الصيغة الخاصة بتوليد جدول:

```
CREATE TABLE table_name
(column1_name column1_data_type column1_constraints,
column2_name column2_data_type column2_constraints,…);
```

سنلاحظ أننا استخدمنا في هذه الصيغة التعبير column_constraints الذي يمثل القيد المراد تطبيقه على الحقل. سنتعرف فيما يلي على مجموعة من القيود على الحقول ونغطي كل منها بمثال مناسب.

من أهم القيود وأكثرها استخداماً:

- Not Null
- Default
- Primary key
- Unique
- Check
- Identity
- Auto_increment

انتبه:

يمكن لأكثر من قيد أن يطبق على نفس الحقل.

القيود على الحقول

إذا عدنا إلى الصيغة الخاصة بإنشاء جدول سنرى أننا استخدمنا التعبير column_ constraints الذي يمثل القيد المراد تطبيقه على الحقل. سنتعرف فيما يلي على مجموعة من القيود على الحقول ونغطي كل منها بمثال مناسب.

من أهم القيود وأكثرها استخداماً:

- Not Null
- Default
- Primary key
- Unique
- Check
- Identity
- Auto_increment

القيد NOT NULL

يستخدم هذا القيد في حال أردنا أن نمنع إدخال القيمة Null في الحقل المراد تقييده. يكفي لتطبيق هذا القيد إضافة التعبير NOT NULL عند إنشاء الجدول.

مثال:

إذا أردنا إنشاء جدول بأسماء الموظفين وتوصيف عملهم بحيث نمنع إعطاء القيمة Null لأي حقل من حقول الجدول نستخدم الصيغة:

```
CREATE TABLE Employees (name varchar(40) NOT NULL ,
Job varchar(50) NOT NULL);
```

إذا حاولنا تنفيذ الاستعلام التالي بغرض إضافة سجل إلى جدول الموظفين:

```
Insert into Employees(name) values('Adel')
```

سنلاحظ هذا التعبير لن يعمل وسيولد رسالة خطأ تفيد بضرورة إعطاء قيمة للحقل Job، كون القيمة التلقائية التي سيأخذها الحقل Job في حال عدم إدخال أي قيمة له هي القيمة Null.

القيد NOT NULL

يستخدم هذا القيد في حال أردنا أن نمنع إدخال القيمة Null في الحقل المراد تقييده. يكفي لتطبيق هذا القيد إضافة التعبير NOT NULL عند إنشاء الجدول.

تحديد قيمة تلقائية لحقل القيد DEFAULT

من القيود المستخدمة أيضاً بكثرة القيد الخاص بتحديد قيمة تلقائية لحقل ما. يأخذ هذا الحقل تلك القيمة حين لا يتم إسناد أية قيمة بديلة.

نستخدم القيد DEFAULT بالصيغة التالية:

```
CREATE TABLE MyTable
(Column1 varchar(50) DEFAULT 'Unknown' ,
Column2 varchar(10) );
```

في هذه الصيغة سيتم إعطاء القيمة 'Unknown' للحقل Column1 في أي سجل جديد يتم إنشاؤه دون تحديد قيمة لـ Column1.

مثال:

نريد توليد جدول على يحتوي وصف للبضائع (Description) وعدد الأيام الذي يلزم لنقلها إلى المستودع الرئيسي(Days) . نعلم مسبقاً أن عدد الأيام اللازم لعملية النقل هو يومان بصورة عامة.

لتوليد الجدول نكتب الصيغة:

```
CREATE TABLE Shipments
(Description varchar(75) Not Null , Days INT DEFAULT 2 Not Null);
```

نلاحظ أننا أنشأنا الجدول Shipments الذي يحتوي:

- الحقل Description ومنعنا إعطاء القيمة Null لهذا الحقل.
- الحقل Days من نمط بيانات الأعداد الصحيحة وحددنا القيمة 2 كقيمة تلقائية للحقل.

إذا حاولنا إدراج سجل ضمن هذا الجدول دون إعطاء قيمة لـ Days كما في الصيغة:

```
INSERT INTO Shipments (Description) Values ('Computer');
```

سيكون شكل السجل الذي سيتم إدراجه في الجدول هو: Computer | 2

حيث أخذ الحقل Days في هذا السجل القيمة 2 علماً أننا لم نحدد هذه القيمة في صيغة إدراج السجل.

تحديد قيمة تلقائية لحقل القيد DEFAULT

يُعتبر القيد DEFAULT من القيود المستخدمة أيضاً بكثرة القيد الخاص بتحديد قيمة تلقائية لحقل ما. يأخذ هذا الحقل تلك القيمة حين لا يتم إسناد أية قيمة بديلة.

القيد PRIMARY KEY

من أهم الخصائص التي تميز الجداول في قواعد البيانات العلائقية والتي تعتبر من الشروط الأساسية للنموذج العلائقي لـ . Codd، ضرورة وجود حقل في كل جدول يلعب قيمه دور مميز لكل سجل من السجلات.

تعمل قيمة هذا المفتاح على تمييز كل سجل من سجلات الجدول بصورة وحيدة ويسمى المفتاح الرئيسي للجدول. لهذا الغرض يُستخدم القيد PRIMARY KEY ليحدد أي الحقول هو حقل مفتاح رئيسي لجدول ما.

يستخدم هذا القيد الصيغة:

```
CREATE TABLE MyTable
(Column1 data_type Not Null , Column2 data_type ,
Constraint myPrimaryKey PRIMARY KEY (Column1));
```

أنشأنا هنا قيد باسم MyPrimaryKey وقيدنا به الحقل Column1 بحيث يجب أن تكون قيمة هذا الحقل وحيدة وتميز كل سجل.

إذا لم نرد تحديد اسم لهذا القيد يمكننا كتابة الصيغة:

```
CREATE TABLE MyTable
(Column1 data_type Not Null , Column2 data_type ,
PRIMARY KEY (Column1));
```

أو بصورة أبسط يمكننا استخدام الصيغة:

```
CREATE TABLE MyTable
(Column1 data_type PRIMARY KEY Not Null , Column2 data_type ,
PRIMARY KEY (Column1));
```

مثال:

نريد إنشاء جدول CreditCards لتخزين أرقام البطاقات الائتمانية cardNumber وأسماء أصحابها cardHolder.

نعلم هنا أن أرقام البطاقات الائتمانية فريدة ولا تتكرر لذا سنعتمدها كمفتاح رئيسي في جدولنا وسنستخدم الصيغة المبسطة لقيد المفتاح الرئيسي:

```
CREATE TABLE CreditCards
(cardNumber varchar(20) PRIMARY KEY Not Null ,
cardHolder varchar(50) Not Null);
```

القيد Primary Key

من أهم الخصائص التي تميز الجداول في قواعد البيانات العلائقية والتي تعتبر من الشروط الأساسية للنموذج العلائقي لـ . Codd، ضرورة وجود حقل في كل جدول يلعب قيمه دور مميز لكل سجل من السجلات.

تعمل قيمة هذا المفتاح على تمييز كل سجل من سجلات الجدول بصورة وحيدة ويسمى المفتاح الرئيسي للجدول. لهذا الغرض يُستخدم القيد PRIMARY KEY ليحدد أي الحقول هو حقل مفتاح رئيسي لجدول ما.

القيد UNIQUE

يُستخدم القيد UNIQUE لمنع تكرار قيمة حقل في أكثر من سجل، ولكن هذا لايعني أن هذا الحقل سيصبح المفتاح الرئيسي للجدول.

الصيغة المستخدمة لهذا القيد هي كالتالي:

```
CREATE TABLE MYTable
(Column1 data_type UNIQUE , Column2 data_type);
```

مثال:

إذا أردنا توليد جدول PhoneBook بأسماء الأشخاص Name وأرقام هواتفهم Phone بحيث يكون رقم الهاتف مفتاح رئيسي واسم الشخص ذو قيمة وحيدة لا تتكرر نستخدم الصيغة:

```
CREATE TABLE PhoneBook
(Name varchar(50) UNIQUE , Phone Primary Key Not Null);
```

القيد UNIQUE

يُستخدم القيد UNIQUE لمنع تكرار قيمة حقل في أكثر من سجل، ولكن هذا لايعني أن هذا الحقل سيصبح المفتاح الرئيسي للجدول.

القيد Check

يعتبر القيد Check من أكثر القيود مرونة لأنه يسمح لنا باستخدام طيف واسع من الشروط على القيم التي يتم إدراجها ضمن الجدول.

تشبه طريقة استخدام شروط القيد Check لطريقة استخدام التعبير Where حيث يتم إعادة تقييم الشرط في كل مرة نضيف فيها سجلاً جديداً أو نعدل سجل موجود.

تكون صيغة استخدام القيد Check كالتالي:

```
CREATE TABLE MyTable
(Column1 data_type ,
 Column2 data_type ,
Constraint Cname CHECK (Condition));
```

حيث تعبر Cname عن اسم القيد وتعبر Condition عن شرط يُستخدم فيه اسم الحقل المراد تقييده بـ . Check.

مثال:

نريد إنشاء جدول Ages يتضمن أسماء Name وأعمار Age مجموعة من الأطفال تتراوح بين سنة و12 سنة.

نكتب الصيغة:

```
CREATE TABLE Ages
(Name varchar(50) Not Null ,
                  Age INT ,
Constraint CheckAge CHECK (Age between 1 And 12));
```

ويمكن أن نكتب صيغة تحقق نفس الغرض مع حذف اسم القيد CheckAge إذا كنا لانريد اسم للقيد.

```
CREATE TABLE Ages
(Name varchar(50) Not Null ,
Age INT CHECK(Age between 1 And 12));
```

يمكن لشرط القيد Check أن يتألف من تعبيرات منطقية تحتوي على عمليات منطقية مثل And أو Or فمثلاً إذا أردنا السماح بإدخال عمر بين 1 و12 أو يساوي 15 نكتب الصيغة:

```
CREATE TABLE Ages
(Name varchar(50) Not Null ,
Age INT CHECK(Age = 15 OR Age between 1 And 12));
```

يمكن وضع أكثر من قيد Check على حقل وحيد فمثلاً إذا أردنا السماح بالقيم للعمر بين 1 و12 عدا العمر 3 نكتب الصيغة:

```
CREATE TABLE Ages
(Name varchar(50) Not Null ,
Age INT,
Constraint CheckAge1 CHECK (Age between 1 And 12),
Constraint CheckAge2 CHECK (Age <> 3));
```

انتبه:

لا يطبق القيد Check عندما لايتلقى الحقل أي إدخالات (أي عندما تكون قيمته Null).

القيد Check

يعتبر القيد Check من أكثر القيود مرونة لأنه يسمح لنا باستخدام طيف واسع من الشروط على القيم التي يتم إدراجها ضمن الجدول.

- تشبه طريقة استخدام القيد Check طريقة استخدام التعبير Where حيث يتم إعادة تقييم الشرط في كل مرة نضيف فيها سجلاً جديداً أو نعدل سجل موجود.
- يمكن لشرط القيد Check أن يتألف من تعبيرات منطقية تحتوي على عمليات منطقية مثل And أو Or.
- يمكن وضع أكثر من قيد Check على حقل وحيد.

القيد IDENTITY و AUTO_INCREMENT

توفر قواعد البيانات آلية لتوليد قيم عددية بصورة آلية كقيم لحقل ما عند إضافة سجل جديد إلى الجدول. يمكن استخدام هذه التقنية بالاشتراك مع القيد PRIMARY KEY لتوليد قيم تسلسلية تلقائية في حقل يكون هو حقل المفتاح الرئيسي للجدول.

تستخدم قواعد البيانات تعابير مختلفة لتخديم نفس الغرض:

- نستخدم في SQL Server التعبير IDENTITY
- نستخدم في MySQL التعبير AUTO_INCREMENT
- نستخدم في DB2 التعبير GENRATED ALWAYS AS IDENTITY
- نستخدم في Access التعبير AUTOINCREMENT
- أما في Oracle فنحتاج لإنشاء متتالية بالأمر Create Sequence

مثال:

نود إنشاء جدول بأرقام تسلسلية للطلاب وأسمائهم:

- في قاعدة بيانات SQL Server:

الأرقام تبدأ من 100 وتتزايد بمقدار 1:

```
CREATE TABLE Students
(Name varchar(50) ,
ID INT IDENTITY (100,1) PRIMARY KEY NOT NULL);
```

في حال لم نحدد البذرة لبدء العد (100) والتزايد (1) تكون القيمة التلقائية للبذرة والتزايد هي (1).

- تصبح الصيغة في حالة MySQL بالشكل:

```
CREATE TABLE Students
(Name varchar(50) ,
ID INT AUTO_INCREMENT PRIMARY KEY NOT NULL);
```

- وتصبح الصيغة في Access:

```
CREATE TABLE Students
(Name varchar(50) ,
ID INT AUTOINCREMENT (100,1) PRIMARY KEY NOT NULL);
```

- أما في DB2 فتصبح الصيغة:

```
CREATE TABLE Students
(Name varchar(50) ,
ID INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY);
```

في هذه الصيغة نستخدم ALWAYS للإشارة إلى أننا لن نسمح بإدخال قيم يدوية لحقل التعداد الآلي. لتمكين إدخال القيم يدوياً يمكننا استخدام BY DEFAULT عوضاً عن ALWAYS.

القيد IDENTITY , AUTO_INCREMENT

توفر قواعد البيانات آلية لتوليد قيم عددية بصورة آلية كقيم لحقل ما عند إضافة سجل جديد إلى الجدول. يمكن استخدام هذه التقنية بالاشتراك مع القيد PRIMARY KEY لتوليد قيم تسلسلية تلقائية في حقل يكون هو حقل المفتاح الرئيسي للجدول.

تستخدم قواعد البيانات تعابير مختلفة لتخديم نفس الغرض:

- نستخدم في SQL Server التعبير IDENTITY
- نستخدم في MySQL التعبير AUTO_INCREMENT
- نستخدم في DB2 التعبير GENRATED ALWAYS AS IDENTITY
- نستخدم في Access التعبير AUTOINCREMENT
- أما في Oracle فنحتاج لإنشاء متتالية بالأمر Create Sequence

القيد IDENTITY وAUTO_INCREMENT

لا تدعم قواعد بيانات Oracle التعابير IDENTITY أو AUTO_INCREMENT المستخدمة في قواعد البيانات الأخرى. لذا تستخدم Oracle تعبير خاص لإنشاء متوالية هو التعبير CREATE SEQUENCE لمحاكاة عمل هذه القيود. تكون صيغة هذا التعبير على الشكل:

```
CREATE SEQUENCE sequence_name
INCREMENT increment_step
START WITH start_seed;
```

- عند إنشاء مثل هذه المتوالية يمكن استخدام القيم التي تأخذها لإدراجها في صيغ SQL أخرى.
- لإعادة قيمة المتوالية وزيادة عداد المتوالية بمقدار الخطوة يكفي استخدام التعبير sequence_name.NextVal

يتم استخدام قيم هذه المتوالية مباشرة في تعبير Insert عند إدراج سجل جديد في جدول ما وذلك بالصيغة:

```
INSERT INTO mytable
(Column1, Column2, Column3)
Values (sequence_name.NextVal, Value2, Value3);
```

نلاحظ هنا أننا أعطينا للحقل Column1 القيمة الخاصة بالمتوالية واستخدمنا الطريقة NextVal لاستجرار القيمة التالية للمتوالية.

مثال:

نريد إنشاء جدول Products في قاعدة بيانات Oracle يحتوي حقلين الأول خاص بأرقام المنتجات productID والثاني وصف المنتجات ProductDescription. كما نود جعل الحقل productID حقل مفتاح رئيسي وجعل قاعدة البيانات تعطي لهذا الحقل أرقام متسلسلة تلقائية.

لإتمام هذا العمل نقوم أولاً بإنشاء الجدول بالصيغة:

```
CREATE TABLE Products
(productID INT PRIMARY KEY NOT NULL ,
productDescription varchar(75));
```

ثم نقوم بإنشاء متوالية وليكن اسمها مثلاً Counter وذلك بالصيغة:

```
CREATE SEQUENCE Counter;
```

حيث لم نحدد هنا بداية العد والخطوة لذلك ستحدد قيمتها تلقائياً بالعدد 1.

عند إدراج سجل في جدولنا الجديد يجب استخدام المتوالية كقيمة للحقل productID وذلك بالشكل:

```
INSERT INTO Products
(productID , productDescription)
Values (Counter.NextVal , 'any Porduct description');
```

لا تدعم قواعد بيانات Oracle التعابير IDENTITY أو AUTO_INCREMENT المستخدمة في قواعد البيانات الأخرى. لذا تستخدم Oracle تعبير خاص لإنشاء متوالية هو التعبير CREATE SEQUENCE لمحاكاة عمل هذه القيود.

عند إنشاء مثل هذه المتوالية يمكن استخدام القيم التي تأخذها لإدراجها في صيغ SQL أخرى.
لإعادة قيمة المتوالية وزيادة عداد المتوالية بمقدار خطوة يكفي استخدام التعبير sequence_name.NextVal

التكامل المرجعي 1

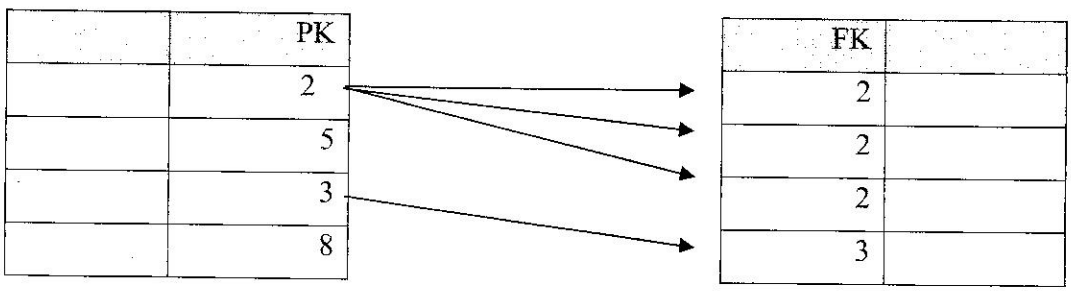
تتألف قواعد البيانات العلائقية من مجموعة من الجداول يتشكل كل من هذه الجداول من مجموعة من السجلات. يميز كل سجل من سجلات جدول قيمة فريدة لحقل أطلقنا عليه اسم حقل المفتاح الرئيسي.

قد ترتبط الجداول بعضها ببعض بعلاقات يمكن لهذه العلاقات أن تأخذ أحد الأشكال:

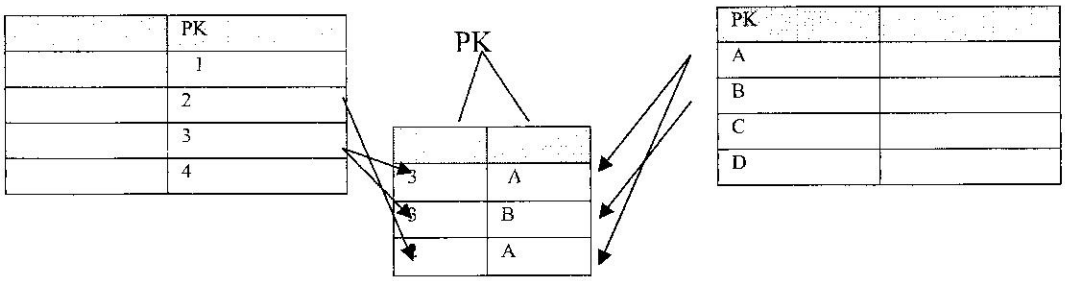
- علاقة واحد لواحد أو سجل لسجل: في هذا النوع من العلاقات يرتبط كل سجل من الجدول الأول مع سجل وحيد من الجدول الثاني.
- علاقة واحد لعديد أو سجل لعدة سجلات: يرتبط كل سجل من الجدول الأول مع مجموعة سجلات من الجدول الثاني وليس العكس.
- علاقة عديد لعديد أو عدة سجلات لعدة سجلات: ترتبط مجموعة من سجلات الجدول الأول مع مجموعة من سجلات الجدول الثاني.

التكامل المرجعي 2

علاقة سجل لعدة سجلات: يتم عادة التعبير عن العلاقة بإدراج ما يسمى المفتاح الثانوي في حقل خاص في الجدول الثاني (الذي يتوضع في الجهة التي توجد فيها مجموعة السجلات المرتبطة بالسجل الوحيد) وبحيث تكون قيم هذا الحقل مأخوذة من قيم حقل المفتاح الرئيسي للجدول الأول.



أما في علاقة عدة سجلات لعدة سجلات فهي تطبّق فعلياً على شكل علاقتين من نوع سجل لعدة سجلات مع استخدام جدول وسيط يحتوي مفتاح أساسي مركب يتكون من قيمة المفتاح الأساسي للسجل المراد ربطه من الجدول الأول مع قيمة المفتاح الأساسي للسجل المراد ربطه من الجدول الثاني.



التكامل المرجعي 2

علاقة سجل لعدة سجلات: يتم عادة التعبير عن العلاقة بإدراج ما يسمى المفتاح الثانوي في حقل خاص في الجدول الثاني (الذي يتوضع في الجهة التي توجد فيها مجموعة السجلات المرتبطة بالسجل الوحيد) بحيث تكون قيم هذا الحقل مأخوذة من قيم حقل المفتاح الرئيسي للجدول الأول.

أما في علاقة عدة سجلات لعدة سجلات فهي تطبّق فعلياً على شكل علاقتين من نوع سجل لعدة سجلات مع استخدام جدول وسيط يحتوي مفتاح أساسي مركب يتكون من قيمة المفتاح الأساسي للسجل المراد ربطه من الجدول الأول مع قيمة المفتاح الأساسي للسجل المراد ربطه من الجدول الثاني.

التكامل المرجعي 3

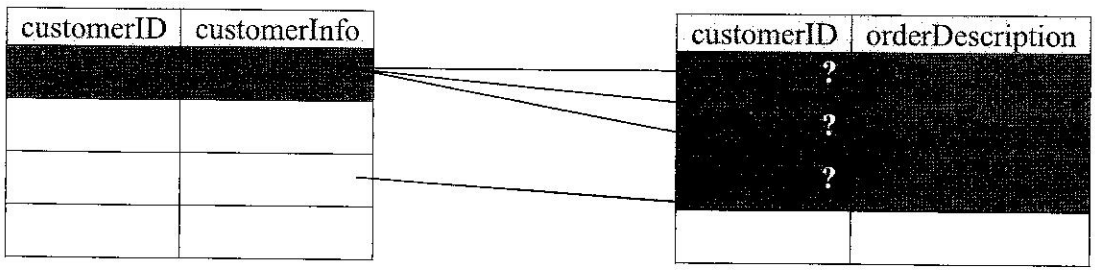
لنفرض أن لدينا قاعدة بيانات علائقية مؤلفة من جدولين يرتبط الأول بالثاني بعلاقة واحد لعديد (سجل من الأول بعدة سجلات من الثاني). سبق وأسلفنا أن طريقة حلّ العلاقة، تتلخص في إدراج مفتاح في الجدول الثاني يسمى المفتاح الثانوي مصدره قيم المفتاح الرئيسي للجدول الأول.

لكن ماذا لو اضطررنا إلى حذف أو تعديل أحد السجلات من الجدول الأول المرتبطة بعدد من السجلات من الجدول الثاني ؟

فعلياً ما سنحصل عليه هو قيم جديدة غير مترابطة ببعضها البعض. وهنا تكمن أهمية المحافظة على ماندعوه التكامل المرجعي في قاعدة البيانات.

مثال:

لنفرض أن لدينا جدول يحتوي معلومات الزبائن وجدول آخر يحتوي طلبات الشراء التي قام بها الزبائن. من الواضح أن العلاقة بين الجدولين هي علاقة سجل من جدول الزبائن لعدة سجلات من جدول الطلبات. فإذا خطر بذهن أحدهم حذف اسم أحد الزبائن من جدول الزبائن وكان لهذا سجلات طلبات في جدول الطلبات فسنحصل بعد عملية الحذف تلك على طلبات في جدول الطلبات لا نعلم الزبون الذي طلبها.



كما نلاحظ في المخطط أعلاه أن حذف السجل من الجدول Customers (إلى اليسار) سيؤدي إلى ضياع مرجعية السجلات في الجدول Orders (إلى اليمين).

التكامل المرجعي 3

لنفرض أن لدينا قاعدة بيانات علائقية مؤلفة من جدولين يرتبط الأول بالثاني بعلاقة واحد لعديد (سجل من الأول بعدة سجلات من الثاني). سبق وأسلفنا أن طريقة حلّ العلاقة، تتلخص في إدراج مفتاح في الجدول الثاني يسمى المفتاح الثانوي مصدره قيم المفتاح الرئيسي للجدول الأول.

لكن ماذا لو اضطررنا إلى حذف أو تعديل أحد السجلات من الجدول الأول المرتبطة بعدد من السجلات من الجدول الثاني ؟

فعلياً ما سنحصل عليه هو قيم جديدة غير مترابطة ببعضها البعض. وهنا تكمن أهمية المحافظة على ماندعوه التكامل المرجعي في قاعدة البيانات.

فإذا فرضنا أن لدينا جدول يحتوي معلومات الزبائن وجدول آخر يحتوي طلبات الشراء التي قام بها الزبائن. من الواضح أن العلاقة بين الجدولين هي علاقة سجل من جدول الزبائن لعدة سجلات من جدول الطلبات. فإذا خطر بذهن أحدهم حذف اسم أحد الزبائن من جدول الزبائن وكان لهذا سجلات طلبات في جدول الطلبات فسنحصل بعد عملية الحذف تلك على طلبات في جدول الطلبات لا نعلم الزبون الذي طلبها.

التكامل المرجعي 4

لضمان عدم كسر قاعدة التكامل المرجعي في SQL، علينا استخدام قيد يعرّف أحد الحقول في جدول على أنه حقل مفتاح ثانوي لجدول آخر بحيث يفشل أي تعبير لا يحترم قاعدة التكامل المرجعي بعد استخدام هذا القيد.

يتم لهذا الغرض استخدام الصيغة:

```
CREATE TABLE myTable
(Column1 Column1Type PRIMARY KEY NOT NULL , Column2 Column2Type ,
Column3 Column3Type ,
CONSTAINT foreign_key_name FOREIGN KEY (Column3)
REFERENCES other_table (other_table_primary_key));
```

نلاحظ في الصيغة السابقة أننا أنشأنا الجدول myTable الذي يحوي الحقول Column1 كحقل مفتاح رئيسي والحقل Column2 والحقل Column3 الذي تم تقييده بالقيد FOREIGN KEY لجعله مفتاح ثانوي مرتبط بالمفتاح الرئيسي للجدول other_table والذي يحمل اسم other_table_primary_key.

مثال:

إذا أردنا إنشاء الجدول Brands الذي يحتوي أسماء وأرقام ماركات أجهزة الحواسب المتوفرة بمتجر، والجدول Models الذي يحتوي الموديلات المتوفرة من كل ماركة من الماركات، وأردنا ربط الجدول Brands مع الجدول Models بعلاقة سجل من جدول Brands إلى عدة سجلات من جدول Models مع ضمان التكامل المرجعي نكتب الصيغة التالية لإنشاء الجدول Brands:

```
brandID INT PRIMARY KEY NOT NULL ,      CREATE TABLE Brands (
brandName varchar (50));
```

لإنشاء الجدول Models نستخدم الصيغة:

```
modelID INT PRIMARY KEY NOT NULL ,      CREATE TABLE Models (
modelName varchar (50) ,
modelBrand INT ,
CONSTRAINT myFK FOREIGN KEY (modelBrand)
REFERENCES Brands (brandID));
```

هذه الصيغة صالحة في SQL Server, Oracle, DB2 و Access.

يمكن كتابة هذه الصيغة بشكل أكثر اختصاراً إذا كنا لا ننوي تحديد اسم القيد بالشكل:

```
modelID INT PRIMARY KEY NOT NULL ,      CREATE TABLE Models (
modelName varchar (50) ,
modelBrand INT ,
FOREIGN KEY (modelBrand)
REFERENCES Brands (brandID));
```

التكامل المرجعي 4

لضمان عدم كسر قاعدة التكامل المرجعي في SQL، علينا استخدام قيد يعرّف أحد الحقول في جدول على أنه حقل مفتاح ثانوي لجدول آخر بحيث يفشل أي تعبير لا يحترم قاعدة التكامل المرجعي بعد استخدام هذا القيد.

التكامل المرجعي في MySQL

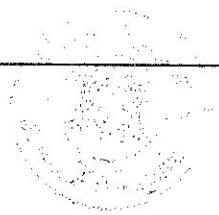
في MySQL تختلف الصيغة بنقطتين أساسيتين:

الأولى هي أن الجداول الداخلة في العلاقة يجب أن تكون من نوع الجداول الخاص في قواعد بيانات MySQL وهو InnoDB، والثانية هي ضرورة تعريف حقل المفتاح الثانوي كفهرس فتصبح الصيغة من الشكل:

```
CREATE TABLE myTable
(Column1 Column1Type PRIMARY KEY NOT NULL , Column2 Column2Type ,
Column3 Column3Type ,
FOREIGN KEY (Column3)
REFERENCES other_table (other_table_primary_key)
INDEX myIndex (Column3))
Type = InnoDB;
```

التكامل المرجعي في MySQL

في MySQL تختلف الصيغة بنقطتين أساسيتين:
الأولى هي أن الجداول الداخلة في العلاقة يجب أن تكون من نوع الجداول الخاص في قواعد بيانات MySQL وهو InnoDB والثانية هي ضرورة تعريف حقل المفتاح الثانوي كفهرس.



نموذج البيانات	طبيعة المدخلات	الحجم التخزيني
Text نص	نص أو تركيبة نصوص وأرقام، كالعناوين. وكذلك الأرقام التي لا تتطلب حسابات، كأرقام الهاتف، أو أرقام الأجزاء،	حتى ٢٥٥ حرفاً. ١بايت لكل حرف.
Currency العملة	قيم العملة. استخدم نوع البيانات "عملة" لمنع حدوث التقريب أثناء إجراء الحسابات. بالضبط ويصل عدد الخانات إلى ١٥ خانة إلى يسار الفاصلة العشرية و ٤ خانات إلى يمينها.	8بايت.
Number رقم	تستخدام البيانات الرقمية للحسابات الرياضية، باستثناء الحسابات المتعلقة بالأموال (استخدام نوع العملة).	1، 2، 4، أو ٨ بايت
Date/Time الوقت التاريخ	تاريخ وزمن	8بايت.
نعم/لا Yes/No	حقول تحتوي فقط على قيمة واحدة أو اثنتين، مثل "نعم/لا"، و"صحيح/خطأ"، و"تشغيل/إيقاف".	1بايت.
AuoNumber ترقيم تلقائي	الأرقام الفريدة المتتالية (التي تزيد بمقدار ١) أو الأرقام العشوائية يتم إدراجها تلقائياً عند إضافة سجل.	4بايت. ١٦ بايت لـــ "معرف"
Memo مذكرة	نص أو أرقام طولية، كالملاحظات أو الوصف.	حتى ٦٤.٠٠٠ حرفاً.
OLE Object كائن OLE	البرامج (مثل مستنداتMicrosoft Word ، أو جداول بياناتMicrosoft Excel ، أو صور، أو أصوات، أو أي بيانات ثنائية أخرى)، التي.تم إنشاؤها في برامج أخرى باستخدام البروتوكولOLE ، والتي يمكن ربطها بجدول في Microsoft Accessأو تضمينها فيه.	حتى ١ جيجا بايت (مقيدة بواسطة مساحة القرص).

أنواع البيانات في MS - Access

Hyperlink ارتباط تشعبي	حقول سوف تقوم بتخزين ارتباطات تشعبية .يمكن أن يكون الارتباط التشعبي مسار UNC أو .URL	حتى ٦٤.٠٠٠ حرفاً.
Lookup معالج البحث	يقوم بإنشاء حقل يسمح لك باختيار قيمة من جدول آخر أو من قائمة قيم باستخدام مربع تحرير وسرد. يؤدي اختيار هذا الخيار في قائمة أنواع البيانات إلى بدء معالج لتعريف هذا النوع من أجلك.	تفس حجم الحقل المفتاح الأساسي والذي هو أيضاً الحقل "بحث"؛ ٤ بايت بالضبط

الجدول (٣-١) نماذج البيانات في MS-ACESS

أكسس _ Microsoft Office Access

برنامج أكسس : نظام لإدارة قواعد البيانات العلائقية (المتضمنة جداول).

قاعدة البيانات: مجموعة من البيانات المنظمة والمخزنة بطريقة نموذجية وبدون تكرار بشكل يسهل استرجاعها.

يسمح نظام إدارة قواعد البيانات (أكسس) بـ :

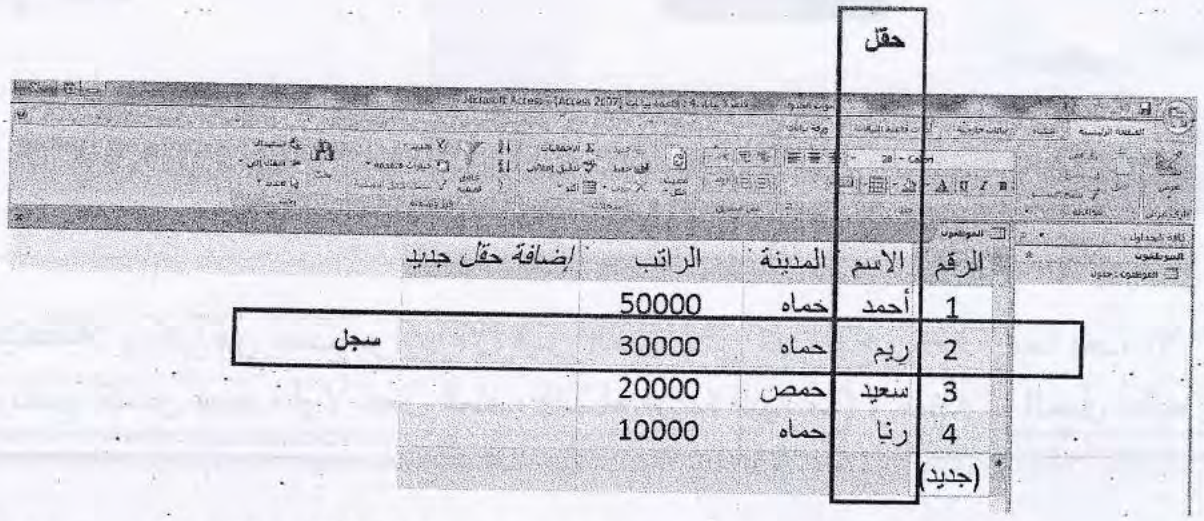
1) إنشاء قاعدة البيانات، وإجراء التعديلات عليها من إضافة/تحديث/حذف
2) تقديم عدة طرق لاسترجاع وعرض البيانات.

أهم مكونات قاعدة البيانات :

1. **الجداول:** الجدول هو المكون الأساسي في قاعدة البيانات ويحتوي الجدول على معلومات عن موضوع معين مثل: الموظفون أو المنتجات ..

يتألف الجدول من مجموعة من **السجلات** (الصفوف)، يحتوي كل سجل على معلومات حول عنصر واحد _موظف معين مثلاً_ ، ويتألف السجل من مجموعة من **الحقول** (الأعمدة) مثل:الاسم ،المدينة، العنوان، الهاتف..

يجب أن تكون بيانات الحقل الواحد من نفس النوع(أحرف، أرقام، تواريخ...)



2. **الاستعلامات**: استرجاع بيانات من جدول أو أكثر وفق معايير محددة.
3. **النماذج**: واجهة تسمح للمستخدم النهائي بالتعامل مع قاعدة البيانات بسهولة .

4. **التقارير**: شكل معين يعرض نتائج تحقق شروط محددة لتتم طباعتها.

ملاحظة : إن أساس قاعدة البيانات هي الجدول ،لا بد أن تحتوي أي قاعدة بيانات على جدول واحد على الأقل، و إلا لا يمكن إنشاء نماذج أو تقارير أو استعلامات لأنها تعتمد على بيانات الجدول.

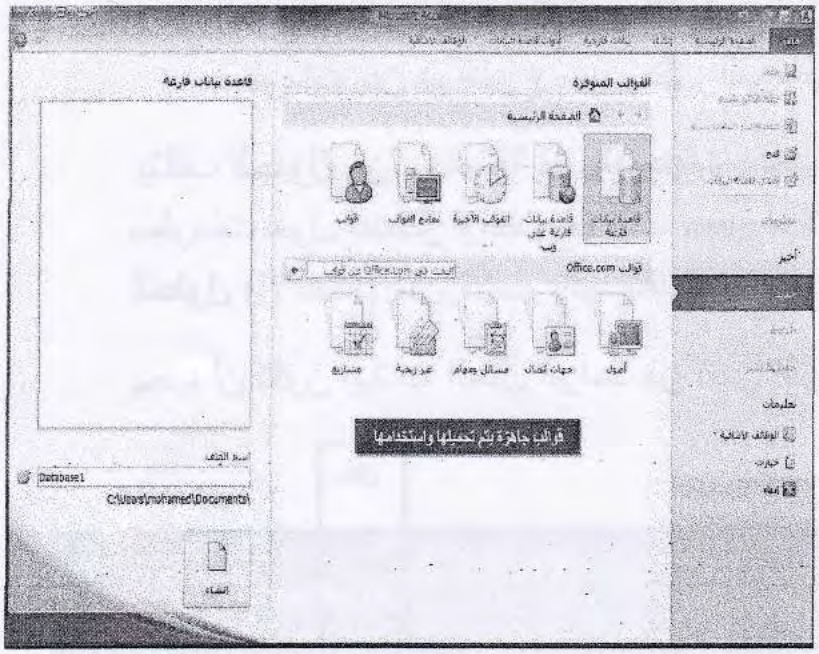
بدء العمل مع أكسس :

الدخول إلى البرنامج :

❖ إنشاء قاعدة البيانات:

تظهر النافذة الرئيسية للبرنامج :

نختار قاعدة بيانات فارغة وبعد كتابة اسم الملف نضغط على إنشاء.



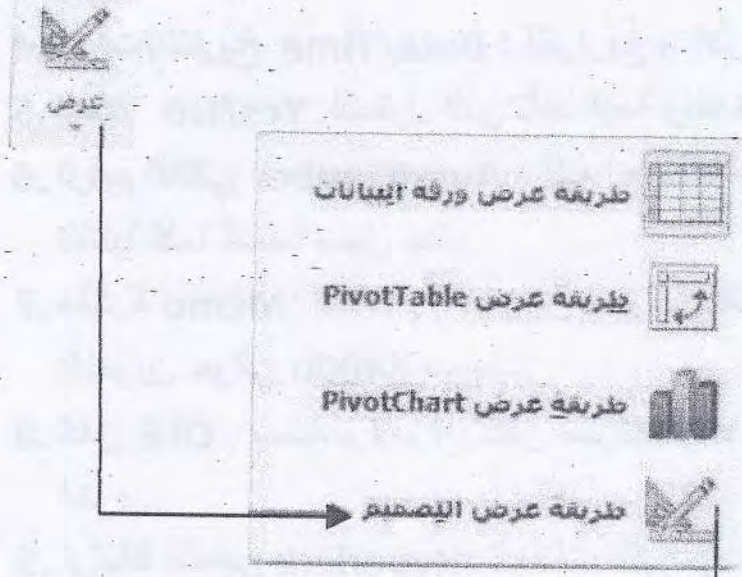
ملاحظة: يمكننا في جميع برامج الأوفيس العمل على البرنامج وحفظه فيما بعد، إلا برنامج أكسس فيجب أولاً عمل الملف (إنشاء قاعدة البيانات) وحفظه ثم العمل عليه .

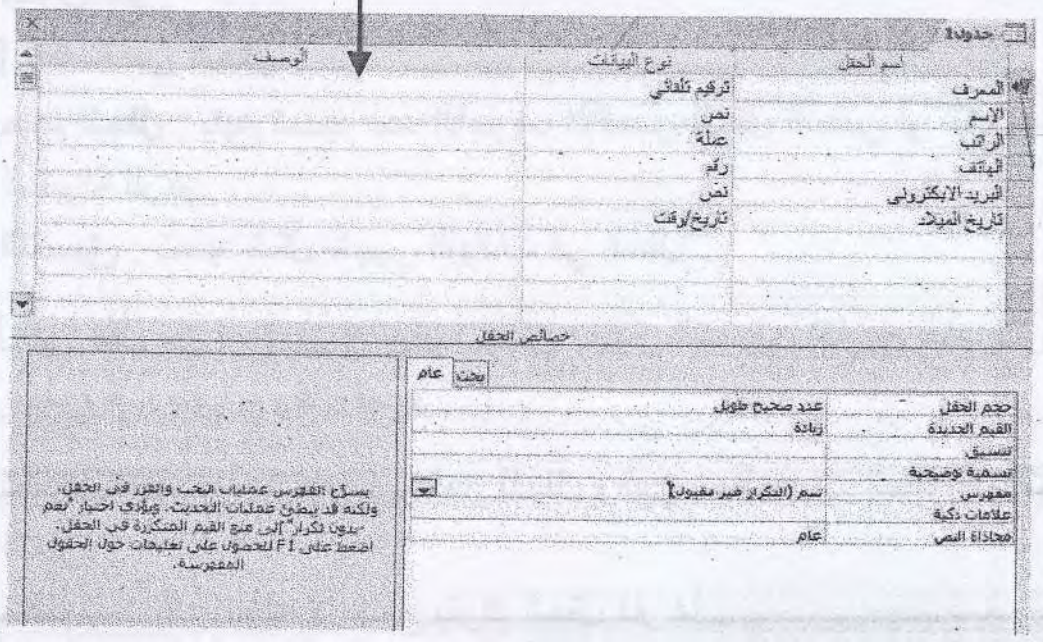
أولاً: الجداول

❖ إنشاء الجدول (بطريقة عرض التصميم):

من خلال تبويب الصفحة الرئيسية _ مجموعة عرض يتم اختيار

طريقة عرض التصميم : حيث يتم تسجيل أسماء الحقول، وتحديد نوع بيانات كل حقل، وخصائص كل حقل.





☒ أنواع بيانات الحقول :

1. **نص_text**: يستخدم للحقول النصية وهي الحروف، أو الأرقام التي لا تتطلب حسابات، حجمه التخزيني حتى 255 محرف .
2. **رقم Number** : يستخدم للحقول الرقمية التي تتطلب حسابات رياضية.
3. **العملة_currency**: تستخدم لحقول المبالغ المالية، لمنع حدوث التقريب عند إجراء الحسابات.
4. **الوقت/التاريخ Date/Time** : للتواريخ و الأزمنة .
5. **نعم/لا Yes/No**: للحقول التي تأخذ قيمة واحدة من قيمتين .
6. **ترقيم تلقائي AutoNumber** : لإدراج سلسلة من الأرقام الفريدة التي تزداد تلقائياً كلما أضفنا سجل جديد.
7. **مذكرة Memo**: تستخدم للنصوص الطويلة كالوصف أو الملاحظات، سعة التخزين حوالي 64000 محرف.
8. **كائن OLE** : يستخدم لإدراج كائن صوتي أو صورة أو ملفات من برامج أخرى.
9. **ارتباط تشعبي Hyperlink**: للبيانات التي من نوع رابط أو مسار .
10. **معالج البحث lookup**: يستخدم من أجل اختيار قيمة من قائمة منسدلة أو من جدول آخر.

☒ خصائص الحقل :

1. **حجم الحقل** : يتيح تحديد عدد الأحرف الأقصى للحقل النصي أو الحدود المقبولة للرقم.
2. **التنسيق** : تحديد شكل ظهور البيانات في الحقل .

مثال في حقل من نوع تاريخ /وقت يمكن اختيار التنسيق: (تاريخ عام، تاريخ متوسط، وقت طويل، وقت قصير...).

3. **تسمية توضيحية:** تحديد اسم آخر للحقل والذي سيظهر في الجدول والنماذج و التقارير.
4. **مطلوب** : عند تفعيله لا يسمح بترك الحقل فارغاً .
5. **مفهرس** : تستخدم لتسريع عملية البحث.

6. **قاعدة التحقق من الصحة** : يسمح بوضع قاعدة لاختبار صحة البيانات المدخلة في حقل معين (تحديد مجال القيم المسموح بإدخالها في الحقل).

عند كتابة قاعدة التحقق من الصحة لا بد من مراعاة الأمور التالية :

أ- التاريخ يوضع بين # #

ب - اسم الحقل يوضع بين قوسين [ ]

ت - النص يوضع بين إشارتي اقتباس " "

ث - يمكننا استخدام معاملات المقارنة والربط المنطقي التالية :

=	يساوي
<>	لا يساوي
>	أكبر تماماً
<	أصغر تماماً
>=	أكبر أو يساوي
<=	أصغر أو يساوي
OR	(أو) لربط أكثر من شرط
AND	(و) لربط شرطين أو أكثر
BETWEEN	بين قيمتين

مثال: علامة الطالب في أي مقرر لا تتجاوز 100 ، لتجنب إدخال علامة تفوق 100 أو تقل عن 0 نكتب في **قاعدة التحقق من الصحة** لحقل (علامة الطالب) :

```
>=0 And <=100
```

أو: `Between 0 And 100`

7. **نص التحقق من الصحة** : لتحديد الرسالة التي ستظهر للمستخدم عند إدخال قيمة مخالفة لقاعدة التحقق من الصحة.
8. **القيمة الافتراضية** : تحديد قيمة افتراضية لتظهر بشكل آلي في الحقل ويمكن تغيير هذه القيمة أثناء إدخال البيانات.

أمثلة على القيمة الافتراضية :

Date()= يتم ظهور **التاريخ** الموجود في نظام الحاسب بشكل تلقائي في حقل من نوع تاريخ/وقت

Now()= يتم ظهور **التاريخ والوقت** الموجود في نظام الحاسب بشكل تلقائي في حقل من نوع تاريخ / وقت

☒ تحديد المفتاح الرئيسي للجدول :

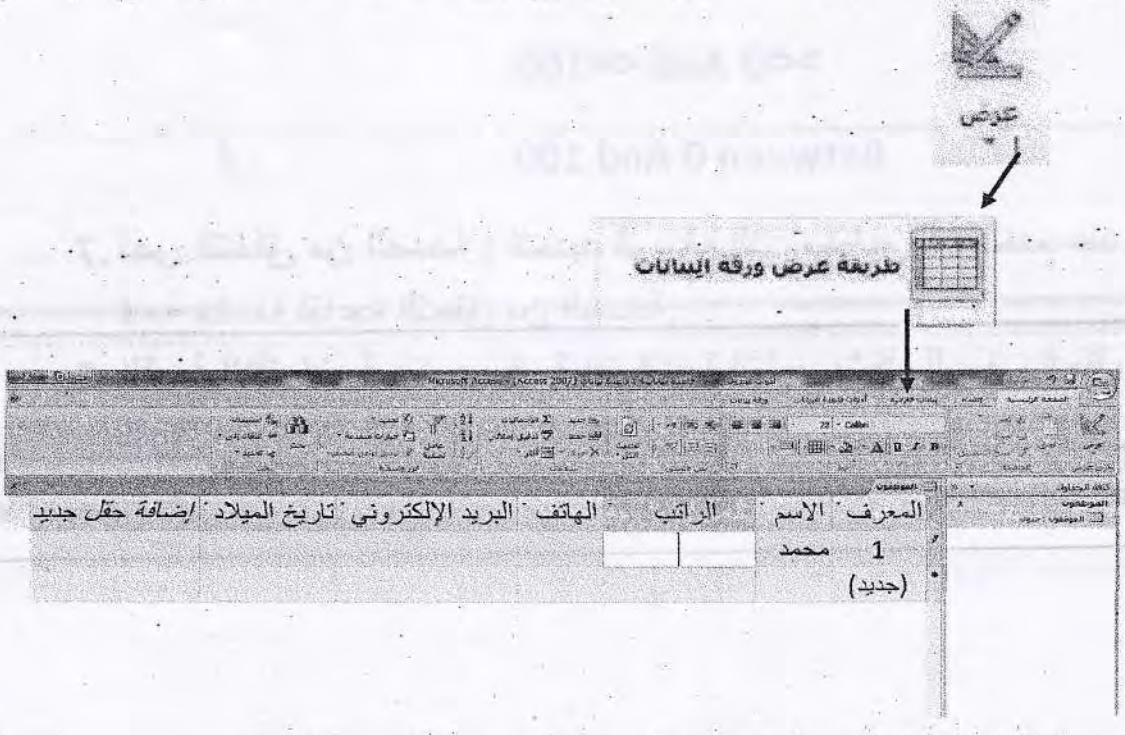
المفتاح الرئيسي: هو حقل أو مجموعة حقول التي تميز أو تعرف سجلات الجدول، مثل (رقم الطالب في جدول الطلاب..) ، خواص المفتاح الرئيسي :

1 -عدم التساوي (أن لا تتكرر القيمة الواحدة أكثر من مرة).
2 -لا يمكن أن يكون فارغاً.
3 -عدم القدرة على الاختزال إذا كان مركباً.

إدخال البيانات في الجدول :

بعد الانتهاء من إدخال أسماء الحقول وتحديد أنواع بيانات كل حقل وخصائصه و تحديد المفتاح الرئيسي في صفحة عرض التصميم ، ننتقل لصفحة ورقة البيانات :

من مجموعة عرض : نختار **طريقة عرض ورقة البيانات** ونقوم بإدخال البيانات في الجدول :



العلاقات بين الجداول

يشترط لربط الجداول وجود حقل مشترك بين الجدولين، ولا يشترط أن يكون للحقلين نفس الاسم، بل **نفس نوع البيانات** .

هناك ثلاثة أنواع للعلاقات بين الجداول :

واحد لواحد (1:1)، واحد لمتعدد (N:1)، متعدد لمتعدد (N:M).

1. علاقة واحد لواحد (1:1) :

تنشأ هذه العلاقة عندما يكون كل سجل من الجدول الأول مرتبط مع سجل واحد فقط من الجدول الثاني.

مثال : لتكن لدينا قاعدة بيانات الطلاب فيها الجدولين :

ذاتية الطلاب (1:1) محصلة مادة الحاسوب

الرقم الجامعي	الاسم	العمر	المدينة
100	محمد	20	حماه
101	ريم	22	حماه
102	نور	21	حمص

الرقم الجامعي	علامة العملي	علامة النظري	المحصلة
100	23	40	63
101	20	30	50
102	30	55	85

نجد هنا أن العلاقة بين الجدولين هي من نوع (1:1) لأن كل سجل في الجدول الأول يقابله سجل واحد فقط في الجدول الثاني (لكل طالب محصلة واحدة فقط).

وكل سجل في الجدول الثاني يقابل سجل واحد فقط في الجدول الأول (كل محصلة تعود لطالب واحد).

إن حقل الرقم الجامعي هو المفتاح الرئيسي في كلا الجدولين (لأنه حقل فريد فكل طالب له رقم مميز لا يمكن تكراره لطالب آخر).وبالتالي يتم الربط بين الجدولين عن طريق حقل الرقم الجامعي (المفتاح الرئيسي).

2. **علاقة واحد لمتعدد (1:N)** : تنشأ هذه العلاقة عندما يرتبط سجل من الجدول الأول مع عدد من سجلات من الجدول الثاني، وكل سجل في الجدول الثاني يرتبط مع سجل واحد فقط في الجدول الأول.

مثال : ليكن لدينا الجدولين :

جدول الأقسام (1:N) **جدول الموظفين**

رقم القسم	اسم القسم
1	الإنتاج
2	التسويق
3	المحاسبة

رقم الموظف	اسم الموظف	رقم القسم
10	سامي	1
11	لمى	1
12	راما	3
13	ماهر	2
14	جمال	2
15	نور	

نلاحظ أن القسم ممكن أن يضم عدد من الموظفين، ولكن كل موظف ينتمي إلى قسم واحد وبالتالي العلاقة بين الجدولين من نوع (1:N)

إن المفتاح الرئيسي في جدول الأقسام هو حقل (رقم القسم)، و المفتاح الرئيسي في جدول الموظفين هو حقل (رقم الموظف).

تمت إضافة حقل (رقم القسم) إلى جدول الموظفين، **وهو مفتاح أجنبي** في جدول الموظفين، (**المفتاح الأجنبي :** هو حقل في جدول مرتبط بجدول أساسي، تكون قيمه مطابقة للمفتاح الرئيسي في الجدول الأساسي، بهدف الربط بين الجدولين، وممكن أن تتكرر قيمه أو أن تكون فارغة)

وبالتالي يتم الربط عن طريق حقل المفتاح الرئيسي في الجدول الأول (حقل رقم القسم في جدول الأقسام) مع حقل المفتاح الأجنبي في الجدول الثاني (حقل رقم القسم في جدول الموظفين)

3. علاقة متعدد لمتعدد (N:M) :

تنشأ هذه العلاقة عندما يرتبط سجل من الجدول الأول مع عدد من سجلات الجدول الثاني، كما يرتبط سجل من الجدول الثاني مع عدد من سجلات الجدول الأول.

مثال : العلاقة بين الجدولين : جدول الطلاب، و جدول المواد

جدول الطلاب

رقم الطالب	الاسم
120	هبة
121	نور
122	سعيد
123	محمد

جدول المواد

رقم المادة	اسم المادة
1	محاسبة
2	اقتصاد
3	إحصاء

(رقم الطالب) هو المفتاح الرئيسي في جدول الطلاب ، كما أن (رقم المادة) مفتاح رئيسي في جدول المواد

نلاحظ أن الطالب ممكن أن يسجل أكثر من مادة ، كما أن المادة الواحدة يسجلها أكثر من طالب لذلك العلاقة بين الجدولين من نوع متعدد لمتعدد (N:M)

لإنشاء علاقة متعدد لمتعدد نحتاج لجدول إضافي وليكن جدول **العلامات**:

رقم الطالب	رقم المادة	العلامة
120	1	70
120	2	65
120	3	78
121	1	55
121	2	90
122	2	81
122	3	43
123	1	66
123	3	80

إن حقل رقم الطالب هو مفتاح أجنبي في جدول العلامات و كذلك حقل رقم المادة .

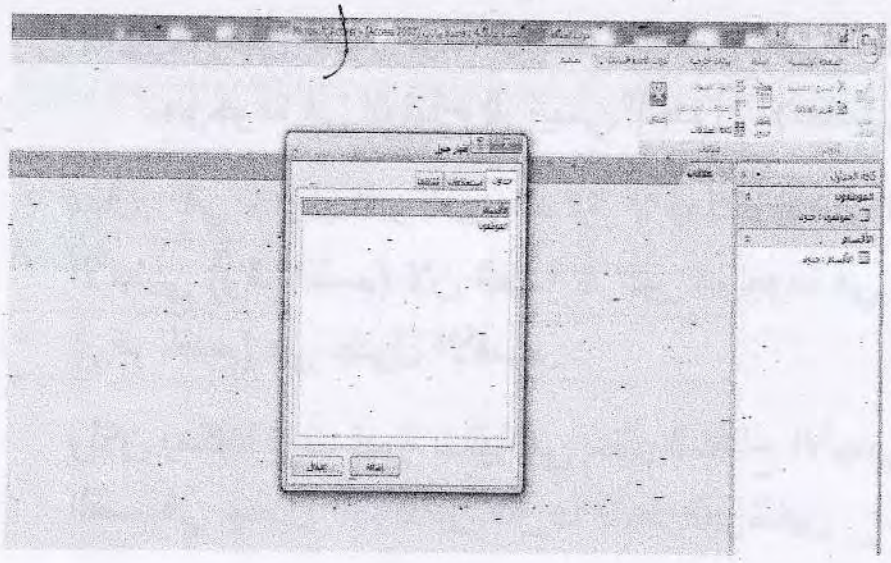
المفتاح الرئيسي في جدول العلامات هو **الحقلين** (رقم الطالب ، رقم المادة) معاً.

و الآن يتم إنشاء علاقة (واحد لمتعدد) بين حقل **رقم الطالب**(المفتاح الرئيسي) في جدول الطلاب، مع حقل **رقم الطالب** (المفتاح الأجنبي) في جدول العلامات.

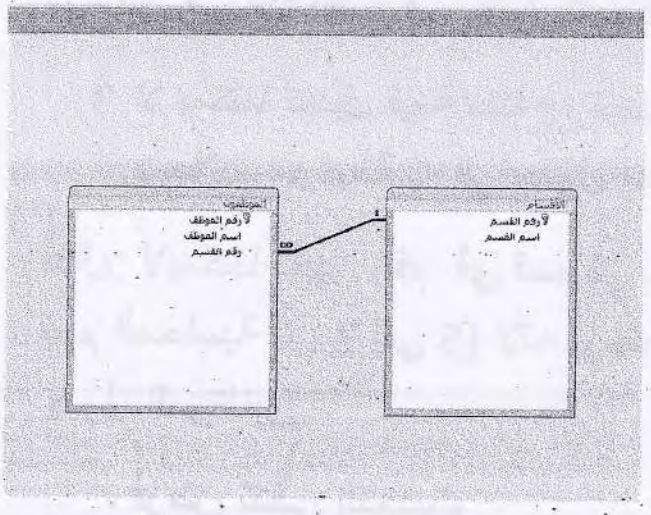
و إنشاء علاقة (واحد لمتعدد) بين حقل **رقم المادة**(المفتاح الرئيسي) في جدول المواد، مع حقل **رقم المادة** (المفتاح الأجنبي) في جدول العلامات.

خطوات إنشاء علاقة ربط بين الجداول في أكسس:

لإنشاء العلاقة بين جدول الأقسام وجدول الموظفين :



1. نغلق الجدولين.
2. من أدوات قاعدة البيانات نختار علاقات
3. يظهر صندوق الحوار، نقوم بإضافة الجدولين



4. بعد ذلك نستطيع إنشاء العلاقة من خلال الحقل المشترك (رقم القسم)

بسحب حقل رقم القسم من الجدول الرئيسي (الأقسام) و الإفلات على حقل رقم القسم في جدول الموظفين.

فرض التكامل المرجعي

يستخدم التكامل المرجعي لضمان صحة العلاقات بين السجلات المرتبطة و ضمان عدم حذف أو تغيير البيانات المرتبطة عن طريق الخطأ.

عند اختيار:

✓ **فرض التكامل المرجعي فقط:**

1. لا يمكن إدخال قيمة في حقل مفتاح أجنبي للجدول المرتبط غير موجودة في المفتاح الرئيسي للجدول الأساسي .

مثال: في جدول الموظفين السابق لا يمكننا إدخال القيمة 4 في حقل المفتاح الأجنبي (رقم القسم) لأن القيمة 4 غير موجودة في حقل المفتاح الرئيسي (رقم القسم) في جدول الأقسام.

ولكن يمكننا ترك قيمة خالية في حقل المفتاح الأجنبي (يمكننا ترك رقم القسم في جدول الموظفين فارغاً لأحد الموظفين _مثل سجل نور_).

2. لا يمكننا حذف سجل من الجدول الرئيسي إذا كان مرتبطاً مع سجلات في جدول آخر.

مثال: لا يمكننا حذف سجل من جدول الأقسام وليكن قسم المحاسبة لأنه يوجد موظفين في هذا القسم في جدول الموظفين.

3. لا يمكننا تعديل قيمة مفتاح رئيسي في الجدول الأساسي إذا كان هناك سجلات مرتبطة به في جدول آخر.

مثال: لايمكننا تغيير رقم أي قسم في جدول الأقسام (مثلا لا يمكن تغيير رقم قسم المحاسبة من 3 إلى 5) لأنه يوجد سجلات للموظفين مرتبطة به في جدول الموظفين.

✓ **فرض تتالي التحديث :**

عند اختيار **فرض التكامل المرجعي مع تتالي التحديث** يتم السماح بتغيير قيمة مفتاح رئيسي بحيث يتم التحديث إلى القيمة الجديدة في الجدول الأساسي و الجدول المرتبط .

مثلاً(عند تغيير رقم قسم المحاسبة في جدول الأقسام إلى 5 سيتم تعديله أيضاً في جدول الموظفين المرتبط به بشكل تلقائي)

✓ **فرض تتالي الحذف:**

عند اختيار **فرض التكامل المرجعي مع تتالي الحذف** يتم السماح بحذف قيمة من حقل مفتاح رئيسي بحيث يتم الحذف في الجدول الأساسي و الجداول المرتبطة .

مثلاً : عند حذف سجل قسم التسويق من جدول الأقسام يتم حذف كافة سجلات الموظفين اللذين يعملون في القسم من جدول الموظفين (يتم حذف سجل ماهر و جمال)

ثانياً : الاستعلامات

الاستعلام هو وسيلة لاسترجاع بيانات داخل جدول أو أكثر وفق شروط معينة، أو إجراء عملية معينة على البيانات.

أنواع الاستعلامات :

1. **استعلامات التحديد :** مهمتها عرض حقول معينة تحقق شروطاً تجيب على استفسار المستخدم (لا تغير في بيانات الجدول).
2. **الاستعلامات الإجرائية أوالتنفيذية** : مهمتها تطبيق عملية معينة على بيانات جدول أو أكثر (كالحذف أو التحديث أو الإضافة أو الإنشاء)

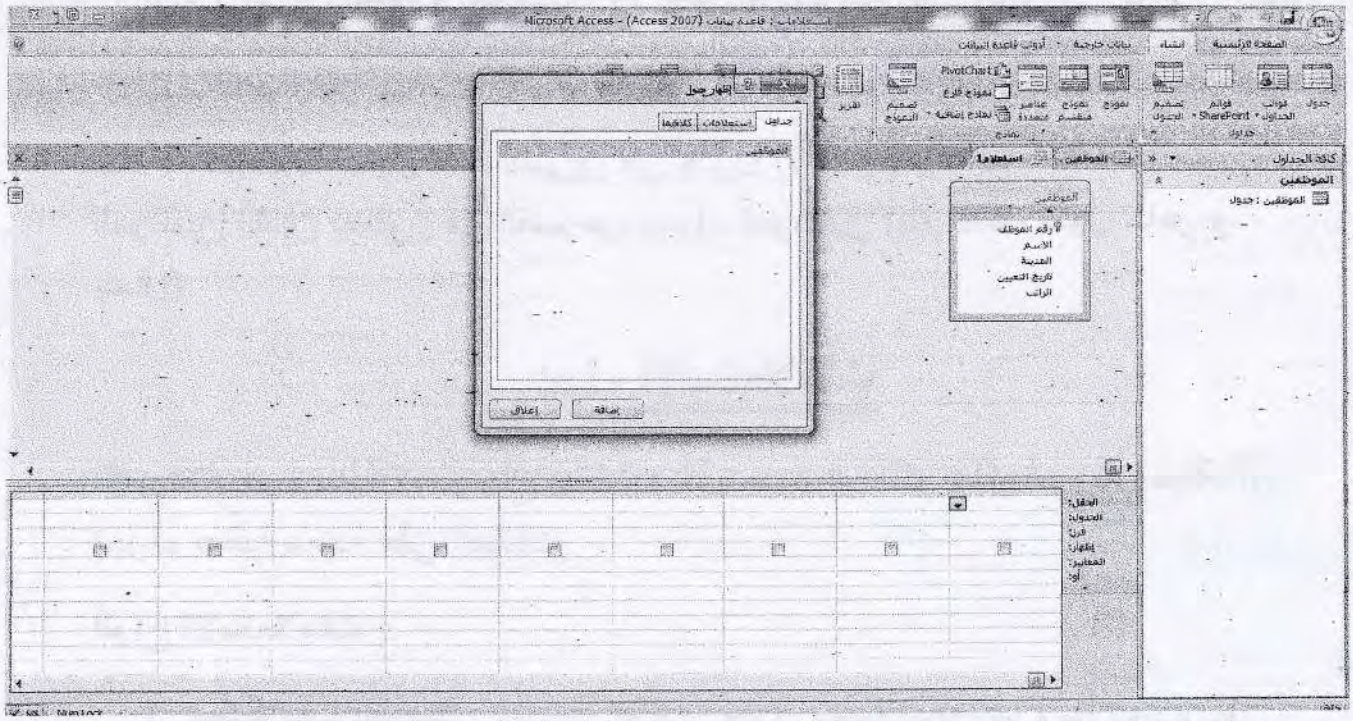
1. استعلامات التحديد

من خلال استعلام التحديد **يمكننا** :

أ - عرض بعض القيم في بعض الحقول حسب الشروط المطبقة .
ب - تجميع البيانات في فئات ثم تطبيق دالة حسابية على كل فئة.
ت - فرز البيانات المستنتجة.
ث - تشكيل حقول محسوبة غير موجودة في الجداول الأصلية.

تصميم الاستعلام :

من **إنشاء** نختار **تصميم الاستعلام** تظهر نافذة نختار منها الجدول الذي سيتم بناء الاستعلام عليه ، نحدد الجدول ونختار إضافة ثم إغلاق.



أقسام النافذة الرئيسية لتصميم استعلام التحديد :

الحقل: لإضافة أسماء الحقول المطلوبة و تشكيل الحقول المحسوبة.

الجدول: يبين أسماء الجداول التي تمت إضافة الحقول منها.

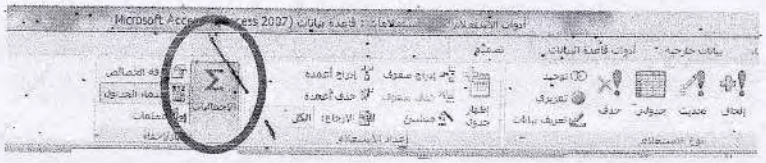
فرز: لتحديد نوع فرز السجلات وفق حقل معين (تصاعدي أو تنازلي).

إظهار : لتحديد الحقول المطلوب عرضها والحقول غير المطلوب عرضها عند تنفيذ الاستعلام.

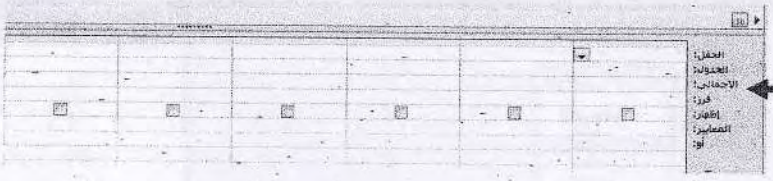
المعايير : لكتابة الشروط الملائمة تحت كل حقل حسب الحاجة .

أو : تستخدم لكتابة أكثر من شرط .

وعند اختيار الإجماليات



سيظهر السطر الإجمالي



(يمكن من خلاله تجميع البيانات إلى فئات ثم تطبيق دالة على كل فئة)

مثال:

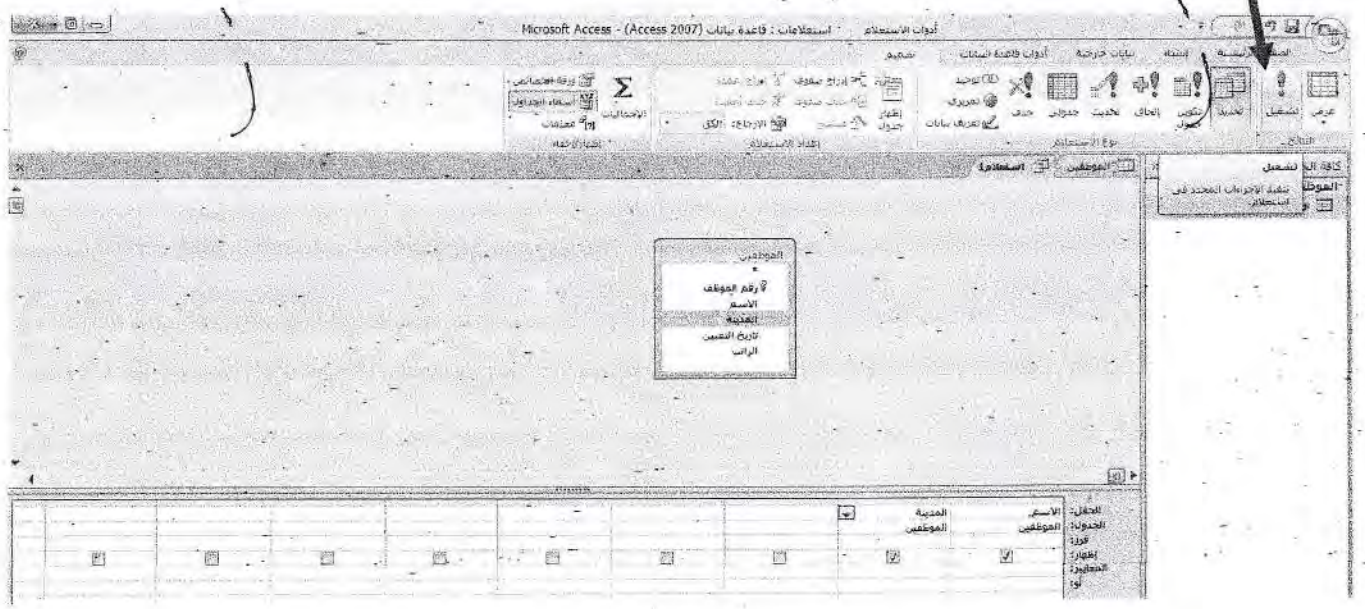
(جدول الموظفين)

رقم الموظف	الاسم	المدينة	تاريخ التعيين	الراتب
1	سعيد	حماه	5/11/2008	40000
2	رامي	حلب	14/3/2000	50000
3	نور	حمص	1/6/2010	35000
4	راما	حماه	1/1/2015	25000
5	سارة	حمص	24/8/2005	37000
6	أحمد	حماه	16/9/2015	25000

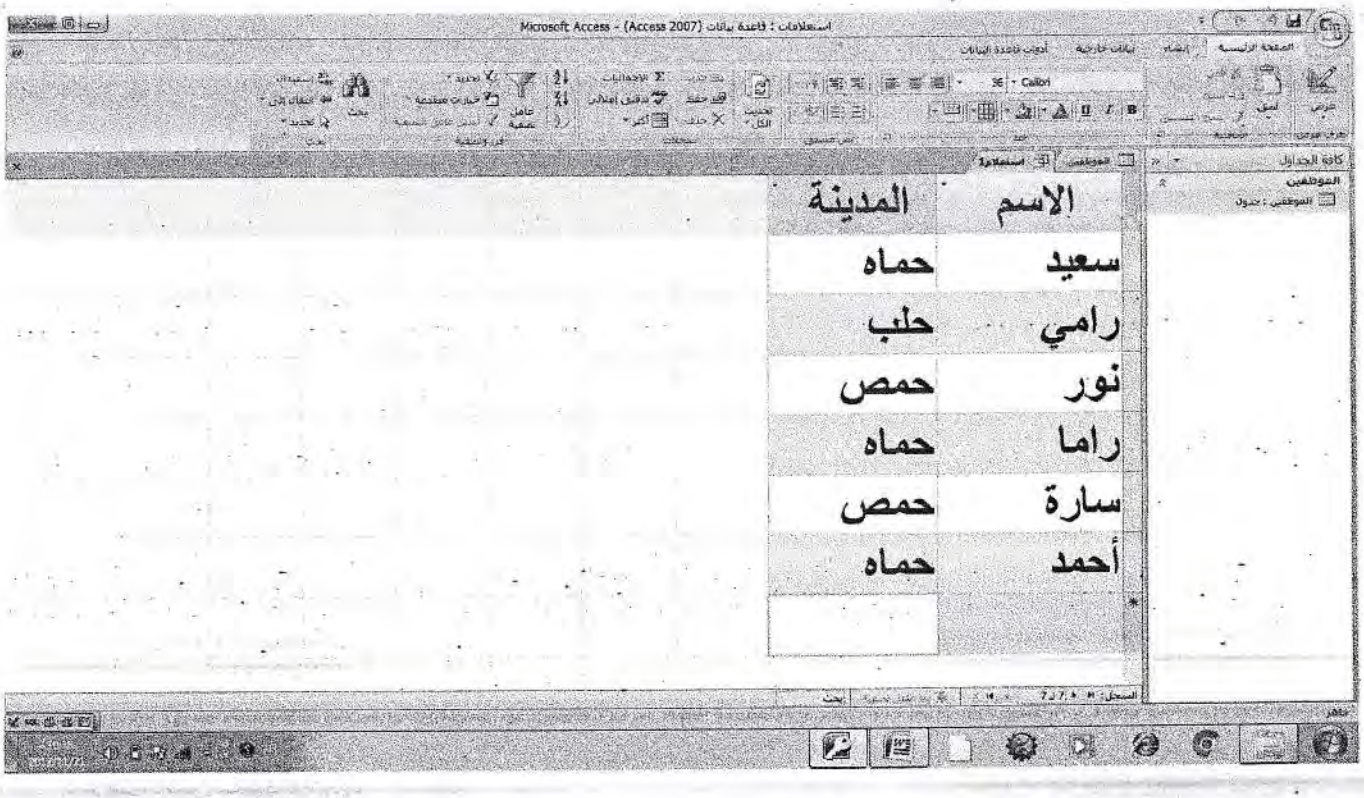
1. لتصميم استعلام يظهر الاسم و المدينة فقط :

نقوم باختيار الحقلين المطلوبين ، و لرؤية نتيجة الاستعلام نضغط على تشغيل

تشغيل:



ستظهر نتيجة الاستعلام كما في الشكل :



الاسم	المدينة
سعيد	حماه
رامي	حلب
نور	حمص
راما	حماه
سارة	حمص
أحمد	حماه

2. لإنشاء استعلام يظهر أسماء الموظفين اللذين مدينتهم حماه بحيث تكون السجلات مرتبة تصاعدياً حسب الاسم

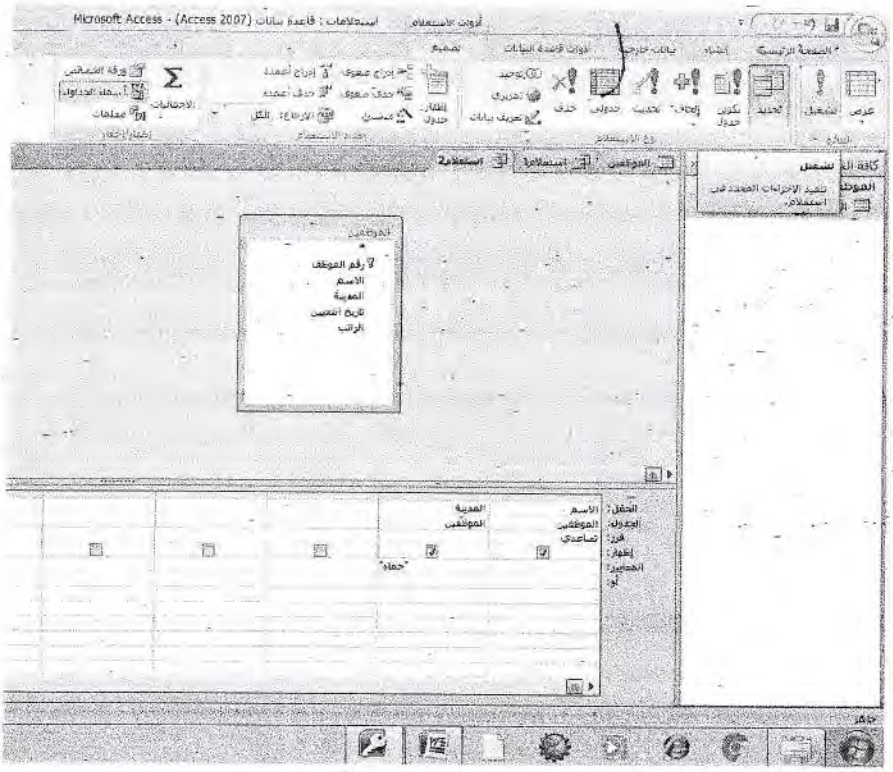
بعد اختيار الحقلين :

الاسم والمدينة،

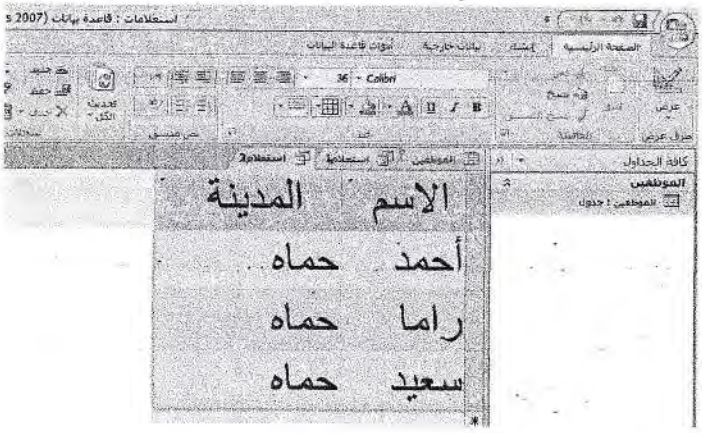
نختار تحت حقل الاسم عند الفرز

فرز :تصاعدي

و تحت حقل المدينة في المعايير: "حماه"



وعند تشغيل الاستعلام تظهر النتيجة :



الاسم	المدينة
أحمد	حماه
راما	حماه
سعيد	حماه

3. لإظهار بيانات الأشخاص اللذين يبدأ اسمهم بحرف السين

نكتب في معايير حقل الاسم : like "س*"

أما لإظهار الأسماء التي تنتهي بحرف الدال :

نكتب في معايير حقل الاسم : like"*د"

4. لعرض الأشخاص اللذين رواتبهم بين 30000 و 40000

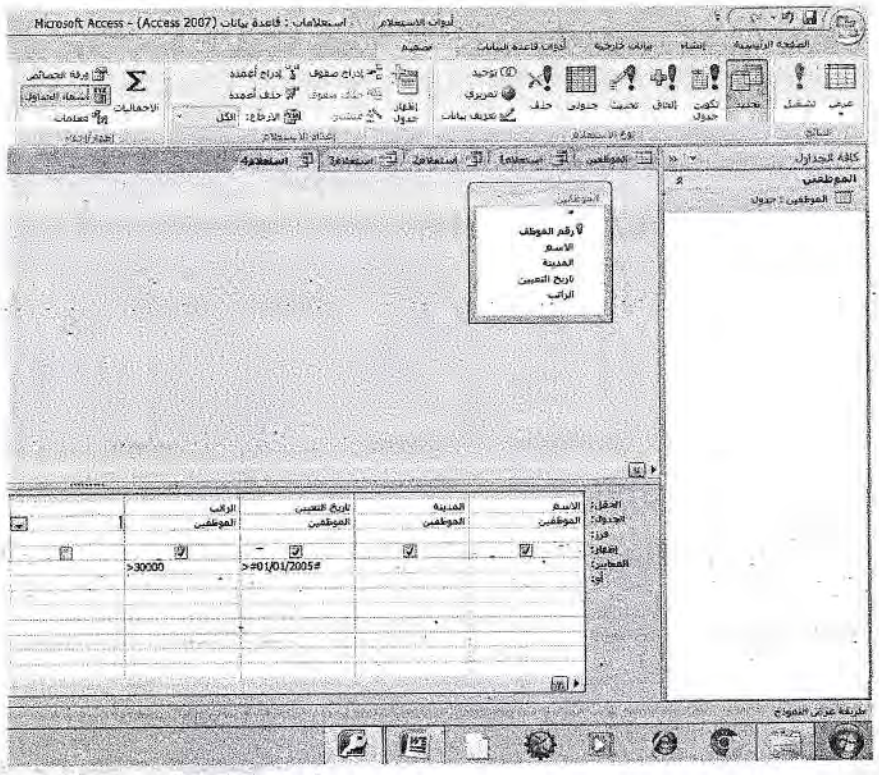
نكتب تحت حقل الراتب في

المعايير : Between 30000 And 40000

وعند التشغيل تظهر نتيجة الاستعلام :

رقم الموظف	الاسم	المدينة	تاريخ التعيين	الراتب
1	سعيد	حماه	05/11/2008	40000
3	نور	حمص	01/06/2010	35000
5	سارة	حمص	24/08/2005	37000
(جديد)				

5. لإظهار الأشخاص اللذين تم تعيينهم بعد تاريخ 1/1/2005 و راتبهم أكبر من 30000



هنا لدينا معياران يجب أن يتحققا **معاً** وبالتالي تتم كتابتهما على نفس السطر

لذا سنكتب تحت حقل تاريخ التعيين في

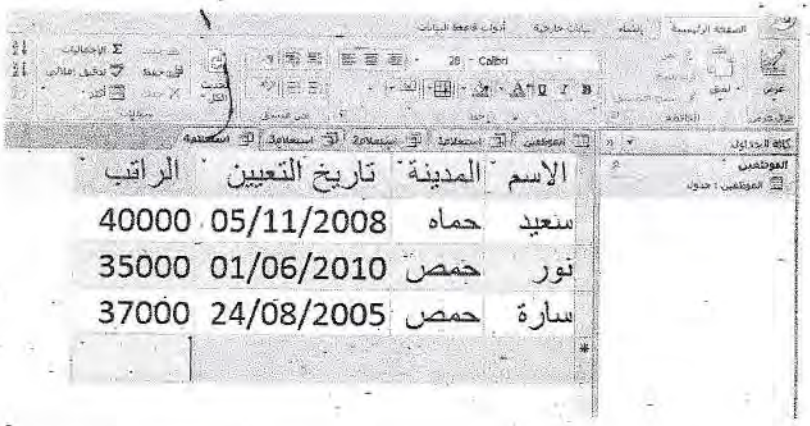
المعايير :

>#1/1/2005#

و نكتب تحت حقل الراتب في

المعايير : >30000

وعند التشغيل تظهر النتيجة:



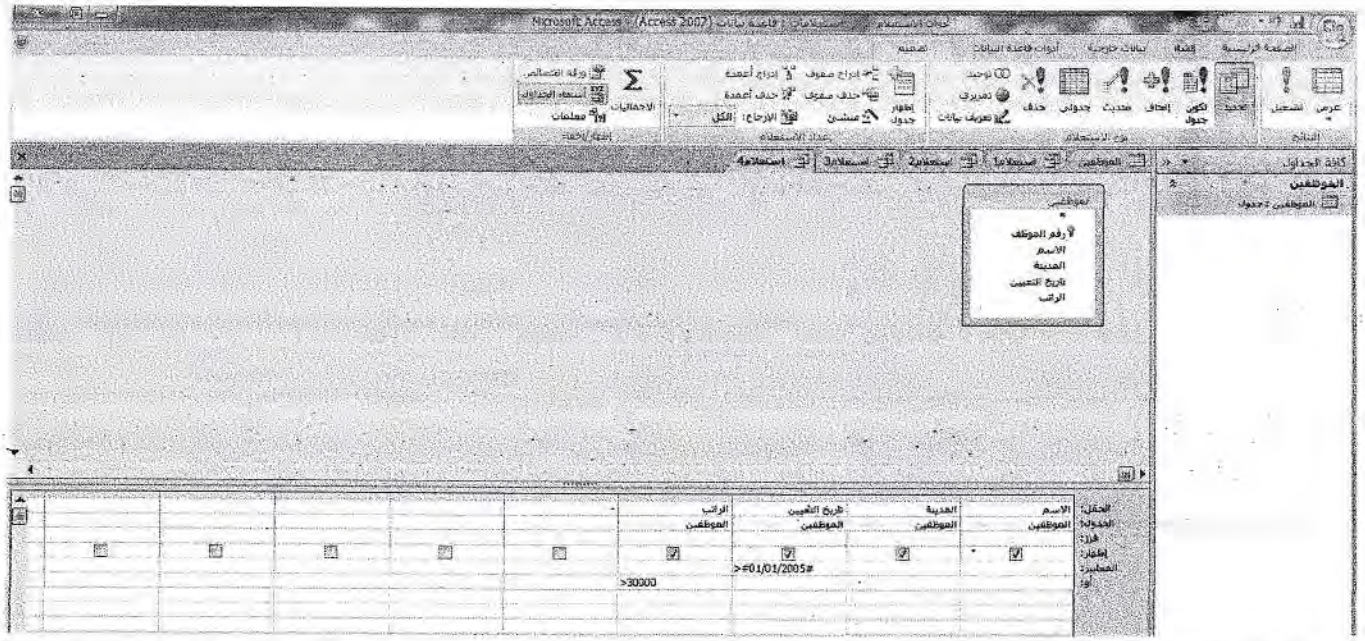
الاسم	المدينة	تاريخ التعيين	الراتب
سعيد	حماه	05/11/2008	40000
نور	حمص	01/06/2010	35000
سارة	حمص	24/08/2005	37000

6. لإظهار الأشخاص اللذين تم تعيينهم بعد تاريخ 1/1/2005 أو راتبهم أكبر من 30000

في هذه الحالة لدينا معياران يجب أن يتحقق واحد منهما على الأقل

يتم كتابة المعيار الأول تحت حقل التاريخ في المعايير: #1/1/2005#>

أما المعيار الثاني الخاص بالراتب نكتبه في سطر **أو** : 30000>



وعند التشغيل تظهر النتيجة :

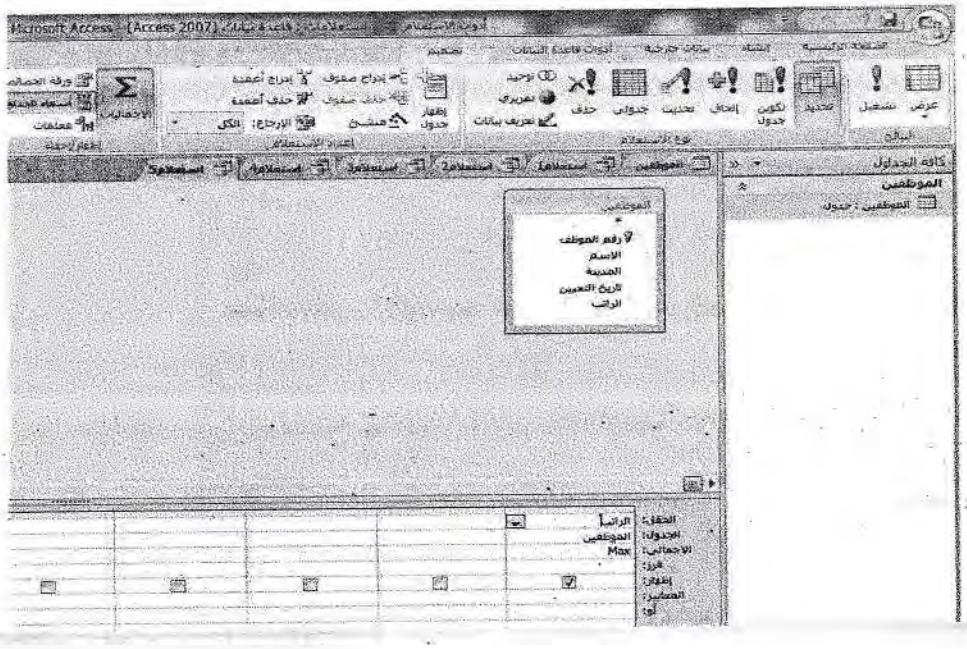
الاسم	المدينة	تاريخ التعيين	الراتب
سعيد	حماه	05/11/2008	40000
رامي	حلب	14/03/2000	50000
نور	حمص	01/06/2010	35000
راما	حماه	01/01/2015	25000
سارة	حمص	24/08/2005	37000
أحمد	حماه	16/09/2015	25000

7. لعرض أكبر راتب:

نضغط على الإجماليات 

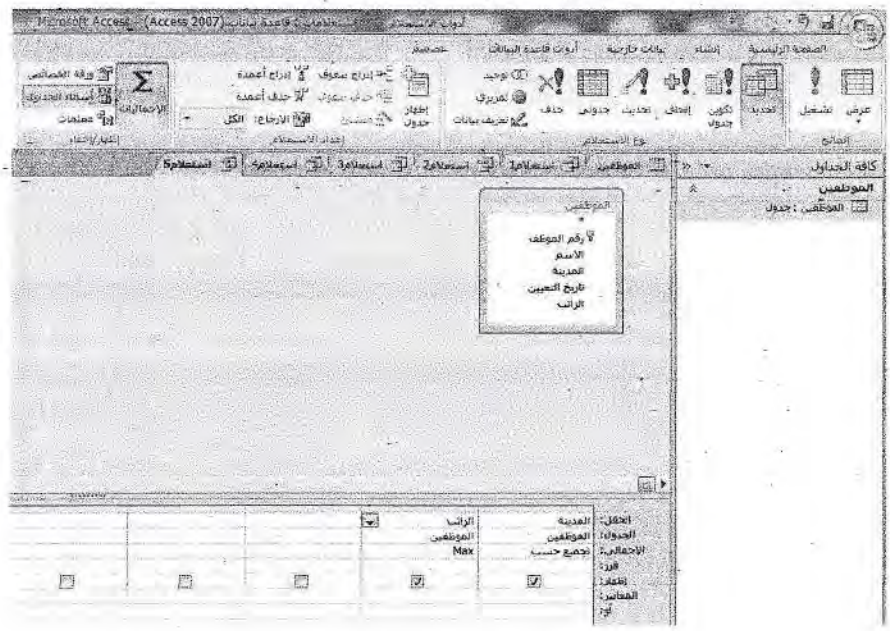
وبالتالي سيضاف سطر جديد بإسم إجمالي

نختار منه الدالة **Max** تحت حقل الراتب



وعند التشغيل سيظهر أكبر راتب:

8. لعرض أكبر راتب في كل مدينة



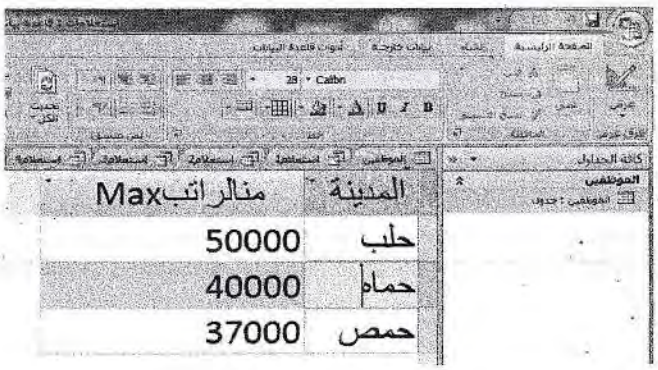
نختار الحقلين : المدينة و الراتب

نضيف سطر الإجمالي

تحت حقل المدينة في سطر الإجمالي نختار: **تجميع حسب**

تحت حقل الراتب في سطر الإجمالي نختار: **Max**

النتيجة :



المدينة	منالراتبMax
حلب	50000
حماه	40000
حمص	37000

يمكن استخدام دوال أخرى مثل (Sum ,Min, Count, Avg, ...)

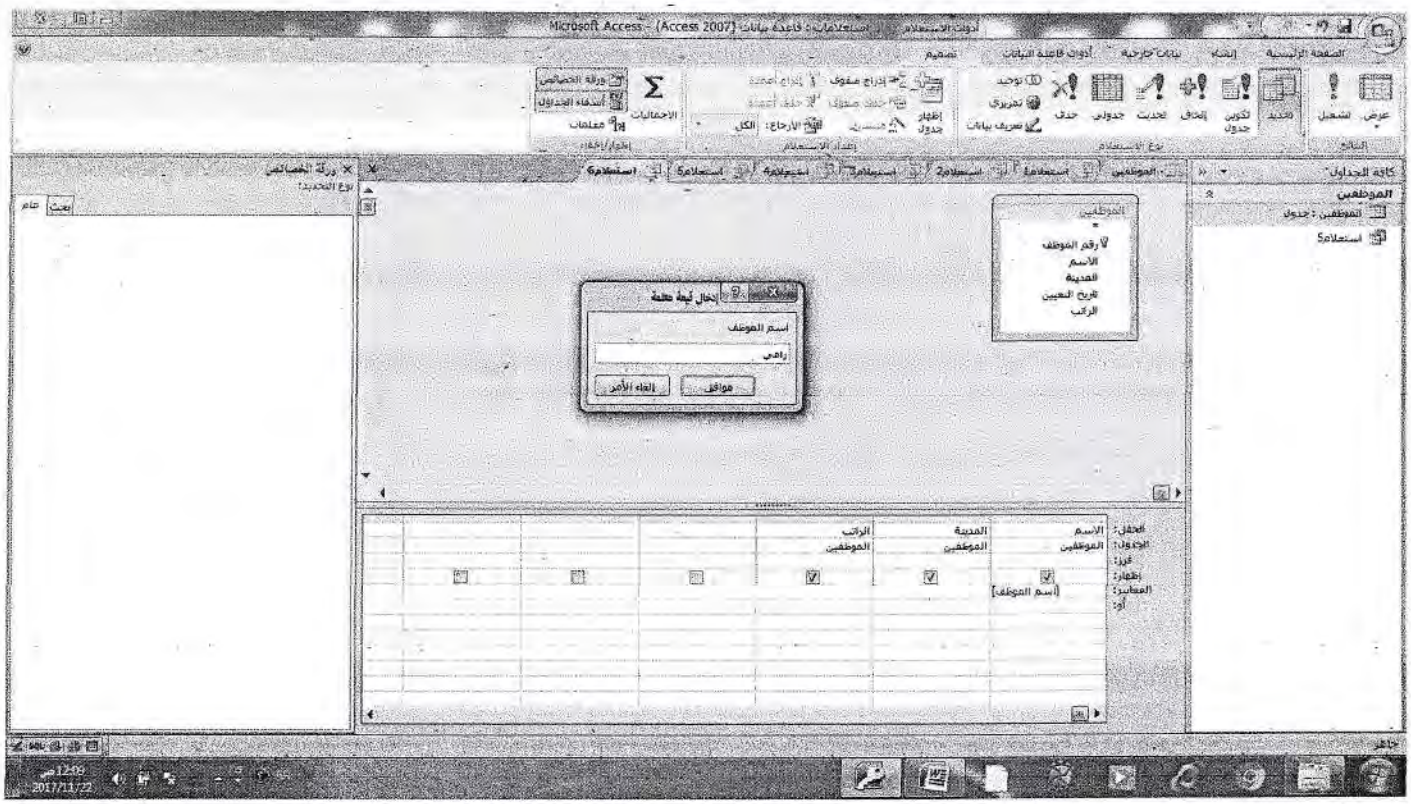
الاستعلام حسب قيمة مدخلة

9. لإنشاء استعلام عن موظف يتم إدخال اسمه عند التنفيذ و ذلك من خلال إظهار رسالة تطلب تسجيل القيمة المطلوبة

نكتب تحت حقل الاسم في سطر المعايير :

[اسم الموظف]

و عند التشغيل يظهر صندوق حوار يطلب تحديد اسم الموظف



و عند تسجيل اسم الموظف المطلوب ينقلنا إلى سجله :

إضافة حقل محسوب إلى الاستعلام :

نكتب اسم الحقل المحسوب متبوعاً بنقطتين ، ثم نكتب العبارة الخاصة بحساب هذا الحقل.

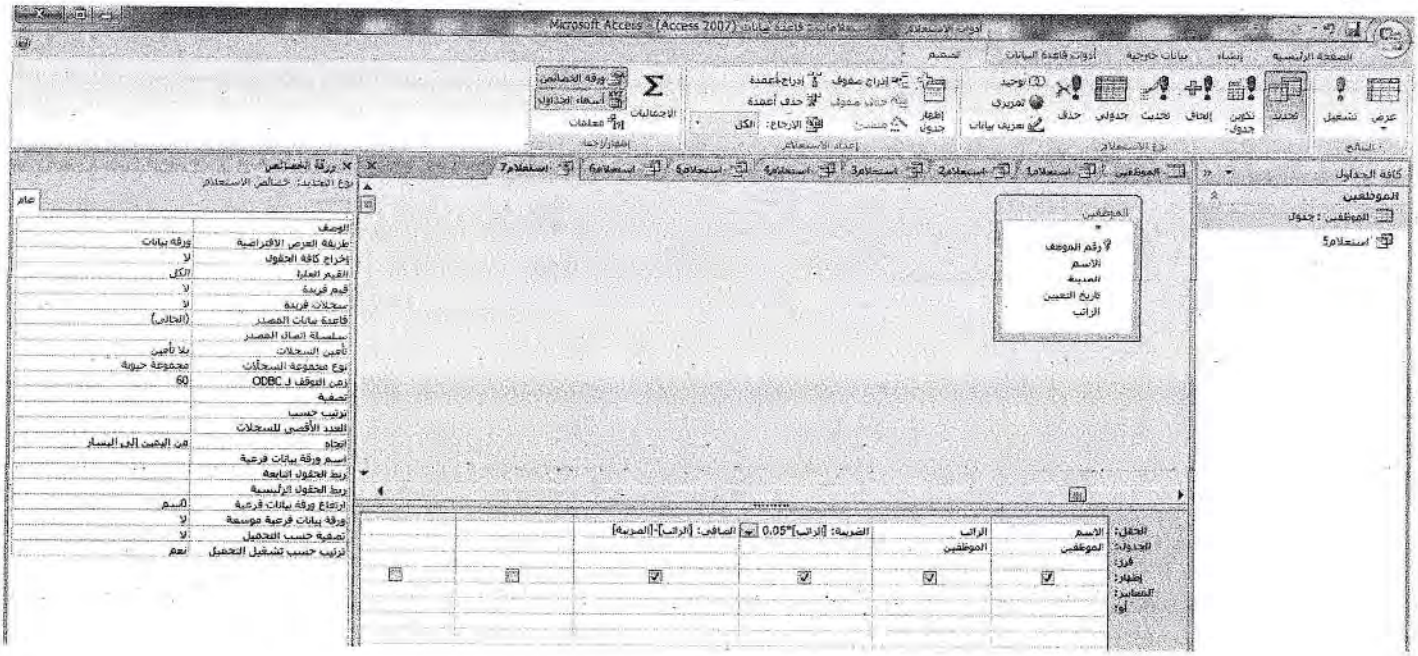
10. لإضافة حقل الضريبة وحقل الراتب الصافي (بفرض أن الضريبة 5% من الراتب).

بعد تصميم استعلام يحتوي على حقل الاسم والراتب، ننشئ الحقل الحسابي بكتابة:

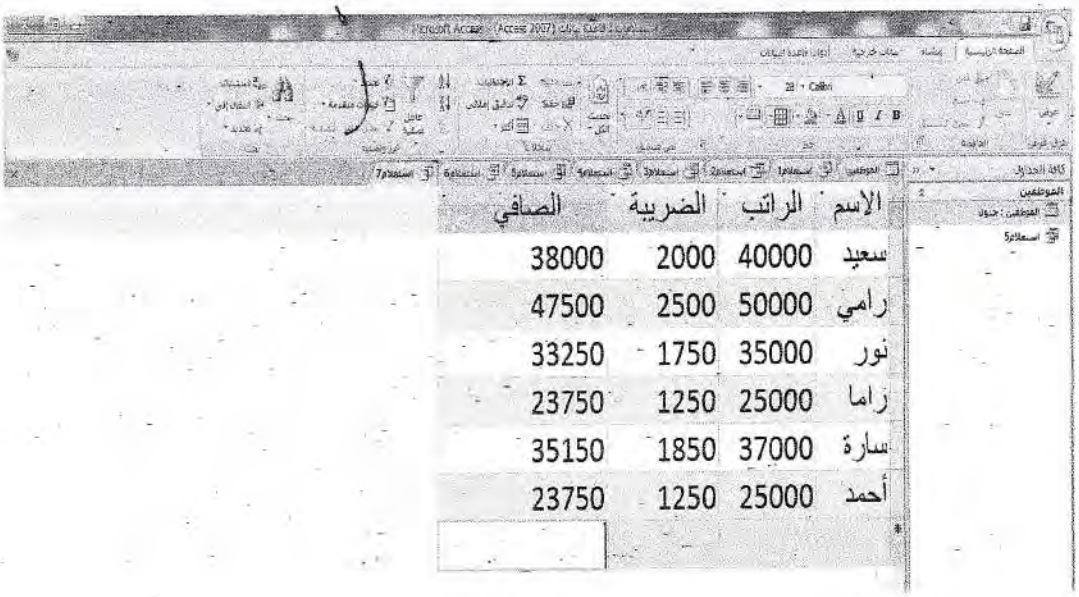
الضريبة:[الراتب]* 0.05

ولإضافة حقل الراتب الصافي نكتب :

الراتب الصافي: [الراتب]-[الضريبة]



وعند تشغيل الاستعلام تظهر النتيجة كما يلي :



الإسم	الراتب	الضريبة	الصافي
سعيد	40000	2000	38000
رامي	50000	2500	47500
نور	35000	1750	33250
راما	25000	1250	23750
سارة	37000	1850	35150
أحمد	25000	1250	23750

2. الاستعلامات الإجرائية

لا تقوم الاستعلامات الإجرائية بعرض البيانات المحققة لشروط معينة كما في استعلامات التحديد، وإنما تقوم بتنفيذ عملية معينة على بيانات الجداول مثل (تحديث البيانات، أو حذفها، أو إضافة بيانات جديدة، أو حتى إنشاء جداول جديدة).

أ - استعلامات التحديث:

يقوم هذا الاستعلام بتحديث بيانات جدول معين أو أكثر وفق شروط معينة .

مثال: (بناءً على جدول الموظفين السابق)

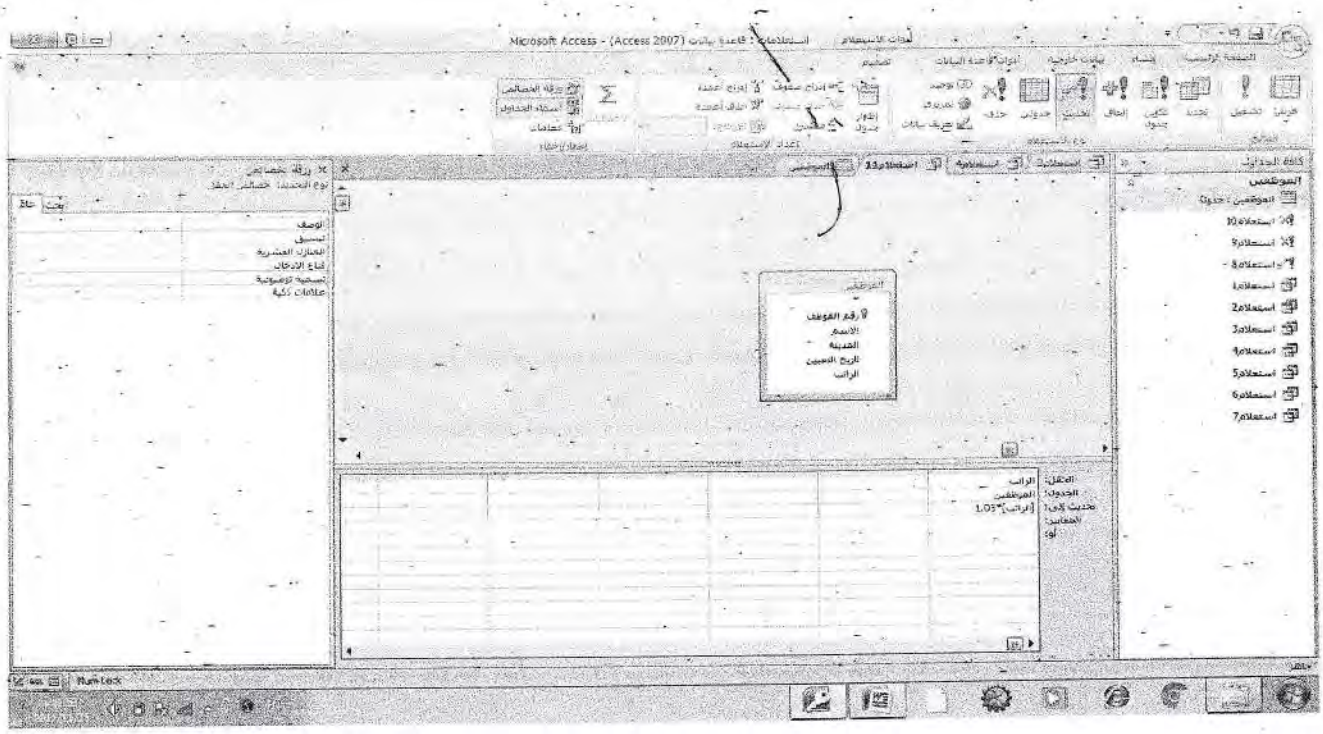
لإنشاء استعلام يقوم بزيادة الرواتب بنسبة 3%

تصميم الاستعلام:

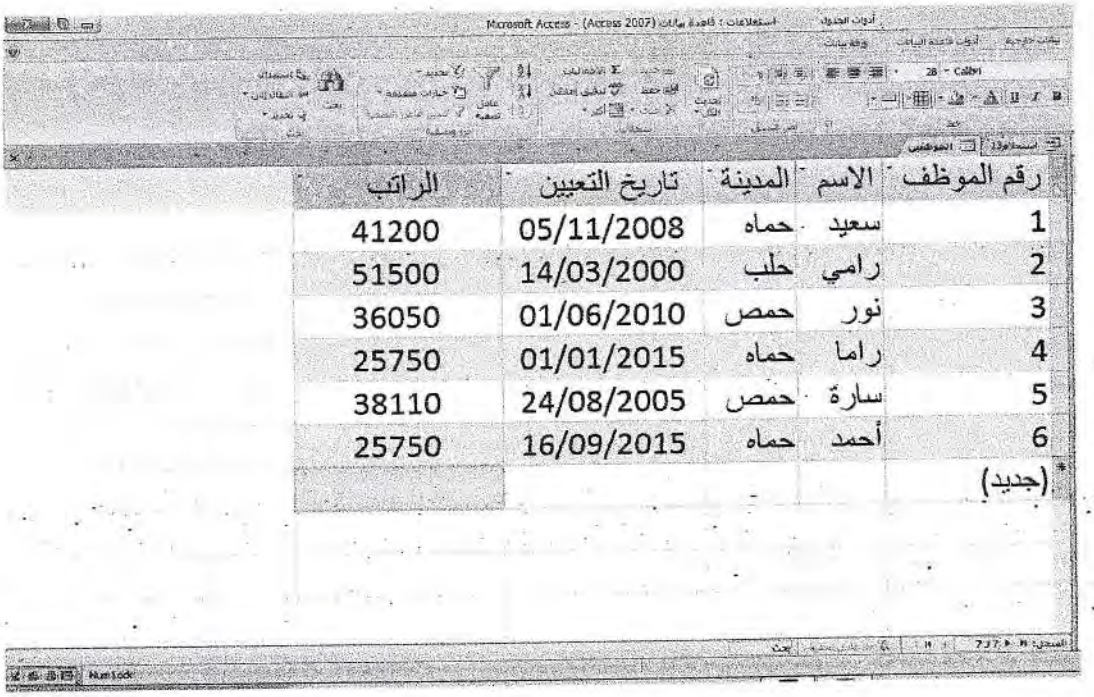
من إنشاء نختار تصميم الاستعلام وبعد إضافة جدول الموظفين نختار نوع الاستعلام **تحديث** وبالتالي سيضاف سطر جديد باسم **تحديث إلى**

نضيف حقل الراتب و نكتب عند **تحديث إلى: [الراتب]+[الراتب]*0.03**

أو ممكن كتابته بالشكل : **[الراتب]*1.03**



بعد تشغيل الاستعلام نقوم بفتح جدول الموظفين فنجد أن الرواتب قد تم تحديثها وزيادتها بنسبة 0.03



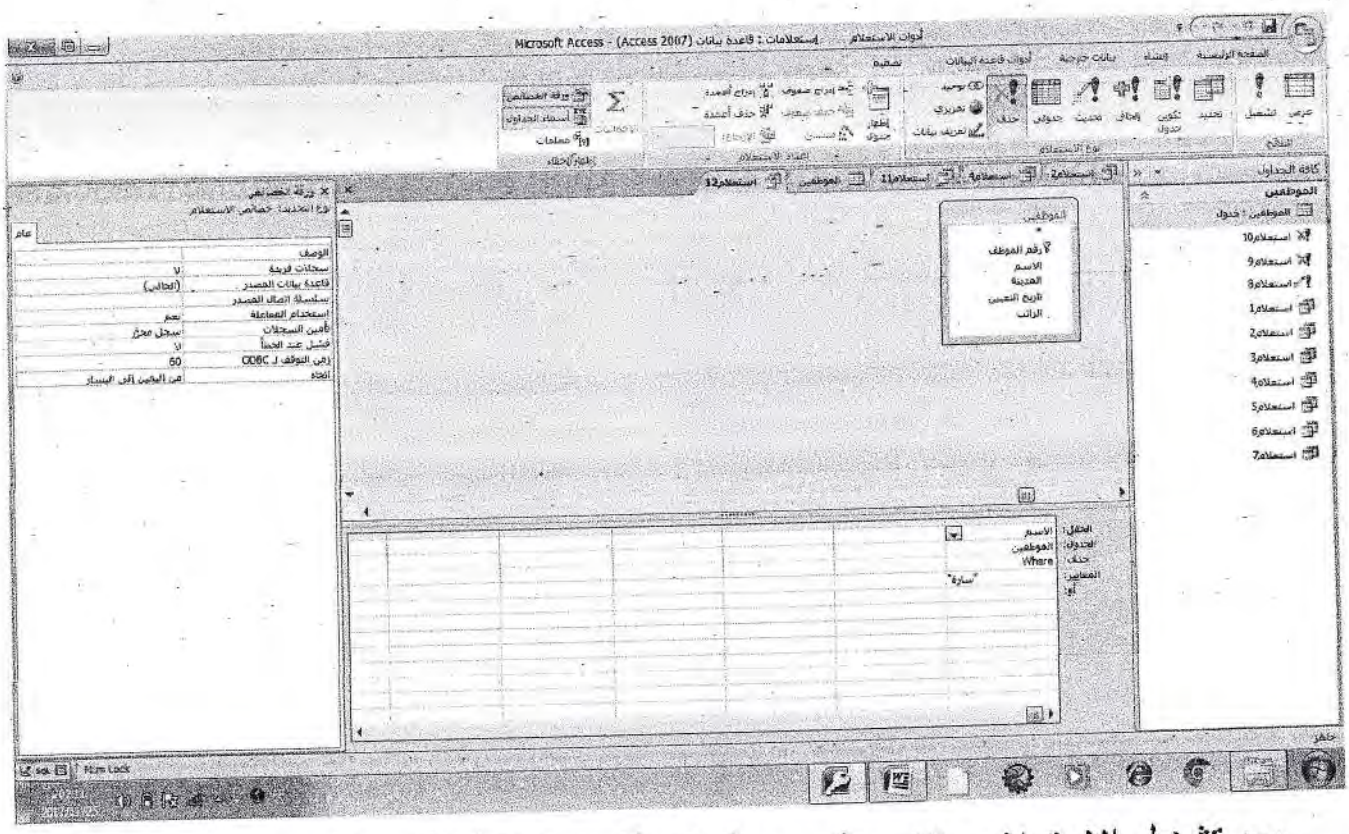
رقم الموظف	الاسم	المدينة	تاريخ التعيين	الراتب
1	سعيد	حماه	05/11/2008	41200
2	رامي	حلب	14/03/2000	51500
3	نور	حمص	01/06/2010	36050
4	راما	حماه	01/01/2015	25750
5	سارة	حمص	24/08/2005	38110
6	أحمد	حماه	16/09/2015	25750
(جديد)				

ب - استعلامات الحذف:

يقوم هذا الاستعلام بحذف بيانات من جدول وفق معايير معينة.

مثال: لحذف سجل سارة من جدول الموظفين، من صفحة تصميم الاستعلام وبعد اختيار جدول الموظفين، نختار نوع الاستعلام **حذف**، سيضاف سطر حذف

ثم نختار حقل الاسم وعند **المعايير نكتب "سارة"**



بعد تشغيل الاستعلام ، نفتح على جدول الموظفين فنلاحظ أن سجل سارة تم حذفه.

ثالثاً: النماذج

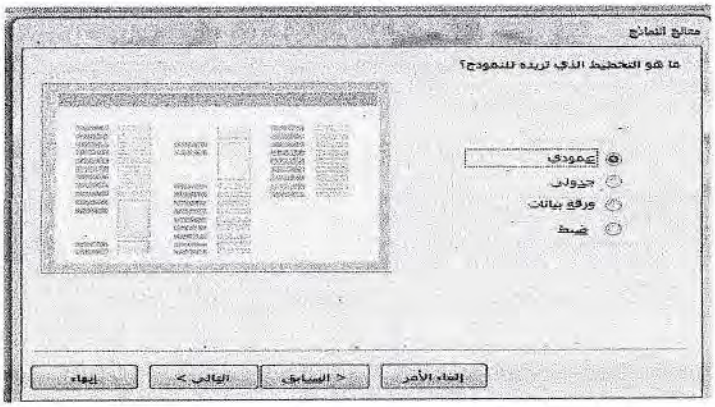
النموذج: هو من كائنات قاعدة البيانات، يستخدم لعرض البيانات أو إضافتها أو تعديلها أو حذفها من جدول أو أكثر أو من استعلام ، فهو الواجهة الأساسية لمستخدم البيانات.

إنشاء نموذج باستخدام معالج النماذج:

من تبويب إنشاء نختار معالج النماذج فتظهر نافذة نختار منها الجدول أو الاستعلام المطلوب بناء النموذج عليه ثم نختار الحقول المطلوبة ثم التالي:

من هذه النافذة نختار التخطيط الذي نريده:

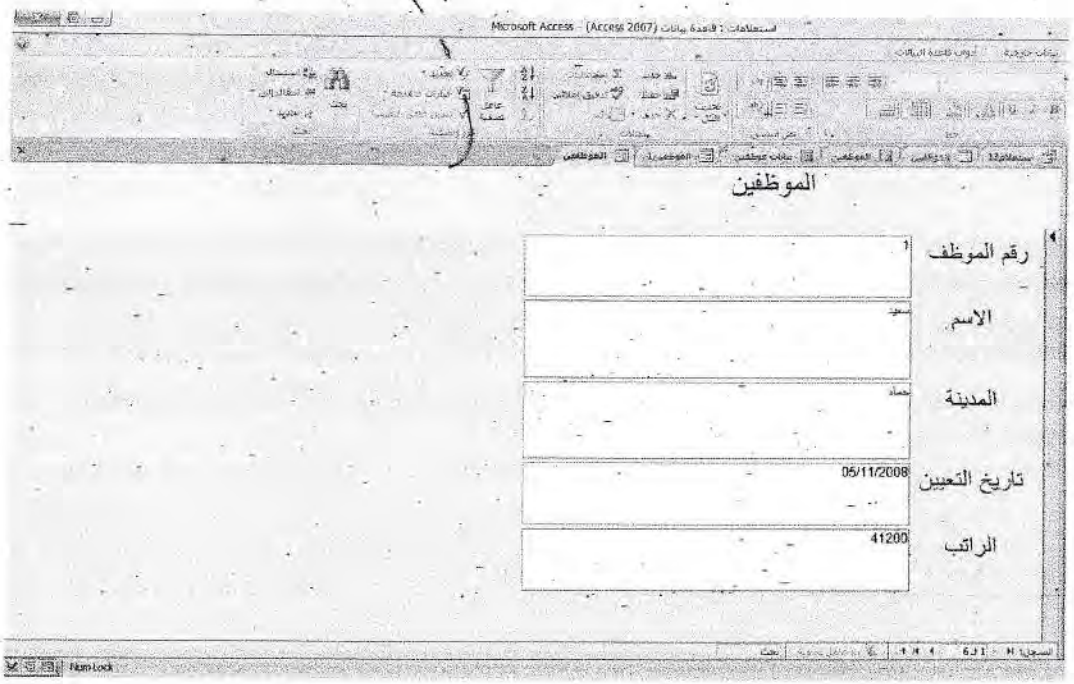
ثم التالي :



نختار النمط ثم التالي

ثم إنهاء.

فيظهر النموذج كما في الشكل:



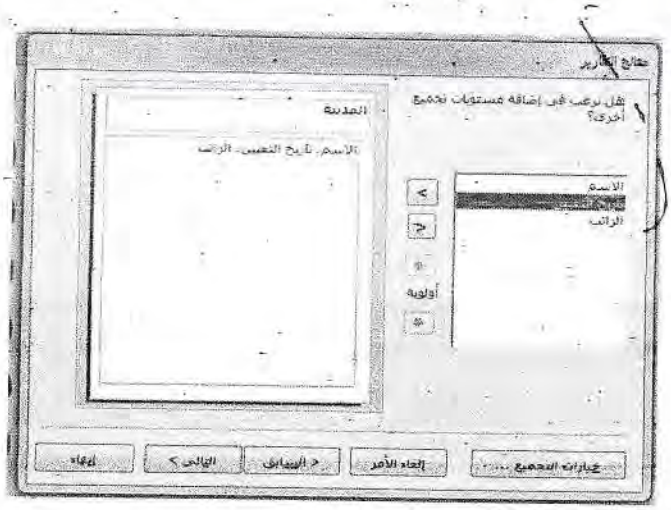
رابعاً: التقارير

يقدم التقرير معلومات مأخوذة من قاعدة البيانات(جدول أو أكثر من جدول أو من استعلام) لتتم طباعتها، ولا يمكن التعديل على البيانات أو حذفها من خلال التقرير.

إنشاء تقرير بطريقة معالج التقارير:

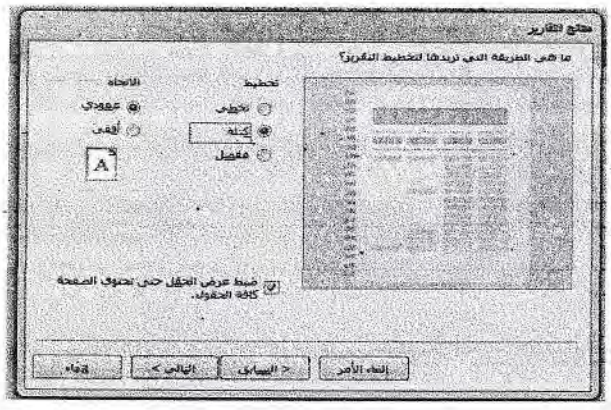
من تبويب إنشاء نختار معالج التقارير فتظهر نافذة نختار منها الجدول أو الاستعلام المطلوب ثم نختار الحقول المطلوبة في التقرير

من هذه النافذة نختار الحقل الذي نريده كنمط للتجميع، وفي مثالنا سنختار حقل المدينة:



ومن هذه النافذة نختار الحقل الذي نريد الفرز بناءً عليه: (وليكن تصاعدي حسب تاريخ التعيين)

نختار نوع التخطيط : ثم التالي ثم نختار النمط :



نكتب اسم للتقرير وليكن(بيانات موظفين) ثم إنهاء:

فيظهر التقرير كما الشكل: