AGASHE

MySQL

حتى الإحتراف

تأليف

م / محمد يوسف

الفهرس

المقدمة 2

الفصل الأول / ما هي SQL 6

الفصل الثاني / التعامل مع DDL و تعريف قواعد البيانات 12

الفصل الثالث / التعامل مع DML من إضافة و حذف و تعديل للبيانات 22

الفصل الرابع / التعامل مع DQL من البحث و توابعه 25

الخاتمة 36

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

السلام عليكم و رحمة الله تعالى و بركاته

أخواني الأعزاء مع التطور الهائل في الحواسيب أصبحت كمية البيانات التي يتم معالجاتها و تخزينها بشكل يومي مهولة و أصبحت قواعد البيانات ضرورة ملحة ، من هذا المنطلق سوف نتابع داخل فصول هذا الكتاب شرح لقواعد البيانات و كيفية التعامل معها بإذن الله.

تعريف قاعدة البيانات (Data Base):

هو ملف يحتوي على أنواع مختلفة من بيانات (نصوص و ارقام ... الخ) ، وفق ترتيب معين يتيح للمستخدم التحكم بهذه البيانات بالإضافة أو التعديل و الحذف طبعا مع البحث و الاستخراج ، و غالبا ما يتم التعامل مع ملف قاعدة البيانات بواسطة برنامج و ليس بشكل مباشر سواء أكان برنامج موجه لسطح المكتب أو للهواتف أو للمواقع أو حتى نظام تشغيل ، و ذلك للحفاظ على سرية البيانات و ترتيبها و تسهيل التحكم بها .

مكونات قاعدة البيانات (Data Base):

أي قاعدة بيانات في الكون تتكون من جداول (Tables) ، الجدول بدوره يحتوي على صفوف - سجلات (records - rows) و أعمدة – حقول (columns – fields) ، تقاطع الأعمدة مع الصفوف يعطي **الخلية** و هي وحدة البناء الأساسية في قواعد البيانات من هذا المفهوم لدينا :

العمود : يشكل معلومة داخل الصف مثل الاسم أو رقم الهاتف أو العنوان ...الخ .

الصف : مجموعة خلايا تشكل ملف عن شيء معين تماما كملفك الدراسي فهو يحتوي على عدد كبير من المعلومات (الأعمدة) .

الجدول : مجموعة الصفوف والأعمدة يعني ملفك مع ملفات زملائك في الفصل مع ملفات زملائك في المرحلة التعليمية ينتج لدينا مثلا جدول للصف الثاني الثانوي .

قاعدة البيانات : و التي تشتمل على كل أنواع البيانات داخل منظومة معينة بغض النظر عن كيفية توزيعها بالمختصر جدول الصف الثاني الثانوي مع الثالث مع الأول و المرحلة المتوسطة (الإعدادية) و الإبتدائية كل هذه الجداول تشكل قاعدة بيانات لطلاب مدرسة ☺ .

الخلاصة :

مفاتيح الجداول (Tables Keys) :

تعتبر المفاتيح من أهم المفاهيم في قواعد البيانات و الغرض الأساسي منها هو الربط بين الصفوف المختلفة بين الجداول ما يهمنا في هذا الكتاب ثلاث منهم :

1 - المفتاح الأساسي (Primary Key) : و هو مفتاح فريد يميز نوع معين من البيانات بحيث أنها تصبح غيرة قابلة للتكرار مثال على ذلك كود الموظف في المثال السابق فلا يمكن لموظفين في الشركة أن يحملا نفس الرقم006 لكن من الوارد أن يحملا نفس الاسم أو نفس تاريخ الميلاد ، كل جدول في الطبيعة يفضل أن يمتلك مفتاح أساسي و نادرا ما ترى جدول بدون مفتاح أساسي .

2 - المفتاح الأجنبي (Foreign Key) : هو مفتاح مرتبط بالمفتاح الأساسي لجدول آخر بمعنى أن كل مفتاح أساسي يمكنه الارتباط مع مفتاح أجنبي في جدول آخر و عادة ما يحمل المفتاح الأجنبي نفس اسم المفتاح الأساسي في الجدول الآخر ، مثلا تريد الربط بين جدول الموظفين و جدول الرواتب فمن البديهي هنا أن يكون كود الموظف هو حلقة الوصل بين الجدولين حيث أنه مفتاح أساسي في جدول الموظفين و مفتاح أجنبي في جدول الرواتب .

3 - الفهارس (Indexes) : تماما كفهرس الكتاب تستعمل لتسريع عملية جلب البيانات كما أنها مفيدة في عمليات البحث .

العلاقات بين الجداول :

نظرا لتشعب البيانات و اختلاف دور قاعدة البيانات الذي تلعبه داخل المنظومة نجد أحيانا ترابط بين الجداول فمثلا جدول الطلاب مع جدول الحصص الدراسية ، و عليه ففي مثل هذه الحالات نحتاج لربط عدة جداول ببعضها البعض و الربط هنا لا يعني بالضرورة ربط كافة الصفوف بل يكفينا ربط صف معين من خلال عمود مثل كود العميل مع نوع البضائع ، من هذا المنطلق فالعلاقات بين الجداول ثلاثة أنواع :

1 - واحد إلى واحد (one to one) : و نجد هنا أن العلاقة بين الجدولين قائمة على ربط صف من جدول مع صف وحيد من جدول آخر و قليلة جدا عدد المرات التي نصادف بها هذا النوع .

2 - واحد إلى متعدد (One to Many) : و هذا من أكثر الأنواع شيوعا فمثلا نريد الربط بين جدول الزبائن و جدول البضائع فتجد هنا كل زبون يمكنه شراء عدة بضائع .

3 - متعدد إلى متعدد (Many to Many) : و هنا تلاحظ أن العلاقة تبادلية فمثلا جدول الموردين و جدول البضائع فنوع البضاعة الواحد يمكن جلبه من أكثر من مورد و أيضا المورد الواحد يمكنه إرسال عدة بضائع .

و هكذا نكون قد أخذنا فكرة عامة عن قواعد البيانات و عن كيفية عملها .

الفصل الأول / ما هي SQL :

هي إختصار لـ Structured Query Language و التي تعني بالعربية " لغة الاستعلام المهيكلة " ، بشكل عام قواعد البيانات كما قلنا سابقا فهي ملفات منفصلة عن البرنامج يمكنك أن تقوم بفتحها يدويا و التعديل عليها لكن هذه الطريقة بالتأكيد غير واقعية ، من هذا المنطلق طور العلماء SQL لتكون لغة تخاطب بين قواعد البيانات لغات البرمجة ، يعني بدل أن يكون لكل لغة برمجة أوامر معينة للتعامل مع قواعد البيانات ، أصبح هناك لغة مشتركة تمكنك من التعامل مع قاعدة البيانات مهما كانت لغتك البرمجية و تذكر دائما أن SQL مخصصة للتعامل مع قواعد البيانات العلائقية (RDBMS) و التي يمكن ربط جداولها بعلاقات كما أشرنا سابقا .

هل الحياة فقط بها SQL ؟؟؟

لا مع الأسف ففي الماضي السحيق و قبل ظهور SQL كانت NoSQL هي الأكثر انتشارا و التي عادت مرة أخرى للأضواء مع ظهور مجال Data Mining ، لن نتحدث عن NoSQL في هذا الكتاب و ربما أيضا خارج الكتاب ☺ ، لكن من الجميل أن تعلم أن NoSQL لا تعتمد على الجداول و لا الصفوف و لا الأعمدة بل لها نظام قائم على ما هو أشبه بتعريف الأصناف في لغات البرمجة .

```
{
        Book_name : "MySQL" ,
        Book_author : "Mohamed" ,
        Book_edition : "3"
}
```

موضوع آخر تماما لا تشغل بالك به ☺

محركات SQL (SQL engins) :

كل لغة برمجة كما نعلم لها مترجم (Compiler) خاص بها ، و المترجمات أنواع و قد تجد بين مترجمات اللغة الواحدة اختلافات كإضافة دوال أو حذف أخرى ، ففي لغة C++ لدينا Visual C++ الخاص بـ Micosoft و هناك GCC التابع لمشروع GNU .

بالمثل لدينا في SQL :

Oracle

MS SQL Server

MS Access

PostgreDB

SQLite

FireBird

MariaDB

و أخير MySQL .

طبعا المحركات لا نهاية لها لكن هذه تقريبا أشهر المحركات التي ستمر عليك و بين صفحات هذا الكتاب سيكون جل إهتمامنا منصب على MySQL نظرا لكونه أكثرهم أهمية و انتشارا .

فقط أحب أن أنوه إلى MongoDB هو أشهر محركات الـ NoSQL و تقريبا لا منافس له ☺

هل تختلف SQL من محرك لآخر ؟

ليس كثيرا لكن بالطبع هناك اختلافات عليك معرفتها قبل استخدام محرك معين .

لماذا اخترنا MySQL في هذا الكتاب ؟

نظرا لطبيعة عصرنا الحالي أغلب تطبيقات قواعد البيانات أصبحنا نصادفها بشكل يومي في مواقع و تطبيقات الإنترنت فلا يخلو موقع مهما كان صغير من قاعدة بيانات مهما كان حجمها أو نوعها ، لهذا محور دراستنا في هذا الكتاب سيكون عن MySQL و هي تعتبر أقوى محركات SQL بالنسبة للإنترنت .

ماذا عن باقي المجالات ؟

مجال سطح المكتب : يمكنك استعمال Oracle أو Access أو SQL Server و ثلاثتهم أفضل من بعض في هذا المجال.

مجال تطبيقات الهواتف : طالما تطبيقك يعمل من خلال الإنترنت هنا يمكنك استعمال قاعدة بيانات مثبتة على خادم (Server) و طبعا MySQL هو أفضل خيار بالنسبة لك ، أما في حالة تخزين البيانات على جهاز المستخدم فالأفضل استعمال SQLite .

ما هي اللغات التي تدعمها MySQL ؟

تقريبا كل اللغات الشهيرة PHP , Pascal , Delphi , Java , C# , Python , Ruby , Perl , C/C++ و تعتبر PHP هي الأكثر شيوعا و غالبا ما يقترن اسم PHP مع MySQL ☺

هل ما زالت MySQL في السباق أم أنها في آخر أيامها ؟

أطمئن عزيزي القارئ فكبرى الشركات FaceBook و Google و twitter و الكثير من الشركات و أغلب المواقع إن لم يكن جميعها لا زالت تستعملها لذا لا داعي من القلق MySQL حديثة و متطورة و تضاهي التقنيات الفضائية ☺

أقسام MySQL :

في لغة MySQL كل مجموعة أوامر تتبع لقسم معين و تقريبا كل قسم يعتبر لغة منفصلة لها وظيفة معينة ، و هم بالترتيب :

1 - لغة تعريف البيانات (Data Definition Language - DDL) : في هذه المجموعة يكون تعاملنا مع قاعدة البيانات و الجداول من إنشاء و تعديل و حذف .

2 - لغة معالجة البيانات (Data Manipulation Language - DML) : و هي المجموعة التي تتعامل مع الصفوف و الأعمدة من إضافة و حذف و تعديل للبيانات داخل الخلايا .

3 - لغة استعلام البيانات (Data Query Language - DQL) : و هي تشمل كل ما له علاقة عن البحث داخل الجداول عن البيانات و ما يترتب عليه من توابع كالربط و البحث بالشروط ... الخ .

4 - لغة تحكم البيانات (Data Control Language - DCL) : و هي تختص بموضوع الحماية و إعطاء الصلاحيات المختلفة للبيانات داخل الجداول .

5 - لغة معاملة البيانات (Data Transaction Language - DTL) : وهي المسؤولة عن الإجراءات (Transactions) و هي باختصار مجموعة أوامر لا يمكن تنفيذ أحدها إلا عندما ينفذ باقي الأوامر وفق ترتيب معين و وظيفة DTL هو الحفاظ على سير تنفيذ هذه الإجراءات .

محركات MySQL :

كما يوجد محركات للـ SQL أيضا يوجد محركات للـ MySQL ، لكنها ليست بنفس الأهمية ، و في الحياة العملية يوجد محركان يتنافسان و الباقي ربما لن تسمع عنهم طوال حياتك ☺ ، الأول هو MyISAM و هو ما سنستعمله في هذا الكتاب نظرا لسرعته و ارتفاع حد التخزين به ، أما منافسه اللدود InnoDB هو جيد و منتشر لكننا لن نتحدث عنه.

بيئة العمل :

طوال صفحات الكتاب سيكون الشرح و التطبيق من خلال سطر الأوامر () الخاص بـ MySQL ، أما بالنسبة للبرنامج فأنا شخصيا أفضل استعمال خادم محلي (Local Server) مثل Appserv ، فـ MySQL تكون مثبتة معه بشكل تلقائي و خاصة أننا نتحدث عن الإنترنت ، لكن إذا كنت مصر فيمكنك استعمال استعمال أي برنامج طالما أنه سيؤدي الغرض .

ملحوظة : إصدار MySQL الذي سنعمل عليه في هذا الكتاب هو 5.0.45 .

تنصيب بيئة العمل :

1 - أولا ندخل لموقع https://www.appservnetwork.com/en/download

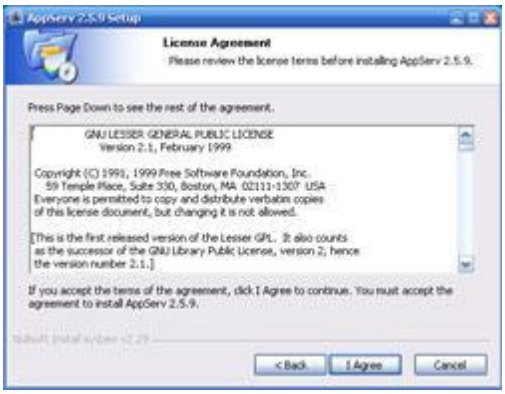
2 - نختار النسخة التي تناسبنا و لا يشترط تحميل أحدث إصدار ، فالموقع يتيح لك تحميل الإصدارات القديمة في حالة إمتلاكك لإنترنت ضعيف ☺ ، جميع الإصدارات ستعمل بلا مشاكل أثناء الشرح .

3 - بعد التحميل سيكون لدينا هذا الملف نقوم بفتحه :

appserv-win32-2.5.9

4 - كتنصيب أي برنامج نضغط التالي :

5 - نوافق على إتفاقية الترخيص :

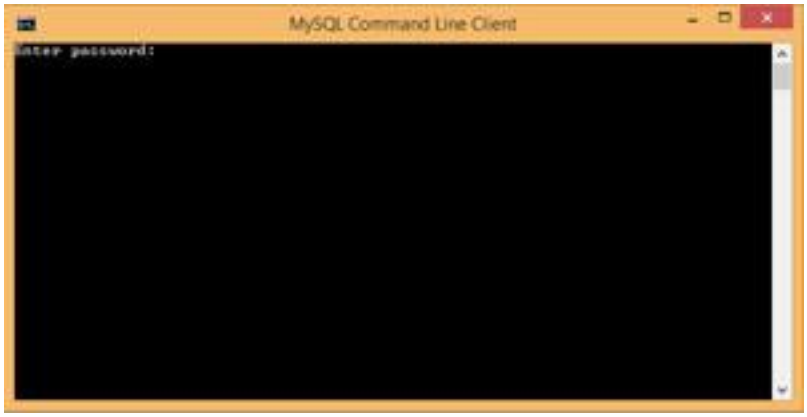


6 - نقوم بتعيين اسم الخادم (Server) و في حالتنا سيكون localhost :

7 - ندخل كلمة المرور (password) سهلة التذكر ☺ مثلا 1234 :

8 - و أخيرا نضغط على finish

9 - نبحث في قائمة إبدأ (Start Menu) عن MySQL Command Line Client فهذه هي الواجهة التي سنتعامل معها طوال فصول الكتاب



بعد أن تقوم بإدخال كلمة المرور التي قمنا باختيارها و الضغط على زر Enter

هكذا نكون قد أعدادنا بيئة العمل و إبتداءًا من الفصل القادم سوف نبدأ العمل الجاد ☺

الفصل الثاني / التعامل مع DDL و تعريف قواعد البيانات :

سنبدأ هذا الفصل أولا بالتعرف على القواعد اللغوية للـ MySQL :

1 - MySQL غير حساسة لحالة الأحرف ، لكن حافظ دائما على أن تجعل أي اسم عنصر تنشئه (Lower Case) .
2 - يفضل دائما أن تكتب الكلمات المحجوزة (Reserved Words) الخاصة باللغة بالأحرف الكبيرة (Upper Case) .
3 - جميع السطور البرمجية تنتهي بالفاصلة المنقوطة (; Semi Colon) .
4 - جميع النصوص و التواريخ تكتب بين علامتي تنصيص مزدوج ، أما الأرقام فلا .
5 - يجوز استعمال جميع الحروف و الأرقام ، علامة $ و _ في تسمية قواعد البيانات و الجداول و الأعمدة .
6 - التعليقات (comments) :

```
# one line
-- one line
/*
Multiple
Line
*/
```

إنشاء قاعدة بيانات :

لإنشاء قاعدة بيانات نستعمل الأمر CREATE كالتالي :

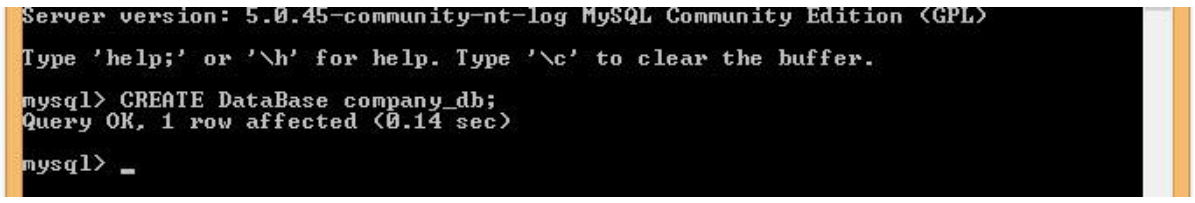
```
CREATE DATABASE DB_name ;
```

CREATE - أمر الإنشاء .

DATABASE - تحديد نوع العنصر الذي سيتم إنشاءه فبعد لحظات سنستعمل نفس الأمر لإنشاء جدول .

DB_name - اسم قاعدة البيانات الذي تختاره .

قم بفتح سطر أوامر MySQL كما تعلمنا و لنجرب إنشاء قاعدة بيانات لشركة مثلا ، فالنسميها company_db .



كما تلاحظ تم تنفيذ الأمر بنجاح "Query OK" و أصبح لدينا قاعدة بيانات ☺

إنشاء جدول :

هذا الجزء طويل قليلا و به الكثير من المعلومات لذا ركز جيدا ☺

```
CREATE TABLE TB_name(

col1  data_type  attribute1  attribute2 . . . . .  attributeN,

col2  data_type  attribute1  attribute2 . . . . .  attributeN,

col1  data_type  attribute1  attribute2 . . . . .  attributeN,

.
.
.
colN  data_type  attribute1  attribute2 . . . . .  attributeN,

Primary Key( col1 )

)Type = engine;
```

الشرح :

1 - CREATE TABLE TB_name - إنشاء جدول بالاسم TB_name .

2 - colN.....col1 , col2 – أسماء الحقول ((الأعمدة)) التي بالجدول كرقم رقم الهاتف و العنوان ..... الخ ، و طبعا عددها غير محدود لذا رمزنا لها بـ N .

3 - data_type - نوع البيانات الذي ستحمله الحقول فمثلا اسم الموظف من نوع نص و مرتبه عدد كسري .

4 - attribute1 attribute2 . . . . . attributeN - كل حقل يمكنه امتلاك عدة خصائص سنعرفها في لحظات .

5 - Primary Key(col1) - كل جدول كما نعلم لابد أن يحتوي على مفتاح أساسي هذا الأمر هو المسؤول عن تعيين المفتاح الأساسي و يمكنك تحديد أي حقل تريد لكن عادة أول حقل في الكود هو المفتاح الأساسي .

6 - Type = engine - هذه الخاصية تحدد نو محرك MySQL الذي سنستعمله ، فكما ذكرنا أن MySQL لديها عدد من المحركات هنا في هذا الكتاب سنستعمل MyISAM .

أنواع البيانات :

Char (size)

Varchar(size)

Text

Longtext

Int

Int(size)

Float

Float(size, length)

Datetime

Enum('1', '2') default ('2')

1 - نص صغير : أقصى عدد حروف يمكنه حملها هو 255 و هو مفيد جدا لتخزين أسماء المستخدمين و كلمات المرور ، و يفضل غالبا استعمال varchar .

2 - النصوص الكبيرة : على حسب عدد الكلمات فالرسائل يمكن تخزينها في النوع Text بينما مواضيع مدونة يمكن تخزينها في Longtext .

3 - الأعداد الصحيحة تخزن في int ، أيضا يمكن تعيين طول العدد بكتابته بين القوسين .

4 - الأعداد الكسرية كلها تخزن في float ، مع العلم أنه يمكن تحديد عدد الخانات التي يمكنه حملها و عدد الخانات بعد العلامة العشرية

Float(5 , 3) → 12.345

5 - الوقت و التاريخ يتم تخزينهم في datetime .

6 - Enum هو نوع من البيانات يمكنه حمل أكثر من اختيار مثل صح و غلط ، مع وضع قيمة افتراضية في حالة أن المستخدم لم يقم بتعيين القيمة الأساسية .

خصائص البيانات :

1 - Null - هذه الخاصية تتيح للحقل أن يحمل قيم فارغة .

2 - Not null - هذه الخاصية تمنع ترك قيمة أي حقل فارغة أثناء عملية إضافة البيانات .

3 - Default - تستعمل مع not null لتعيين القيمة الافتراضية للحقل و غالبا تستعمل في حالة ترك المستخدم للحقل فارغا.

4 - Auto_increment - في المفتاح الأساسي كما نعلم لا يمكن لحقلين أن يتحملا نفس القيمة ، من ناحية أخرى نحن نستعمل المفتاح الأساسي لتمييز السجلات ((الصفوف)) ، مثل كود الموظف ، كود الطالب ،رقم رخصة القيادة ، رقمك القومي (بطاقة الأحوال الشخصية) و العديد من البيانات التي لا يصلح أن يتم تكرارها ، الخلاصة أن هذه الخاصية مرتبطة بالمفتاح الأساسي .

5 - Primary key - يمكن أن يتم كتابتها مع الخصائص لنفس الحقل أو كتابتها منفصلة في نهاية كود الإنشاء كما رأينا في الصيغة الأساسية .

6 - Signed - قيم الحقل يمكن أن تحمل إشارة موجبة أو سالبة ، لتستعمل في العمليات الحسابية .

7 - Unsigned - قيم الحقل لا يمكنها حمل أي إشارات ، و نستعمل هذه الخاصية دائما مع المفتاح الأساسي و كذلك أي أرقام لن تجرى عليها عمليات حسابية مثل أرقام الهواتف .

لتثبيت هذه المعلومات سنستعرض عددا من الأمثلة :

1 - إنشاء جدول الموظفين يحتوي على الحقول التالية :

كود الموظف - عدد صحيح - بدون إشارة ، قابل للزيادة التلقائية ، مفتاح أساسي .
اسم الموظف - نص صغير (50 حرف) - لا يمكن تركه فارغا .
تاريخ ميلاده - تاريخ و وقت - لا يمكن تركه فارغا .
رقم هاتفه المحمول - عدد صحيح (11 رقم) - بدون إشارة .

بعد أن نقوم بعمل هذا الترتيب على الورق حسب ماهية المشروع الذي نعمل عليه نقوم بترجمة كل كلمة لما يقابلها مما تعلمناه :

```
CREATE TABLE employees(
emp_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY ,
emp_name VARCHAR(50) NOT NULL,
emp_date DATETIME NOT NULL,
emp_phone INT(11) UNSIGNED
)Type = MyISAM;
```

2 - جدول مواضيع في مدونة :

كود الموضوع - عدد صحيح - بدون إشارة ، قابل للزيادة التلقائية ، مفتاح أساسي .
عنوان الموضوع - نص قصير (255 حرف) - لا يمكن تركه فارغا القيمة الإفتراضية 'بدون عنوان' .
الموضوع - نص طويل .
تاريخ نشره - تاريخ و وقت .
عدد المشاهدات - عدد صحيح - بدون إشارة .
تقييمه - عدد صحيح - بدون إشارة .

و عليه يكون الكود كالآتي :

```
CREATE TABLE posts(

post_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY ,

post_title VARCHAR(255) NOT NULL DEFAULT 'without title' ,

post_body LONGTEXT,

post_date DATETIME ,

post_readers INT UNSIGNED,

post_rank INT UNSIGNED

)Type = MyISAM;
```

كما تلاحظ DEFAULT كتبنا بعدها القيمة الافتراضية بين علامتين تنصيص فرديتين و الباقي لا غبار عليه .

3 - جدول أعضاء في منتدى :

كود العضو - عدد صحيح - بدون إشارة ، قابل للزيادة التلقائية ، مفتاح أساسي .
اسم العضو - نص قصير (255 حرف) - لا يمكن تركه فارغا .
لقب العضو - نص قصير (255 حرف) .
كلمة المرور - نص - لا يمكن تركه فارغا .
بريده الإلكتروني - نص قصير (255 حرف) - لا يمكن تركه فارغا .
تاريخ انضمامه - تاريخ و وقت .
صلاحياته - صحيح (1 رقم) - بدون إشارة ، القيمة الإفتراضية '3' .

بالنسبة لكلمة المرور فلقد اخترنا و ضعها في نوع نص على فرض أنه بعد تشفريها فقد يزيد طولها عن 255 لتجنب هذا النوع من المشكلات ، لذا قبل إنشاء أي جدول يرجى قياس حساب أدق التفاصيل كي نحصل على أفضل أداء☺

```
CREATE TABLE members(
member_id INT UNSIGNED AUTO_INCREMENT ,
member_name VARCHAR(255) NOT NULL ,
member_nickname VARCHAR(255),
member_password TEXT NOT NULL,
member_email VARCHAR(255) NOT NULL,
member_join_date DATETIME,
member_admin INT(1) UNSIGNED DEFAULT 3,
PRIMARY KEY( member_id )
)Type = MyISAM;
```

- وضعنا المفتاح الأساسي في نهاية التعريف و هذا صحيح .
- فكرة إختيار الصلاحية برقم وحيد لأن مستخدم يقوم بالتسجيل في منتدى يتم ترتيبه كالآتي الأعضاء يأخذون الرقم 3 ، المشرفون يأخذون الرقم 2 ، مدير الموقع الرقم 1 و هكذا كلها مجرد أفكار ☺

تحديد قاعدة البيانات :

قبل أن تقوم بعمل أي إصافة لجداول أو لبيانات أو لإتمام أي عملية على قاعدة البيانات لابد من اختيارها أولا تتم عملية الإختيار بواسطة الأمر USE :

```
USE DB_name;
```

و DB_name هو اسم قاعدة البيانات التي سنستعملها .

```
mysql> USE company_db;
Database changed
mysql> CREATE Table employees(
    -> emp_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    -> emp_name VARCHAR(50) NOT NULL,
    -> emp_date DATETIME NOT NULL,
    -> emp_phone INT(11) UNSIGNED
    -> )Type = MyISAM;
Query OK, 0 rows affected, 1 warning (0.16 sec)
```

ملحوظة : لكتابة عدة سطور متتالية نضغط Enter بدون وضع الفاصلة المنقوطة في نهاية السطر .

```
mysql> CREATE DataBase forum;
Query OK, 1 row affected (0.02 sec)

mysql> USE forum;
Database changed
mysql> CREATE  Table posts(
    -> post_id  INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    -> post_title  VARCHAR(255) NOT NULL DEFAULT 'without title',
    -> post_body  LONGTEXT,
    -> post_date  DATETIME,
    -> post_readers  INT UNSIGNED,
    -> post_rank  INT UNSIGNED
    -> )Type = MyISAM;
Query OK, 0 rows affected, 1 warning (0.14 sec)
```

قمنا بإنشاء قاعدة بيانات forum و قمنا بتحديدها لاستعمالها و أخيرا قمنا بإضافة جدول جديد و سنكر نفس الموضوع مع جدولنا الثالث.

```
mysql> USE forum;
Database changed
mysql> CREATE  Table members(
    -> member_id  INT UNSIGNED AUTO_INCREMENT,
    -> member_name  VARCHAR(255) NOT NULL,
    -> member_nickname  VARCHAR(255),
    -> member_password  TEXT,
    -> member_email  VARCHAR(255) NOT NULL,
    -> member_join_date  DATETIME,
    -> member_admin  INT(1) UNSIGNED  DEFAULT 3,
    -> PRIMARY KEY(member_id)
    -> )Type = MyISAM;
Query OK, 0 rows affected, 1 warning (0.19 sec)
```

استعراض قواعد البيانات :

لمعرفة جميع قواعد البيانات التي قمنا بإنشاءها و لم نحذفها :

```
SHOW DATABASES;
```

و لمعرفة جميع الجداول في قاعدة بيانات معينة ((لا تنسى تحديد قاعدة البيانات أولا بالأمر USE)) :

```
SHOW TABLES;
```

و لمعرفة جميع الحقول في جدول معين :

```
SHOW COLUMNS FROM TB_name;
```

```
mysql> SHOW DataBases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| [illegible]        |
| company_db         |
| [illegible]        |
| forum              |
| [illegible]        |
| [illegible]        |
| movie_store        |
| mysql              |
| phpmyadmin         |
| [illegible]        |
| [illegible]        |
| test               |
| [illegible]        |
| [illegible]        |
+--------------------+
15 rows in set (0.08 sec)
```

```
mysql> USE forum;
Database changed
mysql> SHOW Tables;
+-----------------+
| Tables_in_forum |
+-----------------+
| members         |
| posts           |
+-----------------+
2 rows in set (0.00 sec)
```

```
mysql> SHOW Columns FROM members;
+-----------------+------------------+------+-----+---------+----------------+
| Field           | Type             | Null | Key | Default | Extra          |
+-----------------+------------------+------+-----+---------+----------------+
| member_id       | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| member_name     | varchar(255)     | NO   |     |         |                |
| member_nickname | varchar(255)     | YES  |     | NULL    |                |
| member_password | text             | NO   |     |         |                |
| member_email    | varchar(255)     | NO   |     |         |                |
| member_join_date| datetime         | YES  |     | NULL    |                |
| member_admin    | int(1) unsigned  | YES  |     | 3       |                |
| member_rank     | int(11)          | YES  |     | NULL    |                |
+-----------------+------------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

تعديل الجداول :

لإعادة تسمية جدول :

```
ALTER TABLE TB_name RENAME TB_new_name;
```

حيث : TB_name هو الاسم القديم و TB_new_name هو الاسم الجديد .

لإضافة حقل جديد للجدول :

```
ALTER TABLE TB_name ADD COLUMN col_name data_type attr1 attr2 ……;
```

حيث : …… col_name data_type attr1 attr2 هو تعريف الحقول العادي الذي تعلمنها في إنشاء الجداول .

بشكل عام أي حقل يضاف في نهاية الجدول لكن إذا كنت تفضل إضافة الحقل في بداية الجدول نضع FIRST في نهاية السطر :

```
ALTER TABLE TB_name ADD COLUMN col_name data_type attr1 attr2 …… FIRST;
```

و لإضافته بعد حقل آخر نستعمل AFTER :

```
ALTER TABLE TB_name ADD COLUMN col_name data_type attr1 attr2 …… AFTER anthor_col
```

حيث : anthor_col هو الحقل الذي سنضيفه بعده .

لحذف حقل من جدول :

```
ALTER TABLE TB_name DROP COLUMN col_name;
```

حيث : col_name هو اسم الحقل الذي نريد حذفه .

لإعادة تسمية حقل :

```
ALTER TABLE TB_name CHANGE COLUMN old_col new_col data_type attr1 attr2 ……;
```

حيث : old_col هو اسم الحقل القديم و new_col هو الاسم الجديد و لابد من كتابة كامل التعريف بعد الاسم الجديد .

لتعديل خصائص أو نوع بيانات حقل :

```
ALTER TABLE TB_name MODIFY col_name new_data_type new_attr1 new_attr2 ……;
```

```
mysql> USE forum;
Database changed
mysql> ALTER Table members ADD Column member_rank INT;
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER Table members ADD Column member_followers INT AFTER member_id;
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER Table members DROP Column member_followers;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> USE company_db;
Database changed
mysql> ALTER Table employees RENAME emp;
Query OK, 0 rows affected (0.09 sec)
```

حذف قاعدة بيانات :

```
DROP DATABASE DB_name;
```

حيث : DB_name هو اسم قاعدة البيانات التي نريد حذفها .

حذف جدول :

```
DROP TABLE TB_name ;
```

حيث : TB_name هو اسم الجدول الذي نريد حذفه ، و لا تنسى تحديد قاعدة البيانات الذي يوجد بها .

```
mysql> USE company_db;
Database changed
mysql> DROP Table emp;
Query OK, 0 rows affected (0.01 sec)

mysql> DROP DataBase company_db;
Query OK, 0 rows affected (0.09 sec)
```

طبعا يمكنك التحقق عن طريق الأمر SHOW .

الفصل الثالث / التعامل مع DML من إضافة و حذف و تعديل للبيانات :

درسنا في الفصل السابق كيفية التعامل مع قواعد البيانات نفسها ، في هذا الفصل سنتعرف على مجموعة الأوامر الخاصة بالتعامل مع البيانات .

إضافة البيانات لجدول :

```
INSERT INTO TB_name VALUES (v1 , v2 , ..... , vN);
```

INSERT INTO - أمر الإضافة داخل الجدول .

TB_name - اسم الجدول .

VALUES(v1 , v2 , ... , vN) - القيم التي سيتم إضافتها للحقول و يجب أن تضع ترتيب القيم حسب ترتيب الحقول و تذكر أنه لا يمكنك ترك الحقول ذات الخاصية NOT NULL فارغة و إلا ستظهر رسالة تحذير و يعيب هذه الطريقة أنه لابد من إضافة قيمة المفتاح الأساسي بشكل يدوي و هو أمر متعب .

إضافة البيانات لحقول معينة في الجدول :

كل ما سنفعله هو إضافة أسماء الحقول بعد اسم الجدول .

```
INSERT INTO TB_name (col1 , col2 , ..... , colN) VALUES (v1 , v2 , ..... , vN);
```

ننتقل للأمثلة :

```
INSERT INTO posts VALUES (0 , "programming" , "programming is ..." , NOW( ) , 0 , 0);
```

الدالة NOW( ) تستعمل للحصول على الوقت و التاريخ الحالي .

```
INSERT INTO members

(member_name , member_password , member_email)

VALUES

( "Ahmed" , "12345", "ahmed@example.com" );
```

في هذه الطريقة لا تهتم بالمفتاح الأساسي لأن MySQL تضيف قيمته لكل صف جديد تلقائيا .

```
mysql> INSERT INTO posts VALUES(0, "programming", "programming is ..", NOW(), 0
, 0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO  members
    -> (member_name , member_password , member_email)
    ->  VALUES
    -> ("Ahmed", "12345", "ahmed@example.com");
Query OK, 1 row affected (0.00 sec)
```

تعديل بيانات جدول :

```
UPDATE TB_name

SET col1 = v1 , col2 = v2 , ..... , colN = vN

WHERE column = value;
```

UPDATE - أمر التعديل .

TB_name - اسم الجدول .

SET col1 = v1 – نضع القيمة الجديدة للحقل و نفصل بـ , بين الحقول .

WHERE column = value; - كل سجل له كما نعلم معرف خاص به هو المفتاح الأساسي ، نستعمله غالبا في كامل عمليات البحث و هو دائما يبدأ من الصفر و بهذا أنت تحدد بالضبط السجل الذي سيتم تعديل حقوله ، لكن إذا كنت تريد استعمال أي حقل آخر فلا مشكلة ☺ كل الخيارات متاحة .

```
UPDATE members

SET member_password = "6789" , member_email = "ahmed2@abc.com"

WHERE member_id = 0;
```

أو هكذا :

```
UPDATE members

SET member_password = "6789" , member_email = "ahmed2@abc.com"

WHERE member_name = "Ahmed";
```

```
mysql> UPDATE  members
    -> SET  member_password = "6789" , member_email = "ahmed2@abc.com"
    -> WHERE member_id = 0;
Query OK, 0 rows affected (0.39 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> UPDATE  members
    -> SET  member_password = "6789" , member_email = "ahmed2@abc.com"
    -> WHERE member_name = "Ahmed";
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

حذف بيانات من جدول :

لحذف جميع البيانات من جدول :

```
DELETE FROM TB_name ;

أو

DELETE * FROM TB_name ;
```

لحذف سجل وحيد :

```
DELETE FROM TB_name WHERE col = val;
```

حيث WHERE col = val هو id السجل .

مثال على ذلك :

```
DELETE FROM members WHERE member_id = 0;
```

```
mysql> DELETE FROM members WHERE member_id=0;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM members;
Query OK, 2 rows affected (0.00 sec)
```

الفصل الرابع / التعامل مع DQL من البحث و توابعه :

قبل أن نشرع في التعامل مع البحث و أصدقائه ☺ ، سنقوم بملئ الجدولين posts و members بأي قيم المهم أن يحتوي كل جدول على مجموعة من الصفوف و لا ضرر في تكرار الصف مرتين .

عرض حقل معين من الجدول :

```
SELECT col_name FROM TB_name;
```

SELECT - أشهر أمر في MySQL ((تقريبا المبرمجين يعرفوه و لا يعرفوا MySQL☺)) و هو يختص بالبحث عن السجلات سواء في حقل واحد أو في كامل الجدول مع الكثير من الإضافات التي تقوي عملية البحث سنتعرف عليها بعد قليل .

col_name - اسم الحقل الذي نريد عرض سجلاته ، و طبعا ممكن عدد من الحقول يفصل بينهم علامة " , " .

FROM - لتحديد الجدول أو النطاق الذي سنبحث فيه .

TB_name - اسم الجدول الذي نتعامل معه .

فمثلا إذا أردنا عرض member_name من جدول members يكون الاستعلام كالآتي :

```
SELECT member_name FROM members;
```

و لعرض كامل الحقول في جدول :

```
SELECT * FROM TB_name;
```

```
mysql> SELECT  member_name FROM  members;
+-------------+
| member_name |
+-------------+
| Ali         |
| Omar        |
| Khaled      |
| Ali         |
+-------------+
4 rows in set (0.00 sec)
```

إزالة التكرار من عملية البحث :

```
SELECT DISTINCT col_name FROM TB_name;
```

DISTINCT - هي المسؤولة عن إزالة التكرار من عملية البحث ففي المثال السابق كان لدينا عضوين بالاسم Ali نعدل استعلامنا كالآتي :

```
SELECT DISTINCT member_name FROM members;
```

و ها هو الناتج :

```
mysql> SELECT DISTINCT member_name FROM members;
+--------------+
| member_name  |
+--------------+
| Ali          |
| Omar         |
| Khaled       |
+--------------+
3 rows in set (0.08 sec)
```

البحث بقيم معينة :

```
SELECT * FROM TB_name WHERE col = val;
```

WHERE col = val - هنا يبحث عن تطابق فإن حدث سيعودبناتج عملية البحث سواء أكان حقل واحد أو مجموعة حقول .

فمثلا في جدول posts :

```
SELECT * FROM posts WHERE post_id = 2;
```

```
SELECT post_id FROM posts WHERE post_title = "programming";
```

```
SELECT post_date , post_body FROM posts WHERE post_reader = 0;
```

و لا يشترط أن تعود عملية البحث بناتج واحد ، فقد تعود بألف سجل على حسب تحقق الشرط .

```
mysql> SELECT  *   FROM  posts  WHERE post_id = 2;
+---------+-------------+--------------------+---------------------+-------------
| post_id | post_title  | post_body          | post_date           | post_readers
+---------+-------------+--------------------+---------------------+-------------
|       2 | programming | programming is .. | 2016-07-08 08:25:47 |            0
+---------+-------------+--------------------+---------------------+-------------
1 row in set (0.00 sec)

mysql> SELECT post_id FROM  posts WHERE post_title = "programming";
+---------+
| post_id |
+---------+
|       1 |
|       2 |
|       3 |
|       4 |
|       5 |
+---------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT  post_date , post_body  FROM  posts  WHERE post_readers = 0;
+---------------------+--------------------+
| post_date           | post_body          |
+---------------------+--------------------+
| 2016-07-08 08:15:01 | programming is ?  |
| 2016-07-08 08:25:47 | programming is .. |
| 2016-07-09 08:47:28 | programming is ?  |
| 2016-07-09 08:47:44 | programming is ?  |
| 2016-07-09 08:48:53 | programming is .. |
+---------------------+--------------------+
5 rows in set (0.00 sec)
```

و في حالة عدم وجود نتيجة تطابق المطلوب :

```
mysql> SELECT  *   FROM  posts  WHERE post_id = 20;
Empty set (0.00 sec)
```

المعامل LIKe :

يتيح المعامل like البحث بواسطة جزء معين من النص مثلا للبحث عن جميع الصفوف التي تحتوي على حقل يبدأ بحرف a :

```
SELECT * FROM TB_name WHERE col_name LIKE "a%" ;
```

يحتوي على الحرف a :

```
SELECT * FROM TB_name WHERE col_name LIKE "%a%" ;
```

ينتهي بالحرف a :

```
SELECT * FROM TB_name WHERE col_name LIKE "%a" ;
```

طبعا يمكنك استبدال الحرف a بنص كامل .

جميع الصفوف في جدول members التي تبدأ بالحرف A :

```
SELECT member_id , member_name , member_email FROM members

WHERE member_name LIKE "A%" ;
```

```
mysql> SELECT  member_id , member_name  , member_email  FROM  members  WHERE  me
mber_name LIKE "A%" ;
+-----------+-------------+--------------------+
| member_id | member_name | member_email       |
+-----------+-------------+--------------------+
|         3 | Ali         | ahmed@example.com |
|         6 | Ali         | ahmed@example.com |
+-----------+-------------+--------------------+
2 rows in set (0.00 sec)
```

المعامل ORDER BY :

و ظيفة هذا المعامل تغيير ترتيب عرض النتائج و فق ترتيب حقل معين :

```
SELECT * FROM TB_name ORDER BY col_name ;
```

و لإظهار نتائج البحث معكوسة من تصاعديا فقط نضيف DESC في نهاية الاستعلام :

```
SELECT * FROM TB_name ORDER BY col_name DESC;
```

يوجد أيضا المعامل ASC لعكس العكس ☺ و هو غير منطقي أصلا ، لكن ربما تحتاجه في أحد مشاريعك .

طبعا يمكن استعمال المعامل WHERE و LIKE و أي معامل آخر سنتعلمه .

و ها مثال على جدول posts :

```
SELECT post_id , post_title FROM posts  ORDER BY post_id DESC;
```

```
mysql> SELECT post_id , post_title FROM posts ORDER BY post_id DESC;
+---------+--------------+
| post_id | post_title   |
+---------+--------------+
|       5 | java         |
|       4 | html         |
|       3 | php          |
|       2 | javascript   |
|       1 | programming  |
+---------+--------------+
5 rows in set (0.00 sec)
```

المعاملين AND و OR :

```
SELECT * FROM TB_name WHERE col_name1 = val1 AND col_name2 = val2 AND ……;
```

```
SELECT * FROM TB_name WHERE col_name1 = val1 OR col_name2 = val2 OR ……;
```

طبعا لا تحتاج للشرح فالمعامل AND لن يعود بالقيمة إلا عندما تتحقق كامل الشروط ، أما OR يعود بالقيمة بمجرد أن يتحقق أحد الشروط .

مثلا لنبحث في جدول posts عن id الموضوع الذي يبدأ بحرف j و ينتهي بحرف t :

```
SELECT post_id FROM posts WHERE post_title LIKE "j%" AND post_title LIKE "%t" ;
```

و ها هو الناتج فعلا 2 (يمكنك مراجعة المثال السابق فهو javascript) :

```
mysql> SELECT post_id FROM posts WHERE post_title LIKE "j%" AND post_titl
e LIKE "%t" ;
+---------+
| post_id |
+---------+
|       2 |
+---------+
1 row in set (0.00 sec)
```

و هذا مثال على OR :

```
mysql> SELECT member_id , member_name FROM members WHERE member_name LIKE "A%"
OR member_name = "Omar" ORDER BY member_id DESC ;
+-----------+-------------+
| member_id | member_name |
+-----------+-------------+
|         7 | Ahmed       |
|         6 | Ali         |
|         4 | Omar        |
|         3 | Ali         |
+-----------+-------------+
4 rows in set (0.00 sec)
```

المعامل IN :

```
SELECT * FROM TB_name WHERE col_name IN ( v1 , v2 , ..... );
```

يستعمل المعامل IN تماما مثل swich case في لغات البرمجة ، حيث أنه يقوم بالاختيار من عدة قيم بمعنى في المثال التالي :

```
SELECT member_id , member_name FROM members

WHERE member_name = "Ahmed" OR member_name = "Ali" OR member_name = "omar" ;
```

طالما member_name بين مجموعة من الاحتمالات لذا يمكن كتابته كالتالي :

```
SELECT member_id , member_name FROM members

WHERE member_name IN ("Ali" , "Omar" , "Ahmed");
```

```
mysql> SELECT  member_id , member_name  FROM  members
    -> WHERE  member_name  IN ("Ali" , "Omar" , "Ahmed");
+-----------+-------------+
| member_id | member_name |
+-----------+-------------+
|         3 | Ali         |
|         4 | Omar        |
|         6 | Ali         |
|         7 | Ahmed       |
+-----------+-------------+
4 rows in set (0.00 sec)
```

المعامل BETWEEN .... AND :

نشاهده بكثرة مع التواريخ فهو يبحث داخل نطاق معين :

```
SELECT * FROM TB_name WHERE col_name BETWEEN v1 AND v2;
```

على فرض أنه في جدول المواضيع لدينا هذه التواريخ :

```
+---------------------+
| post_date           |
+---------------------+
| 2016-06-24 00:00:00 |
| 2016-06-26 00:00:00 |
| 2016-06-28 00:00:00 |
| 2016-06-30 00:00:00 |
| 2016-06-30 00:00:00 |
+---------------------+
```

نريد طباعة عناوين المواضيع التي بين يوم 28 و يوم 30 :

```
SELECT post_title , post_date FROM posts

WHERE post_date BETWEEN “2016-6-28” AND “2016-6-30”;
```

```
mysql> SELECT  post_title , post_date  FROM  posts
    -> WHERE  post_date  BETWEEN "2016-6-28" AND "2016-6-30";
+------------+---------------------+
| post_title | post_date           |
+------------+---------------------+
| php        | 2016-06-28 00:00:00 |
| html       | 2016-06-30 00:00:00 |
| java       | 2016-06-30 00:00:00 |
+------------+---------------------+
3 rows in set (0.06 sec)
```

أيضا يمكنك استعمال NOT قبل BETWEEN لنحصل على النتائج خارج نطاق معين فيمكن تعديل المثال السابق ليكون :

```
mysql> SELECT  post_title , post_date  FROM  posts
    -> WHERE  post_date NOT BETWEEN "2016-6-28" AND "2016-6-30";
+-------------+---------------------+
| post_title  | post_date           |
+-------------+---------------------+
| programming | 2016-06-24 00:00:00 |
| javascript  | 2016-06-26 00:00:00 |
+-------------+---------------------+
2 rows in set (0.00 sec)
```

ملحوظة : مع التاريخ و الوقت نكتبهما بالترتيب من اليسار لليمين-> “ year-month-day hours-minutes-seconds “ .

المعامل LIMIT :

للتحكم في عدد السجلات المرجعة من عملية البحث نستعمل LIMIT و لها صيغتين :

الأولى / أن نمرر عدد السجلات مباشرة :

```
SELECT * FROM TB_name LIMIT n:
```

الثانية / أن نمرر رقم السجل الذي نبدأ منه العد متبوعا بعدد السجلات التي نريدها :

```
SELECT * FROM TB_name LIMIT start , offset;
```

```
mysql> SELECT post_title FROM posts LIMIT 3;
+--------------+
| post_title   |
+--------------+
| programming  |
| javascript   |
| php          |
+--------------+
3 rows in set (0.00 sec)

mysql> SELECT post_title FROM posts LIMIT 1 ,  4;
+-------------+
| post_title  |
+-------------+
| javascript  |
| php         |
| html        |
| java        |
+-------------+
4 rows in set (0.00 sec)
```

البحث بين أكثر من جدول :

و يقصد به غالبا الربط بين الجداول :

```
SELECT col_name

 FROM table1 , table2

WHERE table1.col1 = table2.col2;
```

حيث col1 و col2 أحدهما مفتاح أساسي كما درسنا سابقا فوظيفة المفاتيح الأساسية الربط .

سنقوم بإضافة حقل جديد و ليكن id العضو الذي أضاف الموضوع ليكن post_writer_id و نضع فيه قيم أي عضو نختاره :

```
+-----------------+
| post_writer_id  |
+-----------------+
|               7 |
|               6 |
|               5 |
|               7 |
|               6 |
+-----------------+
```

```
SELECT post_title , member_name

FROM posts , members

WHERE posts.post_writer_id = members.member_id;
```

```
mysql> SELECT  post_title , member_name
    -> FROM  posts , members
    -> WHERE  posts.post_writer_id = members.member_id;
+--------------+--------------+
| post_title   | member_name  |
+--------------+--------------+
| programming  | Ahmed        |
| javascript   | Ali          |
| php          | Khaled       |
| html         | Ahmed        |
| java         | Ali          |
+--------------+--------------+
5 rows in set (0.00 sec)
```

دمج جدولين :

نفس ما فعلناه في المثال السابق لكن بصيغة مختلفة :

```
SELECT col_name

 FROM table1

INNER JOIN  table2

ON table1.col1 = table2.col2;
```

```
mysql> SELECT  post_title , member_name
    -> FROM posts
    -> INNER JOIN  members
    -> ON posts.post_writer_id = members.member_id;
+--------------+--------------+
| post_title   | member_name  |
+--------------+--------------+
| programming  | Ahmed        |
| javascript   | Ali          |
| php          | Khaled       |
| html         | Ahmed        |
| java         | Ali          |
+--------------+--------------+
5 rows in set (0.36 sec)
```

كذلك يوجد LEFT JOIN و RIGHT JOIN و هما يعيدان صفوف حتى لو لم يتحقق الشرط لكنها تضع مكان القيمة المفقودة NULL :

```
mysql> SELECT  post_title , member_name
    -> FROM posts
    -> RIGHT JOIN  members
    -> ON  members.member_id = posts.post_writer_id;
+-------------+-------------+
| post_title  | member_name |
+-------------+-------------+
| NULL        | Ali         |
| NULL        | Omar        |
| php         | Khaled      |
| javascript  | Ali         |
| java        | Ali         |
| programming | Ahmed       |
| html        | Ahmed       |
+-------------+-------------+
7 rows in set (0.00 sec)
```

إتحاد الجداول :

لدمج أكثر من عملية بحث في ناتج واحد نستعمل المعامل UNION بالصيغة التالية :

```
SELECT * FROM table1

UNION

SELECT * FROM table2

UNION

SELECT * FROM table3;
```

المهم أن يكون كل جدول من الثلاث يحتوي على نفس عدد الأعمدة حتى لو اختلفت نوع البيانات التي يخزنها كل عمود عن الآخر و في حالة كان هناك حقل ناقص نضع بدلا عنه علامتي تنصيص فارغتين :

```
SELECT col1 , col2 , col3  FROM table1

UNION

SELECT col1 , " " , col3  FROM table2;
```

سنقوم بكتابة بكتابة استعلام يجمع id العضو و اسمه و تاريخ انضمامه مع id الموضوع و عنوانه و تاريخ اضافته .

```
SELECT post_id , post_title , post_date  FROM  posts

UNION

SELECT member_id , member_name , member_join_date   FROM  members;
```

```
mysql> SELECT  post_id , post_title , post_date FROM posts
    -> UNION
    -> SELECT  member_id , member_name , member_join_date FROM members;
+---------+-------------+---------------------+
| post_id | post_title  | post_date           |
+---------+-------------+---------------------+
|       1 | programming | 2016-06-24 00:00:00 |
|       2 | javascript  | 2016-06-26 00:00:00 |
|       3 | php         | 2016-06-28 00:00:00 |
|       4 | html        | 2016-06-30 00:00:00 |
|       5 | java        | 2016-06-30 00:00:00 |
|       3 | Ali         | 2016-07-02 00:00:00 |
|       4 | Omar        | 2016-07-01 00:00:00 |
|       5 | Khaled      | 2016-07-02 00:00:00 |
|       6 | Ali         | 2016-07-03 00:00:00 |
|       7 | Ahmed       | 2016-07-01 00:00:00 |
+---------+-------------+---------------------+
10 rows in set (0.00 sec)
```

طبعا في الواقع أنت لن تحتاج لفعل شيء كهذا ☺ لكن هذا مجرد مثال للتوضيح ليس أكثر .

و في حالة تركك لأحد الحقول :

```
mysql> SELECT  post_id , post_title , post_date FROM posts
    -> UNION
    -> SELECT  member_id , "" , member_password FROM members;
+---------+-------------+---------------------+
| post_id | post_title  | post_date           |
+---------+-------------+---------------------+
|       1 | programming | 2016-06-24 00:00:00 |
|       2 | javascript  | 2016-06-26 00:00:00 |
|       3 | php         | 2016-06-28 00:00:00 |
|       4 | html        | 2016-06-30 00:00:00 |
|       5 | java        | 2016-06-30 00:00:00 |
|       3 |             | 12345               |
|       4 |             | 12345               |
|       5 |             | 12345               |
|       6 |             | 12345               |
|       7 |             | 12345               |
+---------+-------------+---------------------+
10 rows in set (0.01 sec)
```

إلى هنا نكون قد وصلنا إلى نهاية الكتاب و لكن ليس لنهاية MySQL

قراءة هذا الكتاب أو أي كتاب آخر لن تنفعكم يجب أن تحاولوا بأيديكم

و تجربوا كل كلمة موجودة في الكتاب ، حاولوا دائما دمج ما تعلمتومه

في مشاريعكم ، ففي المشاريع تواجه المشاكل و بحل المشاكل

تتولد الخبرة و بكثرة الخبرة نصل للإحتراف

في النهاية أتمنى أن تكونوا قد استفدتم

و إلى اللقاء في كتاب آخر ☺

لا تنسوا مراسلتي على :

modi401@hotmail.com

أو

https://www.facebook.com/mohamed.yossef.583

أو تفضلوا بزيارة موقعي الإلكتروني :

www.agashe.pe.hu