سلسلة الموسوعة العربية للكتب الإلكترونية

جمع و إعداد خليل خليل
تنسيق خليل خليل
عضو بالموسوعة العربية للكمبيوتر و الإنترنت

الطبعة الأولى 2004
نسخة تجريبية غير كاملة

للاستفسار على البريد الإلكتروني
kalel.kalel@mail.sy

إهداء

الإنسان أشبه بذرة اليورانيوم حيث أن وزنها لا يذكر، و لا ترى بالعين المجردة، و ما أن تنفجر هذه الذرة حتى تصدر طاقة هائلة، و لكن كيف لها أن تتفجر ؟
لابد إذن من وجود الصاعق، و هذا الصاعق كان أخي **الدكتور صلاح الدين خليل** الذي قدم لي كل الدعم المادي و المعنوي خلال دراستي، و الذي لولاه بعد الله سبحانه و تعالى لما تمكنت من كتابة هذه السطور لذلك أقدم له هذا الكتاب كهدية معنوية تذكيراً بوفائي لوقوفه بجانبي في السراء و الضراء، و الله وحده يعلم أنني مهما قدمت له لن أعوضه عن جزء من معروفه لي.

و إلى أخي **مرشد محمد** نائب المشرف العام على **الموسوعة العربية للكمبيوتر والانترنت** الذي ساعدني في نشر هذا الكتاب عبر موسوعتنا الحبيبة، و إلى إخواني أعضاء موسوعتنا الكرام.

و أهدي هذا الكتاب أيضاً إلى كل مسلم و مسلمة راجياً من الله تعالى أن تصل الفائدة و المعلومة لكل واحد منهم.

أخوكم في الله أبو عمر

الفهرس

الفصل الأول : مفاهيم في قواعد البيانات العلائقية

الفصل الثاني : العتاد اللازم لتطوير صفحات الويب

الفصل الثالث : لغة تعريف البيانات DDL

الفصل الرابع : لغة معالجة البيانات DML

الفصل الخامس : الاستعلامات في MySQL

مقدمة الكتاب

بسم الله الرحمن الرحيم

الحمد لله الواحد الأحد الفرد الصمد الذي لم يلد و لم يولد و لم يكن له كفواً أحد .. الحمد لله كما ينبغي لجلال وجهه و عظيم سلطانه .. و الصلاة و السلام على سيدنا محمد خاتم النبيين و قائد الغر المحجلين و على آله الطيبين الطاهرين و على صحبه الغر الميامين أجمعين و من تبعهم بإحسان إلى يوم الدين .. اللهم آمين

أما بعد
لم أجد حتى كتابة هذه السطور في مكتبتنا العربية كتاباً يتحدث عن قواعد البيانات **MySQL** بشكل مفصل، حيث وجدت أن معظم الكتب تتحدث عن الـ **PHP** و من ضمنها **MySQL** بشكل بسيط، معتمدين على قواعد بيانات جاهزة تأتي مع برامج جاهزة مثل منتدى **vBulletin** أو عن طريق بناءها بسرعة بواسطة وكيل واجهة التطبيقات الرسومية **PHPMyAdmin** دون الإلمام ببنيتها و كيفية تطويرها و تحديثها، لذلك أحببت أن يكون هذا الكتاب هو الأول من نوعه في الكتب الإلكترونية التي تتحدث بشكل منفصل و مفصل عن قواعد البيانات **MySQL** من الصفر، راجياً من الله تعالى التوفيق و السداد.

و آخر دعوانا أن الحمد لله رب العالمين
أبو عمر

تعريف بالكتاب

تعتبر فصول هذا الكتاب كسلسلة من الحلقات، فإذا فقدت حلقة واحدة فقدت السلسلة كاملة، لذلك أنصحك أخي القارئ بأن تقرأ هذا الكتاب بشكل متسلسل و بتمعن، لأن كل فصل يعتمد على الفصل السابق له، و يتدرج بك الكتاب من الصفر آخذاً بعين الاعتبار أنه ليس لديك أدنى فكرة عن قواعد البيانات العلائقية، فالفصل الأول يشرح المفاهيم الأساسية لقواعد البيانات العلائقية مع تطبيق قاعدة بيانات أثناء التعلم.
و بعد أن يصبح لديك فكرة عن قواعد البيانات ستتعلم في الفصل الثاني كيفية تركيب العتاد اللازم للعمل.
أما باقي الفصول فتتدرج بك لتعلم **MySQL** من الصفر.
و عندما تنتهي من قراءة هذا الكتاب ستكون قد انتقلت إلى نهاية المرحلة المتوسطة في برمجة قواعد البيانات على الإنترنت و ستكون إن شاء الله قادراً على فهم و تحليل قواعد البيانات و بناء قواعد بيانات قوية و كتابه استعلامات معقدة بإذن الله.
أما عن كيفية معالجة المسالك **Threading Handling** و استيراد و تصدير النسخ الاحتياطية لقواعد البيانات، و حماية قواعد البيانات و تكاملها مع **PHP** ، فهذا ما سأكتبه في فصول جديدة، ثم أضيفها إلى كتابي هذا حتى يصبح اسمه بحق **Unleashed MySQL** إن شاء الله.

الفصل الأول
مفاهيم في قواعد البيانات العلائقية

استهلال

إن قاعدة البيانات **Database** هي عبارة عن مجموعة من الجداول **Tables** يتألف كل جدول من أعمدة أو حقول **Columns or Fields** و سجلات أو صفوف أو أسطر **Records** و يؤدي تقاطع العمود مع الصف إلى تكوين الخلية **Cell** حيث يتم تخزين معلومة واحدة فقط في الخلية الواحدة، و هذا تعريف قاعدة البيانات من الناحية الفيزيائية.
أما تعريف قاعدة البيانات من الناحية المنطقية، فهي المستودع الذي يضم و يحوي جميع المعلومات عن منظمة أو شركة أو تنظيم ما ... الخ مهما كان حجم هذا التنظيم أو نوع المعلومات، و لنأخذ على سبيل المثال مدرسة ابتدائية فإن قاعدة البيانات الخاصة بهذه المدرسة ستضم جميع المعلومات التي تتعلق بما يلي:

1. المدرسين: إذ ستضم قاعدة البيانات هذه كل المعلومات عن المدرسين من حيث معلوماتهم الشخصية و المواد التي يقومون بتدريسها و أوضاعهم في المدرسة و كل معلومة تحتاجها المدرسة عن هؤلاء المدرسين، و هذه المعلومات تصنف في جداول و يكون لكل مدرس سجل خاص به.
2. الطلاب: و تضم قاعدة البيانات هذه جميع المعلومات التي تتعلق بالطلاب الدارسين فيها، و أيضاً هذه المعلومات تصنف في جداول، و يكون لكل طالب سجل خاص به.
3. معلومات متفرقة: هي عبارة عن معلومات أخرى تختلف من مدرسة لأخرى كإنجازاتها في مجال الرياضة و المناظرات العلمية و الثقافية ... الخ.

من المؤكد أنك أخي القارئ قد لاحظت وجه الشبه بين قاعدة البيانات التي على الحاسب و قاعدة البيانات الورقية الضخمة الخاصة بأي مدرسة أو منظمة و التي تسمى بالأرشيف، و بالتالي ستدرك مدى أهمية قاعدة البيانات الإلكترونية التي تتمثل في النقاط التالية:

- سهولة تخزين المعلومات في الحاسب.
- سهولة استعادة هذه المعلومات و تطبيق عمليها البحث و الإحصاء عليها و الحصول على التقارير و الرسوم البيانية بسرعة كبيرة جداً.
- سهولة حفظ نسخ احتياطية في مساحة تخزينية صغيرة جداً من حيث الحجم و سهولة نقل هذه النسخة و نقلها إلى حاسب أو شبكة أخرى.

هذا كله في مجال مدرسة فتخيل مدى أهمية قاعدة البيانات في مجال شركة تجارية كبيرة أو مصرف أو وزارة و حتى الوصول إلى الحكومة الإلكترونية.

ما هي قاعدة البيانات العلائقية

هناك الكثير من المعايير التي تتمتع بها قواعد البيانات العلائقية **Relational Databases**، و سأذكر في هذا الكتاب أن شاء الله أهم هذه المعايير، و عند فهمك لها ستصبح بعون الله قادراً على تحليل و تصميم قواعد بيانات متوسطة الحجم و من الممكن أن تحلل و تصمم قواعد بيانات ضخمة، و الله أعلم.
إذن لنبدأ بالتعرف على أهم معايير قواعد البيانات.
ذكرنا سابقاً أن قاعدة البيانات تتألف من جداول و الجداول بدورها تتألف من أعمدة و صفوف، أما الأعمدة فلها بعض الخصائص سنذكرها الآن إن شاء الله.

1. الفتاح الأساسي للجدول **Primary Key**:

في قواعد البيانات العلائقية يجب ألا يتكرر الصف كاملاً بل يجب أن توجد **قيمة فريدة** تميز كل صف عن الصف الآخر، لذلك يجب أن يتواجد في كل جدول عمود واحد على الأقل يحوي قيماً فريدة، و هذا ما نسميه بالمفتاح الأساسي للجدول، و كمثال على ذلك لنفرض أنه لدينا جدول خاص بالموظفين يحوي معلومات شخصية عن الموظفين (الاسم ، العنوان ، الهاتف ... الخ) ستجد أنه من الوارد أن تتكرر معلومات لموظفين على الأقل فمن

الممكن أن يكون لهما نفس الاسم و نفس العنوان و لكن رقم الهاتف هذا غير ممكن، إذن فإن رقم الهاتف هو قيمة فريدة تميز الموظف الأول عن الموظف الثاني و عن الموظف الثالث و هكذا، لذلك سنعتبر رقم الهاتف هو المفتاح الأساسي لجدول الموظفين.

2. المفتاح المركب للجدول:
ندعو المفتاح الأساسي بالمفتاح المركب فقط في حال كان المفتاح الأساسي مكون من أكثر من عمود، فمثلاً إذا اعتبرنا في جدول الموظفين أن المفتاح الأساسي مؤلف من حقل الاسم و الهاتف فإننا ندعوه بالمفتاح الموكب.

3. المفتاح الغريب للجدول **Foreign Key**:
المفتاح الغريب هو اسم يطلق على عمود في جدول، هذا العمود يشير إلى العمود الأساسي في جدول آخر، بحيث يكون للعمود الغريب من الجدول الأول و العمود الأساسي من الجدول الثاني نفس الاسم و النوع و الصفات، فمثلاً لنفرض أنه لدينا جدولين الأول للموظفين و فيه بيانات تتعلق بعملهم و جدول ثان فيه بيانات تتعلق بالمعلومات الشخصية للموظفين، و لنفرض أنك تريد الحصول على كل البيانات المتعلقة بموظف معين سواء الشخصية أو المهنية، فإنك ستكتب استعلاماً يحقق لك ذلك عن طريق المفتاح الأساسي في الجدول الأول و المفتاح الغريب في الجدول الثاني.

4. التكامل المرجعي **referential Integrity**:
إن العلاقة التي ذكرناها سابقاً بين العمود الغريب و العمود الأساسي تدعى بالتكامل المرجعي، فعند إضافة بند إلى الجدول ذي المفتاح الغريب و ليس موجوداً في الجدول ذي المفتاح الأساسي فسيقوم ملقم قواعد البيانات بتنبيهي إلى أنني أقوم بإدراج شيء غير موجود في الجدول الأساسي.

من الأشياء المفيدة التي يقدمها لنا التكامل المرجعي هي عملية الحذف المتسلسل **Cascading Deletes** أي أنك في حال قمت بحذف صف من جدول الموظفين، فإنه تلقائياً سيتم حذف الصفوف المرتبطة معه من جدول عناوين الموظفين، و لكن **MySQL** للأسف لا تدعم هذه الميزة.

5. القيمة **Null**:
في قواعد البيانات العلائقية سنتعرض كثيراً للقيمة **null** و هي تعني اللاقيمة أي لا شيء و هي بالطبع تختلف عن الصفر **Zero** فهو يمثل قيمة و يتم حجز مكاناً له في الذاكرة، و لكن المهم الآن أن تعرف أن **Null** معناها لا قيمة.

6. الفهارس **Indexes**:
الفهرس في قواعد البيانات كالفهرس في الكتاب، و يؤدي نفس وظيفته، ففي الكتاب يساعدنا الفهرس على سرعة الانتقال إلى موضوع معين، و كذلك الأمر في قواعد البيانات فالفهارس تساعدنا على سرعة الانتقال الحصول على معلومات لاستعلام ما.

العلاقات بين الجداول
في قواعد البيانات العلائقية ثلاث أنواع للعلاقات بين الجداول و هي:

1. علاقة رأس برأس **One To One**
هذا النوع من العلاقات يتم بين جدولين على الأقل بحيث يكون صف واحد فقط من الجدول الأول مرتبط مع صف واحد فقط من الجدول الثاني و العكس صحيح، و كمثال على ذلك لنفرض أن لدينا جدول للبلدان و جدول آخر

للعواصم و بين هذين الجدولين يوجد تكامل مرجعي، فكل بلد له عاصمة واحدة فقط و كل عاصمة هي عاصمة دولة واحدة فقط.

علاقة رأس برأس

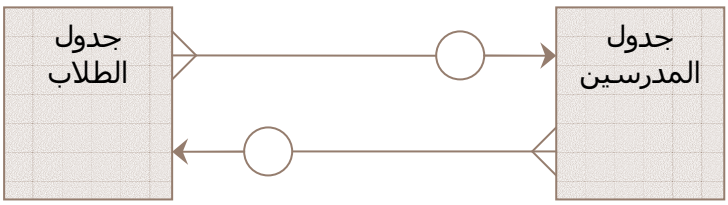
2. علاقة رأس بأطراف **One To Many**

في هذا النوع تكون العلاقة بين الجدولين هي أن كل صف من الجدول الأول له علاقة بصف واحد على الأقل من الجدول الثاني بينما كل صف من الجدول الثاني له علاقة بصف واحد فقط من الجدول الأول، لنأخذ مثال على ذلك جدولين الأول هو جدول المدراء و الثاني هو جدول الموظفين فكل مدير هو مدير على عدة موظفين، بينما كل موظف له مدير واحد مباشر فقط.

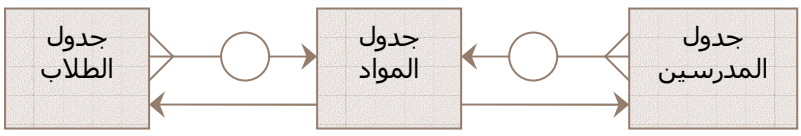
علاقة رأس بأطراف

3. علاقة أطراف بأطراف **Many To Many**

أي أنه كل سجل من الجدول الأول يرتبط بعدة سجلات من الجدول الثاني و العكس صحيح أيضاً، و مثال عليه جدول المدرسين و جدول الطلاب، فكل مدرس هو مدرس لمجموعة طلاب و كل طالب يدرس عليه مجموعة مدرسين، إن هذا النوع من العلاقات هو نوع خاطئ، لذلك يتم في هذه الحالة إنشاء جدول آخر جديد يكون كرابط أو وسيط بين الجدولين السابقين بحيث تكون علاقة هذا الأخير هي علاقة رأس بأطراف مع كل من الجدولين السابقين على حدا.



علاقة أطراف بأطراف



علاقة رأس بأطراف بعد إضافة الجدول الوسيط

لقد لاحظت هذا السهم ذو الدائرة، يدعي هذا السهم بقدم الغراب و هو يستخدم للتعبير عن العلاقة رأس بأطراف.

تحليل و تصميم قاعدة البيانات المثال

خلال تعلمك لهذا الكتاب ستجد أنك قد قمت ببناء قاعدة بيانات الهدف منها هو تطبيق ما ستتعلمه من أوامر **MySQL** عليها.

و في هذا الفصل سنقوم بتحليل قاعدة البيانات ثم تصميمها، و الهدف من ذلك هو أن تأخذ فكرة جيدة عن تحليل و تصميم قواعد البيانات.

إن قاعدة البيانات المثال فهي خاصة بمركز لبيع أفلام الفيديو إلى الزبائن بشكل مباشر أو عب الإنترنت و اسمها **movie_store** و سنقوم بعملية تحليلها و تصميمها بشكل بسيط بعيد عن التعقيد.

كما ذكرت لك هي قاعدة بيانات متخصصة ببيع أفلام الفيديو، فما هي الجداول التي تحتاجها؟

1. ستحتاج إلى جدول يحوي معلومات عن الأفلام التي لديك و هو بمثابة الأرشيف.
2. ستحتاج إلى جدول يحوي معلومات عن الموظفين العاملين في مركزك، فإذا أردت أن تجمع أكبر قدر من المعلومات عنهم فمن الأفضل أن تخصص جدولين لهما بحيث يحوي أحدهما بيانات تتعلق بمجال عملهم و الآخر يحوي بيانات شخصية عنهم.
3. ستحتاج إلى جدول يحوي معلومات عن الموردين الذين تتعامل معهم.
4. جدول يتعلق بالزبائن.
5. جداول خاصة بالحركة المالية تظهر لك المصاريف و الأرباح خلال دورة تحددها أنت إما شهرية أو أسبوعية أو سنوية، بحسب ما تريد (لن نتطرق إلى الجداول المتعلقة بسير الحركة المالية و ذلك لأن الهدف من قاعدة البيانات **movie_store** هو التدريب على أوامر **MySQL**

و بذلك أصبح لدينا قاعدة بيانات اسمها **movie-store** تتألف من خمس جداول هي:

- جدول الأفلام **movies**
- جدول الموظفين **employees**
- جدول عناوين العمال **addresses**
- جدول الزبائن **clients**
- جدول الموردين **suppliers**

و إليك الأعمدة التي تتألف منها هذه الجداول:

الجدول **employee** :

1. رقم الموظف **emp_no** : يخزن في هذا العمود أرقام الموظفين بشكل تسلسلي، و سيشكل هذا العمود الفتاح الأساسي لجدول الموظفين، و يضم **قيماً فريدة** لكل موظف
2. الاسم الموظف **name** : يخزن في هذا العمود اسم الموظف.
3. وظيفته **job** : يخزن في هذا العمود نوع وظيفة الموظف (بائع ، مبرمج ، مدير ... الخ).
4. الراتب المقطوع **salary** : يخزن في هذا العمود قيمة الراتب الذي يتقاضاه الموظف.
5. الحوافز **bonus** : يخزن فيه قيمة الحوافز التي يأخذها الموظف.
6. تاريخ مباشرة العمل **date**

الجدول **address** :

1. رقم عنوان الموظف **add_no** : هذا العمود هو المفتاح الأساسي للجدول، لأنه يضم قيماً فريدة لكل عنوان.
2. رقم الموظف **emp_no** : يحوي هذا العمود أرقام الموظفين العاملين في الشركة، و هو سيكون المفتاح الغريب الذي يشير إلى المفتاح الأساسي في الجدول **employees** لأنه يضم قيمة مرجعية لكل موظف.
4. مسكن الموظف **state**
7. البريد الإلكتروني للموظف **emp_email**

الجدول **supplier** :

1. رقم المورد **sup_no** : يضم قيمة **فريدة** لكل عنوان.
2. اسم المورد **sup_name**.
3. البريد الإلكتروني **sup_email** .

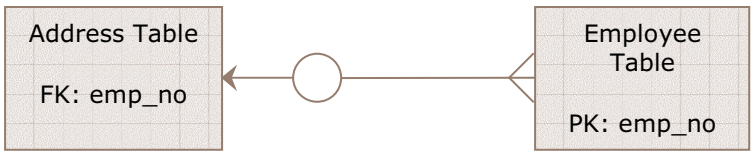
الجدول **client** :
1. رقم الزبون **cli_no** : يضم قيمة **فريدة** لكل زبون.
2. اسم الزبون **cli_name**
3. بريده الالكتروني **cli_email**

الجدول **movie**:
1. رقم الفيلم **mov_no** : يحوي قيماً **فريدة** لكل منتج.
2. اسم الفيلم **mov_name** : يحوي الاسم الخاص بكل منتج.
3. اسم البطل الأول **star_1** : يحوي هذا العمود اسم البطل الرئيسي للفيلم.
4. اسم البطل الثاني **star_2** : يحوي اسم البطل الثاني للفيلم.
6. نوع الفيلم **kind** : هذا العمود يحوي نوع الفيلم (كوميدي، بوليسي، تاريخي، رومانسي، رعب، خيالي، عنف).
7. سعر الفيلم **price** : من المؤكد أنك عرفت أن هذا العمود يضم أسعار المنتجات.

إن قاعدة البيانات هذه لا تضم كل الأعمدة و الجداول المطلوبة أي أن قاعدة البيانات هذه ليس كاملة لأن الهدف الأساسي منها هو تطبيق تعليمات **MySQL** عليها لذلك اخترت الجداول و الأعمدة التي تفي بالغرض فقط.

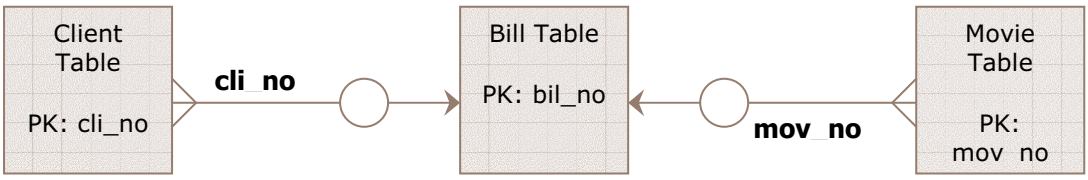
تكوين العلاقات بين قاعدة البيانات المثال

حتى تصبح قاعدة البيانات هذه قاعدة بيانات علائقية يجب أن ترتبط جداولها بعلاقات مع بعضها البعض، و كنت قد تعرفت سابقاً في هذا الفصل على أنواع العلاقات و الآن ستتعلم إن شاء الله كيفية بناء هذه العلاقات.
بالعودة إلى الجداول السابقة نلاحظ أنه يوجد بين الجدولين **employee , address** تكامل مرجعي، إذ أنه يمكن أن يكون لكل موظف أكثر من عنوان بينما العنوان الواحد هو خاص بموظف واحد، إذا العلاقة بين هذين الجدولين هي علاقة رأس بأطراف و التكامل المرجعي بينهما يتم عن طريق العمود **emp_no** كما في الشكل التالي:



إن الرمزين **PK , FK** هما اختصارين يرمزان إلى **Primary Key , Foreign Key** على الترتيب.
و هناك أيضاً علاقة بين الجدولين **movies , clients** حيث أن الزبون عندما يزور موقع المركز فأنه لا يهتم بالموظفين و لا يهتم بالموردين بل يهتم بالمنتجات، فلنفرض أن هناك زبونين يريدان شراء مجموعة من الأفلام بحيث يريد الأول شراء فيلم **face off** و فيلم **assassins** بينما يريد الثاني شراء فيلم **assassins** فقط، لاحظ العلاقة بين **movies , clients** إذ يمكن للزبون الواحد أن يشتري مجموعة أفلام و يمكن للفيلم الواحد أن يشتريه مجموعة زبائن، إذن العلامة بين الجدولين **movies , clients** هي علاقة (أطراف بأطراف) و كما ذكرنا أن هذا النوع من العلاقات هو نوع خاطئ لذلك يتم التحايل عليها عن طريق إنشاء جدول آخر وسيط بين الجدولين السابقين بحيث يشكل هذا الجدول بينهما حلقة وصل.
هذا الجدول اسمه هنا جدول الفواتير **bill** و يتألف من الأعمدة التالية:
1. رقم الفاتورة **bil_no** : حيث يضم هذا العمود قيماً **فريدة** تميز كل فاتورة عن الأخرى مع الأخذ بعين الاعتبار أن الزبون عندما يشترى فيلمين سيسجل فاتورتين و هكذا.
2. رقم الزبون **cli_no** : و يشكل هذا العمود صلة الوصل بين جدول **client** و جدول **bill** أي أنه (**مفتاح غريب**).
3. رقم الفيلم **mov_no** : و يشكل هذا العمود صلة الوصل بين جدول **movie** و جدول **bill**
أي أنه (**مفتاح غريب**).
4. قيمة الفاتورة **paid** : تمثل القيمة المالية للفاتورة.

5. تاريخ تحرير هذه الفاتورة **bil_date** : يذكر التاريخ الذي تم فيه تحرير الفاتورة.
الشكل التالي يظهر العلاقة بين الجداول الثلاث



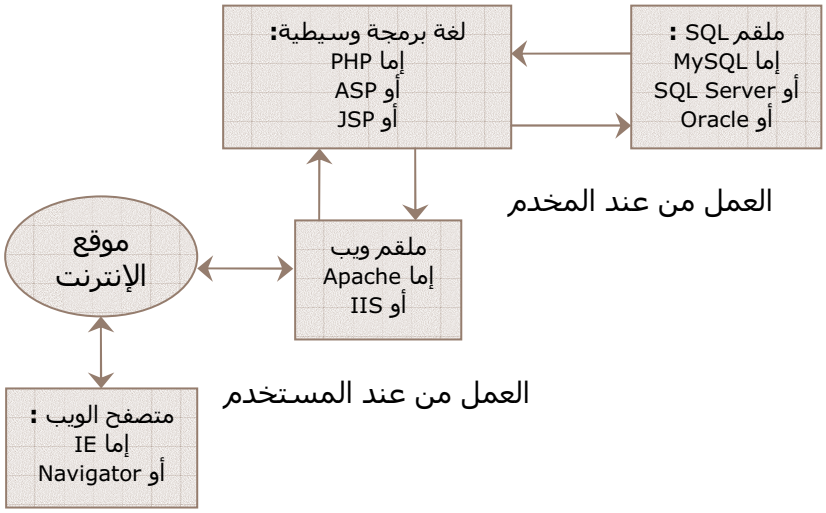
و هكذا أصبح لديك قاعدة بيانات تفي بالغرض و مؤلفة من ستة جداول و هم :
employees , addresses , suppliers , movies , clients , bills
و بهذا نكون بإذن الله قد أخذنا فكرة ميسرة عن تحليل و تصميم قواعد البيانات.

الفصل الثاني
العتاد اللازم لتطوير صفحات الويب

كيف يعمل الويب

بعد أن أخذت فكرة جيدة عن قواعد البيانات العلائقية، ستتعرف الآن على البرمجيات الضرورية للبدء في تعلم **MySQL** ، هناك مجموعة من البرمجيات التي تحقق لك بناء موقع ويب ديناميكي، أي موقع ويب يتفاعل مع الزوار و المستخدمين.

و المخطط التالي يبين لنا الكيفية التي يعمل بها موقع الويب الديناميكي.



من الأخذ بعين الاعتبار ما يلي:

1. **IIS** (**Internet Information Supplier**) : و هو ملقم الويب الخاص بشركة **Microsoft** .
2. **IE** (**Internet Explorer**) : و هو متصفح الإنترنت الخاص بشركة **Microsoft** .
3. **Navigator** : هو متصفح الإنترنت الخاص بشركة **Netscape** .

إذن من خلال المخطط السابق ستجد أنه لتطوير تطبيقات ويب متكاملة ستحتاج إلى العتاد التالي:

1. ملقم قواعد بيانات.
2. لغة برمجة وسيطية **Middleware**
3. ملقم ويب.

ملقم قواعد بيانات:

أفضل الملقمات لقواعد بيانات الإنترنت هو **MySQL** ، و ملقم قواعد البيانات بشكل عام هو ملقم أو مخدم قواعد بيانات **SQL** الذي يحقق التفاعل بين لغة البرمجة الوسيطية و قواعد البيانات، و تقوم ملقمات **SQL** بتخزين البيانات بشكل فعال و باسترجاعها بسرعة كبيرة، فالملقم هو المكان الذي تعشش فيه البيانات، كما و يسهل الملقم عمليات الاستعلام عن هذه البيانات و تقديم معلومات إحصائية حول البيانات التي قمت بتخزينها.

لغة البرمجة الوسيطية:

إن ملقم قواعد البيانات **MySQL** لا يتفاعل مباشرة مع متصفح الإنترنت أو مع موقع الإنترنت كما يظهر لك في المخطط السابق، فهو بحاجة إلى لغة برمجة تعمل كوسيط بين **MySQL** و ملقم الويب، و أفضل لغة برمجة

تحقق التفاعل مع **MySQL** هي لغة **PHP** ، تحقق لغة البرمجة الوسيطية التفاعل بين ملقم الويب و ملقم قواعد البيانات فعندما يكون أمامك نموذج في الويب و تقوم بملئه و ترسله، فإن ملقم الويب يتلقى المعلومات التي يرسلها المستعرض و التي تتضمن معلومات النموذج، و هنا تتدخل البرمجة الوسيطية في أن تلقي نظرة على المعلومات القادمة فإذا كانت صحيحة أي تتطابق مع نوع البيانات الموجودة في ملقم **SQL** فتعيد (أي البرمجة الوسيطية) لك النتائج و إلا فإنها تعيد النموذج نفسه مع رسالة توضح لك أين الخطأ، تماماً كما يحدث معك عندما تسجل عضوية في موقع إنترنت أو تسجل على بريد إلكتروني.

ملقم الويب:

أفضل ملقم ويب في عالم الإنترنت هو **Apache**، و مهمة ملقم الويب هو استدعاء عنوان **IP** (**Internet Protocol address**) أو استدعاء نطاق معين **Domain** عن طريق بروتوكول نقل النصوص التشعبية **HTTP** (**Hypertext Transfer Protocol**) و هي اللغة التي تتكلمها ملقمات الويب مع متصفحات الويب.

إذن لجعل جهازك ملقم ويب متكامل و منسجم مع كل أنظمة التشغيل فأنت بحاجة إلى العتاد التالي:
MySQL , Apache , PHP

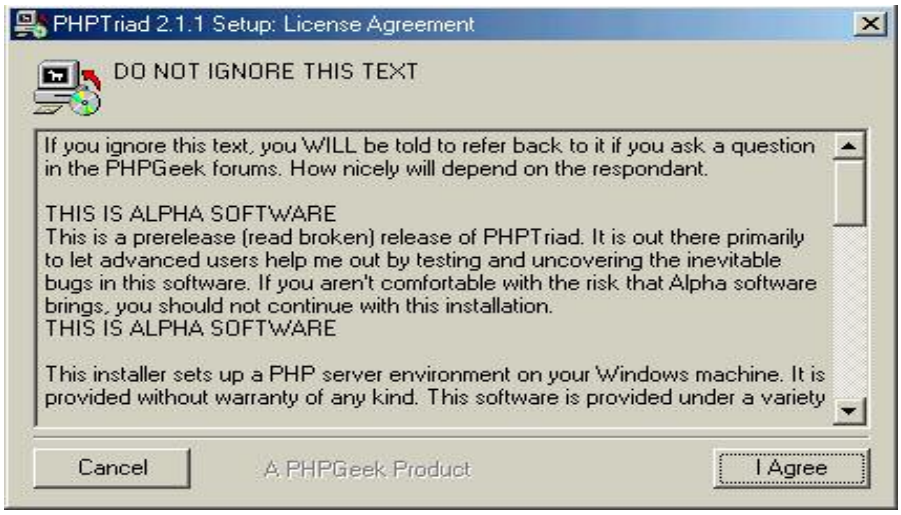
تثبيت العتاد اللازم تحت ويندوز

في هذا الكتاب ستحتاج فقط إلى ملقم قواعد البيانات **MySQL** و لكنني رأيت أنه لا ضير في أن يشمل هذا الكتاب على تعريفك بكيفية تركيب كل من **Apache** و **PHP** كونهما يأتيان مع **MySQL** كطاقم واحد.
هناك طريقتين لتثبيت هذا العتاد على جهازك، و هذا الكتاب يعتمد الطريقة الأولى، أما الطريقة الثانية فسأذكرها فقط من باب العلم بالشيء.

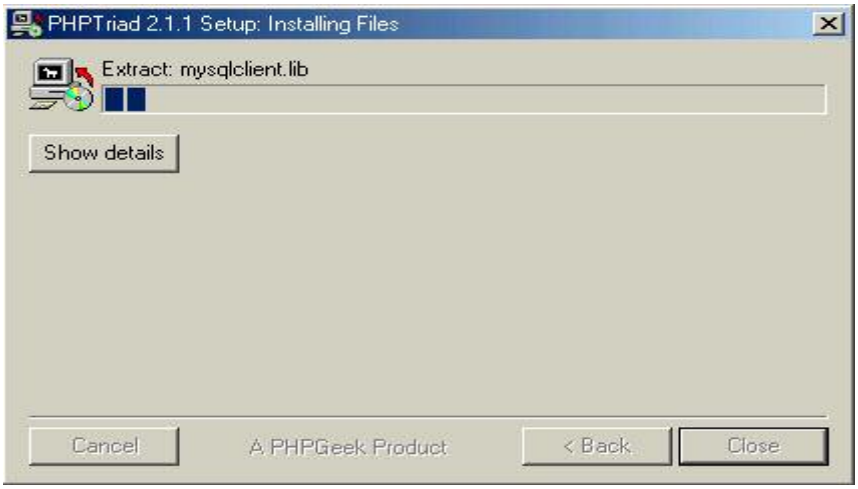
الطريقة الأولى

هذه الطريقة سهلة و بسيطة، و كل ما عليك هو أن تقوم بتنزيل **download** برنامج **PHPTriad** من موقع الإنترنت التالي:
http://sourceforge.net/projects/phptriad

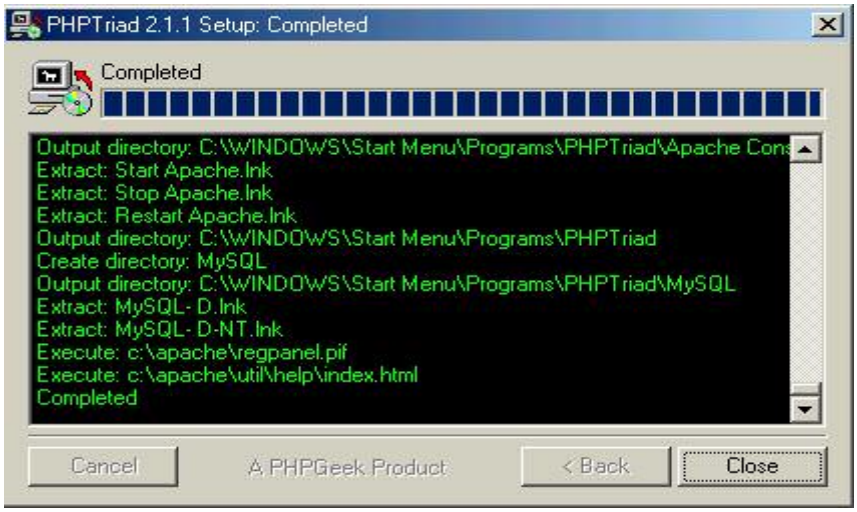
سنعتمد في هذا الكتاب على تثبيت **PHPTriad 2-1-1** على نظام ويندوز.
و إليك الخطوات التثبيت:
بعد تحميل البرنامج **PHPTriad 2.1.1** أو أي إصدار أحدث من الإنترنت، ستجد أنه يتألف من ملف تنفيذي واحد، و بالنقر المزدوج عليه ستظهر لك النافذة التالية



تظهر لك هذه النافذة اتفاقية الترخيص، قم بأخذ الخيار **I Agree** و عندها ستظهر لك النافذة التالية



و عندما تنقر الزر **show details** ستظهر لك نافذة تبين لك بالتفصيل الملفات التي يتم تنصيبها و مسارها كما في الصورة التالية



و عند الانتهاء سيتم تفعيل الزر **Close** قم بالنقر عليه، و بذلك تكون قد قمت بتنصيب كل من **MySQL** و **Apache** و **PHP** بشكل متكامل و لن تحتاج لأن تضيف أي تعدل من إعدادات التنصيب أي شيء.
الآن .. اذهب إلى قائمة ابدأ **start** ثم برامج **programs** ستلاحظ وجود عنصر جديد اسمه **PHPTriad** فيه أربع خيارات كما هو موضح في الصورة التالية

الخيار الأول **Apache Console** يعطيك الخيارات التالية

حيث أن الخيار **Start Apache** يقوم بتشغيل ملقم الويب **Apache** .
أما الخيار **Stop Apache** فيقوم بإيقاف عمل ملقم الويب **Apache** .
و الخيار **restart Apache** فستستخدمه فقط في حال أنك قمت بتشغيل **Apache** و لكنه لم يعمل بشكل سليم.
الخيار الثاني **MySQL** سيعطيك الخيارين التاليين

حيث أن **MySQL-D** هو ما يسمى بحارس **MySQL** أو **MySQL Demon** ، و من دون تشغيله لا تستطيع الدخول و استعمال قواعد بيانات **MySQL** ، و يستخدم هذا الحارس مع أنظمة التشغيل الشخصية مثل **95 , 98 , ME , XP professional** .
أما الخيار الثاني **MySQL -D-NT** فهو نفس الحارس، و لكنه يستخدم مع أنظمة التشغيل الخاصة بالشبكات مثل **NT , 2000 , XP server** .

أما الخيارين الثالث **Config and Data Backup** و الرابع **PHPTriad Control Panel** فهما ليسا من اختصاص هذا الكتاب، إلا اللهم هناك خيار واحد ضمن **PHPTriad Control Panel** و هو **Launch PHPMyAdmin** و الذي سنتحدث عنه لاحقاً إن شاء الله.

التأكد من سلامة التثبيت

حسناً .. بعد تركيب برنامج **PHPTriad** سنتأكد من سلامة كل من **MySQL** و **PHP** و من قدرتهما على التعرف على بعضهما البعض.

أولاً: اذهب إلى المسار التالي **C:\apache\mysql\bin** و ذلك على فرض أنك قد نصبت البرنامج بمساره التلقائي و تأكد من وجود الملفات التالية:

1. **mysqladmin.exe**
2. **mysqld.exe**
3. **mysqlimport.exe**
4. **mysqldump.exe**

ثانياً: اذهب إلى المسار التالي **C:\apache\mysql\data** فستجد مجلدين هما **test , mysql** هذين المجلدين هما قاعدتي بيانات حيث أن **MySQL** تقوم بحفظ قواعد البيانات على شكل مجلدات.
ستجد المجلد **test** فارغ ، ادخل إلى المجلد **mysql** ستجد ملفات متعددة ذات امتدادات ثلاث و هي :

1. الملفات ذات الامتداد **.FRM** هذا النوع يضم توصيفات الجداول.
2. الملفات ذات الامتداد **.MYI** هذا النوع يضم الفهارس.
3. الملفات ذات الامتداد **.MYD** هذا النوع يصم بيانات الجدول.

ثالثاً: من قائمة ابدأ قم بتشغيل كل من **start apache , MySQL-D** ثم اذهب إلى متصفح الإنترنت لديك و حرر في شريط العناوين العنوان التالي:

http://localhost/phpinfo.php

فإذا ظهرت لك الشاشة التالية فإن عملية التنصيب قد تمت بنجاح بإذن الله.

الطريقة الثانية

هذه الطريقة سنطلق عليها اسم الطريقة اليدوية إذا صح التعبير، حيث أنك ستقوم بتنصيب كل من **MySQL** و **Apache** و **PHP** على حدا ثم تقوم بربطهم معاً.

تستطيع الحصول على أحدث نسخة **MySQL** عن طريق الرابط التالي:

http://www.mysql.com/downloads/index.html

تستطيع الحصول على الملقم **Apache** عن طريق الرابط التالي:

http://www.apache.org/dist/httpd

و تستطيع الحصول على أحدث نسخة **PHP** عن طريق الرابط التالي:

http://www.php.net/downloads.php

و بذلك أصبح لديك كل من **MySQL** و **Apache** و **PHP** ، و الآن ستتعلم إن شاء الله كيفية تثبيت هذه المكونات الثلاث.

عندما تقوم بزيارة تلك الروابط التي ذكرناها، قد تجد أن بعضها تغير أو حتى غير موجود، و سبب ذلك هو أن مواقع الإنترنت دائمة التطوير و التحديث، لذلك يمكنك عن طريق محرك البحث **Google.com** أن تبحث عن مرادك و ستظهر لك العشرات النتائج.

قبل أن تباشر بتثبيت المكونات الثلاث ستحتاج إلى تحديث بعض إعدادات **windows** و ذلك حسب نظام التشغيل الذي لديك.

فإذا كنت تعمل على **Windows 95** ستحتاج إلى تحميل **Windows Sockets** من الرابط التالي:

http://www.microsoft.com/windows/downloads/bin/w95ws2setup.exe

و إذا كنت تعمل على **Windows ME , 98** ستحتاج تحميل **MSI** من الرابط التالي:
ftp://ftp.microsoft.com/developr/platformsdk/oct2000/msi/win95/instmsi.exe

و إذا كنت تعمل على **Windows NT , 2000** فأنت بحاجة إلى تحميل **MSI** من الرابط التالي:
ftp://ftp.microsoft.com/developr/platrofmsdk/oct2000/msi/winnt/x86/instmsi.exe

طبعاً هذه التحديثات لن تحتاجها إلا إذا أردت أن يكون جهازك ملقم ويب **Web Server** فعلي، أما من أجل التدريب و التعليم فلن تحتاج لهذه التحديثات.

1. تثبيت MySQL
لن يحتاج منك تثبيت **MySQL** على جهازك سوى بضع نقرات متتالية، يفضل أن يتم تنصيب **MySQL** في مساره الافتراضي **C:\mysql**

2. تثبيت Apache
قم بالنقر المزدوج على الملف التنفيذي، ستظهر لك نافذة تطلب منك تحديد اسم النطاق **Domain** الخاص بموقعك، و اسم موقعك **Server name** و البريد الإلكتروني لمدير الموقع **Administrator's Email** ، إذا كنت ستستخدم المكونات الثلاث للتدريب فقم بإدخال القيمة **localhost** أو **127.0.01** في كل من الحقلين **Network Domain** و **Server Name** و أما الـ **Administrator's Email** فقم بإدخال بريدك الإلكتروني الفعلي.
يوجد في نفس النافذة زري راديو يتيحان لك اختيار تثبيت **Apache** كخدمة **Service** أو تطبيق **Application** ، طبعاً إذا كان نظام التشغيل لديك هو **NT , 2000** قم باختيار الخدمة، أما إذا كان نظام التشغيل لديك **ME , 98 , XP Professional** فقم باختيار التطبيق، ثم أكمل عملية التنصيب، و اترك المسار الافتراضي للتنصيب كما هو.

3. تثبيت PHP
قبل أن تبدأ بتنصيب **PHP** على جهازك، قم بإيقاف كل من **Apache , MySQL** إذا كانا يعملا، ثم قم بالتنصيب و دع مسار التثبيت كما هو **C:\php** .

4. إعداد Apache للتوافق مع PHP
فيما يلي الخطوات الواجب عليك إتباعها كي يتم الإعداد النهائي بشكل سليم، و قبل كل شيء تأكد من أن **Apache** متوقف عن العمل، ثم قم بتنفيذ الخطوات التالية:

- ستجد في المسار **C:\php** ملفاً يدعى **php4st.dll** قم بنسخ هذا الملف إلى المسار **C:\windows\system** إذا كنت تستخدم **Windows ME , 98**، و إذا كنت تستخدم **WINDOWS NT , 2000** قم بنسخه إلى المسار **C:\winnt\system32**

- ستجد في المسار **C:\php** ملف اسمه **php.ini-dist** قم بنسخه إلى المسار **C:\windows** و غير اسمه إلى **php.ini**

- اذهب إلى المسار **C:\program files\Apache group\apache\conf** ستجد ملف اسمه **httpd.conf** ، قبل أن تقوم بتحرير هذا الملف خذ نسخة احتياطية منه، ثم قم بفتحه عن طريق **Notepad** و أضف إليه السطر التالي:

ScriptAlias /php/ "C:/php/"

و الهدف منه هو إعلام **Apache** عن مكان تواجد ملفات الـ **PHP**

- ضمن نفس الملف قم بالبحث عن العبارة **AddType** ثم إضافة السطر التالي:

AddType application/x-httpd-php .php .php3 .html .htm

و الهدف منه هو تمكين **Apache** من التعامل مع هذه الامتدادات كتطبيق **mime**

- ما زلنا ضمن الملف **httpd.conf** قم بالبحث عن المقطع **Action** و أضف السطر التالي:

Action application/x-httpd-php "/php/php.exe"

و الهدف منه هو جعل الملف التنفيذي **php.exe** يقوم بالتحكم بملفات الـ **mime**

- ابحث عن العبارة **DocumentRoot** و تأكد أنها كما يلي:

DocumentRoot "C:\Apache\htdocs"

حيث أن هذا هو المسار الذي ستضع فيه جميع مخطوطات الـ **PHP** ليقوم **Apache** بإرسالها إلى المترجم **php.exe** و من ثم استعادتها و إرسالها للمتصفح لتُعرض كصفحة ويب، مع الأخذ بعين الاعتبار أن هذا الدليل يمكن تغييره.

تسجيل الدخول و الخروج في MySQL

قبل أن تبدأ بتعلم أوامر **MySQL** عليك معرفة كيفية تسجيل الدخول و الخروج إلى **MySQL** .

أولاً: قم بتشغيل حارس **MySQL** عن طريق قائمة ابدأ ثم برامج ثم **PHP Triad** ثم **MySQL** ثم **MySQL-D** في حال كان نظام التشغيل عندك هو ويندوز **95 , 98 , ME** ، أو **MySQL-D-NT** في حال كان نظام التشغيل عندك هو **NT , 2000 , XP** كما شرحنا سابقاً، و تذكر دائماً و أبداً أنك يجب أن تشغل حارس **MySQL** عندما تريد أن تتعامل مع قواعد بياناتك.

و بعد ذلك اذهب إلى موجه الدوس كما يلي:

اضغط مفتاحي **win + R** ثم اكتب داخل المربع الكلمة التالية **command** ثم اضغط إنتر فيفتح لك موجه الدوس، طبعاً هذه الطريقة سهلة و سريعة لذلك كتبتها لك فهي أفضل من طريق قائمة ابدأ.

بعد تشغيل موجه الدوس سيكون شكله كما يلي:

c:\windows\desktop>_

قم بكتابة الأمر التالي: **cd c:\apache\mysql\bin** ثم اضغط إنتر، فعندها سيتغير شكل المحث ليصبح هكذا:

c:\apache\mysql\bin>_

و طبعاً الأمر **cd** هو أمر تغيير المسار **Change Directory** .

و الآن لندخل إلى ملقم قواعد البيانات:

اكتب الأمر التالي:

C:\apache\mysql\bin>mysql -u root

ثم اضغط إنتر فستظهر لك هذه رسالة ترحيبية تعطيك بعض المعلومات المفيدة و إصدار نسخة **MySQL** التي تعمل عليها.

و عندها سيتغير محث الـ **DOS** إلى محث **MySQL** و يصبح شكله كما يلي:

mysql>_

تستطيع أن تشغل حارس ملقم قواعد البيانات **MySQL -D** عن طريق موجه الدوس بكتابة الأمر التالي:

C:\apache\mysql\bin>mysqld ثم إنتر.

و هذا أول أمر من أوامر **MySQL** و هو كيفية تسجل الدخول، حيث أن **-u root** تعرفك على الملقم بأنك المستخدم الأساسي مما يعطيك حرية القيام بأي شيء تريده على أي شيء في قواعد البيانات الموجودة داخل هذا الملقم.

و الآن إلى التعليمة الثانية من أوامر **MySQL** و هي تسجيل الخروج، فلتسجيل خروج من **MySQL** أكتب أحد الأمرين التاليين:

1. **exit** ثم إنتر.
2. **\q** ثم إنتر حيث حرف **q** معناه **quit** أي خروج.

و بهذا تكون قد تعلمت بعون الله كيفية تسجيل الدخول و تسجيل الخروج من و إلى **MySQL**

بما أنك ستعمل كثيراً على الدوس فيجب عليك أن تصل بالذاكرة إلى الحد الأقصى أثناء التعامل مع الدوس ضمن بيئة الويندوز، و ما عليك إلا إتباع الخطوات الآتية :

1. قم بتشغيل الملف **System.ini** من الأمر **Run** .
2. باستخدام شريط التمرير يمكنك الانتقال إلى العبارة [**386ENH**] .
3. قم بإضافة السطر التالي : **LocalLoadHigh=1** ثم احفظ التغير و أعد تشغيل الجهاز.

الفصل الثالث
لغة تعريف البيانات DDL

تقسم تعليمات **MySQL** إلى جزأين هما:
1. لغة تعريف البيانات **DDL** (**Data Definition Language**)
و تقوم هذه اللغة بإنشاء و تعديل قواعد البيانات و الجداول، و أهم أوامرها
create database , create table , alter table
2. لغة معالجة البيانات **DML** (**Data Manipulation Language**)
و هذه اللغة تختص بالتعامل مع البيانات المدرجة داخل الجداول من عمليات حذف و تحرير و إدراج و تعديل .. الخ، و أهم أوامرها **replace , delete , insert , update , select**
وسنتكلم إن شاء الله في هذا الفصل عن لغة تعريف البيانات **DDL** و في الفصل الذي يليه عن لغة معالجة البيانات **DML**

عرض قواعد البيانات

قم بتسجيل الدخول إلى **MySQL** بعدها اكتب الأمر التالي:

```
mysql>show databases;
```

و هذا الأمر يخبر اللقم (ملقم قواعد البيانات) بأن يعرض جميع قواعد البيانات التي لدينا، و انتبه إلى الفاصلة المنقوطة في نهاية الأمر المكتوب و التي ستضيفها من الآن و صاعداً لجميع أوامر **MySQL** ، و ستظهر لنا النتيجة التالية:

```
+--------------+
| Database     |
+--------------+
|  mysql       |
|  test        |
+--------------+
2 rows in set (0.05 sec)
```

و هي تدل على أنه يوجد لدينا قاعدتي بيانات الأولى اسمها **mysql** و الثانية اسمها **test .**
لاحظ الجدول الذي عرض لنا **MySQL** فيه النتائج، تسمى طريقة العرض هذه بطريقة العرض الجدولي، و أخيراً سيعرض لنا **MySQL** سطر نتائج يخبرنا فيه قائلاً إن نتيجة هذا الأمر الذي كتبته هو سطرين، و قد تم معالجة هذا الأمر بزمن و قدره خمسة بالمائة من الثانية.

استخدام قاعدة بيانات محددة

و الآن قم بكتابة الأمر التالي:

```
mysql>use mysql;
```

و هذا الأمر يخبر الملقم بأن يقوم باستخدام قاعدة البيانات التي اسمها **mysql** ، و الهدف من هذا الأمر هو فتح قاعدة البيانات لبدء العمل فيها أي للقيام بعمليات التحديث و الإدراج و الحذف ... الخ من أوامر **MySQL** ، حيث أنك لن تستطيع أن تطبق تعليمات **MySQL** على أية قاعدة بيانات دون أن تستخدمها.
إذن فالصيغة القواعدية لأمر استخدام قاعدة بيانات هي:

```
mysql>use database_name;
```

لذلك إذا أردنا استخدام قاعدة البيانات **test** نكتب:

```
mysql>use test;
```

الأمر **show**

تعرفنا قبل قليل على أول تعليمة من تعليمات الأمر **show** و الآن سنكمل إن شاء الله التعرف على جميع تعليماتها.

تعليمة عرض الجداول

الصيغة القواعدية لهذا الأمر هي:

```
mysql>show tables;
```

و يقوم هذا الأمر بعرض الجداول لقاعدة البيانات التي أنت تستخدمها، لذلك قم بما يلي:

1. اكتب أمر استخدام قاعدة البيانات **mysql**، و ذلك لأن قاعدة البيانات **test** فارغة لا تحتوي جداول.
2. اكتب أمر عرض الجداول التالي:

```
mysql>show tables;
```

و ستكون النتيجة هي كما يلي:

```
+----------------------+
| Tables_in_mysql      |
+----------------------+
|  columns_priv        |
|  db                  |
|  func                |
|  host                |
|  tables_priv         |
|  user                |
+----------------------+
6 rows in set (0.00 sec)
```

هذه الطريقة للعرض هي العرض على شكل جدول، و هناك طريقة أخرى للعرض و هي طريقة العرض الأفقي، و ما عليك إلا أن تضيف العبارة **\G** لأخر الأمر، كما يلي:

```
mysql>show tables \G;
```

مع الأخذ بعين الاعتبار بأن حرف الـ **G** هو حرف كبير، و تكون النتيجة كما هو مبين هنا:

```
*************************** 1. row ***************************
Tables_in_mysql: columns_priv
*************************** 2. row ***************************
Tables_in_mysql: db
*************************** 3. row ***************************
Tables_in_mysql: func
*************************** 4. row ***************************
Tables_in_mysql: host
*************************** 5. row ***************************
Tables_in_mysql: tables_priv
*************************** 6. row ***************************
Tables_in_mysql: user
6 rows in set (0.05 sec)
```

لاحظ اختلاف سطر النتائج بين طريقتي العرض حيث أن الطريقة الأولى أسرع من الثانية.

تعليمة عرض الأعمدة

الصيغة القواعدية لهذا الأمر هي:

mysql>show columns from table_name;

و يقوم هذا الأمر بعرض أعمدة الجدول المذكور، و طبعاً هذا الجدول ينتمي إلى قاعدة البيانات التي طبقنا عليها الأمر **use**، و يقوم هذا الأمر بعرض مجموعة معلومات عن الجدول، فإذا كتبت الأمر التالي:

mysql>show columns from host;

ستكون النتيجة كما يلي:

```
+-----------------------+-----------------------+-------+------+---------+-------+
| Field                 | Type                  | Null  | Key  | Default | Extra |
+-----------------------+-----------------------+-------+------+---------+-------+
| Host                  | char(60) binary       |       | PRI  |         |       |
| Db                    | char(64) binary       |       | PRI  |         |       |
| Select_priv           | enum('N','Y')         |       |      | N       |       |
| Insert_priv           | enum('N','Y')         |       |      | N       |       |
| Update_priv           | enum('N','Y')         |       |      | N       |       |
| Delete_priv           | enum('N','Y')         |       |      | N       |       |
| Create_priv           | enum('N','Y')         |       |      | N       |       |
| Drop_priv             | enum('N','Y')         |       |      | N       |       |
| Grant_priv            | enum('N','Y')         |       |      | N       |       |
| References_priv       | enum('N','Y')         |       |      | N       |       |
| Index_priv            | enum('N','Y')         |       |      | N       |       |
| Alter_priv            | enum('N','Y')         |       |      | N       |       |
+-----------------------+-----------------------+-------+------+---------+-------+
12 rows in set (0.00 sec)
```

لا عليك من تفاصيل هذه المعلومات الآن لأننا سنقوم بإذن الله بشرحها في الدروس القادمة.

هناك أمرين آخرين مرادفين للأمر **show columns** و هما:

mysql>show fields from table_name;

mysql>describe table_name;

جرب كتابة كل أمر و ستلاحظ أنهما يعطياك نفس النتيجة.

تعليمة عرض الفهارس

الصيغة القواعدية لهذا الأمر هي:

mysql>show index from table_name;

حيث يقوم هذا الأمر بعرض معلومات عن جميع الفهارس الموضوعة على هذا الجدول.

فإذا قمت بكتابة الأمر التالي:

mysql>show index from host;

فسيتم عرض النتيجة التالية:

```
+-------+------------+----------+--------------+-------------+-----------+-------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality |
+-------+------------+----------+--------------+-------------+-----------+-------------+
| host  | 0          | PRIMARY  | 1            | Host        | A         | NULL        |
|host   | 0          | PRIMARY  | 2            | Db          | A         | 0           |
+-------+------------+----------+--------------+-------------+-----------+-------------+
2 rows in set (0.00 sec)
```

طبعاً هذا العرض غير كامل بسبب مجال الصفحة، حيث أن الأمر يعرض لك أيضاً ثلاث أعمدة.

تعليمة عرض حالة الجدول
الصيغة القواعدية لهذا الأمر هي :

```
mysql>show table status like 'table_name' ;
```

حيث يقدم هذا الأمر معلومات تفصيلية أكثر عن الجدول، فإذا قمت بكتابة التعليمة التالية:

```
mysql>show table status like 'host' \G;
```

ستحصل على النتيجة التالية:

```
*************************** 1. row ***************************
Name: host
Type: MyISAM
Row_format: Fixed
Rows: 0
Avg_row_length: 0
Data_length: 0
Max_data_length: 579820584959
Index_length: 1024
Data_free: 0
Auto_increment: NULL
Create_time: 2001-10-07 00:33:39
Update_time: 2001-10-07 05:33:40
Check_time: NULL
Create_options:
Comment: Host privileges; Merged with database privileges
```

التعليمة **create**
يستخدم الأمر **create** إنشاء قواعد البيانات و الجداول.
التعليمة **create database**
من الطبيعي أن أقوم بإنشاء قاعدة بيانات قبل أن أقوم ببناء جداولها، و الصيغة القواعدية لأمر إنشاء قاعدة بيانات هو :

```
mysql>create  database DB_name;
```

لذلك قم أنت بكتابة الأمر التالي:

```
mysql>create database movie_store;
```

و بذلك أصبح لديك قاعدة بيانات اسمها **movie_store** ، فإذا ذهبت إلى المسار التالي:
C:\apache\mysql\data ستجد هناك مجلد اسمه **movie_store** ، و عندما تفتحه ستجده فارغاً.
إذن التعليمة **create database** تقوم ببناء قاعدة بيانات يتم تخزينها على شكل مجلد.
حسناً .. للقيام بإنشاء الجداول الخاصة بنا يجب علينا استعمال القاعدة **movie_store** لذلك ستكتب الأمر التالي:

```
mysql>use movie_store;
```

التعليمة **create table**
سنقوم الآن ببناء أول جدول من جداول قاعدة البيانات **movie_store** و هو الجدول **employees** و الصيغة القواعدية العامة لبناء جدول هي:

```
create table table_name(
col_1_name  type  attributes ,
col_2_name  type  attributes ,
col_n_name  type attributes ,
primary key(col_name)) type=table_type ;
```

لا تقلق حول هذه الصيغة فسيتم شرحها الآن بشكل مستفيض إن شاء الله.

هناك خاصية جميلة في **MySQL** و هي في حال أنك أردت كتابة تعليمة طويلة بحيث طولها يزيد عن السطر مثل تعليمات إنشاء الجداول، فإن ذلك ممكن مع **MySQL** بحيث تضغط مفتاح **inter** لتنتقل المشيرة إلى السطر التالي و ستجد في بداية السطر التالي العلامة التالية (**- >**) و هي تدل على أن التعليمة التي تكتبها لم تنته بعد.

في السطر الأول نقول لـ **MySQL** بأن تقوم بإنشاء جدول و نضع له الاسم الذي نريد ثم نفتح قوس و نغلقه كما يلي:

create table table_name( )

و بداخل القوسين نكتب كل المعلومات اللازمة عن عناصر الجدول، حيث أن **col_1_name** هو اسم العمود الأول،و من الممكن أن يكون اسم العمود أي سلسلة محرفية تريدها، لكن بدون فراغات.
أما **type** فهي للدلالة على نوع البيانات المخلة إلى العمود و هناك ثلاث أنواع للأعمدة هي:
1. الأعمدة النصية.
2. الأعمدة الرقمية.
3. أعمدة التاريخ و الوقت.

أما **attributes** فهي خصائص أو صفات أو سمات هذا العمود.
و أخيراً بالنسبة إلى **type=table_type** فهي من أجل إعطاء الجدول نوع معين.

أنواع الأعمدة في **MySQL**

كما ذكرنا أن هناك ثلاث أنواع للأعمدة، و إليك التفصيل.

الأعمدة النصية:

تستخدم الأعمدة النصية لإدراج بيانات محرفية أو خليط من البيانات المحرفية و الرقمية، و هناك عدة أنواع للأعمدة النصية:

1. **char** :

لهذا النوع من الأعمدة طول أعظمي يبلغ 255 محرف.
أما الصيغة القواعدية لاستخدامه فهي:

col_name char(size)

و هذا النوع من الأعمدة هو عمود ذو طول ثابت أي إذا تم إدراج قيمة ما، عدد محارفها أقل من العدد الأعظمي للعمود، فسيتم حشو باقي الحقل بفراغات من اليمين، فمثلاً إذا قمت بتعريف عمود من النوع **char(15)** و قمت بتخزين القيمة **Die Hard** فإن **MySQL** ستقوم عملياً بتخزين تلك القيمة متبوعة بسبع فراغات.
و عيب هذا النوع من الأعمدة هو أنه يقوم بحجز مساحات كبيرة في قاعدة البيانات و معظمها يكون عبارة عن فراغات.
أما أهميته فهو أنه مفيد لإنشاء أعمدة كلمات المرور **Pass Words**.

2. **varchar** :

الطول الأعظمي: 255 محرف.
الصيغة القواعدية:

col_name varchar(size)

و يتميز هذا النوع بأنه ذو طول متغير، أي إذا قمت بتعريف عمود من النوع **varchar(15)** و خزنت القيمة **Die Hard** فيه فلن يتم حشر فراغات إلى يمين القيمة المخزنة بل يقوم بإزالة الفراغات من نهاية السلسلة المحرفية، و لكن **MySQL** تضيف محرفاً واحداً إلى كل عمود من النوع **varchar** حيث يتم فيه تخزين طول الحقل.

3. **text** :
الطول الأعظمي هو : 65535
الصيغة القواعدية:

col_name text

أيضاً هو من النوع متغير الطول، يمكن أن يتم إنشاء فهارس على أول 255 محرف من العمود الذي من النوع **text** كما يمكن أن يتم استخدام الفهارس من النوع **FULLTEXT** ، سنشرح بإذن الله الفهارس لاحقاً.

4. **enum** :
يستخدم هذا النوع من الأعمدة من أجل تحديد خيار واحد من بين عدة خيارات موجودة، و يسمح هذا النوع من الأعمدة باستخدام 65535 قيمة.
الصياغة القواعدية:

col_name enum('val_1' , 'val_2' , ...) default 'val_1'

و لنأخذ المثال التالي:

```
create table temp(
id int auto_increment primary key,
name varchar(30) not null,
gender enum('male' , 'female' ) default 'male' );
```

5. **tinytext** :
الطول الأعظمي له : 255 محرف.
الصياغة القواعدية للاستخدام :

col_name tinytext

و هو أيضاً ذو طول متغير، و هو يؤدي نفس عمل الأعمدة التي من النوع **varchar**

6. **mediumtext** :
الطول الأعظمي : 16777215 محرف.
أما كيفية الاستخدام :

col_name mediumtext

و هو نوع متغير الطول، و يمكن إنشاء فهارس على أول 255 محرف، كما و يمكن استخدام الفهارس التي من النوع **FULLTEXT** عليها.

7. **longtext** :
الطول الأعظمي : 4294967295 محرف.
أما كيفية الاستخدام:

col_name longtext

و هو نوع متغير الطول، و يمكن إنشاء فهارس على أول 255 محرف.

8. **set** :
هذا النوع من الأعمدة مثل النوع **enum** لكنه يقوم بتعريف مجموعة ضخمة من القيم، أما الصياغة القواعدية لهذا النوع فهي:

col_name set('val_1' , 'val_2' , ...)

الأعمدة الرقمية:
يستخدم هذا النوع من أجل إدراج بيانات رقمية أو عددية، و أهم أنواعها:
1. **int/integer** :
طريقة الاستخدام :

col_name integer(size) [zerofill] [unsigned]

إن الأقواس التي باللون الأخضر تعبر عن أن التعليمة الموضوعة داخل القوسين هي تعليمة اختيارية، حيث أن الخيار **zerofill** يستخدم من أجل إدراج أصفاراً إلى يسار القيمة المدخلة، و الخيار **unsigned** يستخدم لتخزين الأرقام الصحيحة من 0 و حتى 4294967295 أما إذا اخترت **signed** فيصبح بمقدورك تخزين أرقاماً من (سالب 2147485648 إلى موجب 2147483647)
و يستخدم هذا النوع من الأعمدة غالباً من أجل أعمدة الترقيم التلقائي، كما في المثال التالي:

```
create table new_table(
id int unsigned auto_increment primary key,
col_2  text);
```

2. **float** :
طريقة الاستخدام:

col_name float(M,D) [zerofill]

حيث **M** هي عدد الخانات التي سيتم حجزها، و **D** هو عدد الفواصل التي ستحدد من هذه الخانات. و هذا النوع لا يمكن أن يكون بدون إشارة، و يستخدم كما يلي:

column_name float(7,3)

أي أن العمود الذي اسمه **column_name** نوعه **float** و عدد خاناته 7 و منها 3 للفاصلة، أي أن أكبر رقم يمكن أن يخزن به هو 9999.999

3. **tinyint** :
طريقة الاستخدام:

tinyint (size) [unsigned] [zerofill]

و في حال اختيارك **unsigned** فإن العمود من هذا النوع يمكن أن يخزن قيماً صحيحة من 0 و حتى 255 ، أما إذا اخترت **signed** فإن المجال يصبح بين 128- و 127 .

4. **mediumint** :
طريقة الاستخدام:

mediumint (size) [unsigned] [zerofill]

إذا اخترت **unsigned** فإن العمود من هذا النوع يمكن أن يخزن قيماً صحيحة من 0 حتى 16777215 أما إذا اخترت **signed** فإن المجال يصبح بين 8388608- و 8388607

5. **bigint** :
طريقة الاستخدام:

bigint (size) [unsifned] [zerofill]

إذا اخترت **unsigned** فإن العمود من هذا النوع يمكن أن يخزن قيماً صحيحة من 0 و حتى 18446766073709551615 أما إذا اخترت **signed** فإن المجال يصبح بين 9223372036854775808- و 9223372036854775807

6. **double/real** :
و هي تشير إلى رقم بفاصلة عائمة بدقة مضاعفة، و لا يمكن أن تكون بدون إشارة.
طريقة الاستخدام:

double [(M,D)] [zerofill]

القيم الممكنة هي بين **-1.7976931348623157E+308** إلى **-2.2250738585072014E-308** و **0** و من **2.2250738585072014E-308** إلى **1.7976931348623157E+308**

7. **decimal** :
طريقة الاستخدام:

decimal [(M,D)] [zerofill]

و يتم تخزين الأرقام في هذا النوع من الأعمدة كمحارف. حيث يتم تخزين كل رقم كسلسلة نصية، محرف مقابل كل عدد في القيمة الرقمية. و إذا كانت **D=0** فإن هذا معناه أن القيم بدون فاصلة عشرية أما المجال الأعظمي للقيم من هذا النوع فهي تطابق مجال قيم النوع **double** و إذا لم توضع قيم لـ **M** فسيكون لها قيمة افتراضية 10.

أعمدة التاريخ و الوقت:
و لها عدة أنواع هي:

1. **date** :
الاستخدام : **col_name date**
و يكون تخزين الوقت في هذا النوع من الأعمدة على الشكل التالي (**YYYY-MM-DD**) حيث أن القيم المسموح بها هي بين 1000-01-01 إلى 9999-12-31

2. **datetime** :
الاستخدام : **col_name datetime**
يكون شكل التنسيق هو (**YYYY-MM-DD HH:MM:SS**)
أما القيم المسموح بها فهي بين: 1000-01-01 00:00:00 و 9999-12-31 23:59:59
و من سيئات هذين النوعين هو أنه ستقوم أنت بإضافة التاريخ، لذلك أفضل استخدام النوع الثالث.

3. **timestamp** :
الاستخدام **col_name(size)**
هذا النوع من الأعمدة هو متعدد الاستخدامات كما أنه يقوم تلقائيا بتسجيل تاريخ و وقت أحدث التغييرات سواء أكان هذا التغيير إدراجاً أو تحديثاً، أما الوسيط **size** فهو يأخذ القيم الزوجية التي بين العددين 2 و 14 حيث يكون التنسيق كما يلي:

Size	Format
2	YY
4	YYMM
6	YYMMDD
8	YYYYMMDD
10	YYMMDDHHMM
12	YYMMDDHHMMSS
14	YYYYMMDDHHMMSS

4. **time** :
الاستخدام: **time**
يتم هنا تخزين الوقت بالتنسيق **HH:MM:SS** و هو يقع ضمن المجال **-838:59:59** و حتى المجال **838:59:59**
أما سبب وجود قيم كبيرة فهو إمكانية استخدام نوع العمود **time** لتخزين نتيجة معادلات حسابية تتضمن الوقت.

5. **year** :
الاستخدام: **year[(2/4)]**
التنسيق برقمين يأخذ المجال بين 1970 و حتى 2069 أما التنسيق بأربع خانات يأخذ المجال بين 1901 و حتى 2155.

خصائص الأعمدة في MySQL

1- **NULL / NOT NULL** :
تخبر هذه الخاصية **MySQL** فيما إذا سيتم السماح باستخدام القيم الفارغة(اللاقيمة) أم لا في العمود. حيث أن **NOT NULL** تخبر **MySQL** بعدم السماح بوجود اللاقيمة، و **NULL** تخبره بالسماح بوجود اللاقيمة و الشكل الافتراضي هو **NULL** .

الشكل الافتراضي **Default** هي الطريقة التي تعتمدها نظام تشغيل أو برنامج ما في حال عدم تنبيهنا له عن كيفية الطريقة الواجب إتباعها، و لتوضيح ذلك دعنا نأخذ هذا المثال: عندما تفتح مجلد يضم العديد من الملفات ذات الامتدادات المختلفة **Different Suffixes** ستلاحظ أن ويندوز قد قام من تلقاء نفسه بترتيبها أبجدياً، و هذا هو المقصود بالشكل الافتراضي.

2- **DEFAULT** :
و هي القيمة الافتراضية التي تريدها أنت أن يتم إدراجها في حال لم يقوم المستخدم بإدراج قيمة، لنأخذ هذا المثال:

```
create table exam(
name varchar(30) primary key not null default 'He has no name' ) ;
```

لاحظ ففي حال لم يدخل المستخدم أو الزبون اسمه في النموذج فستظهر الجملة **He has no name** في الخانة الخاصة باسمه من النموذج.

3- **auto_increment** :
و هي ميزة الترقيم المتزايد التلقائي و تأخذ أرقاماً من الواحد إلى اللانهاية و هذه الخاصية مفيدة من أجل المفتاح الأساسي **Primary Key** .

4- **Primary Key** :
كل جدول من جداول قواعد البيانات يجب أن يحوي عموداً مميز يسمى بالمفتاح الأساسي **primary key** و هذا العمود يجب ألا يحوي قيماً متكررة، لنأخذ المثال التالي:
قم بإنشاء جدول اسمه **names** مكون من عمودين بحيث الأول هو مفتاح أساسي و اسمه **id** و نوعه **int** و الثاني يحوي أسماء أشخاص و اسمه **name** و نوعه **varchar** :

```
create table names (
id int primary key auto_increment,
name varchar(20) not null);
```

و بالتالي عندما تقوم بإدراج اسم ما فإن هذا الاسم تلقائياً يأخذ رقم، و تبدأ الأرقام من الواحد و صعوداً، و عند إضافة اسم جديد فأنه يأخذ تلقائياً الرقم اثنين ... و هكذا.
كما و يمكن أن تستخدم السمة **primary key** بشكل منفصل كما يلي:

```
create table names(
id int auto_increment,
name varchar(20) not null,
primary key (id));
```

و هذا الشكل أفضل لأنه في بعض الحالات قد تحتاج أن يكون المفتاح الأساسي مؤلف من عمودين أو أكثر كما يلي:

```
primary key (id , name);
```

أنواع الجداول في MySQL

هناك عدة أنواع للجداول في **MySQL** و هي:

1. **MyISAM** :

و هو النوع الافتراضي الذي يضعه **MySQL** للجدول في حال عدم تعيينك لنوع الجدول الذي تقوم بإنشائه، و هذا النوع من الجداول سريع جداً و مستقر.

2. **Heap** :

هذا النوع من الجداول تكون مقيمة في الذاكرة، أي أنها غير مخزنة في أي مكان فيزيائي، لذلك فإنها تتبخر في حال انقطاع التغذية، و لكن كونها تتوضع على الذواكر فهي بغاية السرعة، و الفائدة منها هو إمكانية بناء جداول مؤقتة لتتوضع على الذواكر من أجل الاستعلامات السريعة.

و هناك أنواع أخرى هامة هي **InnoDB , BDB , Gemini** و لكل منها مزايا عديدة و مختلفة بحيث يتوقف النوع الذي ستختاره على نوع الوظيفة التي سيقوم بها الجدول الذي تقوم ببنائه، و سنشرح ذلك لاحقا إن شاء الله، لكن الآن سنعتمد على أن كل جداولنا هي من النوع **MyISAM**
و لنأخذ هذا المثال:

```
create table my_table(col_name type attributes) type=myisam;
```

بناء قاعدة البيانات المثال برمجياً

و الآن و بعد أن عرفت كيفية إنشاء جدول شبه متكامل، سنقوم بتطبيق عمليات إنشاء الجداول لقاعدة البيانات المثال **movie_store** خطوة بخطوة حتى تتعرف بشكل عملي على كيفية بناء قاعدة بيانات من الصفر.

الجدول **employees :**

طبعاً قبل أن تقوم ببناء الجداول ستسجل دخولاً إلى **MySQL** كمستخدم أساسي ثم تقوم باستخدام قاعدة البيانات **movie_store** التي قمت بإنشائها سابقاً.
حسناً .. سنقوم الآن بإنشاء الجدول **employees** و الذي يحوي الأعمدة التالية:

1. رقم العامل **emp_no :** نمط هذا العمود هو **int** و يضم قيمة **فريدة** لكل عامل لذلك سنجعله مفتاحاً أساسياً و يأخذ قيم الترقيم التلقائي أي **auto_increment .**
2. اسم العامل **name :** نوع العمود هو محرفي متغير الطول قياسه 25 محرف **varchar(25)** لا يقبل الـ **NULL** ، و سنضع فيه القيمة الافتراضية في حال أن المستخدم لم يقوم بإدخال اسمه هي **no name.**

3. الوظيفة **job :** هذا العمود يضم اسم الوظائف الخاصة بالموظفين فهو إذن عمود محرفي متغير الطول قياسه 20 محرف.
4. راتبه المقطوع **salary :** عمود يتضمن راتب الموظف، نوع العمود هو رقمي **float** قياسه 8 خانات منها خانتين للكسور و لا يقبل الـ **NULL** و القيمة الافتراضية هي 8000 .
5. الحوافز **bonus :** و هذا العمود يضم الحوافز و المكافآت التي يأخذها الموظف و نوع العمود هو **float(7,2)** و يقبل الـ **null** أي أنه من الممكن ألا يكون للموظف حوافز أو مكافآت.
6. تاريخ مباشرة العمل **date :** عمود يحوي التاريخ الذي انتسب فيه الموظف إلى الشركة و نوع العمود هو **timestamp** و قياس العمود هو 8 أي **YYYY-MM-DD** و يقبل الـ **NULL** .

هذا الكلام الذي ذكرناه تواً يتم تدوينه على المسودة و عند وضع اللمسات الأخيرة على تصميم الجدول، و بعد ذلك تقوم بتنسيقه كما هو موضح في الشكل التالي:

Table Name : Employees

اسم العمود	نوع العمود	NULL	المفتاح	القيمة الافتراضية	سمات أخرى
emp_no	int		primary		auto_increment
name	varchar(25)			no name	
job	varchar(20)			seller	
salary	int (6)			8000	
bonus	int (5)	yes			
date	timestamp(8)	yes			

قد تعتقد أن هذا الشكل معقد بعض الشيء لكنه ليس كذلك فاسم العمود هو مألوف لديك و هو الاسم الذي ستضعه للعمود.
أما نوع العمود فالمقصود به هو نوع البيانات المدخلة في هذا العمود، فعمود **name** تكون بياناته أسماء أعلام و بالتالي فإن نوع البيانات المدخلة فيه هي محارف أي **char** و لكني في الجزء الأول من درس (أنواع الأعمدة و الجداول في **MySQL**) كنت قد شرحت أن النوع **varchar** أفضل من النوع **char** .
أما **null** فلاحظ الأعمدة التي تقبل القيمة **NULL** مكتوب عندها **yes** و التي لا تقبلها لا يوجد شيء مكتوب، و قد منعت الأعمدة (**name , salary , job**) من قبول القيمة **null** لأنه ليس من المعقول أن يكون موظف بلا اسم أو راتب أو وظيفة.
و بالنسبة إلى المفتاح فهناك نوعين من المفاتيح و هي المفتاح الأساسي **Primary key** و المفتاح الغريب **foreign key** و كنت قد شرحت عنهما في الجزء الأول من درس مفاهيم في قواعد البيانات العلائقية.
أما القيمة الافتراضية و سمات أخرى، أيضاً شرحتهما في درس (استخدام التعليمة **Create** في قواعد بيانات **MySQL**).
هذا هو الشكل الرئيسي لإنشاء أي جدول فعندما تعمل أنت مع فريق عمل برمجي و يكون المطلوب منك هو تصميم قاعدة البيانات التي تصنعها أنت و مجموعتك، فعندها تقدم الشكل السابق.
و الآن سأقوم بكتابة الكود اللازم لبناء هذا الجدول:

```
1.  create table employees(
2.  emp_no int auto_increment,
3.  name varchar(25) not null default 'na name',
4.  job varchar(20) not null default 'seller',
5.  salary int(6) not null default 8000,
6.  bonus int(5),
7.  date timestamp(8),
8.  primary key(emp_no),
9.  index index_on_name(name)
10. ) type=myisam;
```

- أولاً :
لاحظ أنه من السطر رقم 2 إلى السطر رقم 8 يوجد فاصلة عند نهاية الأمر للفصل بين كل أمر و الأمر التالي له، أما السطر 9 فلا يوجد فاصلة و ذلك لأنها هي آخر تعليمة داخل القوسين الحمر.
- ثانياً :
لاحظ في السطر 8 حيث أن المفتاح الأساسي وضعته كسطر منفرد حيث أنه من الممكن أن أضعه كسمة للعمود **emp_no** كما يلي:

emp_no int auto_increment primary key,

و لكن قد تقول ما الفائدة من وضعه بسطر منفرد؟؟ الفائدة هي أنه في بعض الحالات تحتاج فيها أن يكون المفتاح الأساسي على أكثر من عمود، ففي مثالنا هذا إذا كان المفتاح الأساسي على العمودين **emp_no , name** فيتم ذلك كما يلي:

primary key (emp_no , name),

- ثالثاً:
في السطر التاسع قمنا بإنشاء فهرس، لكن الآن يجب أن تعرف الآن أن إنشاء الفهرس يكون وفق الصيغة القواعدية التالية:

index index_name(indexed_col)

- رابعاً:
في السطر 10 و الأخير قمنا بتعيين نوع الجدول و قد اخترته من النوع **myisam** و إذا لم أقم بوضع نوع للجدول فإن **MySQL** سيقوم بوضع هذا النوع افتراضياً.
و بهذا تكون قد تمكنت من إنشاء أول جدول في قاعدة البيانات **movie_store** .

الجدول addresses:

و الآن سنقوم ببناء الجدول الثاني و هو الجدول **addresses** .
بداية خذ هذا التصميم لجدول **addresses** كما هو موضح في الجدول التالي:

Table Name : addresses

اسم العمود	نوع العمود	NULL	المفتاح	القيمة الافتراضية	سمات أخرى
add_no	int		primary		auto_increment
emp_no	int(11)		foreign key refers employees(emp_no)		
state	varchar(30)			Syria	
emp_email	varchar(40)	yes			

- أولاً:
لاحظ بأن العمود **emp_no** هو عمود مفتاحي و نوعه مفتاح غريب **foreign key** ، و كنت قد شرحت عنه في الفصل الأول في مقطع ما هي قاعدة البيانات العلائقية، أما عن كيفية إنشاء المفتاح الغريب فستتعلم ذلك الآن إن شاء الله.
- ثانياً:
الكود الخاص بالجدول **addresses**

```
create table addresses(
add_no int primary key auto_increment,
emp_no int(10),
state varchar(30) not null default 'Syria',
emp_email varchar(40),
foreign key(emp_no) references employees(emp_no));
```

طبعاً الأسطر الخمسة الأولى باتت مألوفة لديك، و لاحظ في السطر الثاني كيف أن**primary key** وضعناها على شكل سمة.
أما بالنسبة للسطر الأخير فهو لإنشاء المفتاح الغريب، حيث أن جملة **foreign key** تقوم بإعلام الملقم بأنني أريد أن أنشئ عموداً هو مفاتح غريب، و بين القوسين نقوم بوضع اسم العمود الذي أريده أن يكون مفتاح غريب، أما الكلمة **references** فمعناها (يشير إلى) ثم أكتب اسم الجدول الذي أريد أن يشير المفتاح الغريب إليه و بداخل القوسين أضع اسم العمود الذي يشير إليه المفتاح الغريب.

كيف لك أن تعرف بأن هذا المفتاح الغريب يشير إلى ذاك العمود من ذاك الجدول؟ الجواب هو أنه يجب أن تكون ملماً بشكل جيد بتحليل و تصميم قواعد البيانات، أما هذه الدروس فهي لتعليم كيفية برمجة قواعد بيانات الإنترنت. راجع الفصل الأول كي تأخذ فكرة عامة عن تحليل و تصميم نظم قواعد البيانات.

و بالعودة إلى الجدولين **employees , addresses** ستجد أنك تعلمت بعون الله كيف تبني الجداول برمجياً و تضع أعمدتها و تحدد نوع كل عمود و سماته، كما تعلمت أيضاً كيفية إنشاء المفتاح الأساسي، و المفتاح الغريب، و بقي عليك شيء واحد هو الفهارس، أعود و أذكرك بأن الفهارس ستتعلمها لاحقاً إن شاء الله.
يوجد في **MySQL** تعليمة مفيدة تظهر لك الكيفية التي تم بواسطتها بناء الجدول و هي:

```
show create table table_name \G;
```

طبعاً الخيار **\G** هو ليس جزءاً من التعليمة بل هو من أجل تغيير طريقة عرض نتيجة هذا الأمر من الشكل الجدولي إلى الشكل الأفقي، و لكي ترى خطوات بناء الجدول **employees** أكتب:

```
show create table employees;
```

و الفائدة من هذه التعليمة هو أنه في حال أردت أن تقوم بعملية صيانة للجدول فأنك يجب أن تعرف هيكليته كي تتمكن من تغير خصائصه أو سماته أو أي شيء فيه.
حسناً .. لقد تم بناء جدولين أمامك، و الآن عليك أن تقوم ببناء الجداول التالية لوحدك.
و إليك هيكلية الجداول التي نريد منك بناءها:

1. الجدول **Movies :**

Table Name : Movies

أسم العمود	نوع العمود	**NULL**	المفتاح	القيمة الافتراضية	سمات أخرى
mov_no	int		primary		auto_increment
mov_name	varchar(20)			no movie	
star_1	varchar(30)			no star	
star_2	varchar(30)			no star	
kind	varchar(15)			violence	
price	int(6)			500	

2. الجدول **Clients :**

Table Name : Clients

اسم العمود	نوع العمود	**NULL**	المفتاح	القيمة الافتراضية	سمات أخرى
cli_no	int		primary		auto_increment
cli_name	varchar(30)			no name	
cli_email	varchar(40)			have not email	

3. الجدول **Suppliers :**

Table name : Suppliers

اسم العمود	نوع العمود	NULL	المفتاح	القيمة الافتراضية	سمات أخرى
sup_no	int		primary		auto_increment
sup_name	varchar(30)			no name	
sup_email	varchar(40)			haven't email	

لاحظ أن القيمة الافتراضية للعمود **sup_email** تحوي الفاصلة العلوية و هذا المحرف هو من العلامات الخاصة بـ **SQL** و التي سنتعرف عليها بإذن الله لاحقاً، و لكي يدرك ملقم **MySQL** بأن هذه الفاصلة هي محرف و ليس علامة خاصة، يجب أن تسبقها علامة الهروب **back slash** لذلك هكذا تكتب عند إنشاء الجدول :

sup_email varchar(40) not null default 'haven\'t email'

و لا تكتب هكذا:

sup_email varchar(40) not null default 'haven't email'

4. الجدول **Bills :**

Table Name : Bills

اسم العمود	نوع العمود	NULL	المفتاح	القيمة الافتراضية	سمات أخرى
bil_no	int		primary		auto_increment
cli_no	int(11)		foreign refers clients(cli_no)		
mov_no	int(11)		foreign refers movies (mov_no)		
paid	int(4)			250	
bil_date	timestamp(12)				

إذن هذه هي الجداول التي نريدها لبناء قاعدة البيانات الخاصة بنا، قم ببناء هذه الجداول كما تعلمت في هذا الفصل، و بذلك نكون قد انتهينا من بناء قاعدة بياناتنا **movie_store** .

التعليمة **alter table**

يتم تعديل بنية الجداول عن طريق التعليمة **alter table** ، و تبدأ هذه التعليمة دائماً بـ **alter table** **table_name** أما تتمة هذه التعليمة فتتغير بتغير الهدف منها.

تغيير اسم الجدول:

الصيغة القواعدية لهذه التعليمة:

alter table table_name rename new_table_name;

فمثلاً لنفرض أنك تجد الاسم **employees** هو اسماً طويلاً و تريد تغييره إلى اسم مختصر قصير و لنفرض **emp** فيمكنك عمل ذلك كما يلي:

alter table employees rename emp;

و هناك طريقة أخرى لتغيير اسم الجدول و لكنها تستخدم مع الإصدار **3.23.27** من **MySQL** أو الإصدارات الأحدث و هي:

rename table table_name to new_table_name;

و بتطبيق ذلك على مثالنا نكتب:

rename table employees to emp;

إضافة الأعمدة:
الصيغة القواعدية لإضافة عمود إلى جدول، هي:
alter table table_name add column col_name type attributes;

فعلى سبيل المثال أنك تريد أن تضيف عموداً إلى الجدول **suppliers** يتم فيه تخزين أرقام الجوالات الخاصة بالموردين فيتم ذلك كما يلي:
alter table suppliers add column mobile varchar(15) not null;

كما أنه يمكنك في **MySQL** أن تحدد ترتيب موقع العمود أثناء إضافته، فإذا أرت أن تضعه في أول الأعمدة تكتب **first** في آخر التعليمة كما يلي:
alter table suppliers add column mobile varchar(15) not null first;

أما إذا أردت أن تضعه بعد عمود ما و ليكن مثلا **sup_name** يكتب **after sup_name** في نهاية التعليمة كما يلي:
alter table suppliers add column mobile varchar(15) not null after sup_name;

حذف الأعمدة:
الصيغة القواعدية لعملية حذف عمود هي:

alter table table_name drop column col_name ;

فمثلا لحذف العمود **mobile** الذي قمنا بإضافته نكتب:

alter table suppliers drop column mobile ;

لاحظ أنه عند الحذف لا تضع نوع و سمات العمود المراد حذفه.

تغيير خصائص العمود :
ربما تريد في بعض الأحيان أن تغير بعض السمات أو نوع أو اسم عمود ما، إن ذلك ممكن في **MySQL** و يتم ذلك عن طريق أحد الأمرين **change , modify** و إليك الصياغة القواعدية لهذا الأمر:

alter table table_name change old_col_name new_col_name type attributes ;

إن الإمكانيات المتاحة لك خلال هذا الأمر هي:
1. من الممكن أن تغير فقط اسم العمود: لذلك عند استخدام هذه التعليمة ستقوم فقط بتغيير اسم العمود مع المحافظة على النوع و السمات.
2. من الممكن أن تغير فقط النوع أو السمات أو كلاهما: و يتم ذلك عن طريق وضع الاسم الجديد نفس الاسم القديم مع تغيير النوع أو السمات أو كلاهما، و هنا ستفضل استخدام الأمر **modify** الذي يستخدم كما يلي:
alter table table_name modify col_name new_type new_attributes;

لنفرض أنك تريد أن تغير في الجدول **emp** طول العمود **name** من **varchar(25)** إلى الطول **varchar(20)** فستكتب:
alter table emp modify name varchar(20);

3. من الممكن أن تغير كل من السمات و النوع و الاسم: و ذلك بكتابة اسم و نوع و سمات جديدة.

إضافة فهرس:
يمكنك إضافة فهرس إلى جدول ما باستخدام أحد الأوامر **index , unique , primary key** و إليك الصيغة القواعدية للأوامر الثلاث التالية:

1. **index :**

```
alter table table_name add index index_name(indexed_col);
```

مع الأخذ بعين الاعتبار أنه من الممكن أن يمتد الفهرس على أكثر من عمود أي كما يلي:

```
alter table table_name add index ind_name(indexed_col_1 , indexed_col_2 , ... );
```

2. **unique :**

حيث أن **unique** تستخدم للتعبير عن القيم الفريدة غير المتكررة.

```
alter table table_name add unique index_name(col_name);
```

3. **primary key :**

```
alter table table_name add primary key (col_name);
```

حذف فهرس:
يتم حذف الفهرس باستخدام التعليمة **drop** كما يلي:

```
alter table table_name drop index index_name;
```

إن وضع اسماً للفهرس هو أمراً اختيارياً، و لكن من الأفضل أن تضع له اسماً ، فلو أنك قمت بإنشاء فهرس و لم تضع له اسماً و أردت بعد ذلك حذفه ماذا ستفعل؟
لا تقلق فإن **MySQL** ستقوم بإعطاء اسماً افتراضياً للفهرس و هو نفس العمود الذي يتم فهرسته، و تستطيع معرفة اسم الفهرس عن طريق الأمر **show index** الذي يقوم بعرض الفهارس الموضوعة على جدول.

إضافة مفتاح غريب:
لنفرض أننا نريد أن نضيف مفتاح غريب إلى الجدول **bills** على العمود **cli_no** ليشير إلى العمود **cli_no** من الجدول **clients** ، سيتم ذلك كما يلي:

```
alter table bills add foreign key(cli_no) references clients(cli_no) ;
```

التعليمة **drop**
تستخدم التعليمة **Drop** لحذف الجداول و قواعد البيانات.

حذف جدول:
في بعض الأحيان يكون أحد جداولك بحاجة إلى صيانة كبيرة و لا يوجد بداخله بيانات تخسرها، لذلك من الأفضل أن تقوم بحذفه من أساسه، ثم إعادة بناء واحد جديد،و تستطيع أن تحذف الجداول باستخدام التعليمة **drop** كما يلي:

```
drop table table_name;
```

كما و تستطيع أن تحذف عدة جداول معاً

```
drop table table_1 , table_2 , table_3 ... ;
```

حذف قاعدة بيانات:
يتم حذف قاعدة بيانات بما تحتويه من جداول و بيانات وفق التعليمة التالية:

```
drop database DB_name;
```

و بذلك نكون بعون الله قد انتهينا من مجمل تعليمات القسم الأول من **MySQL** و هي لغة **DDL** (**Data Definition Language**).

الفهارس indexes

قد تتساءل ما الفائدة من الفهارس و ماذا تقدم لنا الفهارس؟
لنفرض أنك تبحث في قاموس عن كلمة ما و لنفرض أنها كلمة **System** فإنك ستفتح باب حرف الـ **S** ثم تتجاوز عدة صفحات حتى تصل إلى الحرف الثاني **Y** و كذلك الأمر بالنسبة إلى باقي الحروف، إذن لقد وصلت إلى مبتغاك بسرعة، حسناً .. تصور أنك تبحث عن هذه الكلمة في قاموس عشوائي**!** إذن ستقوم بالبحث في كل صفحة و كل كلمة، و هذه هي الصورة بالنسبة للفهارس في قواعد البيانات، دعنا نأخذ المثال التالي، لنفرض أن لديك جدول مؤلف من عمودين الأول مفهرس و الثاني غير مفهرس كما يلي:

```
create table my_table(
col_1 varchar(15) not null primary key,
col_2 varchar(15) not null );
```

ثم قمت بإدخال 20000 صف في الجدول، ثم قمت بكتابة استعلامين متشابهين بالصياغة القواعدية كما يلي:

```
select * from my_table where col_1=3;
select * from my_table where col_2=3;
```

لا تهتم الآن بالاستعلامين لأننا سنتكلم عن الاستعلامات في **MySQL** لاحقاً إن شاء الله.
في الاستعلام الأول سيستخدم **MySQL** فهرساً من أجل التعليمة **where** بينما في الاستعلام الثاني سيتم البحث في كل الصفوف، لذلك فإن الاستعلام الأول أسرع بعدة مرات من الاستعلام الثاني على الرغم من أن كلاهما استعلامات سريعة، لكن سيختلف الحال مع ملقم ويب نشط يقوم بمعالجة مئات الاستعلامات في الثانية أضف إلى ذلك ازدياد حجم الجداول و الاستعلامات، و هنا تبرز أهمية الفهرس.
و الصيغة القواعدية لإنشاء فهرس هي:

```
index index_name(indexed_col);
```

طبعاً هذه الصيغة تستخدمها مع التعليمة **create table** .
أن **indexed_col** هو اسم العمود الذي تمت فهرسته، و من الممكن أن يمتد الفهرس على أكثر من عمود، و الحد الأقصى هو 16 عمود، و يتم إنشاء الفهرس على أكثر من عمود كما يلي:

```
index index_name(col_1,col_2, ... ,col_16);
```

أما مبدأ عمل الفهرس فيمكنك أن تتخيل أن العمود المفهرس هو كخط مستقيم و يتم تخزين قيمة وسطية في الفهرس فإذا جاء استعلام يطلب قيمة أكبر من القيمة الوسطية فإن الفهرس يبحث باتجاه اليمين فرضاً و إذا جاء استعلام يطلب قيمة أصغر القيمة الوسطية فسيبحث الفهرس باتجاه اليسار.

أنواع الفهارس:

للفهارس ثلاث أنواع و هي:

1. **primary key :**

حيث أن **MySQL** يقوم بفهرسة المفتاح الأساسي **Primary Key** لذلك فهو يعد نوع من أنواع الفهارس.

2. **unique :**

كلمة **unique** معناها وحيد أو فريد و هي في **MySQL** تدل فعلاً على معناها، و عندما نصرح عن عمود أنه فريد فإننا نقصد بذلك أن القيم التي يتم إدخالها في هذا العمود هي قيما فريدة أي غير متكررة، و على سبيل المثال إذا كان لدينا عموداً يحوي البريد الإلكتروني فمن المستحيل أن يكون هناك بريدين إلكترونيين متشابهين، لذلك هذا العمود يضم قيماً غير متكررة أي فريدة، أما إذا كان لدينا عمود يحوي أسماء فمن الممكن جداً أن تتكرر الأسماء و بالتالي فإن هذا العمود يضم قيما متكررة.
كيف أتمكن من إضافة عمود فريد **unique** : تتم إضافته تماماً مثل المفتاح الأساسي أي أنك تستطيع أن تضيفه كما لو أنه سمة كما في المثال التالي:

```
create table test( col_1 varchar(10) not null unique );
```

أو بشكل منفصل كما يلي:

```
create table test(
col_1 varchar(10) not null,
unique index_name(indexed_col));
```

ما الفرق بين الطريقتين: في الطريقة الأولى هناك سيئتين الأولى أنك لن تستطيع أن تمد الفهرس على أكثر من عمود و الثانية أن هذا الفهرس سيكون بلا اسم (لكن **MySQL** يعطيه افتراضياً نفس اسم العمود الذي تمت فهرسته)، وفي الطريقة الثانية هناك حسنتين الأولى أنك تستطيع أن تمد الفهرس على أكثر من عمود و الثانية أنك تستطيع وضع اسماً لهذا الفهرس.

3. **index :**

ذكرنا في الفقرة (إنشاء جدول قاعدة بيانات المثال) من الفصل الثاني (تعليمات **MySQL**) كيفية إنشاء الفهرس من النوع **index** .

كما قلنا أنه من الممكن أن ننشئ فهرساً على أكثر من عمود، و أيضاً من الممكن أن ننشئ فهرساً على جزء من عمود، أي أنه في حال كان لدينا عموداً نوعه **varchar** و قياسه 70 محرف أي **varchar(70)** ، فيمكننا أن ننشئ فهرساً على أول 10 محارف فرضاً، أو على أول 23 محرف، و ذلك حسب ما تريد.

حسناً لنأخذ هذا المثال التوضيحي، بفرض أنه لدينا جدولاً كما يلي:

```
create table my_table(
varchar_col varchar(120) not null,
text_col text not null,
index index_on_char_col(char_col(25)),
index index_on_text_col(text_col(200)));
```

بالنظر إلى هذا المثال ستلاحظ بأن الفهرس الأول لم يتم إنشاءه على كامل العمود بل على القسم الاستهلالي من العمود **varchar_col**.

و الفهرس الثاني تم إنشاءه فقط على أول 200 محرف من العمود **text_col** .

و الخلاصة أنه يمكن أن تنشئ فهرساً على القسم الاستهلالي من العمود الذي يكون نوعه **char , varchar** ، و أيضاً يمكنك إنشاء فهرساً على أول 255 محرف من الأعمدة ذات النوع **text , tinytext , longtext , mediumtext** هكذا تقريباً تكون الصورة قد اكتملت عن الفهارس.

قد تقول لماذا لا أنشئ فهارس على جميع الأعمدة و تنتهي المشكلة. طبعاً هذا الكلام خطأ، ذلك لأن الفهارس تستولي على جزء من مصادر النظام، وفي كل مرة يتم تشغيل أحد تعليمات **update , insert** سيتم تحديث الفهارس التي تم تطبيقها على هذا الجدول، فإذا كان الجدول مستخدماً من أجل عمليات الإدراج فقط، فإن وجود الفهارس غير ضروري على ذاك الجدول لأنه سيسبب بطئاً في أداء **MySQL** و بالإضافة إلى أنها ستستحوذ على جزءاً من ذاكرة الوصول العشوائي **RAM**

الفصل الرابع
لغة معالجة البيانات DML

سنتكلم في هذا الفصل عن لغة معالجة البيانات **DML** إن شاء الله.

التعليمة **insert**

إن قاعدة البيانات **Movie_Store** التي قمت بإنشائها سابقاً، كانت خالية من البيانات لذلك ستتعلم كيفية إدراج البيانات أو حذفها أو التعديل عليها بإذن الله.

إن التعليمة **insert** تمكنك من إدراج البيانات في الجداول، و إليك الصيغة القواعدية لهذا الأمر:

```
insert into Table_name(col_1,col_2,...,col_n) values('val_1' , 'val_2' , ... , 'val_n');
```

أعتقد أنك تقول بأن الصيغة سهلة لكنها غير واضحة تقريباً، حسناً سنوضحها بتطبيق عملي:

سنقوم بإدراج معلومات عن فيلم ما في الجدول **movies** ، طبعاً يجب أن تعرف ما هي الأعمدة الموجودة في هذا الجدول، و الأعمدة هي:

1. **mov_no :** و هذا العمود هو عمود ترقيم تلقائي، و طبعاً هذا العمود لن ندرج به قيمة لأنه يأخذ أرقاماً تسلسلية تلقائية.
2. **mov_name :** سنضع فيه اسم الفيلم المراد إدراجه، و ليكن فيلم **assassins** .
3. **Star_1 :** سنضع فيه اسم البطل الأول للفيلم، و ليكن **Sylvester stallone** .
4. **Star_2 :** سنضع فيه اسم البطل الثاني للفيلم إذا كان هناك بطل ثاني. أدخل إلى بنية هذا الجدول باستخدام التعليمة **show columns from movies** و لاحظ أن هذا العمود يقبل القيمة **null** ، و هنا البطل الثاني اسمه **Antonio Banderas**.
5. **Kind :** سنضع فيه نوع الفيلم و المقصود بنوع الفيلم إما رعب أو عنف أو بوليسي أو رومانسي ... الخ من أنواع الأفلام التي تعرفها، لاحظ القيمة الافتراضية هي **violence** باعتبار أن معظم الأفلام هي أفلام عنف، لذلك دائماً اجعل القيمة الافتراضية هي القيمة الأكثر استخداماً، و في حالتنا هذه فإن فيلم **assassins** هو فيلم عنف أي **violence** .
6. **price :** و هنا سنضع سعر الفيلم و ليكن 250 .

و بالعودة إلى التعليمة **insert** سنقوم بإدراج قيم في الأعمدة التالية فقط: **mov_name , star_1 , star_2 , price**

سأكتب الآن تعليمة الإدراج و قارن بينها و بين الصيغة القواعدية لها:

```
insert into movies(mov_name,star_1,star_2,price) values
('assassins' , 'sylvester stallone' , 'A. banderas' , 250 );
```

لاحظ أن القيم المحرفية يتم إحاطتها بعلامة اقتباس مفرده **single quote** أما القيم الرقمية و العددية فلا يتم ذلك.

قم بكتابة الاستعلام التالي:

```
select * from movies;
```

ستظهر لك إن شاء الله النتيجة التالية:

```
+----------+------------------+------------------+---------------------+----------+---------------+
| mov_no | mov_name       | star_1           | star_2              | kind     | price         |
+----------+------------------+------------------+---------------------+----------+---------------+
| 1        | assassins        | sylvester stalon | antonio banderas | violence | 250.00        |
+----------+------------------+------------------+---------------------+----------+---------------+
```

لاحظ كيف أن **mov_no** قد أخذ القيمة واحد تلقائياً علما بأننا لم ندرج قيمة، و هناك صفة سيئة للأعمدة التي من النوع **auto_increment** إذ أنه يمكنك أن تدرج رقماً عن طريق تعليمة **insert** حتى لو لم يكن متسلسل.
و لاحظ أيضاً كيف أن العمود **kind** أخذ تلقائياً القيمة الافتراضية **violence** عندما لم يتم إدراج قيمة.
و لاحظ أيضاً أن العمود **price** قد أخذ الرقم الذي أعطيته إياه و وضع له فاصلة ثم صفرين و ذلك لأن نوع هذا العمود من النوع **float** مع دقة بمقدار رقمين بعد الفاصلة.
و هناك خاصية حلوة في **MySQL** هي أنه يمكنك أن تدرج عدة صفوف مع وضع فاصلة بين كل صف، و يتم ذلك كما يلي:

```
insert into table_name(col_1 , ... , col_n) values('val_1' , ... , 'val_n') , ('another_val' , ... , 'another_n');
```

و بهذه التعليمة تكون قد قمت بإدراج صفين معاً.
و أعود و أذكر بأنه في حال أردت أن تدرج محارف خاصة في عمود ما مثل علامة الاقتباس المفردة **Single quote** أو علامة الاقتباس المزدوجة **Double quotation** أو علامة الهروب **back slash** فيجب أن تستخدم علامة الهروب كما يلي:

```
insert into my_table(col_1) values ('it\'s for you');
insert into my_table(col_1) values ('c:\\windows');
```

بعد أن تعلمت جميع خيارات التعليمة **Insert** قم الآن بإدراج البيانات التالية في الجدول **emp** كما يلي:

name	job	salary	bonus	date
omar	manager	12000	3000	2000-1-1
nour	programmer	---	500	2000-1-15
ahmad	---	5000	---	2000-2-1
ameen	---	5000	0	2000-2-3
osama	programmer	9000	250	2001-5-22

و يتم الإدراج وفق تعليمات **insert** التالية:

```
insert into emp(name, job, salary, bonus, date)
values('omar' , 'manager' , 12000 , 3000 , '2000-1-1' );
insert into emp(name, job, bonus, date) values('nour' , 'programmer' , 500 , '2000-1-15');
insert into emp(name, salary, date) values('ahmad' , 5000 , '2000-2-1');
insert into emp(name, salary, bonus, date) values('ameen' , 5000 , 0 , '2000-2-3');
insert into emp(name, job, salary, bonus, date)
values('osama' , 'programmer' , 9000 , 250 , '2001-5-22' );
```

ثم قم بإدراج البيانات في الجدول **adrs** كما يلي:

emp_no	state	emp_email
1	Syria	omar@mail.sy
2	Egypt	nour@lycos.com
3	Syria	---
4	Yemen	ameen@lycos.com

أما تعليمات الـ **insert** التي ستكتبها فهي كما يلي:

```
insert into adrs(emp_no,state,emp_email)
values(1, 'Syria', 'omar@mail.sy') , (2, 'Egypt', 'nour@lycos.com);
insert into adrs(emp_no,state) values(3,'Syria');
insert into adrs(emp_no,state,emp_email) values(4, 'Yemen', 'ameen@lycos.com');
```

و الآن ستقوم بإدراج البيانات في باقي الجداول لوحدك كما يلي:

1. الجدول **movies :**

mov_name	star_1	star_2	kind	price
last man standing	brouce wiles	---	---	250
city of angels	nicolas cage	---	romance	300
urban legend	---	---	horror	300
the fist of legend	get lee	---	---	250

2. الجدول **clients :**

cli_name	cli_email
mohammed	mohammed@hotmial.com
anas	anas@hotmail.com
khalid	khalid@yahoo.com

3. الجدول **suppliers :**

sup_name	sup_email
salem	salem@myway.com
azez	azez@mail.sy
hanan	---

العبارة **where**

تعرفت إلى الآن على الكثير من تعليمات **MySQL** ، التي تستخدم معظمها العبارة **where** لذلك رأيت أن تتعرف على هذه التعليمة قبل أن تغوص في بحر الـ **DML** حتى تحسن استخدامها هناك،
العبارة **where** في **MySQL** يقصد بها كلمة "حيث"، و سأوضح لك معناها و استخدامها من خلال الاستعلام التالي:

```
select * from movies;
```

هذا الاستعلام سيظهر لك كل أعمدة الجدول **movies** ، كل ما تحويه من بيانات، و لكنك لو كتبت الاستعلام التالي:

```
select * from movies where mov_name='assassins';
```

فإنه سيعطيك فقط جميع المعلومات عن الفيلم **assassins** الموجودة في الجدول **movies** ، لاحظ الصيغة القواعدية لهذا الاستعلام ستجد أنه يقسم إلى ثلاث أقسام :
القسم الأول هو **الاستعلام** و هو **select * from movies** ، دعك من هذا الجزء الآن.
القسم الثاني هو **الشرط** و هو **mov_name='assassins'** و هذا القسم هو الشرط الذي تضعه أنت كمبرمج على القسم الأول، أي الشرط الذي تضعه على الاستعلام، لاحظ أن هذا الشرط يحوي على **معامل مقارنة** و هو (=) إشارة المساواة، و معامل المقارنة هذا بدوره يربط بين اسم عمود و قيمة فيه حيث أن **mov_name** هو اسم عمود، و **assassins** هو قيمة في ذلك العمود، لاحظ أيضاً أن القيمة هنا وضعت ضمن علامتي اقتباس، لماذا؟ لأنها قيمة محرفية أو سلسلة محرفية، و لو كانت هذه القيمة قيمة عددية لما وضعناها ضمن علامتي اقتباس كما في الاستعلام التالي:

```
select * from movies where mov_no=1;
```

القسم الثالث هو العبارة **where** حيث تربط هذه العبارة الاستعلام بالشرط.
و أعتقد أن فائدة العبارة **where** قد أصبحت جلية لك، فالهدف منها هو تحديد عدد الصفوف التي ستتأثر بالاستعلام أو الأمر المكتوب، لفهم هذا الكلام بشكل أفضل قارنه بالاستعلامين السابقين ستتضح لك الصورة بشكل أفضل إن شاء الله.
و الآن عرفت أهمية استخدام **where** ، لتتعلم الآن كيفية استخدامها.

يرتبط استخدام العبارة **where** بالشرط الموضوع على الاستعلام، و الشرط الموضوع على الاستعلام يتألف من اسم عمود و قيمة في هذا العمود و **معامل المقارنة**، إليك الجدول التالي الذي يضم معاملات المقارنة **Comparable Operators** :

المعامل	المعنى	مثال
=	يساوي	**where col_1 = 5**
>	أكبر من	**where col_1 > 5**
>=	أكبر أو يساوي	**where col_1 >= 5**
<	أصغر من	**where col_1 < 5**
<=	أصغر أو يساوي	**where col_1 <= 5**
!=	لا يساوي	**where col_1 != 5**
is null	يحوي **null**	**where col_1 in null**
is not null	لا يحوي **null**	**where col_1 is not null**
between	يشير إلى مجال ما	**where col_1 between 1 and 5** **where col_1 between 'a' and 'b'**

هذه هي أهم استخدامات العبارة **where** و سنقدم أمثلة تطبيقية عليها حتى يزداد استيعابك لها، و سيكون الجدول المعتمد لتطبيق خيارات **where** عليه هو الجدول **emp** ، و لكن قبل أن نطبق الخيارات قم بعمل استعراض لمحتويات الجدول **emp** حتى تتعرف عليه عن كثب، و قم بكتابة الأمر التالي:

```
select * from emp;
```

سيظهر لك الناتج التالي:

```
+----------+-------------+---------------+-------------+-------------+-------------+
| emp_no | name       | job           | salary      | bonus       |date         |
+----------+-------------+---------------+-------------+-------------+-------------+
| 1        | Omar       | manager       | 12000.00    | 3000.00     | 20000101    |
| 2        | nour       | programmer    | 8000.00     | 500.00      | 20000115    |
| 3        | ahmad      | seller        | 5000.00     | null        | 20000201    |
| 4        | amen       | seller        | 5000.00     | 0.00        | 20000203    |
+----------+-------------+---------------+-------------+-------------+-------------+
```

لاحظ أن عمود الـ **date** قد يكون معقد بعض الشيء، و ذلك لأنه من التنسيق ذي النوع **timestamp**، لكن لتفهمه قم بقراءته من اليسار عندها ستلاحظ أن الموظف أمين كان قد انتسب إلى الشركة في تاريخ 03-02-2000 أي في اليوم الثالث من الشهر الثاني من عام 2000 ، والآن سنقوم إن شاء الله بتطبيق المعاملات السابقة على هذا الجدول بالترتيب و سنقوم بشرح الاستعلام الذي يوجب الشرح و المشكلات التي قد تواجهنا أثناء تطبيق الاستعلام حتى تصبح الصورة واضحة تماماً عن العبارة **where** ، و أما الاستعلام البسيط فأعتقد بأنك قادر بعون الله على تحليله لوحدك.

1. معامل المساواة:

```
select * from emp where emp_no=1 ;
select * from emp where name='omar' ;
```

لاحظ في الاستعلام الأول أنه في حال استخدام القيم العددية لا داعي لاستخدام علامتي الاقتباس.
لاحظ في الاستعلام الثاني أنه تم استخدام علامتي الاقتباس، و لاحظ أيضا أن **MySQL** لم يتأثر لحالة الأحرف عندما استخدمنا القيمة **omar** حيث أن هذه القيمة مخزنة بالشكل التالي **Omar** .

2. معامل أكبر من:

```
select * from emp where salary>8000;
```

3. معامل أكبر أو يساوي:

```
select * from emp where salary >= 8000;
```

4. معامل أصغر من:

```
select * from emp where salary < 8000;
```

5. معامل أصغر أو يساوي:

```
select * from emp where salary <= 8000;
```

6. معامل **is null** :

```
select * from emp where bonus in null;
```

7. معامل **is not null** :

```
select * from emp where bonus is not null;
```

8. معامل **between** :

```
select * from emp where salary between 6000 and 15000;
select * from emp where emp_no between 2 and 4;
```

9. المعامل **!=** :

```
select * from emp where emp_no != 1 ;
```

الأمر بسيط جدا أليس كذلك؟ لكن الخطورة ليس هنا بل الخطورة تكون حيث يصطدم هذا المعامل مع الـ **null** !! كيف ذلك؟ أنا سأخبرك، قم بكتابة الاستعلام التالي:

```
select * from emp where bonus = 3000;
```

لاحظ النتيجة، قام **MySQL** بعرض سجل السيد عمر فقط أليس كذلك؟ و الأمور إلى الآن بخير، حسناً قم بكتابة الاستعلام التالي:

```
select * from emp where bonus != 3000;
```

هل لاحظت الخطأ الذي نتج؟ سيقوم **MySQL** فقط بعرض سجلي الموظفين نور و أمين بينما لم يعرض سجل الموظف أحمد! لماذا؟ لأن قيمة العمود **Bonus** لديه هو **Null** لذلك لم يتم عرضه، و للتخلص من هذه المشكلة هناك حلين:
الأول: هو أن تتجنب قدر الإمكان استخدام **null** في جداولك.
الثاني أن تضيف الجزء التالي **or col_name in null** إلى كل استعلام تتوقع أن يحتوي على **null** ، و بالعودة إلى الاستعلام السابق سيصبح شكل الاستعلام كما يلي:

```
select * from emp where bonud != 3000 or in null ;
```

و عندها سيقوم **MySQL** بعرض السجلات الثلاث بخير و سلام.
تتيح العبارة **where** وضع أكثر من شرط على الاستعلام، عن طريق التجميع بين الشروط باستخدام الكلمة المفتاحية **and** أو **or** ، فإذا كتبنا استعلام وظيفته إظهار معلومات عن الموظفين الذين أرقامهما 1 و 3 فقط نكتب:

```
select * from emp where emp_no=1 or emp_no=3 ;
```

لاحظ أخي القارئ أنك يجب ألا تكتب الاستعلام السابق كما يلي:

```
select * from emp where emp_no=1 and emp_no=3;
```

فالاستعلام الأول معناه هو (قم يا ملقم **MySQL** بالاستعلام عن الموظفين الذين أرقامهم 1 أو 3)، عندها سيقوم الملقم بإظهار المعلومات عن الموظفين ذوي الأرقام 1 أو 3 .

أما الاستعلام الثاني فمعناه هو (قم يا ملقم **MySQL** بالاستعلام عن الموظفين الذين أرقامهم 1 و 3) و هذا مستحيل لأن كل موظف له رقم واحد فقط و ليس رقمين.
و أظنك قد أدركت الفرق بين **and** و **or** حيث أن الكلمة المفتاحية **and** تجمع عدة شروط لسجل واحد فقط، بينما الكلمة المفتاحية **or** تجمع عدة شروط لعدة سجلات.
ستقول لي ما الفائدة من الكلمة **and** ، حسناً الفائدة هي أنها تخولك من جمع أكبر عدد من الشروط لحصر البحث، أي ليظهر لك البحث أقل عدد ممكن من النتائج، و لتوضيح ذلك لاحظ عمليات البحث في محرك بحث و ليكن **Google** مثلاً ، هناك خيار اسمه "بحث متقدم" ، و تستطيع عن طريق هذا الخيار أن تكثر من شروط البحث بحيث تحصل على مرادك.
و بذلك نكون قد قدمنا صورة شبه متكاملة عن العبارة **where** و أتمنى أن تكون قد أصبحت مألوفة لديك، حيث ستزداد أهميتها في الفصول القادمة و خاصة في عمليات الاستعلام المتقدم.

التعليمة update

في قاعدة البيانات **movie_store** لديك جدول اسمه **movies** و يعد هذا الجدول هو أرشيفك الخاص لكل الأفلام التي تمتلكها أو لك حقوق بيعها، فإذا أردت يوماً ما أن تدرج فيلماً جديداً إلى أرشيفك، فإنك ستقوم بذلك عن طريق التعليمة **insert** ، أما لتغيير (تحديث) معلومات ما خاصة بفيلم معين، فما سبيلك إلى ذلك؟ تقدم لنا **MySQL** التعليمة المناسبة للتحديث و هي **update** و إليك الصيغة القواعدية العامة لهذا الأمر:

```
update Table_name set col_1=val_1 , col_2=val_2 , ... , col_n=val_n where ... ;
```

لنقوم بتجربة هذه التعليمة عملياً..
لنفرض أنك ستختصر اسم بطل فيلم **assassins** من **sylvester stallone** إلى **S. Stallone** ، ستتمكن من ذلك عن طريق استخدام التعليمة **update** كما يلي:

```
update movies set star_1='S. Stallone' where mov_name='assassins';
```

لاحظ أن هذه التعليمة بسيطة جداً و هي تشرح نفسها بنفسها.
حسناً لنقوم بإجراء تحديث آخر لنفس الجدول، فلنفرض أنك تريد أن تغير قيمتين أو أكثر في آن واحد لنفس السجل، مثلاً تريد أن تغير اسم الفيلم **urban legend** إلى الفيلم **others** و تجعل البطل الأول له هو **nicole kidman** ، فهذه التعليمة تفي بالغرض:

```
update movies set mov_name='others' , star_1='nicole kidman'
where mov_name='urban legend';
```

كما و يمكنك إجراء نفس التحديث السابق كما يلي:

```
update movies set mov_name='others' , star_1='nicole kidman' where mov_no=4;
```

لاحظ المرونة الكبيرة التي تؤمنها لنا **MySQL** باستخدام العبارة **where** .
و يمكنك أيضاً تحديث أكثر من صف باستخدام تعليمة **update** واحدة، فلنفرض أنك تريد أن ترفع سعر كل أفلام العنف **violence** إلى القيمة 300 فما عليك سوى أن تكتب التعليمة التالية:

```
update movies set price=300 where kind='violence';
```

في حال عدم وضع الشرط **where** سيؤدي ذلك إلى تغيير كل شيء فمثلاً لو كتبت:

```
update movies set price=300;
```

سيؤدي ذلك إلى تغيير كل القيم الموجودة في العمود **price** و يجعلها تساوي القيمة 300
لذلك انتبه جيداً عند استخدامك التعليمة **update**

التعليمة replace

إن التعليمة **replace** هي تعليمة مقتصرة على قواعد بيانات **MySQL**، كما أنها ليست جزءاً من لغة **SQL** القياسية أو المعيارية.
أما الصيغة القواعدية لهذه التعليمة فهي:

```
replace into table_name(col_1 , col_2 , ... ) values(val_1 , val_2 , ... ) ;
```

لاحظ أن الصيغة القواعدية لهذه التعليمة تشبه تماماً الصيغة القواعدية للتعليمة **insert** ، و تعمل هذه التعليمة مع صف تعرف فيه قيمة المفتاح الأساسي، فعندما تقوم باستخدام التعليمة **replace** فإن **MySQL** سيبحث عن صف قيمة المفتاح الأساسي له تساوي قيمة المفتاح الأساسي المشار إليها (أي القيمة) في التعليمة، و بالتالي فإذا وجد تطابق سيتم التحديث (أي سيشغل تعليمة **update**) و إذا لم يجد تطابق فسيتم إدراج صف جديد (أي سيشغل التعليمة **insert**).
و ندرك من هذا الكلام أن التعليمة **replace** تعمل ضمنياً وفق الخوارزمية التالية:

إذا كانت القيمة الموجودة في التعليمة **replace** تطابق القيمة الموجودة في المفتاح الأساسي
قم باستخدام التعليمة **update**
و إلا
قم باستخدام التعليمة **insert**

كما أن التعليمة **replace** تشكل خطورة على البيانات، و لتوضيح آلية عمل **replace** و الخطورة الممكنة دعنا نأخذ المثال التالي:
سنقوم بتعديل اسم الفيلم **the fist of legend** إلى **fist of legend** ، مع العلم أن رقم الفيلم هو 5 حيث أن رقم الفيلم هو المفتاح الأساسي، و ذلك باستخدام التعليمة **replace** :
من البديهي أنك ستقوم بكتابة السطر التالي:

```
replace into movies(mov_no , mov_name) values(5 , 'fist of legend');
```

أولاً: آلية عمل **replace** : سيقوم ملقم قواعد البيانات بالبحث عن المفتاح الأساسي الذي يحمل القيمة 5 فإذا وجدها قام بإجراء التغييرات المناسبة، و إذا لم يجدها قام بإجراء عمليات الإدراج.
ثانياً: الخطورة الكامنة: الخطورة تتجسد عندما يجد الملقم القيمة لذلك سيتم تبديل قيم كل الأعمدة الموجودة في التعليمة و أما قيم الأعمدة الغير موجودة في التعليمة فسيتم إدراج القيم الافتراضية لها. كيف ذلك؟ حسناً .. بالعودة إلى التعليمة التي كتبتها فإنه سيتم تغيير القيمة المخزنة في **mov_no** إلى 5 و القيمة المخزنة في **mov_name** إلى **fist of legend** و أما قيم الأعمدة **star_1 , star_2 , kind , price** فستتغير القيم التي كانت مخزنة فيها إلى القيم الافتراضية، لذلك فإما أن تستخدم التعليمة **update** أو أن تضع كل الأعمدة في حال استخدامك التعليمة **replace** .
قد يتبادر إلى ذهنك سؤال ما فائدة التعليمة **replace** ؟ إن فائدتها تكمن في شيئين اثنين هما:
1. أنك بدلا من استخدام التعليمتين **update** و **insert** معاً، ستستخدم تعليمة واحدة.
2. و هو الأهم، إن طول المخطوطة **script** المكتوبة بلغة **PHP** سيكون أقصر بكثير و أقل تعقيداً في حال استخدامك تعليمة **replace** بدلاً من تعليمتي **update , insert** .
على العموم لا تقلق، فمع كثرة الاستخدام ستعرف إن شاء الله أي تعليمة ستستخدم و في مكانها المناسب، و بقي نقطة أخيرة هي أن التعليمة **replace** لا تقبل العبارة **where** .

التعليمة delete

تقوم التعليمة **delete** بإزالة الصف أو الصفوف من جدول، و الصيغة القواعدية لهذه التعليمة هي:

```
delete from table_name where ... ;
```

فلإزالة الصف الذي تكون فيه قيمة **mov_no=5** يمكنك استخدام الأمر التالي:

```
delete from movies where mov_no=5;
```

كن جذر عند استخدام التعليمة **delete** إذ أن البيانات المحذوفة لا يمكن استرجعها، إلا في حال كان لديك نسخة احتياطية عن بياناتك.

مخطوطة الـ **PHP**

فيما يلي ملف **PHP** بسيط قمت بكتابته لتوضيح فكرة الإدراج و الحذف و التعديل، و إليك التعليمات اللازمة لعمل هذه المخطوطة بشكل فاعل على جهازك:

- قم بنسخ هذه المخطوطة إلى أي محرر نصوص و ليكن **notepad** و احفظه باسم **test1.php** .
- قم بحفظ نسخة من هذا الملف في المسار التالي: **C:\apache\htdocs** و ذلك بفرض أن السواقة **C:** هي المكان الذي قمت بتنصيب **apache** عليه.
- من قائمة ابدأ قم بتشغيل كل من **apache , MySQL-D**
- قم بفتح متصفح الإنترنت لديك، و حرر في شريط العناوين العنوان التالي: **http://localhost/test1.php**

عندها سيظهر لك نموذج **PHP** بسيط، تعرف عليه بنفسك و جرب كل الأشياء الممكنة.

و فيما يلي شيفرة المخطوطة **test1.php**

```
<?php
$conn=mysql_connect("localhost" , "root" , "");
$db=mysql_select_db("movie_store");
if (! empty($mov_no) && empty($submit) )
        {
                $result = mysql_query("select mov_name , star_1 , star_2 , kind , price
                from movies where mov_no = $mov_no " ) ;
                list($mov_name,$star_1,$star_2,$kind,$price)=mysql_fetch_array($result);
        }
elseif (!empty($mov_no) && $submit=="update" )
        {
                $sql="replace into movies(mov_no,mov_name,star_1,star_2,kind,price) values
                ('$mov_no' , '$mov_name' , '$star_1' , '$star_2' , '$kind' , '$price')";
                mysql_query($sql) ;
                header ("Location: test1.php");
        }
elseif (empty($mov_no) && $submit == "update")
        {
                $sql = " insert into movies
                        (mov_name , star_1 , star_2 , kind , price) values
                        ('$mov_name' , '$star_1' , '$star_2' , '$kind' , '$price') ";
                mysql_query($sql) ;
                header ("Location: test1.php");
        }
elseif ( ! empty($mov_no) && $submit=="delete" )
        {
                $sql = " delete from movies where mov_no = $mov_no " ;
                mysql_query ($sql) ;
                header ( " Location: test1.php " ) ;
        }
echo ( " للحصول على نموذج فارغ أنقر <a href=\"test1.php\">هنا</a> لو سمحت " ) ;
echo '<br><br>';
$result = mysql_query( "select * from movies" );
while ( $row = mysql_fetch_array($result) )
```

```
        {
                echo "<a href=\"test1.php?mov_no=".$row["mov_no"]."\">";
                echo $row["mov_name"]." ... ".$row["star_1"]."<br>";
                echo "</a>" ;
        }
?>
<html>
<style type="text/css">
body{ font-family:"tahoma";
     font-weight:"bold";
     font-size:"10 pt";
     color:"#306800" }
</style>
<body>
<br><br>
<table>
<form method="get">
<td>رقم الفيلم</td>
<td><input type="text" name="mov_no" value="<?php echo $mov_no; ?>"></td>
<tr><td>اسم الفيلم</td>
<td><input type="text" name="mov_name" value="<?php echo $mov_name; ?>"></td>
</tr>

<tr><td>البطل الأول</td>
<td><input type="text" name="star_1" value="<?php echo $star_1; ?>"></td>
</tr>

<tr><td>البطل الثاني</td>
<td><input type="text" name="star_2" value="<?php echo $star_2; ?>"></td>
</tr>

<tr><td>تصنيف الفيلم</td>
<td><input type="text" name="kind" value="<?php echo $kind; ?>"></td>
</tr>

<tr><td>سعر الفيلم</td>
<td><input type="text" name="price" value="<?php echo $price; ?>"></td>
</tr>

<tr>
<td><input type="submit" name="submit" value="update"></button></td>
<td><input type="submit" name="submit" value="delete"></button></td>
</tr>
</form></table>
</body></html>
```

و فيما يلي الخوارزمية التي تعمل وفقها الشيفرة السابقة:

اتصل بملقم **MySQL** و اختر قاعدة بيانات
إذا كان لـ **mov_no** قيمة و لم يتم ضغط زر الإرسال
أعد قيمة كل عمود في الصف المشار إليه بـ **mov_no**
أسند أسماء الأعمدة هذه إلى المتحولات بلغة البرمجة
أما إذا كان لـ **mov_no** قيمة و تم ضغط الزر **update**
قم بتشغيل استعلام **update**
قم بإعادة تحميل الصفحة باستخدام قيمة لـ **mov_no**
أما إذا لم تكن لـ **mov_no** قيمة و تم الضغط على زر **update**
قم بتشغيل استعلام **insert**
قم بإعادة تحميل الصفحة بدون أي قيمة لـ **mov_no**
أما إذا كان لـ **mov_no** قيمة و تم الضغط زر **delete**
قم بتشغيل استعلام **delete**
نهاية شرط إذا
قم بعرض الارتباط إلى نموذج فارغ (بدون إظهار **mov_no**)
قم بعرض سرد لجميع المدخلات في الجدول، التي ترتبط بـ **mov_no**
قم بعرض نموذج. من الضروري أن يتضمن النموذج قيماً لكل عمود من الصف (للتحرير)
إذا كان **mov_no** متوفراً.

الفصل الخامس
الاستعلامات في MySQL

لقد تعلمت إلى الآن كيف تقوم ببناء قاعدة بيانات علائقية خاصة بالويب بشكل جيد.

استهلال
لقواعد البيانات العلائقية وظيفتين أساسيتين هما:
1. تخزين و حفظ البيانات: أي أرشفة البيانات، و كل ما تعلمته في الدروس السابقة هو لتحقيق هذه الأرشفة.
2. سهولة عمليات البحث و الاستعلام: و هذا ما تحققه التعليمة **select** ، إذن فهي تشكل نصف أهمية قواعد البيانات العلائقية، و الاستعلام البسيط أو الركيك يسبب بطئ أثناء التعامل مع قواعد البيانات.
و من باب العلم بالشيء .. يميل بعض مبرمجي قواعد البيانات إلى تقسيم قواعد البيانات العلائقية لثلاث أقسام هي لغة **DDL** و لغة **DML** و لغة الاستعلام **Select** ، و بعضهم الآخر يميل إلى أن يضم **Select** مع لغة الـ **DML** (راجع درس "تسجيل الدخول و الخروج في **MySQL** " لمعرفة الفرق بينهما).

التعليمة **select**
سأفترض في هذا الدرس أنك لم تحرر أو تكتب أي استعلاماً بشكل يدوي مسبقاً، فإذا كنت تعرف كتابة الاستعلامات يدوياً، فإنك تستطيع أن تتجاوز هذا الفصل و إلا فعليك أن تقرأه بتمعن و ترو لأن الاستعلامات ستتطور معك خلال الدروس القادمة إن شاء الله حتى تصل إلى ما يسمى بالربط الذاتي و الربط اليميني و الربط اليساري و الاستعلامات المتداخلة أو المعششة، فكيف ستعرف كتابة هذه الأنواع من الاستعلامات و أنت لا تعرف مفهوم الاستعلام!؟ .. إذن لندخل في عالم الاستعلامات خطوة بخطوة و نتعلم كيفية كتابة الاستعلامات في **MySQL** و ننتقل بها حتى نصل إلى الاستعلامات المعقدة، لنبدأ على بركة الله.
تعرفت سابقاً على أبسط و أعم شكل من أشكال الاستعلامات، و هو:

```
select * from table_name;
```

حيث أن المقصود بـ (*) هو اختيار كل الأعمدة من الجدول المذكور في الاستعلام.
الآن سننتقل إلى أشكال مختلفة للاستعلامات، إن الصيغة القواعدية العامة لتعليمة **select** هي:

```
select col_1,col_2, ... , col_n from table_name where ... ;
```

و معنى هذه الكتابة أنه قم (أي الملقم) بإظهار القيم المخزنة في الأعمدة **col_1 , col_2 , ... , col_n** من الجدول **table_name** بحيث تحقق الشروط كذا و كذا.
لاحظ أنه عندما تريد اختيار مجموعة أعمدة فعليك ذكر أسمائها أما عندما تريد اختيار كل الأعمدة فيمكنك أن تكتب أسماء كل الأعمدة أو تستخدم نجمة (*) بدلاً من ذلك و هو الأفضل.
لنأخذ مثالاً تطبيقياً على ذلك و نقوم بتحليله، و مع كثرة تعاملك مع الاستعلامات ستصبح الاستعلامات بالنسبة لك من أيسر الأمور إن شاء الله.

مثال تطبيقي:
أكتب استعلاماً تحصل فيه على اسم البطل الرئيسي و سعر الفيلم الذي عنوانه **assassins**
المشكلة : تريد اسم البطل الرئيسي لفيلم **assassins** و سعره.
التحليل :
- ما المعلومات التي تريدها؟
اسم البطل الرئيسي + سعر الفيلم

- في أي جدول ستبحث؟
في الجدول الذي يحوي هذه المعلومات و هو الجدول **movies**
- ما هي الشروط الموضوعة؟
شرط واحد و هو أن تلك المعلومات خاصة فقط بفيلم القتلة **assassins**
الحل خوارزمياً :
نقصد بالحل الخوارزمي هو أن تضع خطوات الحل بصيغة كلامية، و بالتالي سيكون الحل خوارزمياً كما يلي:

شغل استعلاماً بحيث يعيد لي
اسم البطل الأساسي ، سعر الفيلم
من جدول أرشيف الأفلام
بحيث أن
اسم الفيلم هو القتلة

لاحظ أن هذه الخوارزمية إذا ترجمناها إلى اللغة الإنكليزية ستكون أسهل بكثير و السبب في ذلك واضح جداً هو أن البرمجة تعتمد على اللغة الإنكليزية، و إليك الترجمة:

Run a query that give me
star_1 , price
from movies
where
mov_name is assassins

الحل برمجياً :
عندما تريد نقل الحل من الشكل الخوارزمي إلى الشكل البرمجي، كل ما عليك هو أن تترجم من اللغة الإنكليزية إلى لغة **MySQL** ، و إليك الترجمة:

```
select
star_1 , price
from movies
where
mov_name = 'assassins' ;
```

و تصبح الصيغة النهائية على الشكل التالي:

```
select star_1 , price from movies where mov_name='assassins' ;
```

استخدام العبارات **in / not in**
قلنا سابقاً أن العبارة **where** تستخدم لوضع الشروط و يترافق معها كلمات محجوزة سنتكلم في هذا الفصل عن خيارات العبارة **where** مع التعليمة **select**
اكتب استعلاماً يعيد لنا جميع الأفلام التي أنواعها كما يلي **violence , horror** :
طبعاً ستكتب الاستعلام التالي:

```
select * from movies where kind='violence' or kind='horror';
```

حسناً في حال أن الاستعلام طلب منك الأنواع التالية **romance , horror , comedy , historical** فهل ستكتب الاستعلام كما في السابق؟ طريقة طويلة طبعاً لذلك فإن **MySQL** قدمت لك العبارتين **in , not in** حيث أن العبارة **in** تعني أن الشرط هو ضمن المجال التالي، أما **not in** فتعني أن الشرط هو في كل شيء عدى المجال التالي.
لذلك سيصبح الاستعلام السابق كما يلي:

```
... where kind in ('violence' , 'horror') ;
```

و قد يطلب منك كتابة استعلام يعيد لك كل الأفلام عدى أفلام العنف، فستكتب الاستعلام التالي:
... where kind not in ('violence');

و طبعاً تستخدم العبارتين **in/not in** مع القيم المحرفية فقط، و في حال القيم الرقمية أو العددية فستستخدم العبارتين **between/not between** .
و يكون ذلك كما في المثال التالي:
اكتب استعلاما يعيد لنا جميع الأفلام التي سعرها بين 200 و 400 ، ستكون صيغة الاستعلام كالتالي:
select * from movies where price between 200 and 400;

اكتب استعلاماً يعيد أسعار الأفلام التي خارج المجال 200 و 400 ، و ستكون صيغة الاستعلام كالتالي:
select * from movies where price not between 200 and 400;

استخدام العبارة **like**
الصيغة القواعدية لهذه الكلمة هي:
... where col_name like 'val_name' ;
الكلمة **like** تستخدم لمطابقة ما بعدها مع القيم المخزنة في اسم العمود المذكور قبلها، مع الأخذ بعين الاعتبار أن القيم المحرفية تستخدم علامتي الاقتباس المفردة و أن القيم العددية يمكن أن تستخدم أو لا تستخدم علامات اقتباس معها.
اكتب استعلاما يعيد لنا جميع الموظفين الذين أسمائهم هو عمر:
select * from emp where name like 'omar';

طبعاً **MySQL** غير حساسة لحالة الأحرف، فلو كتبت **omar** أو **Omar** كله واحد.
حسناً لنفرض أنك لا تعرف كيفية كتابة كلمة **Omar** كاملة ماذا ستفعل في هذه الحالة؟ تؤمن لك **MySQL** محارفاً خاصة للتعبير عن المحارف، تماماً كما في عمليات البحث، و هذه المحارف هي:
1. علامة النسبة المئوية %:
تستخدم هذه العلامة للتعبير عن صفر محرف أو أكثر، و إليك الاستعلامات التالية و معانيها:
select * from emp where name like 'a%';

و معناه أعد لي كل السجلات التي تكون القيم المخزنة في العمود **name** تبدأ بالحرف **a** و مهما كان الأحرف التي بعدها.
select * from emp where name like '%a%';
أي أعد السجلات التي تحوي على حرف **a** في الوسط مهما كانت عدد الأحرف قبلها أو بعدها.
select * from emp where name like '%a' ;
أعد السجلات التي تحوي على الحرف **a** في آخر السلسلة المحرفية.
2. علامة الشرطة السفلية:
و تستخدم هذه العلامة للتعبير عن محرف واحد فقط، فلنفرض أنك ستبحث عن موظف أنت متأكد أن اسمه مؤلف من 4 محارف و أن الحرف الأول هو **n** فيمكنك كتابة الاستعلام التالي:
select * from emp where name loke 'n___';
حيث أن الخط الذي بعد الحرف **n** هو عبارة عن ثلاث شرطات سفلية تمثل ثلاث محارف.
هناك الكثير من السلاسل المحرفية التي تحوي خلالها على الشرطة السفلية مثل البريد الإلكتروني، فللتعبير عن الشرطة السفلية أو أي محرف محجوز في **MySQL** كمحرف، فإنك ستستخدم علامة الهروب **** قبل هذا المحرف.
فلنفرض أنك ستبحث عن كل بريداً إلكترونياً يحوي الشرطة السفلية ضمن الجدول **adrs** ، فإنك ستكتب الاستعلام التالي:
select * from adrs where emp_email like '%_%';

الفرق بين **and** و **or**

1. اكتب استعلاماً يعيد لنا جميع المعلومات عن الفيلم الذي اسمه **assassins** و الفيلم الذي رقمه في أرشيفنا (3) .

عندما تقوم بترجمة هذا التمرين ترجمة حرفية فإنك ستكتب الاستعلام التالي:

```
select * from movies where mov_name='assassins' and mov_no=3;
```

طبعاً هذا الاستعلام صحيح قواعدياً و لكنه سيعطيك نتيجة غير المتوقعة، و بدلاً من أن يعيد لك معلومات عن الفيلمين الذين طلبتهما بالاستعلام، فإنه سيعيد لك العبارة التالية **Empty set** ، و السبب في ذلك هو الكلمة **and** حيث أن عمل هذه الكلمة هو تجميع الشروط المطروحة بعد العبارة **where** لنفس السجل أو السطر، و بالتالي فإن الملقم سيبحث عن فيلماً اسمه **assassins** و بنفس الوقت رقمه (أي الفيلم **assassins**) هو 3 ، و هذا السجل غير موجود في جدولنا لذلك سيعطينا العبارة السابقة.

إذن ما هو الحل؟ الحل هو أن تستبدل الكلمة المفتاحية **and** بالكلمة المفتاحية **or**، و بالتالي سيصبح شكل الاستعلام السابق كما يلي:

```
select * from movies where mov_name='assassins' or mov_no=3;
```

و عندها سيعيد لك هذا الاستعلام المعلومات التي طلبتها عن الفيلمين، حيث أن هذا الاستعلام يقول لملقم البيانات أظهر لنا كل البيانات من الجدول **movies** بحيث اسم الفيلم **assassins** أو رقم الفيلم 3.

2. اكتب استعلاماً يعيد لنا الفيلم الذي اسمه **assasins** أو **others** حيث نوع الأول هو **violence** و الثاني : **horror**

ستكتب الاستعلام بالصيغة التالية:

```
... where mov_name='assassins' and kind='violence' or mov_name='others' and kind='horror';
```

طبعاً هذه الصيغة صحيحة و ستعيد لك مرادك، و لكن من الأفضل أن تجمع عناصر الاستعلام ضمن أقواس أي كما يلي:

```
... where (mov_name='assassins' and kind='violence') or
(mov_name='others' and kind='horror');
```

فعندما تكتب استعلاماً معقداً، فإنك ستحتاج إلى الأقواس حتماً، حيث أن أفضلية التنفيذ هي **and** ثم **or** .

1. الكلمة **and** تستخدم لتجميع الشروط لسجل واحد، بينما تستخدم الكلمة **or** لوضع الشروط على أكثر من سجل.

2. عند كتابة الاستعلام بشروط عديدة و تحوي كل من **and , or** فمن الأفضل وضع الشروط ضمن أقواسها المناسبة، مع الأخذ بعين الاعتبار أن أفضلية التنفيذ في **MySQL** هي للكلمة **and** ثم الكلمة **or** .

3. اكتب استعلاماً يعيد لنا أسماء و أسعار جميع أفلام العنف.

إن صيغة هذا الاستعلام هي كالتالي:

```
select mov_name , price  from movies where kind='violence';
```

يفيد هذا الاستعلام في إعادة جميع السجلات ذات الصفة الواحدة.

استخدام العبارة order by

إن العبارة **order by** تعني (مرتباً أو مفروزاً حسب)، و تستخدم هذه العبارة للفرز إما التصاعدي **asc** أو التنازلي **desc** ، و الصيغة العامة لهذه العبارة هي:

```
select ... from table_name where ... order by col_name_1 asc , col_name_2 desc ;
```

و معناها قم باختيار الأعمدة كذا و كذا من الجدول الفلاني حيث الشروط هي كذا و كذا بحيث تكون النتائج مرتبة تصاعدياً وفق العمود **col_name_1** و مرتبة تنازلياً حسب العمود **col_name_2** ، و القصد في أن الفرز حسب العمود الأول تصاعدي و العمود الثاني تنازلي هو أنه في حال تساوت القيم في العمود الأول سيتم الفرز تنازلياً حسب قيم العمود الثاني، و طبعاً أنت لست مكرهاً على هذا التسلسل في الفرز، فقد تفرز حسب عمود و احد فقط أو حسب أكثر من عمودين أو كل الفرز تصاعدياً أو كله تنازلياً، أي أنه لك مطلق الحرية في استخدام عدد الأعمدة التي تريد و طريقة الفرز، و إليك المثال التالي:

اكتب استعلاماً يعيد لنا اسم الفيلم و رقمه و اسم البطل الأساسي مرتبة تصاعدياً حسب اسم الفيلم و السعر:

```
select mov_no , mov_name , star_1 from movies order by mov_name asc , price asc ;
```

طبعاً تستطيع الاستغناء عن العبارة **asc** (الفرز التصاعدي) لأن الفرز الافتراضي في **MySQL** هو الفرز التصاعدي.

استخدام العبارة limit

تفيد هذه العبارة في إعادة جزء من النتائج و ليس كامل النتائج، و الفائدة من ذلك هو أنه في حال كانت النتائج فرضاً أكثر من 500 نتيجة، فهنا قد يتجاوز تحميل الصفحة التي تحوي هذه النتائج أكثر من 5 دقائق، و كمثال على استخدام العبارة **limit** هو محرك البحث **Google** و المنتديات عندما تشاهد عدة صفحات لعرض النتائج.

أما الصياغة اللغوية للعبارة **limit** فهي كما يلي:

```
select * from movies limit 1;
```

و هنا تطلب من الملقم أن يعرض فقط أول نتيجة، و سيعرض لنا فقط معلومات الفيلم **assassins** ، أما لعرض النتيجة التالية فستكتب الاستعلام التالي:

```
select * from movies limit 1 , 1 ;
```

و هنا فقط سيعرض لنا الملقم معلومات الفيلم **last man standing** ، أما لعرض معلومات الفيلم الثالث **city of angels** فنكتب الاستعلام التالي:

```
select * from movies limit 2 , 1 ;
```

و لعرض معلومات الفيلمين **others , fist of legend** فإنك ستكتب الاستعلام التالي:

```
select * from movies limit 3 , 2;
```

لاحظ أن الرقم الأول و هو هنا 3 يقوم بإعلام الملقم بأن يبدأ بالبحث اعتباراً من السطر (السجل) الرابع و ذلك لأن العبارة **limit** تبدأ العد من الصفر و ليس من الواحد، و الرقم الثاني و هو هنا 2 يقوم بإعلام الملقم بأنني أريد فقط نتيجتين.

لذلك فإذا كتبت الاستعلام التالي فهو صحيح:

```
select * from movies limit 0 , 3;
```

أي قم بإظهار ثلاث نتائج فقط اعتباراً من السطر الأول.

و هو يساوي الاستعلام التالي:

```
select * from movies limit 3;
```

لأنه في حال عدم ذكر السطر الذي يتوجب على **MySQL** أن يبدأ البحث من عنده، سيعتبر **MySQL** افتراضياً أن بداية إظهار النتائج هو من عند الصف رقم 0 أي الصف الأول.

استخدام العبارة **distinct**

لنفرض أنك تريد أن تشاهد جميع أنواع الأفلام التي في أرشيفك لذلك ستكتب الاستعلام التالي:

```
select kind from movies [order by kind desc];
```

طبعاً هذان القوسان يدلان على أن ما بداخلهما هو اختياري، حسناً سيقوم الاستعلام السابق بعرض النتيجة التالية:

```
+---------------+
|kind           |
+---------------+
|violence       |
|violence       |
|violence       |
|romance        |
|horror         |
+---------------+
```

فلو كان أرشيفك زاخراً بالأفلام فستحصل على تكرار في الخرج، و أنت تريد فقط أن يظهر لك الأنواع، لذلك فإن **MySQL** تقدم لك العبارة **distinct** و وظيفتها هي إزالة القيم المكررة من النتيجة، و يتم استخدامها كما يلي:

```
select distinct kind from movies order by kind asc ;
```

و ستظهر النتيجة كما يلي:

```
+---------------+
|kind           |
+---------------+
|horror         |
|romance        |
|violence       |
+---------------+
```

الفصل السادس
الاستعلامات المتقدمة في MySQL

بعد أن تعرفت أخي القارئ خلال الفصل السابق على الاستعلامات و كيفية عملها سننتقل الآن إلى استعلامات أكثر تقدماً، لنبدأ على بركة الله.

استهلال

تستخدم الاستعلامات المتقدمة مفاهيم خاصة بقواعد البيانات، هذه المفاهيم تدعى بالروابط **joins** و لهذه الروابط عدة أنواع و هي:

1. الرابطة المشتركة **equal join**
2. الرابطة الداخلية **inner join**
3. الرابطة الخارجية **outer join**
4. الاستعلامات المتداخلة **sub-selects**
5. الاتحادات **unions**
6. الربط الذاتي **self join**

و سنتكلم عنها خلال هذا الفصل بشكل مستفيض إن شاء الله.

الرابطة المشتركة **equal join**

يُستخدم هذا النوع من الروابط لكتابة استعلامات تطلب نتائج من أكثر من جدول، أي من جدولين على الأقل، معتمدة في ذلك على المفاتيح الغريبة **foreign keys** للجداول.

قلنا سابقاً أن **MySQL** لا تدعم خاصية المفتاح الغريب بشكل قوي مثل قواعد بيانات **Oracle** و غيرها ، لذلك ستنجح معك الاستعلامات التي ستكتبها بـ **MySQL** حتى في حال عدم وجود مفاتيح غريبة في الجداول التي تطبق عليها الاستعلام.

لنأخذ هذا المثال أولاً ثم نقوم بتحليله:
اكتب استعلاماً يعيد لنا ما يلي (اسم الموظف و رقمه و اسم الولاية التي ينتمي إليها) ..
نحن نعلم أن اسم الموظف و رقمه موجودان في الجدول **emp** من قاعدة البيانات **movie_store** و أن الولاية التي ينتمي إليها موجودة في الجدول **adrs** لذلك عندما نريد أن نكتب هذا الاستعلام فإننا بحاجة إلى ربط الجدولين **emp , adrs** مع بعضهما البعض، و يتم ذلك عن طريق الرابطة المشتركة كما يلي:

```
select emp.emp_no , name , state from emp , adrs
where
emp.emp_no = adrs.emp_no ;
```

الشيء الجديد عليك في هذا الاستعلام هو معامل النقطة، ماذا يعني ؟ و متى نستخدمه ؟
إن معامل النقطة يعني أن العمود المذكور بعده (أي **emp_no**) ينتمي إلى الجدول المذكور قبله (أي الجدول **emp**) و بالتالي إذا أردنا أن نقول بأن العمود **mov_name** الذي ينتمي إلى الجدول **movies** نكتب **movies.mov_name** ، أما عن استخدامه فإننا نستخدمه في حال كان اسم العمود المطلوب (و هو في مثالنا

هذا العمود **emp_no**) موجود في أكثر من جدول من الجداول الداخلة في الاستعلام، لنفرض أننا كتبنا الاستعلام السابق دون استخدام معامل النقطة كما يلي:

```
select emp_no , name , state from emp , adrs
```

عندها سيظهر لنا ملقم **MySQL** رسالة خطأ و يخبرنا بأنه يوجد هناك عمود اسمه **emp_no** في كل من الجدولين **emp , adrs** فأي واحد منهما تقصد ؟
لذلك يجب أن نحدد أن هذا العمود نريده من الجدول الفلاني، و يتم ذلك عن طريق معامل النقطة.
و ستكون نتيجة الاستعلام السابق هي كالتالي:

```
+---------------+--------------+---------------+
|emp_no         |name          |state          |
+---------------+--------------+---------------+
|1              |Omar          |Syria          |
|2              |nour          |Egypt          |
|4              |ameen         |Yemen          |
|3              |ahmad         |Syria          |
+---------------+--------------+---------------+
```

كما أنه يطلق على الرابطة المشتركة أيضاً اسم الرابطة المباشرة **straight join**

الرابطة الداخلية **inner join**

الرابطة الداخلية هي عبارة عن شكل آخر للرابطة المشتركة **equal join** أي أنهما يؤديان نفس الوظيفة و يستخدمان لنفس الهدف، أما الفارق الوحيد بينهما فهو الصيغة القواعدية فقط.
فالمثال السابق تم حله عن طريق الرابطة المشتركة، و الآن سنحله عن طريق الرابطة الداخلية، كما يلي:

```
select emp.emp_no , name , state from emp
inner join adrs on emp.emp_no = adrs.emp_no ;
```

قارن بين الحلين، ستجد أن الجدول الذي جاء بعد الكلمة المفتاحية **from** هو نفس الجدول الذي اخترناه لتحديد العمود **emp_no** و هو الجدول **emp** ثم استخدمنا العبارة **inner join** و كتبنا اسم الجدول الثاني ثم كتبنا الشرط المطلوب.
حسناً .. كيف سيصبح الأمر إذا كان الاستعلام من أكثر من جدولين ؟؟
لن يتطلب الأمر منا سوى كتابة ربط داخلي بحسب عدد الجداول التي لدينا في الاستعلام، و لتوضيح هذا الأمر دعنا نأخذ المثال التالي:
قم بإدراج البيانات التالية في الجدول **bills** حسبما تعلمت عن طريق التعليمة **insert** و البيانات هي:
mov_no = 1 و **cli_no = 1** و **paid = 250**
الآن .. قم بكتابة استعلام يعيد لنا ما يلي:
اسم الزبون و اسم الفيلم الذي اشتراه و الثمن المدفوع للفيلم و تاريخ تحرير الفاتورة مع العلم أن الرقم التسلسلي لهذه الفاتورة هو واحد.
تحليل الاستعلام:
اسم الزبون **cli_name** نأخذه من الجدول **clients** .
اسم الفيلم **mov_name** نأخذه من الجدول **movies** .
الثمن المدفوع **paid** نأخذه من الجدول **bills** .
تاريخ تحرير الفاتورة **bil_date** نأخذه من الجدول **bills** .
الرقم التسلسلي للفاتورة **bil_no** نأخذه من الجدول **bills** .
و بالتالي سيكون الشكل النهائي للاستعلام هو:

```
select cli_name , mov_name , paid , bil_date from bills
inner join clients on bills.cli_no = clients.cli_no
inner join movies.mov_no = bills.mov_no and bil_no =1;
```

و سيكون الناتج كما يلي:

```
+--------------+----------------------+------+----------------------+
|cli_name      |mov_name              |paid  |bil_date              |
+--------------+----------------------+------+----------------------+
|mohammed      |last man standing     |250   |040914220644          |
+--------------+----------------------+------+----------------------+
1 row in set (0.33 sec)
```

و هناك صيغة أخرى بالنسبة للشروط في الرابطة الداخلية، إذ تستطيع أن تكتب:

```
select cli_name , mov_name , paid , bil_date from bills
inner join clients on bills.cli_no = clients.cli_no
inner join movies.mov_no = bills.mov_no
where bil_no =1;
```

لاحظ أننا بدلنا العبارة **and bil_no = 1** بالعبارة **where bil_no=1** فكلا الطريقتين صحيحتين.
قد تتساءل كيف لي أن أعرف بأن الجدول الذي سأضعه بعد الكلمة **from** هو الجدول **bills** ؟
في الحقيقة لا يوجد أي فرق سواءً وضعت اسم الجدول **bills** أو الجدول **clients** أو الجدول **movies** لكن مع مراعاة تغير صيغ الشروط كما يلي:

```
select cli_name , mov_name , paid , bil_date from clients
inner join movies on movies.mov_no  = bills.mov_no
inner join bills on bills.cli_no = clients.cli_no
where bil_no = 1 ;
```

و لكن من الأفضل أن نضع بعد **from** الجدول الذي يطابق العبارة **where** ، ففي مثالنا هذا كان الشرط **where bil_no=1** و هو مأخوذ من الجدول **bills** لذلك من الأفضل أن نكتب **from bills**

الرابطة الخارجية **outer join**

أكتب لنا استعلاماً يعيد ما يلي:
اسم الموظف **name** و راتبه **salary** و كل المعلومات المتعلقة به و الموجودة في الجدول **adrs**
طبعاً ستكتب أنت الاستعلام التالي:

```
select name , salaty , adrs.* from emp , adrs
where
emp.emp_no = adrs.emp_no ;
```

ستكون النتيجة هي:

```
+--------------+-------------+--------------+--------------+--------------+----------------------+
|name          |salary       |add_no        |emp_no        |state         |emp_email             |
+--------------+-------------+--------------+--------------+--------------+----------------------+
|Omar          |12000        |1             |1             |Syria         |omar@mail.sy          |
|nour          |8000         |2             |2             |Egypt         |nour@lycos.com        |
|ameen         |5000         |3             |4             |Yemen         |ameen@lycos.com       |
|ahmad         |5000         |4             |3             |Syria         |NULL                  |
+--------------+-------------+--------------+--------------+--------------+----------------------+
```

طبعاً أنت لاحظت عدم وجود السجل الخاص بالموظف أسامة الذي يملك سجلاً في جدول الموظفين **emp** و ليس له سجل في الجدول **adrs**، و بالتالي فإن الرابطة المشتركة لن تفي بالغرض و كذلك الأمر بالنسبة للرابطة الداخلية كونها (كما قلنا سابقاً) أنها عبارة عن شكل آخر للرابطة المشتركة.
قدمت لنا **MySQL** ما يسمى بالرابطة الخارجية و التي تدعى بالرابطة اليسارية **left join** أيضاً، و التي تستطيع أن تظهر لنا السجل الخاص بالموظف أسامة، سنكتب هذا الاستعلام أولاً ثم نعلق عليه.

```
select name , salary , adrs.* from emp
left join adrs on emp._emp_no = adrs.emp_no ;
```

و ستكون النتيجة كما يلي:

```
+---------------+--------------+---------------+---------------+---------------+----------------------+
|name           |salary        |add_no         |emp_no         |state          |emp_email             |
+---------------+--------------+---------------+---------------+---------------+----------------------+
|Omar           |12000         |1              |1              |Syria          |omar@mail.sy          |
|nour           |8000          |2              |2              |Egypt          |nour@lycos.com        |
|ahmad          |5000          |4              |3              |Syria          |NULL                  |
|ameen          |5000          |3              |4              |Yemen          |ameen@lycos.com       |
|osama          |9000          |NULL           |NULL           |NULL           |NULL                  |
+---------------+--------------+---------------+---------------+---------------+----------------------+
```

لاحظ أنه عندما كتبنا الاستعلام استخدمنا العبارة **left join** و ليس العبارة **outer join** ، حيث أن استخدام الأخيرة سوف يظهر لك خطأ قواعدي **syntax error** .
إذن تقوم الرابطة **left join** بمقارنة تحقق الشرط بين الجدول الوارد بعد العبارة **from** (و الذي سنسميه الجدول الرئيسي و هو في مثالنا هذا الجدول **emp**) و الجدول الوارد بعد العبارة **left join** (و الذي سنسميه بالجدول الثانوي و هو في مثالنا هذا الجدول **adrs**) فيعيد السجلات التي تطابق الشرط، و أما السجلات التي لا تطابق الشرط فإنها (أي الرابط اليسارية) تفعل ما يلي:

1. تقوم بطباعة الحقول (أي الأعمدة) المطلوبة من الجدول الرئيسي.
2. يقوم بطباعة الكلمة **NULL** مكان الحقول المحددة في الاستعلام من الجدول الثانوي.
3. يهمل جميع السجلات التي توجد في الجدول الثانوي و الغير موجودة في الجدول الرئيسي.

و بالتالي فإن التبديل بين مواقع الجداول في الاستعلامات ذات الروابط اليسارية تعطينا نتائج مختلفة و ليس مثل الروابط الداخلية التي لا تؤثر عملية التبديل في النتائج.
حسناً .. كيف سنعرف أن هذا الجدول هو جدول رئيسي و أن ذاك الجدول هو جدول ثانوي؟
الأمر بسيط جداً إذ أن الجدول الثانوي هو الجدول الذي يحوي على المفتاح الغريب **foreign key** و الذي يشير إلى المفتاح الأساسي في الجدول الآخر، و الذي هو بدوره يكون الجدول الأساسي في الاستعلامات ذات الرابطة اليسارية.

عندما تكتب استعلاماً من جدولين أو أكثر، فإنه يفضل استخدام الروابط اليسارية **left join** على استخدام الروابط المشتركة **equal join** أو الروابط الداخلية **inner join** ، و ذلك تلافياً لظهور نتائج غير متوقعة أو مغلوطة.

الاستعلامات المتداخلة sub-selections

الاستعلامات المتداخلة أو الفرعية **sub-selects** هي استعلامات تستخدم أحد العبارتين التاليتين **in / not in** وفق الصيغة القواعدية التالية:

```
select ... where ... in (تكتب هنا الاستعلام الفرعي) ;
select ... where ... not in (هنا الاستعلام الفرعي) ;
```

إصدارات **MySQL** الأقدم من **4.1** لا تدعم خاصية الاستعلام الفرعي **sub-selects** ، و لمعرفة نسخة **MySQL** التي لديك اكتب الاستعلام التالي **select version();**
إذا كنت تملك إصداراً أقدم من **4.1** يمكنك التحايل على هذا النوع من الاستعلامات عن طريق استخدام الرابطة اليسارية **left join**، و ستعرف بعد قليل هذه الطريقة إن شاء الله.

لنأخذ مثالاً على الاستعلامات المتداخلة:

مثال (1):

أكتب استعلاماً يعيد لنا جميع المعلومات عن الموظفين السوريين في الشركة:

التحليل:

أنت تعرف أن العمود **state** و الذي يحوي فيه على ولايات الموظفين يوجد في الجدول **adrs** و الاستعلام يطلب منك معلومات من الجدول **emp** ، إذن كيف سأحقق ذلك؟

يتم ذلك عن طريق الـ **sub-select** كما يلي:

```
select * from emp where emp_no in (select emp_no from adrs where state like 'Syria' );
```

و ستكون النتيجة كما يلي:

```
+---------------+--------------+--------------+--------------+--------------+--------------+
|emp_no         |name          |job           |salary        |bonus         |date          |
+---------------+--------------+--------------+--------------+--------------+--------------+
|1              |Omar          |manager       |12000         |3000          |20000101      |
|3              |ahmed         |saller        |5000          |NULL          |20000201      |
+---------------+--------------+--------------+--------------+--------------+--------------+
```

لاحظ أن العمود **emp_no** الذي باللون الأحمر هو مفتاح أساسي في الجدول **emp** ، بينما العمود **emp_no** الذي باللون الأخضر هو مفتاح غريب في الجدول **adrs**.

مثال (2):

أكتب استعلاماً يعيد لنا جميع المعلومات عن الموظفين الغير سوريين في الشركة:

```
select * from emp where emp_no not in (select emp_no from adrs where state like 'syria');
```

ذكرت سابقاً في الملاحظات أنه يمكنك أن تتحايل على الاستعلامات المتداخلة (و ذلك في حال كانت نسخة الـ **MySQL** لديك قديمة أو غير قديمة) عن طريق الرابطة اليسارية(الخارجية)، و يتم ذلك كما يلي:

```
select emp.* , state , emp_email  from emp
left join adrs on emp.emp_no = adrs.emp_no
where state='Syria' ;
```

```
+---------------+--------------+--------------+--------+-------+--------------+-------+--------------+
|emp_no         |name          |job           |salary  |bonus  |date          |state  |emp_email     |
+---------------+--------------+--------------+--------+-------+--------------+-------+--------------+
|1              |Omar          |manager       |12000   |3000   |20000101      |Syria  |omar@mail.sy  |
|3              |ahmad         |saller        |5000    |NULL   |20000201      |Syria  |NULL          |
+---------------+--------------+--------------+--------+-------+--------------+-------+--------------+
```

الاتحادات unions

بفرض أنك تريد أن ترسل بطاقات معايدة إلكترونية إلى كل من زبائنك و مورديك، عندها ستكتب استعلاماً يعيد لك أسماء الزبائن و الموردين و عناوين بريدهم الإلكتروني.
لذلك ستضطر إلى أن تكتب الاستعلامين التاليين:

```
select cli_name , cli_email from clients ;
select sup_name , sup_email from suppliers ;
```

و سترتبك خلال تنقلك بين النتيجتين اللتين حصلت عليهما.
تؤمن **MySQL** عملية دمج للنتائج بحيث تظهر كنتيجة واحدة، و ذلك عن طريق التعليمة **union** التي تستخدم كما يلي:

```
select cli_name , cli_email from clients
union
select sup_name , sup_email from suppliers ;
```

و بذلك ستظهر لك النتائج في قائمة واحدة. لكنك يجب أن تراعي الشروط التالية:
1. يجب أن يكون عدد الأعمدة في الجزأين متساوي.
2. يجب أن تكون الأعمدة من نفس النوع.

للأسف فإن إصدارات **MySQL** الأقدم من الإصدار **4.1** لا تدعم هذه الخاصية، و لكن تستطيع الالتفاف عليها عن طريق إنشاء الجداول المؤقتة **temporary tables** كما ستتعلم في الفقرة القادمة إن شاء الله.

الجداول المؤقتة

تفيد الجداول المؤقتة في إنشاء نسخ أخرى من جداول قواعد البيانات ليتم وضعها (أي الجداول المؤقتة) على الذواكر العشوائية **RAM** و ذلك لجعل الاستعلامات أسرع، و يتم إنشاء الجداول المؤقتة عن طريق الجملة **create temporary table** وفق الصيغة القواعدية التالية:

```
create temporary table table_name
select col_1, ... col_n from existed_table where .... ;
```

و بالعودة إلى مثالنا السابق، قلنا أنه يمكننا الالتفاف على الاتحاد **union** عن طريق استخدام الجداول المؤقتة لذلك ستقوم بإنشاء جدول مؤقت و تخزن فيه البيانات التي تريد:

```
create temporary table temp
select cli_name as names , cli_email as emails from clients
where cli_no=1;
```

قد تستغرب وجود العبارة **as** ، هذه العبارة تدعى بعبارة الاسم المستعار **alias** ، و لتوضيح الأمر أكثر قم بكتابة الاستعلام التالي:

```
select * from temp;
```

عندها ستظهر لك هذه النتيجة

```
+--------------+-----------------------------+
|names         |emails                       |
+-------------- +-----------------------------+
|mohammed   | mohammed@hotmail.com |
+--------------+-----------------------------+
```

لاحظ أنه قد تغيرت أسماء الأعمدة (طبعاً فقط في الجدول المؤقت) إذن العبارة **as** تستخدم لوضع اسم مستعار لاسم العمود بحيث يدل الاسم المستعار على معنى أوضح، و ستستخدم العبارة **as** بكثرة عندما تدرس توابع **MySQL** و العمليات الرياضية عليها لاحقاً إن شاء الله.

تأخذ أعمدة الجداول المؤقتة نفس نوع و خصائص أعمدة الجدول الأصلي، و لتتأكد من ذلك قم بكتابة الأمر التالي:

```
show create table temp \G ;
```

و ستظهر لك بنية الجدول **table** و ستجد أنها مطابقة تماماً لبنية الجدول **clients** .

و الآن سنقوم بإدراج البيانات في الجدول المؤقت **temp**، و يتم إدراج البيانات في الجداول المؤقتة كما يلي:

```
insert into temp(names , emails)
select cli_name , cli_email from clients where cli_no>1;
```

و سنقوم أيضاً بإدراج بيانات الموردين:

```
insert into temp(names , emails)
select sup_name , sup_email from suppliers;
```

و بذلك يصبح لدينا جدولاً مؤقتاً يحوي أسماء و إيميلات جميع الزبائن و الموردين.

تفقد قواعد البيانات الجداول المؤقتة **temporary tables** عند انقطاع التغذية الكهربائية، أو عند الخروج من قاعدة البيانات بشكل تلقائي، و ذلك لأن الهدف من الجداول المؤقتة هو وضع نسخ عن جداول قواعد البيانات التي لديك على الذواكر المؤقتة **RAM** لتسريع عمليات الاستعلام.

تعرفنا قبل قليل على طريقة جديدة في إنشاء الجداول و إدراج البيانات إليها، تفيدنا هذه الطريقة في نسخ بنية الجدول و البيانات التي فيه إلى جدول آخر (ليس من الضروري أن يكون هذا الجدول جدولاً مؤقتاً).

لنفرض أنك تريد أن تنشئ جدولاً اسمه **old_employees** مثلاً، و له نفس بنية الجدول **emp** ، لكنك لا تريد أن تنسخ بيانات الجدول **emp** لذلك ستكتب الاستعلام التالي:

```
create table old_employees
select * from emp
where 1=0;
```

و بالتالي فإن ملقم **MySQL** سيقوم بنسخ بنية الجدول **emp** إلى الجدول **old_employees** و عندما يتحقق من الشرط سيجد أن نتيجته **false** أي غير محقق، عندها لن يقوم بإدراج أي بيانات فيه، و بالتالي ستحصل على جدول جديد اسمه **old_employees** لا يحوي أي بيانات.

حسناً لنفرض أنك تريد أن تدرج فيه بيانات من جدول موجود لديك سابقاً، و لنفرض أنك تريد أن تنسخ البيانات الموجودة في الجدول **adrs** إلى هذا الجدول، فهل تستطيع ذلك؟

طبعاً لا .. لأن بنية الجدول **adrs** تختلف عن بنية الجدول **old_employees** لذلك عندما تريد أن تنسخ بيانات إلى هذا الجدول (الجدول **old_employees**) يجب أن تكون تلك البيانات إما من الجدول الأصلي و هو الجدول **emp** أو من جدول بنيته مطابقة تماماً لبنية الجدول **old_employees**، قم الآن بنسخ بيانات الجدول **emp** إلى الجدول **old_employees**

كما شاهدنا سابقاً أن النسخ يتم باستخدام الأمرين **insert** و **select** كما يلي:

```
insert into old_employees
select * from emp;
```

و بذلك يتم نسخ جميع بيانات الجدول **emp** إلى الجدول **old_employees** ، قد تريد أن تنسخ بعض البيانات و لبعض الموظفين فقط، و بالتالي يمكنك استخدام الاستعلام التالي:

```
insert into old_employees(name , job)
select name ,job from emp where emp_no=2 ;
```

و بذلك نكون قد انتهينا من هذه الطريقة الجديد في نسخ الجداول.

عودة إلى الاتحادات unions

و بالعودة إلى الاتحادات **unions** نذكر بأن الهدف منها هو تجميع البيانات الناتجة من الاستعلامات في تنسيق واحد مناسب، لا تستخف بهذه الوظيفة، و لتوضيح هذا الأمر دعنا ندرس مثالاً تطبيقياً **:**
بفرض أنه لديك الجدولين التاليين

الأول اسمه **cashes** :

cli_no	pay_cash
1	**25**
2	**50**

و الثاني اسمه **credits** :

cli_no	pay_credit
2	**75**
3	**88**

و تريد كتابة استعلاماً يظهر لك النتائج كما يلي:

cli_no	pay_cash	pay_credit	total
1	**25**	**NULL**	**25**
2	**50**	**75**	**125**
3	**NULL**	**88**	**88**

التمرين معقد بعض الشيء لذلك يجب أن تركز جيداً، فلنبدأ على بركة الله.

التحليل :

1. بما أن **cli_no=1** غير موجود في الجدول **credits** و أيضاً **cli_no=3** غير موجود في الجدول **cashes** فأنت إذن بحاجة إلى شيئين هما:
   - أن تكتب استعلامين.
   - أن تستخدم في كلاهما الرابطة اليسارية **left join**
2. بما أنك تريد أن تظهر كلا الاستعلامين في استعلاماً واحداً لذلك عليك أن تستخدم العبارة **union** بينهما، و بالتالي سنحلل كل استعلام على حدا.
3. تذكر الخصائص الثلاث للرابطة اليسارية، و التي كنت قد ذكرتها في درس (الاستعلامات المتقدمة في **MySQL** - الجزء الثالث)

تحليل الاستعلام الأول :

ستكتب استعلاماً يظهر لك رقم الزبون، و بما أن الزبون الثالث غير موجود في الجدول **cashes** ستستخدم هنا رابطة يسارية على الجدول الذي يحوي الرقم الزبون الثالث و هو الجدول **credits** كي تتخلص من الزبون الثالث، و بما أن رقم الزبون **cli_no** موجود في كلا الجدولين فإنك ستحدد من أي جدول تريد أن تختار رقم الزبون.
ثم ستظهر رقم الدفع النقدي **pay_cash** .
ثم ستظهر قيمة الدفع عبر بطاقة الائتمان **pay_credit** و في حال لم يوجد **pay_credit** ستقوم الرابطة اليسارية تلقائياً بوضع **NULL** مكانها.
ثم ستجمع **pay_cash+pay_credit** ليوضع المجموع في عمود اسمه **total** ، لكنك تعرف سلفاً أن مجموع أي قيمة مع **NULL** سيعطي **NULL** لذلك أنت بحاجة على التابع **ifnull()** لا عليك منه الآن، و سنتكلم عن التوابع في **MySQL** بشكل مستفيض في الفصل التاسع.

و الآن لنقوم بترجمة هذا الكلام إلى صيغة برمجية، فسيظهر معك الاستعلام التالي:

```
select cashes.cli_no , pay_cash , pay_credit , (pay_cash+ifnull (pay_credit , 0 ) ) as total
from cashes
left join credits on cashes.cli_no = credits.cli_no ;
```

عندها ستظهر لك النتيجة التالية:

```
+--------------+-------------+--------------+-------+
|cli_no        |pay_cash     |pay_credit    |total  |
+--------------+-------------+--------------+-------+
|1             |25           |NULL          |25     |
|2             |50           |75            |125    |
+--------------+-------------+--------------+-------+
```

تحليل الاستعلام الثاني :

وظيفة هذا الاستعلام هو أن تظهر سجل الزبون الثالث، لذلك ستبدأ استعلامك بالعبارات التالية:

```
select credits.cli_no , pay_cash , pay_credit , pay_credit as total from credits -->
```

لاحظ أننا وضعنا **pay_credit as total** و لم نستخدم تابعاً و ذلك لأن جميع السجلات التي في الجدول الثاني و التي تحتاج إلى جمع قد عالجناها في الاستعلام الأول.

و الآن ستكمل استعلامك و تكتب:

```
--> left join cashes on credits.cli_no = cashes.cli_no -->
```

حسناً إذا توقفت إلى هذا الحد فإن هذا الاستعلام سيظهر لك سجل الزبون الثاني أيضاً و بالتالي ستحصل على تكرار، لذلك ستضع شرطاً و هو أن قيمة الـ **pay_cash** هي **NULL**

```
--> where pay_cash is null;
```

و بتجميع أجزاء هذا الاستعلام نحصل على النتيجة التالية:

```
select credits.cli_no , pay_cash , pay_credit , pay_credit as total from credits
left join cashes on credits.cli_no = cashes._cli_no
where pay_cash is null;
```

و ستحصل على النتيجة التالية:

```
+--------------+----------------------+----------------------+------+
|user_id       |payment_credit        |payment_cash          |total |
+--------------+----------------------+----------------------+------+
|3             |88                    |NULL                  |88    |
+--------------+----------------------+----------------------+------+
```

و الآن بقي عليك الخطوة الأخيرة و هي توحيد الاستعلامين السابقين بواسطة العبارة **union** كما يلي:

```
select cashes.cli_no , pay_cash , pay_credit , (pay_cash+ifnull (pay_credit , 0 ) ) as total
from cashes
left join credits on cashes.cli_no = credits.cli_no ;
union
select credits.cli_no , pay_cash , pay_credit , pay_credit as total from credits
left join cashes on credits.cli_no = cashes._cli_no
where pay_cash is null;
```

الربط الذاتي self join

بفرض أنه لدينا الجدول التالي:

table name: employee

id	name	mgr
1	Mohammed	null
2	omar	1
3	ahmed	1
4	rasheed	2

حيث العمود **id** يمثل رقم الموظف، و هو المفتاح الأساسي لهذا الجدول.
و العمود **name** يمثل اسم الموظف.
و العمود **mgr** يمثل رقم مدير هذا الموظف.
لاحظ وجود القيمة **null** في سجل الموظف محمد عند العمود **mgr** و هذا يعني أنه المدير العام، أما وجود الرقم 1 في سجل الموظف عمر فهذا يعني أن مديره المباشر هو محمد و هكذا.
حسناً .. أكتب استعلاماً يعيد لنا بيانات المدراء في شركتنا؟
هنا ستكتب أنت الاستعلام التالي:

```
select * from employee where id=mgr;
```

ستفاجئ عندما تجد أن هذا الاستعلام لا يعيد لك أي نتيجة، على الرغم من أنه صحيح منطقياً، و السبب في ذلك هو أن **SQL** (و ليس **MySQL**) لا تستطيع أن تقارن العمود الأساسي **حصراً** مع أي عمود آخر في نفس الجدول، فلو كان العمود **mgr** في جدول آخر لكان الأمر بسيطاً، و أصبحت عملية المقارنة هي بين جدولين.
طبعاً بقولنا حصراً فهذا يعني أن **SQL** تستطيع أن تقارن بين أي عمودين في الجدول الواحد، لكن بشرط ألا يكون أحدهما على الأقل عموداً أساسياً في هذا الجدول.
إذن ما الحل لجعل الاستعلام السابق يعمل؟
الحل هو أن نقوم بإنشاء نسختين من الجدول **employee** و نقارن بينهما عن طريق الرابطة المشتركة **equal join** ، أو الرابطة الداخلية **inner join** .
تسمى عملية (إنشاء نسختين عن جدول ما) باسم الربط الذاتي **self join**، و بالتالي ستكتب الاستعلام السابق كما يلي:

```
select t1.* from employee t1 , employee t2
where t1.empno=t2.mgr;
```

لاحظ كيف تم إنشاء نسختين هما **t1 , t2** من الجدول **employee** ، فكل ما عملناه هو أن كتبنا اسم الجدول ثم فراغاً ثم اسم النسخة و طبعاً اسم النسخة اختياري.
بعد كتابتك للاستعلام السابق ستظهر لك النتيجة كما يلي:

```
+------+--------------+------+
|id    |name          |mgr   |
+------+--------------+------+
|1     |Muhammed      |NULL  |
|1     |Muhammed      |NULL  |
|2     |omar          |1     |
+------+--------------+------+
```

و كي تقوم بإلغاء هذا التكرار في النتيجة يمكنك أن تستخدم العبارة **distinct** و التي تكلمنا عنها سابقاً (راجع الدرس الاستعلامات في **MySQL** - الجزء الرابع) ، و بالتالي يصبح الشكل النهائي للاستعلام كما يلي:

```
select distinct t1.* from employee t1 , employee t2
where t1.id=t2.mgr;
```

و ستظهر لك النتيجة السابقة و لكن بعد حذف (طبعاً من النتيجة فقط) السجلات المكررة.
و بذلك نكون قد انتهينا من الروابط في **MySQL** بإذن الله.

الفصل السابع
التوابع الرياضية في MySQL

ما هو التابع
سأفترض أنك أخي القارئ لا تعرف أي شيء عن التوابع، فإذا كنت على دراية بماهية و عمل التابع، فإنك تستطيع أن تنتقل مباشرة إلى الفقرة التالية.

يتلخص التابع في النقاط التالية:

- التابع أو الدالة كما يسمى في بعض المناطق العربية هو تعريب لكلمة **function** (و تعني وظيفة) و بالتالي من اسمها نعرف أن التابع يقوم بعمل شيء واحد فقط، أي أننا نقدم له الأدوات و العتاد اللازم (ندعو هذه الأدوات و العتاد في البرمجة باسم الوسطاء **parameters**) فيقوم هو بعمله داخلياً و يعيد لنا النتيجة جاهزة.
- بما أن اسمه تابع فهذا يعني أنه يتبع الوسطاء **parameters** التي نقدمها له، أي عندما نعطيه وسطاء معينة، سيعيد لنا نتيجة معينة، و عندما نعطيه وسطاء جدد سيعطينا نتيجة أخرى جديدة، و هكذا.
- هذه الوسطاء من الممكن أن تكون متحولات **variables** أو أسماء أعمدة من جداول، أو قيم ثابتة إما رقمية مثل (1 ، 2 ، ... الخ) أو نصية مثل العبارات و الأحرف الأبجدية.
- لا تهمك البنية الداخلية للتوابع في **MySQL** حصراً، فكل ما عليك هو أن تحفظ اسم التابع و ما يقوم به و كم عدد الوسائط التي يأخذها.

فعلى سبيل المثال هناك تابع في **MySQL** اسمه **ucase()** وظيفته هي تحويل الوسيط الذي نقدمه له إلى أحرف كبيرة، فإذا كتبنا :

```
select ucase('By the name of Allah') ;
```

فإنه سيعيد لنا النتيجة التالية:

```
+-------------------------------------+
|ucase('by the name of Allah')        |
+-------------------------------------+
|BY THE NAME OF ALLAH                 |
+-------------------------------------+
```

و إذا أعطيناه وسيطاً آخراً سيعيد لنا نتيجة أخرى و هكذا.
لاحظ ما يلي:

1. أنه بعد اسم التابع يأتي قوسين و بداخلهما توجد الـ **parameters** .
2. التابع **ucase()** قدمنا له وسيطاً واحداً، و هو قيمة ثابتة نصية **string**.
3. القيم النصية توضع داخل علامتي اقتباس، إما مفردتين هكذا **' string '** ، أو مزدوجتين هكذا **" string "**.
4. لاحظ أنه في النتيجة قد ظهرت في الخلية العليا العبارة التالية **ucase('by the name of Allah')** و كي تتخلص منها بحيث تظهر لك كلمة أو جملة مختصرة تستخدم العبارة **as** ، كما يلي:

```
select ucase('By the name of Allah') as Basmala;
```

عندها ستظهر النتيجة كما يلي:

```
+-----------------------------+
|Basmala                      |
+-----------------------------+
|BY THE NAME OF ALLAH         |
+-----------------------------+
```

هناك توابع في **MySQL** تأخذ أكثر من وسيطاً واحداً مثل التابع **ifnull()** الذي يأخذ وسيطين، الأول اسم العمود و الثاني قيمة ثابتة عددية، ففي حال كان التابع يأخذ أكثر من وسيط يتم الفصل بين هذه الوسطاء بفاصلة عادية، مثال:

ifnull(bonus,0)

لا تهتم الآن بهذا التابع فستتعرف إليه عن كثب في الفصل التاسع إن شاء الله، بهذا نكون قد شرحنا مفهوم التابع بشكل كامل إن شاء الله.

التابع **count()**

يأخذ هذا التابع وسيطاً واحداً و هو اسم عمود، مثل **count(name)** .

و يقوم هذا التابع بحساب مجموع الخلايا الموجودة في العمود المذكور.

مثال تطبيقي:

select count(name) from emp;

ستظهر النتيجة التالية:

```
+--------------+
|count(name)|
+--------------+
|7             |
+--------------+
```

عندما نكتب هكذا :

select count(*) from emp;

ستظهر لنا نفس النتيجة السابقة، إذن ما الفرق بين الطريقتين؟

قلنا أن الطريقة الأولى يقوم التابع بحساب مجموع الخلايا التي تنتمي إلى العمود المقدم على شكل وسيط، أما في الطريقة الثانية فإنه (أي التابع) يقوم بحساب مجموع السجلات الموجودة في الجدول، طبعاً هناك فارق كبير، و لتعرف ذلك قم بكتابة الاستعلام التالي:

select count(bonus) as "total bonus" from emp ;

ستفاجئ عندما تظهر لك النتيجة التالية:

```
+--------------+
|total bonus   |
+--------------+
|5             |
+--------------+
```

ألم نقل أن التابع **count()** يقوم بحساب مجموع الخلايا في حال كان الوسيط هو اسم عمود !؟ نعم .. لكنه لا يقوم بحساب الخلايا التي تحوي على **NULL** ، لذلك يجب الحذر عندما تستخدم هذا التابع.

لاحظ الجملة **total bonus** لقد وضعناها بين علامتي اقتباس لأنه في حال كان هناك أكثر من كلمة متصلة بعد العبارة **as** يجب وضعها ضمن علامتي اقتباس.

قد تريد أن تقوم بجمع سجلات جدولين أو أكثر عن طريق التابع **count()** ، فقد تكتب الاستعلام التالي: **select count(*) from emp , adrs;**

عندها سيقوم **MySQL** بضرب عدد سجلات الجدول **emp** بسجلات الجدول **adrs** و يعطيك النتيجة فإذا كان في كل جدول منهما سبع سجلات فإن التابع **count()** لن يعيد لك الرقم 14 بل سيعيد الرقم 49

لا يتوقف استخدام التابع **count()** عند الحدود التي ذكرناها، فعلى سبيل المثال قد تريد أن تعرف عدد الموظفين لديك الذين رواتبهم أكبر أو تساوي 5000، عندها ستحتاج للتابع **count()** :

```
select count(*) as 'salary big than 5000' from emp
where salary>=5000;
```

ستظهر لك النتيجة كما يلي:

```
+------------------------------+
|Salary big than 5000          |
+------------------------------+
|6                             |
+------------------------------+
```

إذن التابع **count()** يستخدم لإعطائنا معلومات إحصائية عن البيانات التي في قواعد بياناتنا.

في كثير من الأحيان تقوم بكتابة استعلام، يكون صحيح منطقياً، و يعيد لك نتيجة، و لكنها تحوي على خطأ غير متوقع، و هنا تكمن الخطورة، فتخيل أنك تقوم بالاشتراك في أحد المنتديات، و عند الإرسال سيظهر لك رابط لتعديل بياناتك قد تدخل إلى هذا الرابط، فتشاهد أنه قد تم وضع العنوان مكان اسم المشترك و رقم الهاتف بدلاً من العنوان ... الخ، إذن لم يظهر أي خطأ برمجي و لكن ظهر خطأ في النتيجة، دعنا الآن نأخذ مثالاً صحيحاً برمجياً، و عند التنفيذ لا يعطينا أي خطأ، بل سيعطينا نتيجة، و لكنها مغلوطة.

قم بكتابة استعلاماً يعيد لنا ما يلي:

1. رقم و اسم و راتب الموظف.
2. مجموع الموظفين ذوي الرواتب المتساوية.

مع الأخذ بعين الاعتبار أن الراتب أكبر أو يساوي 5000 و مفروزين حسب الراتب تنازلياً.

الحل:

```
select emp_no , name , salary , count(*) as 'the same salary'
from emp where salary >= 5000
group by salary desc;
```

عندها ستظهر لك النتيجة التالية:

```
+-------------+--------------+--------------+----------------------+
|emp_no       |name          |salary        |the same salary       |
+-------------+--------------+--------------+----------------------+
|1            |Omar          |12000         |1                     |
|5            |osama         |9000          |1                     |
|2            |nour          |8000          |2                     |
|3            |ahmad         |5000          |2                     |
+-------------+--------------+--------------+----------------------+
```

لا تعر اهتماماً الآن إلى العبارة الجديدة **group by** فسنتكلم عنها في الفقرة القادمة إن شاء الله.

لاحظ أن هذا الاستعلام قد أعاد لك عدد الموظفين ذوي الراتب 8000 هو اثنان، و عدد الموظفين ذوي الراتب 5000 هو اثنان و بالمقابل في خانة الأسماء أعاد لك اسماً واحداً، فأين اسم الموظف الآخر؟ يعتبر هذا الخطأ خطأ فادح.

كيف يتم تلافي مثل هذه الأخطاء ؟

ببساطة يتم تلافي مثل هذه الأخطاء عن طريق تحليل الاستعلام الذي ستكتبه، مع الأخذ بعين الاعتبار أن هناك بعض الاستعلامات المعقدة التي تحتاج لإتمامها بشكل سليم إلى لغة برمجة مثل **PHP** ، لكن هناك استعلامات (مثل استعلامنا هذا) هي بحد ذاتها خاطئة، فكيف تطلب مجموع الموظفين و بنفس الوقت تطلب اسم الموظف و رقمه في نفس الاستعلام ؟!

لذلك و حتى يعيد لنا الاستعلام السابق نتيجة صحيحة يجب أن يكون طلباً صحيحاً، فيمكنك أن تقول اكتب استعلاماً يعيد مجموع الموظفين ذوي الرواتب التي أكبر أو تساوي 5000 مفروزين حسب الراتب، و بالتالي سيكون شكل استعلامك الصحيح كالتالي:

```
select salary , count(*) as 'the same salary' from emp
where salary >= 5000
group by salary desc ;
```

و عندها ستكون النتيجة كما يلي:

```
+--------------+---------------------+
|salary        |the same salary      |
+--------------+---------------------+
|12000         |1                    |
|9000          |1                    |
|8000          |2                    |
|5000          |2                    |
+--------------+---------------------+
```

العبارة group by

تستخدم العبارة **group by** مع التوابع الرياضية في **MySQL** من أجل عمليات الفرز حسب النتائج، و هي تختلف عن العبارة **order by** حيث أن الأخيرة تستخدم لفرز السجلات الصادرة عن الاستعلام، بينما العبارة **group by** تستخدم لفرز النتائج الصادرة عن التوابع الرياضية في الاستعلام،

دعنا نأخذ المثال التالي:

قم بكتابة استعلام يعيد اسم الولاية و عدد الولايات.

```
select state , count(state) as total from adrs
group by state;
```

ستظهر النتيجة التالية:

```
+--------------+-------+
|state         |total  |
+--------------+-------+
|Egypt         |1      |
|Syria         |2      |
|Yemen         |1      |
+--------------+-------+
```

مع الأخذ بعين الاعتبار أن الفرز الافتراضي يكون تصاعدياً **asc**، و لجعل الفرز تنازلياً عليك أن تضيف العبارة **desc** في آخر الاستعلام.

طبعاً لا يمكن الفرز حسب العبارة **count(state)** أو الاسم المستعار لها، و في كلا الحالتين سيظهر خطأ كما هو مبين في المثالين التاليين:

الخطأ 1:

```
select state , count(state) as total from adrs
group by count(state);
```

ستظهر رسالة الخطأ التالية:

```
ERROR 1111: Invalid use of group function
```

الخطأ 2:

```
select state , count(state) as total from adrs
group by total;
```

ستظهر رسالة الخطأ التالية:

```
ERROR 1056: Can't group on 'total'
```

يمكن الفرز حسب عمود آخر غير الأعمدة الواردة في الاستعلام و لكنك ستحصل على نتيجة مختلفة عن التي تصبو إليها، و إليك هذا المثال:

```
select state , count(state) as total from adrs group by emp_no;
```

عندها ستظهر النتيجة التالية:

```
+--------------+-------+
|state         |total  |
+--------------+-------+
|Syria         |1      |
|Egypt         |1      |
|Syria         |1      |
|Yemen         |1      |
+--------------+-------+
```

طبعاً ظهرت نتيجة و لكنها ليست كالتي تتوقعها.

التابع **sum()**

يستخدم هذا التابع لإنجاز عمليات الجمع، و يأخذ التابع **sum()** وسيطاً واحداً هو اسم عمود، و يكون استخدامه كما يلي:

```
select sum(bonus) as 'total bonus' from emp;
```

و ستظهر النتيجة كالتالي:

```
+--------------+
|total bonus   |
+--------------+
|6250          |
+--------------+
```

لاحظ أن التابع لم يظهر لنا **NULL** على الرغم من وجود بعض الخلايا في العمود **bonus** تحوي **NULL**، و كنا قد قلنا أن مجموع أي قيمة مع **NULL** هو **NULL** ، هذا الكلام صحيح و لكن في حال أنك تستخدم العمليات الحسابية و ليس التوابع، و إلا فما الفائدة من التوابع ما لم تختصر علينا بعض الخطوات و المشاكل.

قبل أن نستمر تذكر هذه النقاط الهامة:

1. لا يقف استخدام التوابع عند هذه الحدود البسيطة.
2. لفرز النتائج تستخدم مع التوابع العبارة **group by** و ليس العبارة **order by**.
3. معظم التوابع الرياضية في **MySQL** تستخدم من أجل المعلومات الإحصائية.

الآن .. سنأخذ هذا التمرين الجيد.

أكتب استعلاماً يعيد لنا عدد الفواتير المسجلة باسم كل زبون و المجموع المالي لهذه الفواتير.

تكمن هنا أهمية التوابع الجاهزة في **MySQL** فلو أردت أن تحقق مثل هذا الاستعلام بلغة برمجة، سيحتاج منك الأمر إلى ما يزيد عن 30 سطر برمجي، و العديد من حلقات **for** و عبارات **if/else** .

إليك الحل عن طريق التوابع الرياضية الجاهزة في **MySQL :**

```
select cli_no , count(bil_no) , sum(paid) from bills group by cli_no;
```

و تسكون النتيجة كما يلي:

```
+--------------+----------------------+--------------+
|cli_no        |count(bil_no)         |sum(paid)     |
+--------------+----------------------+--------------+
|1             |4                     |1350          |
|2             |2                     |800           |
|3             |1                     |250           |
+--------------+----------------------+--------------+
```

و تبين النتيجة أن الزبون الأول قد اشترى بمقدار أربع فواتير و مجموع ما اشتراه هو 1350 ، و كذلك بالنسبة لبقية الزبائن.

حسناً دعنا نطور هذا الاستعلام و نضيف له اسم الزبون و بريده الإلكتروني.

هنا ستحتاج إلى ربط الجدولين **bills , clients**

الحل باستخدام الرابطة المشتركة:

```
select clients.cli_no , cli_name , cli_email , count(bil_no) , sum(paid) from clients , bills
where clients.cli_no = bills.cli_no group by clients.cli_no ;
```

الحل باستخدام الرابطة الداخلية:

```
select clients.cli_no , cli_name , cli_email , count(bil_no) , sum(paid) from clients
inner join bills on clients.cli_no = bills.cli_no
group by clients.cli_no ;
```

الحل باستخدام الرابطة الخارجية:

```
select clients.cli_no , cli_name , cli_email , count(bil_no) , sum(paid) from clients
left join bills on clients.cli_no = bills.cli_no group by clients.cli_no ;
```

و في كل الحالات السابقة ستظهر لنا النتيجة التالية:

```
+--------------+--------------+------------------------------+----------------------+--------------+
|cli_no        |cli_name      |cli_email                     |count(bil_no)         |sum(paid)     |
+--------------+--------------+------------------------------+----------------------+--------------+
|1             |mohammed      |mohammed@hotmail.com          |4                     |1350          |
|2             |anas          |anas@hotmail.com              |2                     |800           |
|3             |khalid        |khalid@yahoo.com              |1                     |250           |
+--------------+--------------+------------------------------+----------------------+--------------+
```

تطوير قدراتك الشخصية في التعامل مع قواعد البيانات

بما انك وصلت إلى هذا الدرس، فهذا معناه أنك قادر بإذن الله على أن تبني قاعدة بيانات، و قادر على أن تضيف إليها البيانات و تعدل البيانات التي بداخلها، و قادر على استخدام التوابع في مكانها المناسب، و قادر أيضاً إن شاء الله (عندما ترى مخطوطة برمجية مكتوبة بلغة الـ **PHP** أو غيرها من لغات البرمجة تحوي على استعلامات و توابع جديدة عليك مكتوبة بـ **MySQL**) قادر أن تعرف ما هو المطلوب من هذه الاستعلامات و التوابع الجديدة، فإذا وجدت أنك غير قادر على ذلك و لا تعرف ما هو السبب، فلا تقلق، فإن الحل يكمن في أحد النقاط التالية إن شاء الله:

- إذا كنت غير قادر على كتابة الاستعلامات بشكل جيد مع قاعدة بياناتك، و يكثر معك أن تكتب استعلاماً فيعطيك نتائج غير متوقعة، فهذا يعني أنك غير ملم ببنية قاعدة البيانات التي تعمل عليها، و الحل أن تراجع الفصلين الثالث و الرابع، قم بمراجعتهما بشكل جيد و كأنك تمر عليهما لأول مرة، ففيهما الشرح الكافي و الوافي عن كيفية بناء قواعد البيانات و فهم بنيتها، لذلك لم أتطرق فيما سبق إلى (وكيل واجهة التطبيقات الرسومية **PHPMYADMIN**) و السبب في ذلك هو أن **MYADMIN** قد يساعدك في بناء قواعد بيانات بسرعة لكن دون فهم لبنيتها، و على ذلك لن تستطيع استخدام الاستعلامات و التوابع بشكل سليم بتاتاً.

- إذا كنت غير قادر على بناء قاعدة بيانات متينة و مترابطة، فالسبب في ذلك هو عدم قدرتك على تحليل و تصميم قواعد البيانات، و الحل هو الرجوع إلى الفصل الأول الذي يتكلم عن مفاهيم في قواعد البيانات.

قاعدة البيانات المتينة

هي تلك القاعدة التي لا تحوي على جداول مكررة، أو أعمدة مكررة، فمثلاً قد يكون لك زبائن من القارات الخمس، فلا داعي لإنشاء خمس جداول تمثل القارات الخمس، بل يكفي أن توجد جدولاً واحداً للزبائن و تضع فيه عمود مثل اسم الدولة التي ينتمي لها الزبون، قد تقول بأنه في كلا الحالتين سيكون الحجم واحد، نعم هذا صحيح، و لكن عندما تريد أن تكتب الاستعلامات ستواجه صعوبات كبيرة و ربما لن تستطيع أن تكتب بعض الاستعلامات المعقدة.

قاعدة البيانات المترابطة

هي تلك القاعدة التي يكون ربط الجداول فيها قائم على أساس سليم و منطقي، فمثلاً لا يوجد أي داع لأن تربط جدول الأفلام بجدول الموردين، على الرغم من أن الموردين هم من يزودوك بالأفلام، إلا أن المورد لا يهمه ما اسم الفيلم الذي يرسله لك و لا يهمه إذا بعت هذا الفيلم أم لا، بل يهمه مقدار النقود التي قبضها و متى تم ذلك و ما له و ما عليه من نقود، و أنت أيضاً يهمك فقط متى استلمت طلبية الأفلام من المورد و كم أخذ من النقود و هل سددت له المبلغ كاملاً أم جزءاً منه أو كامل المبلغ على حساب مدين، لذلك يكفي أن تضيف أربع أعمدة في جدول الموردين الأول هو (ما للمورد) و الثاني هو (ما عليه) و الثالث (تاريخ تقديم الطلبية) و الرابع (تاريخ استلام الطلبية)، و من خلال هذه الأعمدة تستطيع أن تعرف أن هذا المورد سريع في إنجاز الطلبيات التي طلبتها منه، و أن ذاك المورد لا يقبل إلا التعامل بالدفع النقدي، و بذلك تستطيع أن تعرف أنه من المجدي أن تتعامل مع المورد الفلاني، و أنه من غير المجدي أن تتعامل مع المورد الآخر، و في الحقيقة هذا أحد الأهداف الأساسية و الجوهرية لاستخدام قواعد البيانات، فلا تقتصر وظيفة قواعد البيانات على نقل البيانات من الشكل الورقي إلى الشكل الإلكتروني، بل و من أجل سهولة و سرعة إيجاد البيانات و وضع التحليلات و الإحصاءات و الدراسات بشكل دقيق و سريع و بالتالي تحسين منحى عمل المؤسسة بسرعة وسهولة.

التابع **avg()**

يستخدم هذا التابع لحساب المتوسط الحسابي **average** لمجموعة قيم.
أكتب استعلاماً يعيد لنا السعر الوسطي للأفلام التي لدينا، مفروزة حسب نوع الفيلم:

```
select kind , avg(price) as 'average price' from movies group by kind;
+--------------+----------------------+
|kind          |average price         |
+--------------+----------------------+
|horror        |300.0000              |
|romance       |300.0000              |
|violence      |332.5000              |
+--------------+----------------------+
```

التابعان **max() , min()**

يعيد التابع **min()** القيمة الدنيا **minimum** لمجموعة قيم، و كما يعيد التابع **max()** القيمة العليا **maximum** ، فمثلاً للحصول على أعلى سعر و أدنى سعر للأفلام نكتب الاستعلام التالي:

```
select kind , min(price) , max(price) from movies
group by kind;
+--------------+--------------+--------------+
|kind          |max(price)    |min(price)    |
+--------------+--------------+--------------+
|horror        |300           |300           |
|romance       |300           |300           |
|violence      |430           |250           |
+--------------+--------------+--------------+
```

الإسناد **having**

عندما تستخدم العبارة **having** ستجد أنها تشابه في عملها العبارة **where** ، لكنها (أي العبارة **having**) تستخدم لوضع شروطاً على الصفوف التي **تنتج عن استخدام** العبارة **group by** ، أما العبارة **where** فإنها تستخدم لوضع شروطاً على الصفوف التي **ستستخدمها** العبارة **group by** .

لنأخذ هذين المثالين للتمييز بين كلا العبارتين.

مثال **1 :**

اكتب استعلاماً يعيد لنا السعر الوسطي للأفلام التي يزيد سعرها عن 250 .

بما أن نص التمرين يطلب منك تجاهل الأفلام التي سعرها اقل من 250 فهذا يعني أنك ستتجاهل **السجلات** التي فيها خلية السعر أقل من 250، لذلك ستستخدم العبارة **where** لأن تعاملك هو مع السجلات.

```
select kind , avg(price) from movies
where price>250
group by kind ;
+--------------+--------------+
|kind          |avg(price)    |
+--------------+--------------+
|horror        |300.0000      |
|romance       |300.0000      |
|violence      |415.0000      |
+--------------+--------------+
```

مثال **2 :**

اكتب استعلاماً يعيد لنا أنواع الأفلام التي يكون متوسط الحسابي لسعرها أكبر من 300

إذن هذا الاستعلام يطلب منك أن تتجاهل **المتوسطات الحسابية** الأقل من 300 لذلك ستستخدم العبارة **having** لأن تعاملك هو مع النتائج وليس مع السجلات.

```
select kind , avg(price)  from movies
group by kind
having avg(price)>300 ;
+--------------+--------------+
|kind          |avg(price)    |
+--------------+--------------+
|violence      |332.5000      |
+--------------+--------------+
```

توابع التقريب

تستخدم توابع التقريب **Rounding Functions** مع النتائج التي تحوي على الفاصلة العائمة **floating Comma** و تسمى هذه التوابع أيضاً باسم توابع التدوير و طبعاً هذه الكلمة (التدوير) هي ترجمة حرفية غير محبذة لكلمة **round** و المقصود بها هو التقريب.
لنكتب الاستعلام التالي:

```
select avg(price) as average from movies;
+---------------+
|average        |
+---------------+
|321.6667       |
+---------------+
```

لاحظ عدد الأرقام بعد الفاصلة، و هناك حالات يصل فيها عدد الأرقام بعد الفاصلة إلى 24 رقم، و للتخلص من هذه الأرقام ستستخدم أحد توابع التقريب

التابع **round()** :

يأخذ هذا التابع وسيطين الأول هو القيمة التي نريد تقريبها و الثاني هو وسيط اختياري وظيفته هي تحديد عدد المنازل العشرية التي نريدها أن تظهر في نتيجة التقريب.
أكتب نفس الاستعلام السابق بحيث تقرب الناتج إلى رقمين بعد الفاصلة العشرية.

```
select round(avg(price),2) as average from movies;
+----------+
| average |
+----------+
| 321.67  |
+----------+
```

و كي تتخلص من الأرقام التي بعد الفاصلة العشرية، يمكنك أن تضع الرقم صفر في الوسيط الثاني للتابع **round()** ، كما يلي:

```
select round(avg(price),0) as average from movies;
```

كما و يمكنك ألا تضع وسيطاً ثانياً كما يلي:

```
select round(avg(price)) as average from movies;
```

يقوم التابع **round()** أولاً بمقارنة الرقم الذي في أقصى اليمين، فإذا وجده أكبر من 5 يقوم بحذفه و زيادة واحد إلى الرقم الذي قبله، أما إذا وجده أصغر من 5 يقوم فقط بحذفه، و هكذا حتى يصبح عدد الخانات التي بعد الفاصلة العشرية مساوياً للوسيط الثاني للتابع **round()** ، أما الرقم 5 فإنه يقوم بحذفه مباشرة دون زيادة أو إنقاص الرقم الذي قبله، و في حال تكرر الرقم خمسة مرتين متتاليتين فإنه يقوم بحذفهما و إضافة واحد إلى الرقم إلى قبلهما.

في الجدول التالي مجموعة استعلامات تستخدم التابع **round()** ، لاحظ خرجها في العمود **result**

the query	the result
select round(3.46111 , 2) ;	**3.46**
select round(3.4177 , 2) ;	**3.42**
select round(3.41177 , 2) ;	**3.41**
select round(3.415 , 2) ;	**3.41**
select round(3.4155 , 2) ;	**3.42** لاحظ زيادة الرقم 1 عند تكرر الرقم 5 مرتين متتاليتين
select round(3.4155,3) ;	**3.415**

التابع **floor()** :
يعيد هذا التابع أقرب عدد صحيح أقل من القيمة المعطاة، مثال:

```
select floor(2.9) ;    //2
```

التابع **ceiling()** :
يعيد أقرب عدد صحيح أكبر من القيمة المعطاة، مثال:

```
select ceiling(2.1) ;    //3
```

التابع **truncate()** :
يقوم هذا التابع بإزالة المنازل العشرية دون تقريب، و يأخذ هذا التابع وسيطين الأول هو القيمة التي سيقربها، و الوسيط الثاني هو عدد المنازل العشرية التي نريدها أن تظهر في النتيجة، مثال:

```
select truncate(3.4567354 , 3) ;      //3.456
```

العمليات الحسابية

تستخدم العمليات الحسابية كما يتم استخدامها في أي لغة برمجة وللعمليات الحسابية نفس الأفضلية الموجودة في لغات البرمجة الأخرى. لنأخذ أمثلة توضح كيفية استخدام العمليات الحسابية:

```
select 4*10+12;   //52

select 4*(10+12);    //88

select price+bonus as total from emp order by total desc;
```

لاحظ أنه يمكن أن تجمع عمودين، و طبعاً يقوم **MySQL** بجمع كل خليتين متقابلتين على حدا.

التوابع المثلثية

لن تحتاج إلى التوابع المثلثية في أغلب الأحيان عندما تبرمج قواعد بيانات موقع إنترنت، و لكنني سأذكرها في هذا الدرس من باب العلم بالشيء.

The Function	It's meaning
cos()	تجيب الزاوية
sin()	جيب الزاوية
tan()	ظل الزاوية

الفصل الثامن
توابع التاريخ و الوقت في MySQL

استهلال

تعد توابع التاريخ و الوقت من أهم التوابع في قواعد البيانات سواءً في الإنترنت أو في أي مجال من مجالات الحواسب، و تبدي **MySQL** الكثير من المرونة عند التعامل مع التاريخ و الوقت، حيث يمكنك أن تدرج التاريخ إما كسلسلة محرفية أو كقيمة رقمية صحيحة دون أن يعيد لك ملقم **MySQL** أي خطأ.

أنظر إلى إدراجات التاريخ التالية:

```
insert .... (20040819);
insert .... ('2004-08-19');
insert .... ('20040819');
insert .... ('04-04-19');
```

كل هذه الإدراجات صحيحة، و بعدها يقوم **MySQL** بترتيب التاريخ في العمود الخاص بالتاريخ حسب التنسيق الذي حددته أنت له عندما قمت بإنشاء الجدول.

لا تُعر أي اهتمام لتنسيق التاريخ و الوقت في جداولك، و كل الذي يهمك هو تنسيق التاريخ و الوقت عند كتابة الاستعلامات، و هذا ما ستتعلمه إن شاء الله في هذا الدرس.

معظم قواعد بيانات **MySQL** تستخدم التنسيق **timestamp** لجداولها، لأن هذا التنسيق يقوم بإدراج التاريخ و الوقت بشكل تلقائي دون الحاجة إلى إدراج التاريخ يدوياً.

أنظر إلى الاستعلام التالي و نتيجته

```
select name , date from emp where emp_no=1;
```

```
+----------+------------+
| name     | date       |
+----------+------------+
| Omar     | 20000101 |
+----------+------------+
```

لاحظ أن قراءة التاريخ صعبة بعض الشيء، و خاصة عندما يظهر هذا التنسيق لشخص غير مختص، لذلك يوجد في **MySQL** التابع **date_format()** الذي يظهر لك التاريخ و الوقت في الاستعلامات بالتنسيق الذي تريده.

التابع **date_format()**
يأخذ هذا التابع وسيطين الأول هو اسم عمود التاريخ الذي تريد تنسيقه، و الثاني هو شكل التنسيق الذي تريده أن يظهر، و يتم عن طريق استخدام رموز خاصة.

فيما يلي جدول الرموز الخاصة بالتابع **date_format()**

الرمز	المعنى
%a	اختصار لسم اليوم **Sun , Mon ... etc.**
%b	اختصار لسم الشهر **Jan , Feb ... etc.**
%c	الشهر، خانة بين 1 و 12
%D	ترتيب اليوم في الشهر **1st** أو **2ed** أو **3rd** ... الخ
%d	ترتيب اليوم في الشهر، خانتين بين 00 و 31
%e	ترتيب اليوم في الشهر، خانة بين 0 و 31
%f	أجزاء الثانية (000000 .. 999999)
%H	الساعة، خانتين بين 00 و 23
%h	الساعة، خانتين بين 01 و 12
%I	الساعة، خانتين بين 01 و 12
%i	الدقائق، خانتين بين 00 و 59
%j	ترتيب اليوم في السنة بين 001 و 366
%k	الساعة، خانة بين 0 و 23
%l	الساعة، خانة بين 1 و 12
%M	اسم الشهر **January, February ... etc.**
%m	الشهر، خانتين بين 00 و 12
%p	**AM** أو **PM**
%r	الوقت 12 ساعة بالتنسيق **hh:mm:ss AM or PM**
%S	الثواني، خانتين 00 و 59
%s	الثواني، خانتين 00 و 59
%T	الوقت 24 ساعة بالتنسيق **hh:mm:ss** بدون **AM** أو **PM**
%U	السنة بالأسابيع، الأحد اليوم الأول، خانة من 00 و حتى 53
%u	السنة بالأسابيع، الاثنين اليوم الأول، خانة من 00 و حتى 53
%V	السنة بالأسابيع، الأحد اليوم الأول، خانتين من 01 إلى 53، يستخدم مع **%X**
%v	السنة بالأسابيع، الاثنين اليوم الأول، خانتين من 01 إلى 53، يستخدم مع **%x**
%W	اسم اليوم **Sunday, Monday ... etc.**
%w	ترتيب اليوم في الأسبوع بحيث اليوم الأول هو الأحد
%X	السنة، عدد صحيح أربع خانات، يستخدم مع **%V**، الأحد هو اليوم الأول
%x	السنة، عدد صحيح أربع خانات، يستخدم مع الرمز **%v**، الاثنين هو اليوم الأول
%Y	السنة، عدد صحيح ذو أربع خانات
%y	السنة، عدد صحيح خانتين
%%	لإظهار الرمز **%** كما هو في الخرج

طبعاً إن حفظ هذا الجدول كاملاً هو أمر مرهق، لذلك أنصحك بألا تحفظ سوى الرموز الأربعة التالية:
%W , %M , %d , %Y

فهي تعطيك تنسيق شامل عن التاريخ.

إليك المثال التالي الذي يوضح كيفية استخدام هذه الرموز:

```
select bil_no , date_format(bil_date , '%W: %M %d, %Y' ) as 'date' from bills;
```

لاحظ أن الرمزين **:** و **,** هما من أجل تحديد نوع الفاصل بين عناصر التاريخ لذلك تستطيع أنت أن تضع أي فاصل تريد.

ستظهر النتيجة كما يلي:

```
+-------+------------------------------+
|bil_no | date                         |
+-------+------------------------------+
| 1     | Thursday: August 19, 2004 |
+-------+------------------------------+
```

التابع now()

التابع **now()** لا يأخذ أي وسيط ، و هو يقوم بإظهار التاريخ و الوقت الحاليين في جهازك أو المخدم الخاص بك.

أكتب الاستعلام التالي و شاهد النتيجة:

```
select now();
+------------------------------+
|now()                         |
+------------------------------+
|2004-09-23  13:24:46       |
+------------------------------+
```

سيظهر لك التاريخ و الوقت الحاليين، لاحظ أن التنسيق الافتراضي للتابع **now()** هو:

```
 %Y-%m-%d %T
```

التابع curdate()

يعمل هذا التابع نفس عمل التابع **now()** و لكنه يظهر فقط التاريخ الحالي بدون الوقت كما يلي:

```
select curdate();
+---------------+
|curdate()      |
+---------------+
|2004-09-23  |
+---------------+
```

طبعاً هذا التابع هو اختصار للجملة **current date** أي التاريخ الحالي.

التابع curtime()

يعيد الوقت الحالي

```
select curtime();
+---------------+
|curtime()      |
+---------------+
|13:24:46       |
+---------------+
```

حساب مجالات التاريخ

إن مجالات التاريخ من أهم الأمور في الإحصاءات و قواعد البيانات، فقد ترغب مثلاً بإنشاء نموذج يتم من خلاله حساب مبيعات شركتك خلال الشهر الفائت أو خلال الأسبوع الفائت ... الخ، و قد ترغب أيضاً بأن تحصل على تقرير يتضمن معلومات تفصيلية عن فعاليات موقعك في الـ 24 ساعة الماضية.

يمكنك في **MySQL** حساب التاريخ باستخدام إشارات الجمع و الطرح مع الإشارة إلى الفترة الزمنية (سنة أو شهر أو ساعة ...الخ) التي تريد أن تستخدمها في الحساب.

يعيد الاستعلام التالي التاريخ و الوقت قبل 24 ساعة من تشغيل هذا الاستعلام.

```
select now()-interval 24 hour as 'last day' ;
+-----------------------------+
|last day                     |
+-----------------------------+
|2004-09-22 13:47:56          |
+-----------------------------+
```

لاحظ العبارة **interval** ، هذه العبارة هامة جداً في حساب مجالات التاريخ، فهي تساعد في أن يظهر لنا الوقت و التاريخ بشكل منسق، و أيضاً تجعل عملية الطرح لا تتجاهل الكلمة التي تأتي بعد الرقم المراد طرحه من الوقت.
و لتوضيح ما ذكرنا جرب اكتب الاستعلامين التاليين:

الاستعلام الأول: بدون **interval**

```
select now()-24 hour as 'last day' ;
ERROR 1064: You have an error in your SQL syntax near 'as 'last day'' at line 1
```

لاحظ أن الاستعلام لم يقبل العبارة **as** و اعتبرها خطأ قواعدي.

الاستعلام الثاني: بدون **interval** و بدون **as**

```
select now()-24 hour ;
+----------------------+
|hour                  |
+----------------------+
|20040923141883        |
+----------------------+
```

لاحظ أنه لم يتم طرح 24 ساعة بل تم طرح الرقم 24، إذن العبارة **Interval** ضرورية جداً في حساب مجالات التاريخ.
فيما يلي جدولاً يتضمن قائمة بالمجالات التي يمكنك استخدامها في العمليات الخاصة بالتاريخ في **MySQL**

المجال	المعنى	مثال
second	الثواني	select now()-interval 3600 second ;
minute	الدقائق	select now()-interval 120 minute ;
hour	الساعات	select now()-interval 24 hour ;
day	الأيام	select now()-interval 21 day ;
month	الأشهر	select now()-interval 6 month ;
year	السنوات	select now()-interval 1 year ;
minute_second	الدقائق و الثواني	select now()-interval '30:20' minute_second ;
hour_minute	الساعات و الدقائق	select now()-interval '23:40' hour_minute ;
day_hour	اليوم و الساعات	select now()-interval '3:2' day_hour ;
year_month	السنوات و الأشهر	select now()-interval '4:4' year_month ;
hour_second	ساعات، دقائق، ثواني	select now()-interval '3:23:50' hour_second ;
day_minute	أيام، ساعات، دقائق	select now()-interval '3:23:21' day_minute ;
day_second	أيام، ساعات، دقائق، ثواني	select now()-interval '4:16:23:2' day_second ;

فإذا أردت أن تتعرف على المبيعات التي تمت خلال فترة من الفترات الماضية من الجدول **bills**، فيمكنك ذلك باستخدام توابع التاريخ الوقت كما هو مبين في الأمثلة التالية:

أحسب عدد الفواتير و المبيعات خلال الثلاث أيام الماضية:

```
select count(bil_no) , sum(paid) from bills
where bil_date > (now() - interval 3 day) ;
+----------------------+---------------+
|count(bil_no)         |sum(paid)      |
+----------------------+---------------+
|6                     |2300           |
+----------------------+---------------+
```

حساب عدد الفواتير و مجموع المبيعات خلال الشهر الماضي مفروزة حسب رقم الزبائن:

```
select cli_no , count(bil_no) , sum(paid) from bills
where bil_date>(now() - interval 1 month)
group by cli_no ;
+---------------+---------------------+---------------+
|cli_no         |count(bil_no)        |sum(paid)      |
+---------------+---------------------+---------------+
|1              |5                    |1800           |
|2              |2                    |800            |
|3              |1                    |250            |
+---------------+---------------------+---------------+
```

اكتب استعلاماً يعيد لنا اسم و رقم و البريد الإلكتروني للزبون و مجموع فواتيره و مجموع مشترياته، و ذلك لكل الزبائن خلال الشهر الماضي.

```
select clients.cli_no , cli_name , cli_email , count(bil_no) , sum(paid) from clients
left join bills on clients.cli_no=bills.cli_no
and bil_date>(now()-interval 1 month)
group by clients.cli_no ;
+---------------+--------------+-----------------------------+----------------------+---------------+
|cli_no         |cli_name      |cli_email                    |count(bil_no)         |sum(paid       |
+---------------+--------------+-----------------------------+----------------------+---------------+
|1              |mohammed      |mohammed@hotmail.com         |5                     |1800           |
|2              |anas          |anas@hotmail.com             |2                     |800            |
|3              |khalid        |khalid@yahoo.com             |1                     |250            |
+---------------+--------------+-----------------------------+----------------------+---------------+
```

كما و يمكنك مثلاً أن تستخدم مجالات التاريخ و الوقت لمعرفة التواريخ القادمة، فالاستعلام التالي يوجد التاريخ الذي يسبق التاريخ الحالي بيومين و خمس ساعات و عشر دقائق و ثلاثون ثانية:

```
select now() + interval '2:5:10:30' day_second ;
```

توابع تنسيق التاريخ

هي توابع تأخذ وسيطاً واحداً هو اسم العمود الذي يكون من النمط التاريخ و الوقت، أو تابع تاريخ، أو تاريخ ما، فإذا أردت أن تعرف مثلاً اسم اليوم الذي انتسبت فيه الموظفة نور إلى الشركة، فإنك ستستخدم التابع **dayname()** كما يلي:

```
select emo_no , name , date , dayname(date) as 'day name' from emp
where name='nour';
+----------+----------+-------------+-----------------+
|emp_no | name    | date        |day name         |
+----------+----------+-------------+-----------------+
| 2        | nour    | 20000115 |Saturday         |
+----------+----------+-------------+-----------------+
```

و إذا أردت أن تعرف ما اسم هذا اليوم.

```
select dayname(now()) ;
+----------------------+
|(dayname(now))     |
+----------------------+
|Friday                |
+----------------------+
```

و إذا أردت أن تعرف ما هو اسم اليوم الذي كان تاريخه **2001-9-11**

```
select dayname('2001-9-11');
+------------------------------+
|dayname('2001-9-11')      |
+------------------------------+
|Tuesday                       |
+------------------------------+
```

و كذلك الأمر في حال أردت أن تعرف اسم الشهر، و كل ما يجب عليك هو أن تبدل التابع **dayname()** بالتابع **monthname()** ، إذن توابع التاريخ هذه بسيطة جداً، و في ما يلي قائمة بأسماء توابع التاريخ و الوقت.

التابع	القيمة التي يعيدها التابع
dayname()	يعيد اسم اليوم
monthname()	يعيد اسم الشهر
week()	يعيد عدد صحيح هو ترتيب الأسبوع في السنة
year()	يعيد عدد صحيح يمثل السنة
yearweek()	السنة و الأسبوع بالتنسيق التالي: **YYYYWW**
dayofweek()	يعيد عدد صحيح هو رقم اليوم في الأسبوع **Sunday=1, Monday=2 ...**
weekday()	نفس التابع السابق **Monday=0, Thursday=1 ...**
dayofmonth()	يعيد عدد صحيح يمثل ترتيب اليوم في الشهر
dayofyear()	يعيد عدد صحيح يمثل ترتيب اليوم في السنة
month()	يعيد عدد صحيح يمثل ترتيب الشهر في السنة
hour()	الساعة، من 0 حتى 23
minute()	الدقائق، من 0 حتى 59
second()	الثواني، من 0 حتى 59
quarter()	يعيد قيمة صحيحة تمثل ترتيب فصل السنة

اكتب استعلاماً يعيد لنا عدد طلبات الشراء التي تمت في جميع أيام الأربعاء من شهر أيار من عام 2004

```
select count(*) as 'total orders' from bills where
dayname(bil_date)='Wednesday' and monthname(bil_date)='may' and year(bil_date)='2004';
```

اكتب استعلاماً يعيد لنا تقريراً بسيطاً عن سير عمليات البيع التي تمت خلال الأسبوع الماضي

```
select dayname(bil_date) as 'day' , count(bil_no) as 'Orders per Day' , sum(paid) as 'total sale'
from bills
where bil_date>=(now()-interval 7 day)
group by dayname(bil_date);
```

و ستظهر النتيجة كما يلي:

```
+-------------+----------------------+-------------+
|day          |orders per day        |total sale   |
+-------------+----------------------+-------------+
|Thursday     |1                     |450          |
|Wednesday    |5                     |1850         |
+-------------+----------------------+-------------+
```

في الحقيقة توابع التاريخ و الوقت كثيرة جداً، لكن يمكنك الاستغناء عنها، فما تعلمته في هذا الفصل من توابع التاريخ و الوقت كافياً جداً.

الفصل التاسع
التوابع العالية في MySQL

توابع التحكم بالتدفق
كما هو واضح من اسمها أنها تستخدم للتحكم في سير البرنامج، و أهم تابعين يستخدمان للتحكم بالتدفق في **MySQL** هما **ifnull() , if()** و سنتكلم عنهما الآن إن شاء الله.

التابع **ifnull() :**
لنفرض أنك ستكتب استعلاماً يعيد مجموع كل من الراتب **salary** و الحوافز **bonus** لكل موظف، فإنك ستكتب الاستعلام التالي:

```
select name , salary + bonus from emp;
```

ستفاجئ عندما يظهر لك مجموع راتب و حوافز بعض الموظفين هو **NULL** فكيف حصل ذلك؟ إن الاستعلام السابق يقوم بجمع الراتب مع الحوافز، لكن هناك بعض الموظفين الذين حوافزهم هي **null** و أنت تعرف أن نتيجة جمع أي قيمة مع **NULL** هي **null** و للتخلص من هذه المعضلة فأنت بحاجة إلى الخوارزمية التالية:

قم بتشغيل استعلام **MySQL** على الجدول **emp**
يعيد **name** و مجموع **salary + bonus**
فإذا كان **null = bonus**
اطبع فقط **salary**
و إلا
اطبع **salary + bonus**

إذا أردت كتابة هذه الخوارزمية بأحد لغات البرمجة مثل **PHP** فهي ستكون طويلة، لذلك توفر لنا **MySQL** التابع **ifnull()** الذي يعمل عمل الخوارزمية السابقة. يأخذ هذا التابع وسيطين، و هو يقوم باختبار قيمة الوسيط الأول فإذا لم تكن مساوية لـ **null** فسيعيد قيمة الوسيط الأول، و إذا كانت قيمة الوسيط الأول مساوي لـ **null** فإن التابع سيعيد قيمة الوسيط الثاني، لذلك فإن الاستعلام السابق سيأخذ الشكل التالي:

```
select name , salary + ifnull(bonus , 0) as 'total' from emp;
```

التابع **if() :**
يأخذ التابع **if()** ثلاث وسطاء حيث أن الوسيط الأول سيكون شرطاً، و من الممكن أن يكون أي معامل حسابي أو سلسلة محرفية أو عبارة **like** ، و في حال كان الشرط محققاً فإن التابع سيعيد الوسيط الثاني، و إلا فإنه سيعيد الوسيط الثالث، دعنا نأخذ المثال التالي:
لنفرض أنك تريد أن تقدم حسماً قدره 15% على كل أفلام العنف، فباستخدام التابع **if()** سيكون شكل الاستعلام كما يلي:

```
select mov_name , kind, if(kind ='violence' , price * .85 , price) as price from movies ;
```

لاحظ أن الاستعلام سيبحث في العمود **kind** فإذا وجد فيها القيمة **violence** سيقوم بضرب القيمة المخزنة بالعمود **price** بـ **0.85** و إذا وجد أن القيمة المخزنة في العمود **kind** ليست **violence** فسيطبع القيمة المخزنة في العمود **price** كما هي.

و بالتالي ستكون نتيجة الاستعلام السابق كما يلي:

```
+----------------------+--------------+-------------+
|mov_name              |kind          |price        |
+----------------------+--------------+-------------+
|last man standing     |violence      |212.50       |
|city of angels        |romance       |300          |
|urban legend          |horror        |300          |
|the fist of legend    |violence      |212.50       |
|assassins             |violence      |340.00       |
|face off              |violence      |365.50       |
+----------------------+--------------+-------------+
```

توابع الإخفاء

تستخدم توابع الإخفاء من أجل تخزين المعلومات بشكل مشفر مثل كلمات المرور أو أرقام بطاقات الائتمان و ذلك للحفاظ على سرية هذه المعلومات الحساسة.

التابعين password() و md5() :

يَستخدم هذان التابعان ما يسمى بخوارزمية الاتجاه الواحد **(one-way algorithm)** حيث تقوم خوارزمية الاتجاه الواحد ببعثرة سلسلة محرفية بطريقة تجعل إعادة تجميعها غير ممكنة، لذلك فلو اخترق أحد العابثين ملقمك فإنه لن يجد طريقة لتحديد شكل كلمة المرور الأصلية.

يقوم التابع **password()** بإنشاء سلسلة محرفية (خليط من أرقام و أحرف) مكونة من 16 محرفاً، و ذلك مهما كان طول كلمة المرور التي تقوم أنت كمستخدم و ليس كمبرمج بإدخالها، أنظر إلى الاستعلامات التالية

```
select password ('a');                          //60671c896665c3fa
select password ('aa');                         //077baf58491e1952
select password ('by the name of Allah');       //34b8da5a29c9317c
```

يفضل أن يكون نوع العمود الذي يستخدم التابعين **password() , md5()** النوع **char** و ليس من النوع **text** أو **varchar**، و يتم إنشاء عمود كلمة المرور مثل أي عمود نصي آخر، أما عن كيفية إدخال البيانات فيه، فإليك هذا المثال

سنقوم بإضافة عمود خاص بكلمات المرور اسمه **pwl** اختصار (**Pass Word Log**) إلى الجدول **clients**

```
alter table clients add column pwl char(16) not null;
```

لنقم بإدراج سجل لزبون جديد، و لاحظ كيفية إدراج كلمة المرور

```
insert into clients values (4 , 'Rami' , 'Rami@mail.sy' , password('SAR')) ;
```

إذن تم إدراج كلمة المرور عن طريق تابع، و وسيط هذا التابع كانت كلمة المرور، لاحظ أن طول كلمة المرور التي أدخلتها كان ثلاث أحرف، أما طول السلسلة المحرفية التي سيقوم التابع **password()** بتكوينها سيكون 16 محرف، قم الآن بكتابة الاستعلام التالي:

```
select * from clients where cli_no=4;
+--------------+-------------+----------------------+----------------------+
|cli_no        |cli_name     |cli_email             |pwl                   |
+--------------+-------------+----------------------+----------------------+
|4             |Rami         |rami@mail.sy          |7f3fd4a026a892ac      |
+--------------+-------------+----------------------+----------------------+
```

أما لكتابة استعلام يعيد لنا السجل الخاص بالزبون رامي، فيتم كما يلي

```
select * from clients where cli_name='rami' and pwl=password('SAR');
```

طبعاً توابع كلمة المرور حساسة لحالة الأحرف، قم بكتابة نفس الاستعلام السابق مع تبديل الأحرف **SAR** بأحرف صغيرة **sar** ، سيعيد لك الاستعلام الجملة التالية:

```
Empty set (0.00 sec)
```

التابع **md5()** يختلف عن نظيره السابق بأنه يعيد سلسلة محرفية مكونة من 32 محرفاً، أما باقي المزايا فهي واحدة.

التابعان **encode()** و **decode() :**

قبل كل شيء يجب أن تعرف أن هذان التابعان يعملان مع بعضهما البعض، أي أن التابع **encode()** يقوم بالتشفير، و التابع **decode()** يفك التشفير.

يأخذ التابع **encode(str , pass_str)** وسيطين، الأول هو السلسلة المحرفية التي تريد تشفيرها (كلمة المرور)، و الوسيط الثاني سلسلة محرفية أيضاً لترشد التابع **encode()** على كيفية التشفير، و طبعاً كلا الوسيطين اختياريين، و لتوضيح الأمر أنظر إلى سلسلة الاستعلامات التالية

```
select encode('khalil' , 'my pass');
select encode('khalil' , 'zoom');
select encode('khalil' , 'good');
```

ستكون نتيجة هذه الاستعلامات معقدة، و هي تشبه الصورة عندما تريد فتحها في محرر نصوص، أي أن التابع **encode()** يقوم بإنشاء كائناً ثنائياً (و ليس مثل التابعين **password() , md5()** اللذين يبنيان سلسلة نصية)، لذلك فإن نوع العمود الذي سيحوي كلمات المرور المستخدمة مع التابع **encode()** سيكون من النوع **text** الذي يدعم الكائنات الثنائية و ليس من النوع **char** أو **varchar** اللذان يدعما السلاسل المحرفية.

التابع **decode(crypt_str , passs_str)** يأخذ وسيطين أيضاً، الأول هو اسم العمود الذي يحوي كلمات المرور المستخدمة مع التابع **encode()** ، و الوسيط الثاني هو السلسلة المحرفية نفسها المستخدمة مع التابع **encode()** و التي قلنا عنها أنها ترشد التابع **encode()** على كيفية التشفير، سنأخذ هذا المثال لتوضيح كيفية عمل كلا التابعين:

أولاً: قم بإنشاء الجدول التالي

```
create table temp(
id int(3) not null primary key,
name varchar(15) not null,
login text not null ) ;
```

ثانياً: قم بإدراج سجل إلى هذا الجدول و لاحظ كيف يتم إدراج كلمة المرور الثنائية

```
insert into test_table values (1 , 'rasheed' , encode('tiger', 'pass'));
```

ثالثاً: قم بتشغيل الاستعلام التالي و شاهد النتيجة

```
select id ,  name from temp where decode(login, 'pass')='tiger' ;
```

توابع معالجة السلاسل المحرفية

تحوي **MySQL** على توابع لمعالجة السلاسل المحرفية، على الرغم من أن **PHP** قادرة على معالجة السلاسل المحرفية بشكل سهل، و لكن لا ضير في أن تتعرف على توابع **MySQL** بحيث يمكنك استخدامها عندما تحتاج إليها.

التابعان **ucase()** و **lcase() :**

يقوم التابع **ucase()** بتحويل الأحرف الإنكليزية إلى أحرف كبيرة، و يأخذ وسيطاً واحداً.

```
select ucase('I am learning MySQL') as Capitals ;
+------------------------------+
|Capitals                      |
+------------------------------+
|I AM LEARNING MYSQL           |
+------------------------------+
```

و التابع **lcase()** أيضاً له نفس وضع التابع **ucase()** إلا أنه يحول الأحرف الإنكليزية إلى أحرف صغيرة.

```
select ucase('I am LEARNING MySQL') as 'Small Letters' ;
+----------------------+
|Small Letters         |
+----------------------+
|i am learning mysql   |
+----------------------+

select ucase(name) from emp where emp_no=1;
+----------------------+
|ucase(name)           |
+----------------------+
|OMAR                  |
+----------------------+
```

و إذا أردت أن تجعل كلمات المرور غير حساسة لحالة الأحرف، فيمكنك أن تجعل جميع الحروف كبيرة، و ذلك باستخدام التالي ضمن الاستعلام:

```
select cli_email from clients where
cli_name='rami' and pwl=password(ucase('sar'));
```

التابعان **left()** و **right() :**

يستخدم التابع **left()** لإظهار مجموعة محارف من سلسلة محرفية ما، حيث يأخذ هذا التابع وسيطان الأول هو سلسلة محرفية، أو اسم عمود، و الوسيط الثاني هو عدد المحارف التي تريد إظهارها.

```
select left('Syrian Arab Republic' , 6) a 'My String' ;
+--------------+
|My String     |
+--------------+
|Syrian        |
+--------------+
```

```
select mov_name, left(star_1, 4) as description from movies ;
```

أما التابع **right()** فإنه لا يختلف عن التابع **left()** بأي شيء سوى أنه يقوم بإظهار المحارف من الجانب اليميني للسلسلة المحرفية.

```
select right('Syrian Arab Republic' , 8) as 'My String' ;
+--------------+
|My String     |
+--------------+
|Republic      |
+--------------+
```

التابع **concat() :**

يأخذ التابع **concat()** عدداً غير محدد من الوسطاء، و يقوم بربط السلاسل المحرفية ببعضها البعض، بمعنى أنه يقوم بإضافة سلسلة محرفية أولى إلى نهاية سلسلة محرفية أخرى و هكذا.

```
select concat('www.' , 'c4arab' , '.com') as 'Arabic Encyclopedia';
+----------------------+
|Arabic Encyclopedia |
+----------------------+
|www.c4arab.com     |
+----------------------+
```

مثال آخر

```
select mov_name , concat(star_1, ' \\' , star_2) as stars from movies;
+----------------------+------------------------------+
|mov_name              |stars                         |
+----------------------+------------------------------+
|last man standing     |brouce wiles \ no name        |
|city of angels        |nicolas cage \ no name        |
|urban legend          |no name \ no name             |
|the fist of legend    |get lee \ no name             |
|assassins             |S. Stallone \ A. Banderas     |
|face off              | N. Cage \ J. Travolta        |
+----------------------+------------------------------+
```

التوابع **ltrim()** و **rtrim() :**

أحياناً عندما يقوم زوار الموقع بإدخال معلومات بواسطة تطبيق أو نموذج ستجد أن بعض الحقول قد تم أضافة فراغات إما قبل أو بعد السلسلة المحرفية التي يكتبوها، و هذه التوابع مخصصة لإزالة تلك الفراغات الزائدة.

يأخذ كل من التابعين **ltrim() , rtrim()** وسيطاً واحداً و يزيلا الفراغات من يمين و يسار السلسلة الحرفية على الترتيب.

```
select rtrim ('Khalil          ');
+------------------------------+
| rtrim ('Khalil          ')   |
+------------------------------+
|Khalil                        |
+------------------------------+
```

التابع **trim() :**

التابع **trim()** أكثر إرباكاً من التابعين **rtrim() , ltrim()**، و له عدة استخدامات:

1. إذا استخدمت هذا التابع مع وسيطاً واحداً فقط، فستتم إزالة الفراغات من كلا جانبي السلسلة المحرفية

```
select trim(mov_name) from movies;
select trim('    Khalil          ');
+------------------------------+
| trim('    Khalil          ') |
+------------------------------+
|Khalil                        |
+------------------------------+
```

2. يمكن أن يستخدم التابع لإزالة السلاسل المحرفية إما من بداية (**leading**) أو نهاية(**trailing**) أو كلا جانبي (**both**) السلسلة المحرفية، و لهذا التابع الشكل التالي:

```
trim( [both | leading | trailing] [string_to_strip] from string) ;
```

حيث أن **string_to_strip** هي الجملة المراد حذفها من السلسلة النصية **string**، و يمكن أن يكون الجزء **strin_to_strip** إما محرفاً واحداً أو عدة محارف.

فمثلاً لإزالة المحرفين **"er"** من نهاية العمود **job** يمكنك ذلك كما يلي:

```
select tirm(trailing 'er' from job) as 'job' from emp;
+--------------+
|job           |
+--------------+
|manag         |
|program       |
|sall          |
|sall          |
|program       |
|sall          |
|sall          |
+--------------+
```

فتظهر لنا النتيجة بعد حذف الحرفين **er**، و بذلك نكون قد انتهينا من توابع معالجة السلاسل المحرفية بإذن الله

توابع مفيدة في MySQL

التابع **version() :**
لقد تعرفت سابقاً على التابع **version()** و هو يقوم بإظهار نسخة الـ **MySQL** التي تعمل عليها، و لا يأخذ هذا التابع أي وسائط.

```
select version();
+----------------------+
|version()             |
+----------------------+
|3.23.47-max-debug |
+----------------------+
```

التابع **database() :**
يظهر لك هذا التابع اسم قاعدة البيانات التي تعمل عليها أنت الآن، و لا يأخذ هذا التابع أي وسائط.

```
select database();
+--------------+
|database()    |
+--------------+
|movie_store |
+--------------+
```

التابع **user() :**
يظهر لك هذا التابع اسم المستخدم و اسم المخدم الذين تعمل عليهما الآن، و ستتعرف، ستعرف مدى أهمية هذه التوابع في الدروس القادمة إن شاء الله

```
select user();
+----------------------+
|user()                |
+----------------------+
|root; @localhost      |
+----------------------+
```

التابع **rand() :**
يقوم هذا التابع بتوليد أرقاماً عشوائية **random** تتراوح بين 0 و 1 ، و تستطيع أن تعطي هذا التابع رقماً واحداً كوسيط له، مع العلم أنه يمكن ألا يأخذ وسطاء، و يستخدم هذا التابع بشكل أكبر ضمن عبارة **order by** عندما تريد أن تظهر نتائج استعلام بترتيب عشوائي.
لنفرض أنك تريد أن تقدم مكافئة مالية لثلاث موظفين لديك و ستختارهم بشكل عشوائي، عندها ستستخدم الاستعلام التالي:

```
select * from emp order by rand() limit 3 ;
+--------------+--------------+--------------+--------------+--------------+--------------+
|emp_no        |name          |job           |salary        |bonus         |date          |
+--------------+--------------+--------------+--------------+--------------+--------------+
|5             |osama         |programmer |9000        |250           |20010522      |
|3             |ahmad         |saller        |5000        |NULL          |20000201      |
|2             |nour          |programmer |8000        |500           |20000115      |
+--------------+--------------+--------------+--------------+--------------+--------------+
```

الفصل العاشر
البحث المتقدم و الفهرسة المتقدمة في MySQL

الفصل الحادي عشر
المسالك المتعددة و الإجراءات في MySQL