Introduction to MySQL

COMP-3077 Web-Based Data Management

Dr. Saja Al-Mamoori

Introduction

- MySQL is one of the **most popular** database management system for web servers.
- Developed in the **mid-1990s**, it’s now a mature technology that powers many of today’s most-visited Internet destinations.
- One reason for its success must be the fact that, like PHP, it’s free to use.
- But it’s also extremely powerful and exceptionally fast—it can run on even the most basic of hardware.
- It hardly puts a burden on system resources. MySQL is also highly scalable, which means that it can grow with your website.

MySQL Basics

- A database is a **structured collection of records** or data stored in a computer system and organized in such a way that it can be quickly searched and information can be rapidly retrieved.
- The SQL in MySQL stands for **Structured Query Language**.
- This language is loosely based on English and also used in other databases such as Oracle and Microsoft SQL Server.
- It is designed to allow simple requests from a database via commands such as:

```
SELECT title FROM publications WHERE author = 'Charles Dickens';
```

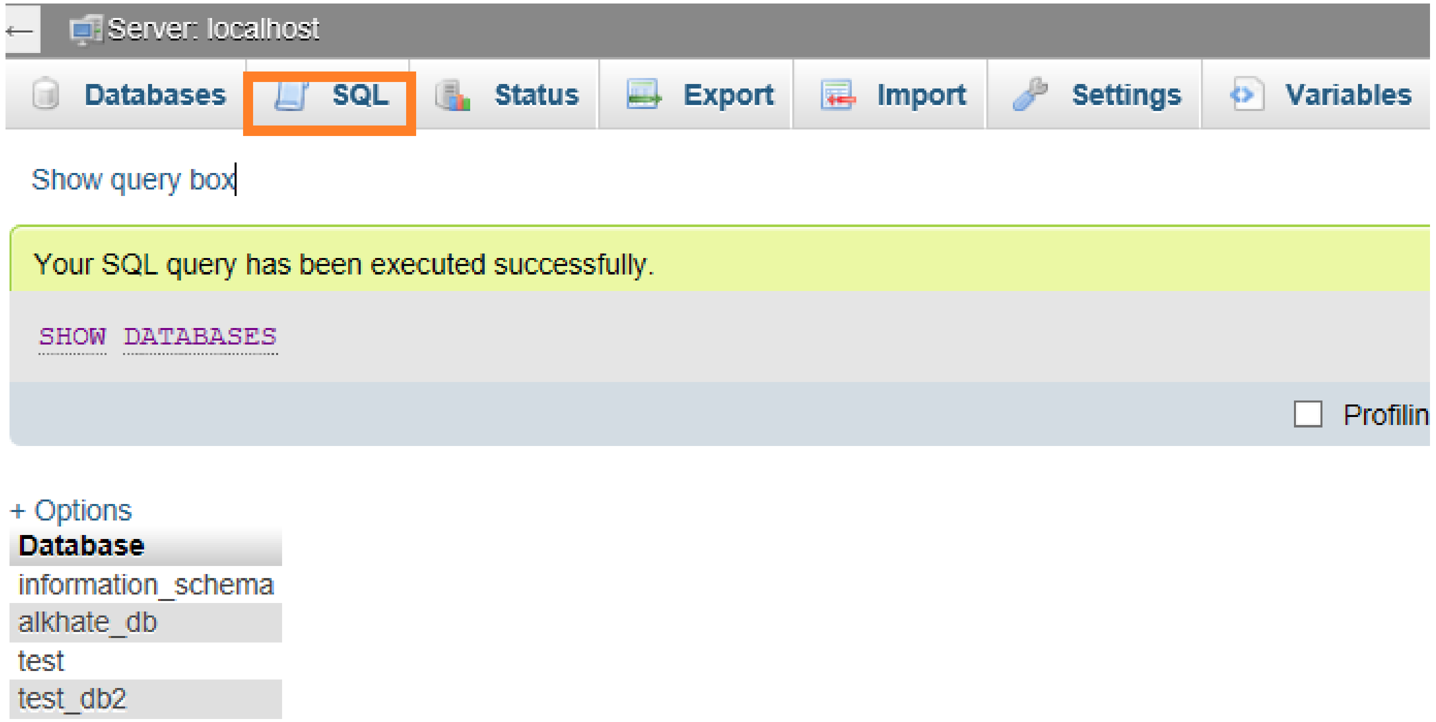
Summary of Database Terms

- ***Database:*** The overall container for a collection of MySQL data
- ***Table:*** A subcontainer within a database that stores the actual data
- ***Row:*** A single record within a table, which may contain several fields
- ***Column:*** The name of a field within a row

Example- publications table

Author	Title	Type	Year
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
Charles Darwin	The Origin of Species	Non-Fiction	1856
Charles Dickens	The Old Curiosity Shop	Fiction	1841
William Shakespeare	Romeo and Juliet	Play	1594

Show databases



Your SQL query has been executed successfully.

```
SHOW DATABASES
```

Database
information_schema
alkhate_db
test
test_db2

MySQL Commands

Command	Action
ALTER	Alter a database or table
BACKUP	Backup a table \c
CREATE	Create a database
DELETE	Delete a row from a table
DESCRIBE	Describe a table’s columns
DROP	Delete a database or table
GRANT	Change user privileges
INSERT	Insert data

MySQL Commands

Command	Action
LOCK	Lock table(s)
RENAME	Rename a table
SHOW	List details about an object
SOURCE	Execute a file
TRUNCATE	Empty a table
UNLOCK	Unlock table(s)
UPDATE	Update an existing record

MySQL Commands

- SQL commands and keywords are **case-insensitive**. CREATE, create, and CrEaTe all mean the same thing. However, for the sake of clarity, the recommended style is to use **uppercase**.
- Table names are **case-sensitive on Linux and OS X**.
- but **case-insensitive on Windows**.
- So for portability purposes, you should always choose a case and stick to it. The **recommended style is to use lowercase** for tables.

Creating a database

- create a new database called publications using SQL tab in phpMyAdmin:

  CREATE DATABASE uwinid_pbl;

  ***pbl is abbreviation of publications

- for some administrations setup, on myweb you cannot create, remove databases or users using script, creating new database or user you need to follow the instruction on:

  http://site-helper.com/mysql.html#create

The same steps in the Orientation slides, which you have followed earlier.

Creating a database

- Using script, now that you've created the database, you want to work with it, so issue the following:

```
USE uwinid_pbl;
```

- Now you are in uwinid_pbl database

Creating a user

- As you probably won’t want to grant your PHP scripts root access to MySQL; it could cause a real headache should you get hacked
- To create a user, issue the GRANT command, which takes the following form
- GRANT PRIVILEGES ON database.object TO 'username'@'hostname' IDENTIFIED BY 'password';
- This should be pretty straightforward, with the possible exception of the data base.object part, which refers to the database itself and the objects it contains, such as tables

Granting access to the created user

- Lets assume that we have created the following user:

```
GRANT PRIVILEGES ON database.object TO 'jim'@'localhost' IDENTIFIED BY 'mypasswd';
```

- We need to grant ‘jim’ an access to uwin_publications database.

```
GRANT ALL ON uwinid_pbl.* TO 'jim'@'localhost'  IDENTIFIED BY 'mypasswd';
```

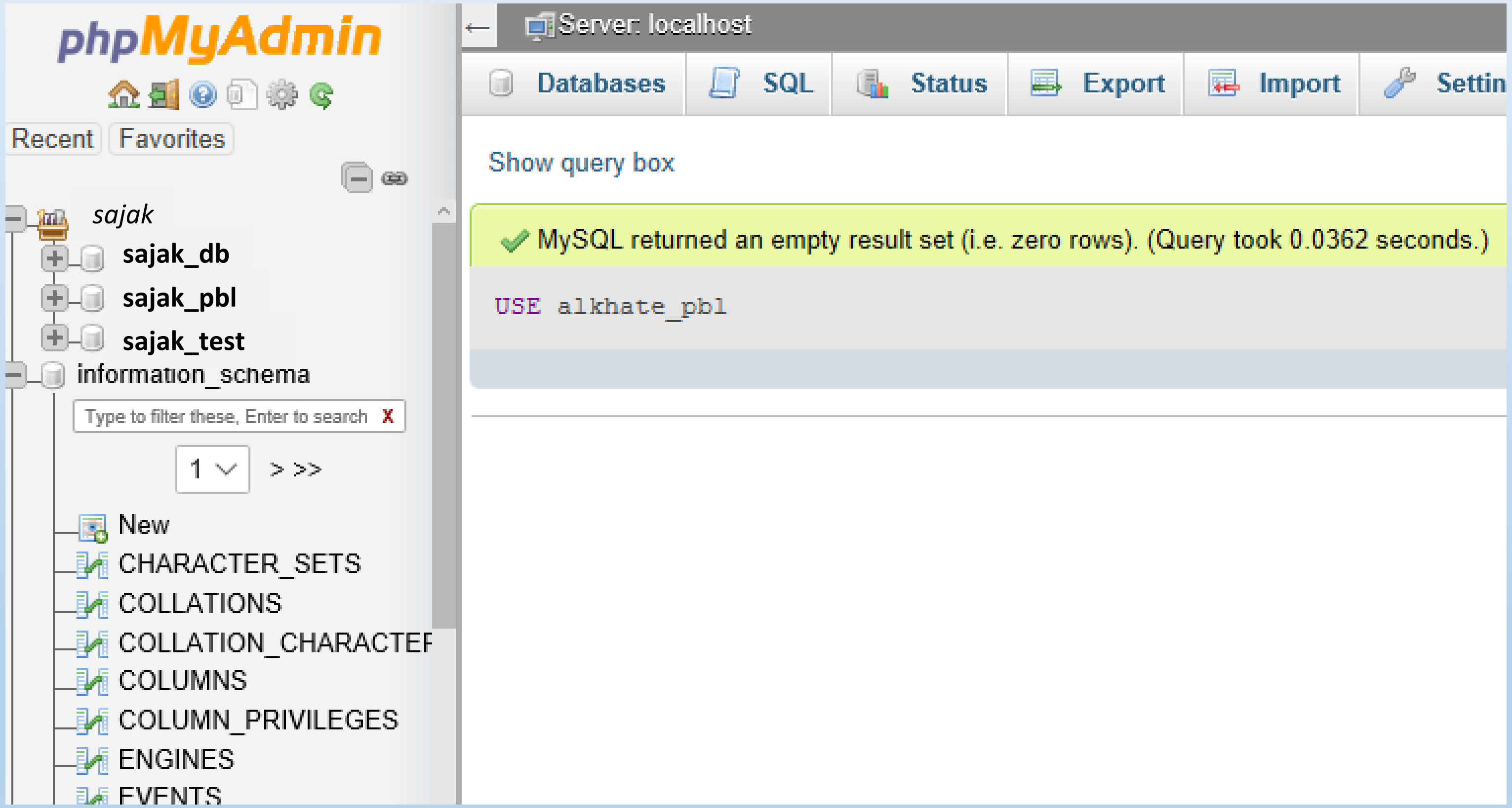
- Creating a user and grant it a privileges will be done automatically once you create a user using myweb→ MySQL databases → create a new database → create a new user

Creating a table

```
CREATE TABLE classics
( author VARCHAR(128),
title VARCHAR(128),
type VARCHAR(16),
year CHAR(4))
ENGINE MyISAM;
```

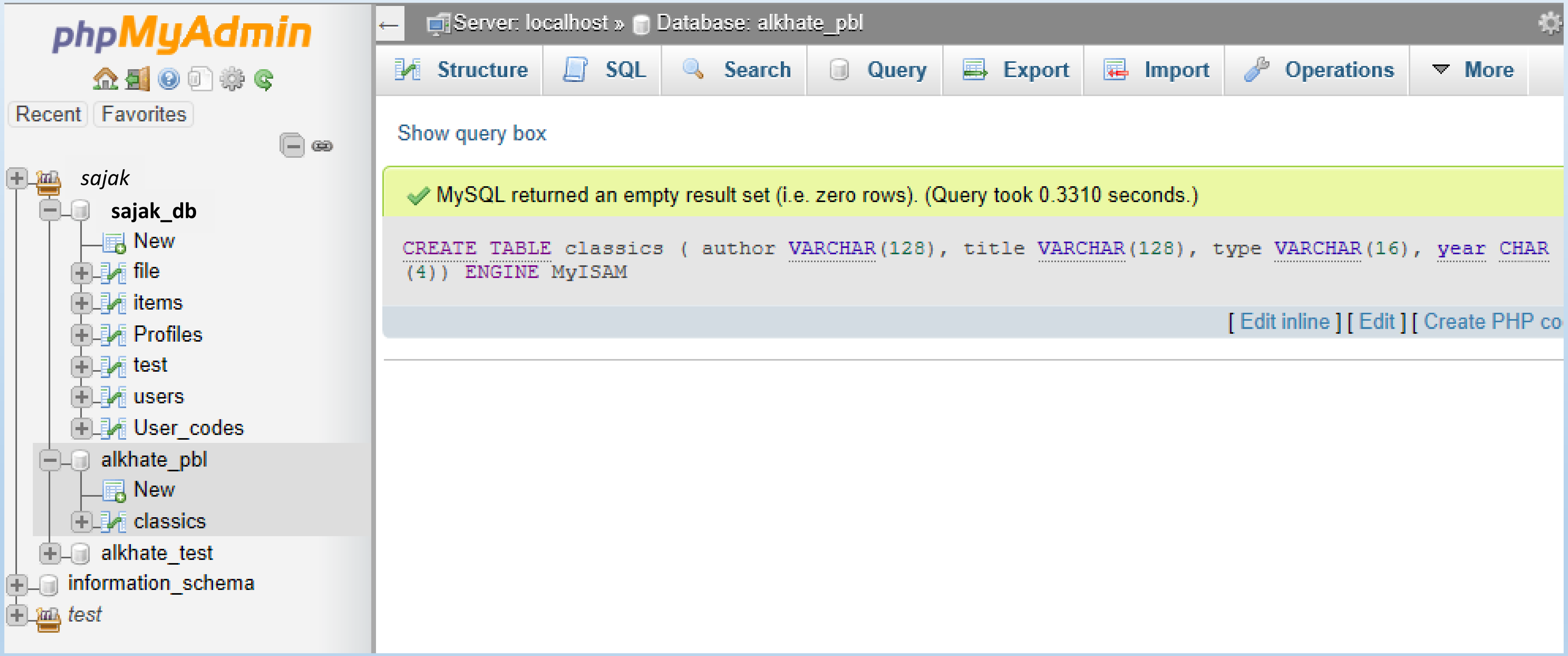
- ENGINE MyISAM tells MySQL the type of database engine to use for this table.

```
DESCRIBE classics;
```



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0362 seconds.)

```
USE alkhate_pbl
```



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.3310 seconds.)

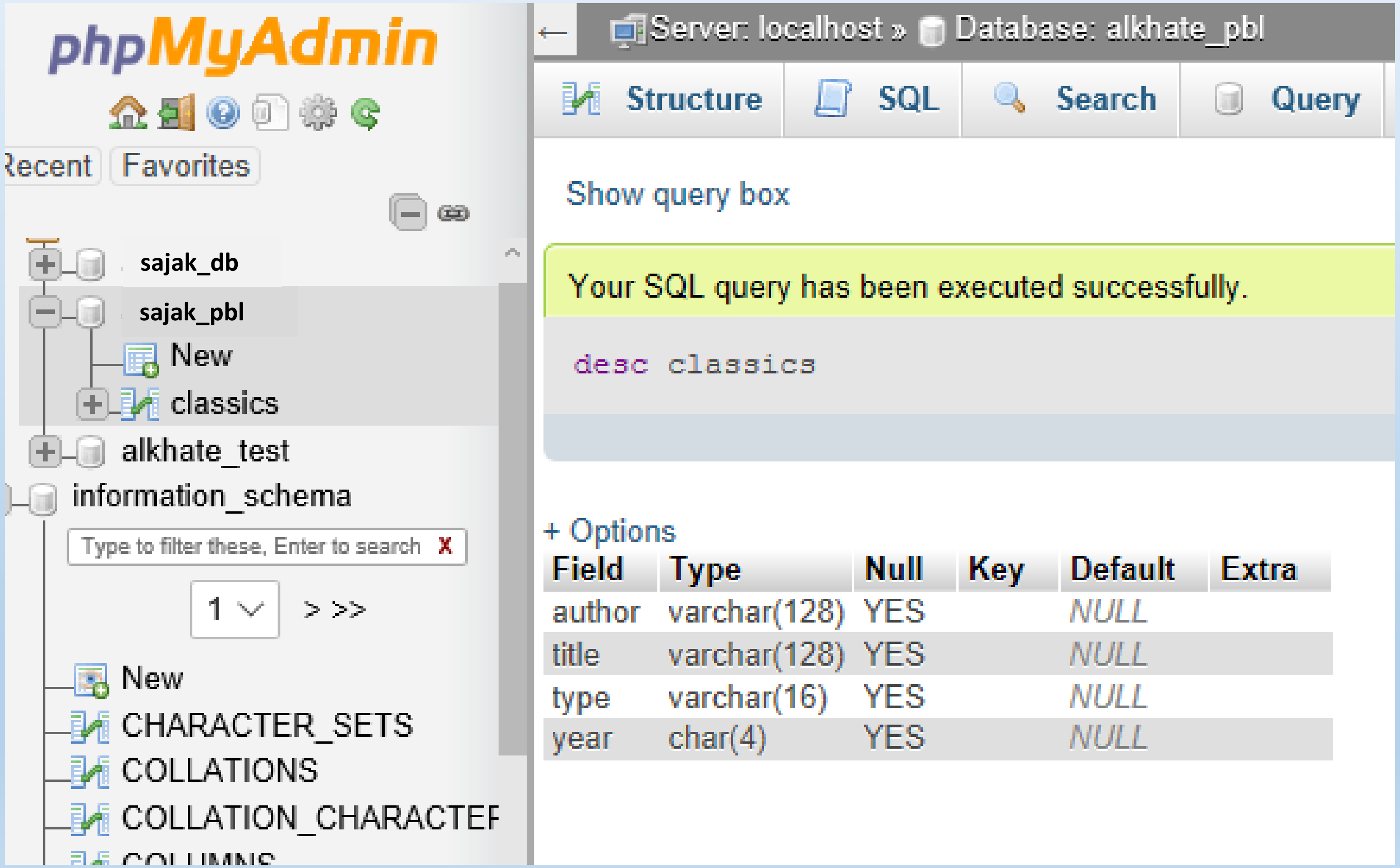
```
CREATE TABLE classics ( author VARCHAR(128), title VARCHAR(128), type VARCHAR(16), year CHAR
(4)) ENGINE MyISAM
```

[Edit inline] [Edit] [Create PHP co

We use DESCRIBE:

1- to check the table has been created CORRECTLY.

2- to remember the fields of the table, If they are needed for development or query



Your SQL query has been executed successfully.

```
desc classics
```

Field	Type	Null	Key	Default	Extra
author	varchar(128)	YES		*NULL*	
title	varchar(128)	YES		*NULL*	
type	varchar(16)	YES		*NULL*	
year	char(4)	YES		*NULL*	

Data Types - The CHAR data type

- All these types offer a parameter that sets the maximum (or exact) length of the string allowed in the field

Data type	Bytes used	Examples
CHAR(n)	Exactly n (<= 255)	CHAR(5)“Hello” uses 5 bytes CHAR(57)“Goodbye” uses 57 bytes
VARCHAR(n)	Up to n (<= 65535)	VARCHAR(7) “Morning” uses 7 bytes VARCHAR(100) “Night” uses 5 bytes

- CHAR more efficient when the length fixed or mostly fixed e.g year field
- VARCAR saves more spaces when the values in the columns vary.

The TEXT and VARCHAR data types

Data type	Bytes	used Attributes
TINYTEXT(n)	Up to n (<= 255)	Treated as a string with a character set
TEXT(n)	Up to n (<= 65535)	Treated as a string with a character set
MEDIUMTEXT(n)	Up to n (<= 1.67e+7)	Treated as a string with a character set
LONGTEXT(n)	Up to n (<= 4.29e+9)	Treated as a string with a character set

The TEXT and VARCHAR data types

- The differences between TEXT and VARCHAR are small:
  - ➢TEXT fields cannot have default values.
  - ➢MySQL indexes only the first n characters of a TEXT column (you specify n when you create the index).
- VARCHAR is the better and faster data type to use if you need to search the entire contents of a field.
- If you will never search more than a certain number of leading characters in a field, you should probably use a TEXT data type

Data Types - The BINARY data type

- The BINARY data type is used for storing strings of full bytes that do **NOT** have an **associated character set**.
- For example, you might use the BINARY data type to store a GIF image

Data type	Bytes used	Examples
BINARY(n) or BYTE(n)	Exactly n (<= 255)	As CHAR but contains binary data
VARBINARY(n)	Up to n (<= 65535)	As VARCHAR but contains binary data

The BLOB data type

- The term BLOB stands for Binary Large Object and, therefore, as you would think, the BLOB data type is most useful for binary data in ***excess*** of 65,536 bytes in size.
- The main other difference between the BLOB and BINARY data types is that BLOBs ***cannot*** have default values

Data type	Bytes	used Attributes
TINYBLOB(n)	Up to n (<= 255)	Treated as binary data—no character set
BLOB(n)	Up to n (<= 65535)	Treated as binary data—no character set
MEDIUMBLOB(n)	Up to n (<= 1.67e+7)	Treated as binary data—no character set
LONGBLOB(n)	Up to n (<= 4.29e+9)	Treated as binary data—no character set

Numeric data types

- MySQL supports various numeric data types from a single byte up to double precision floating-point numbers.
- Although the most memory that a numeric field can use up is 8 bytes, you are well advised to choose the smallest data type that will adequately handle the largest value you expect.
- Your databases will be small and quickly accessible.

Numeric data types

Data Type	Bytes used	Minimum value		Maximum	
		Signed	Unsigned	Signed	Unsigned
TINYINT	1	–128	0	127	255
SMALLINT	2	–32768	0	32767	65535
MEDIUMINT	3	–8.38e+6	0	8.38E+06	1.67E+07
INT/INTEGER	4	–2.15e+9	0	2.15E+09	4.29E+09
BIGINT	8	–9.22e+18	0	9.22E+18	1.84E+19
FLOAT	4	–3.40e+38	n/a	3.40E+38	n/a
DOUBLE/REAL	8	–1.80e+308	n/a	1.80e+308	n/a

Numeric data types

- To specify whether a data type is signed or unsigned, use the UNSIGNED qualifier, default is SIGNED.
- CREATE TABLE tablename (fieldname INT UNSIGNED);
- you can also pass an optional number as a parameter:
- CREATE TABLE tablename (fieldname INT(4));
- Unlike BINARY and CHAR data types, the size parameter does not indicate the number of bytes of storage to use.
- It actually represents is the **display width** of the data in the field when it is retrieved.

DATE and TIME

- The main remaining data types supported by MySQL relate to the date and time

Data type	Time/date format
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	'0000-00-00 00:00:00'
TIME	'00:00:00'
YEAR	0000 (Only years 0000 and 1901–2155)

DATE and TIME

- The DATETIME and TIMESTAMP data types display the same way.
- The main difference is that TIMESTAMP has a very narrow range (from the years 1970 through 2037),
- DATETIME will hold just about any date you’re likely to specify, unless you’re interested in ancient history or science fiction.
- TIMESTAMP is useful, however, because you can let MySQL set the value for you. If you don’t specify the value when adding a row, the current time is automatically inserted.

The AUTO_INCREMENT function

- Sometimes you need to ensure that every row in your database is guaranteed to be unique.
- ISBN (International Standard Book Number) as an example.

```
CREATE TABLE classics ( author VARCHAR(128),
 title VARCHAR(128),
type VARCHAR(16),
year CHAR(4),
 id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY) ENGINE MyISAM;
```

- An auto-increment column is useful as a key, because you will tend to search for rows based on this column.

Add column to a table

- Changing structure of a table needs ALTER TABLE command
- If you already have created classics table in slide 15, and you want to add ID
- You can do so as:

```
ALTER TABLE classics ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY;
```

Adding data to a table

- To add data to a table, use the INSERT command

```
INSERT INTO classics(author, title, type, year) VALUES('Mark Twain','The Adventures of Tom Sawyer','Fiction','1876');
INSERT INTO classics(author, title, type, year) VALUES('Jane Austen','Pride and Prejudice','Fiction','1811');
INSERT INTO classics(author, title, type, year) VALUES('Charles Darwin','The Origin of Species','Non-Fiction','1856');
INSERT INTO classics(author, title, type, year) VALUES('Charles Dickens','The Old Curiosity Shop','Fiction','1841');
INSERT INTO classics(author, title, type, year) VALUES('William Shakespeare','Romeo and Juliet','Play','1594');
```

Renaming a table

- like any other change to the structure or meta information about a table, is achieved via the ALTER command.

```
ALTER TABLE classics RENAME pre1900;
```

- If you tried that command, you should revert the table name by entering the following, so that later examples in this chapter will work as printed:

```
ALTER TABLE pre1900 RENAME classics;
```

Changing the data type of a column

- Changing a column's data type also makes use of the ALTER command, this time in conjunction with the MODIFY keyword.
- So to change the data type of column year from CHAR(4) to SMALLINT

```
ALTER TABLE classics MODIFY year SMALLINT;
```

- If the conversion of data type makes sense to MySQL, it will automatically change the data while keeping the meaning

Adding a new column

- Here's how to add the new column pages, which will be used to store the number of pages in a publication:

```
ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
```

- UNSIGNED SMALLINT data type, sufficient to hold a value of up to 65,535, hopefully, it is enough!!

Renaming a column

- A column named ‘type’ can be confusing, because that is the name used by MySQL to identify data types.
- let’s change its name to category, like this:

```
ALTER TABLE classics CHANGE type category VARCHAR(16);
```

Removing a column

- you might decide that the page count column pages isn't actually all that useful for this particular database
- To remove it:

```
ALTER TABLE classics DROP pages;
```

- Remember that DROP is irreversible and you should always use it with caution, because you could inadvertently delete entire tables (and even databases) with it if you are not careful!

Deleting a table

- I don't want you to have to reenter all the data for the classics table, let's quickly create a new table

```
CREATE TABLE disposable(trash INT);

DESCRIBE disposable;
```

- Lets drop it now

```
DROP TABLE disposable;

SHOW tables;
```

Indexes

- As things stand, the table classics works and can be searched without problem by MySQL
- Until it grows to more than a couple of hundred rows, that is.
- At that point, database accesses will get slower and slower with every new row added.
- because MySQL has to search through every row whenever a query is issued.

Creating an Index

- It can be done when creating a table or at any time afterward.
- You must decide which columns require an index.
- Judgment that requires you to predict whether you will be searching any of the data in that column.
- You can combine multiple columns in one index
- The Pages column that we deleted, is an example of columns that usually people don search, so no need to index it.

Creating an Index

- In classics table , all those columns can be searched- lets create indexes for them:

```
ALTER TABLE classics ADD INDEX(author(20));
ALTER TABLE classics ADD INDEX(title(20));
ALTER TABLE classics ADD INDEX(category(4));
ALTER TABLE classics ADD INDEX(year);
DESCRIBE classics;
```

- The first two commands create indexes on both the author and title columns, limiting each index to only the first 20 characters to optimize the search speed.
- E.g. The Adventures of Tom Sawyer =only=> The Adventures of To
- For category column it can be reduced to 1 (F for Fiction, N for Nonfiction, and P for Play)

Using CREATE INDEX

- Create index independently from creating table

```
ALTER TABLE classics ADD INDEX(author(20));

CREATE INDEX author ON classics (author(20));
```

- Can not be used to create primary key index
- If you create index afterward of creating table, creating index is time consuming, especially for large amount of records

Adding indexes when creating tables

- CREATE TABLE classics (author VARCHAR(128),

```
title VARCHAR(128),
category VARCHAR(16),
year SMALLINT, INDEX(author(20)),
INDEX(title(20)),
INDEX(category(4)),
INDEX(year))
ENGINE MyISAM;
```

Primary keys

- Primary key columns are indexed columns
- The slide “The AUTO_INCREMENT data type” , briefly introduced the idea of a primary key when creating the auto-incrementing column id, which could have been used as a primary key for this table.
- However, we are going to use ISBN

```
ALTER TABLE classics ADD isbn CHAR(13) PRIMARY KEY;
```

- Unique, You cannot duplicate this value (give same ISBN for 2 different publications
- You cannot insert/update NULL as an entry value to this column

UPDATE .. SET .. WHERE

- If you wish to change the contents of one or more fields, you need to first narrow in on just the field or fields to be changed.
- Luckily, each of the years is unique in the current set of data, so we can use the year column to identify each row for updating.

```
ALTER TABLE classics ADD isbn CHAR(13);

UPDATE classics SET isbn='9781598184891' WHERE year='1876';
UPDATE classics SET isbn='9780582506206' WHERE year='1811';
UPDATE classics SET isbn='9780517123201' WHERE year='1856';
UPDATE classics SET isbn='9780099533474' WHERE year='1841';
UPDATE classics SET isbn='9780192814968' WHERE year='1594';

ALTER TABLE classics ADD PRIMARY KEY(isbn);

DESCRIBE classics;
```

Creating primary key when create a table

- CREATE TABLE classics1 (author VARCHAR(128),
  title VARCHAR(128),
  category VARCHAR(16),
  year SMALLINT,
  isbn CHAR(13),
  INDEX(author(20)),
  INDEX(title(20)),
  INDEX(category(4)),
  INDEX(year),
  PRIMARY KEY (isbn)) ENGINE MyISAM;

Classics1, or rename classics to classics1 then recreate it using this slide command

Creating a FULLTEXT index

- Unlike a regular index, MySQL's FULLTEXT allows super-fast searches of entire columns of text
- FULLTEXT indexes can be used only with MyISAM tables.
- MySQL has tens of different engine, to change ENGINE use:

  ALTER TABLE tablename ENGINE = MyISAM;.
- FULLTEXT indexes can be created for CHAR, VARCHAR, and TEXT columns only.
- FULLTEXT INDEX can take place either in CREATE TABLE or ALTER TABLE

```
ALTER TABLE classics ADD FULLTEXT(author,title);
```

Querying a MySQL Database

- So far, we’ve created a MySQL database and tables, populated them with data, and added indexes to make them fast to search. Now it’s time to look at how these searches are performed.

<u>SELECT:</u> is used to extract data from a table

- SELECT something FROM tablename;
- the simplest form to select all records from a table

```
SELECT * FROM classics;
```

SELECT

```
SELECT author , title FROM classics;
SELECT title , isbn FROM classics;
```

COUNT: counting rows

```
SELECT COUNT(*) from classics;
```

DISTINCT: to select the unique value of each instance (row in the result of SELECT)

To have a good example please insert the following to add another row for Charles Dickens

```
INSERT INTO classics(author, title, category, year, isbn) VALUES('Charles Dickens','Little Dorrit','Fiction','1857', '9780141439969');
```

Now:

```
SELECT author FROM classics;
SELECT DISTINCT author FROM classics;
```

Charles Dicken has to appear 1 time with DISTINCT twice without DISTINCT

MATCH...AGAINST

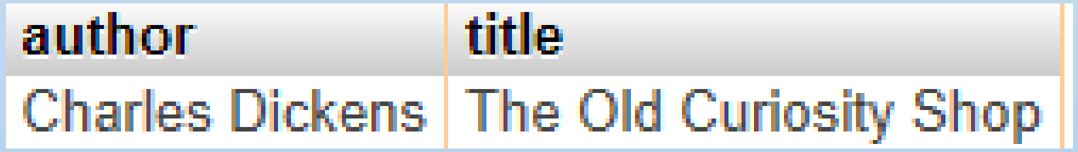
- The MATCH...AGAINST construct can be used on columns that have been given a FULL TEXT index
- It uses Natural Language algorithm.
- Unlike the use of WHERE.. It let you enter multiple words in a search query and checks them against all words in the FULLTEXT columns.
- It is case-insensitive
- FULLTEXT index can search all words, except a list of common word “and, is ,or ..etc” called stopword list, MySQL ignore searcing stopword words.

MATCH...AGAINST

- SELECT author,title FROM classics WHERE MATCH(author,title) AGAINST('and');

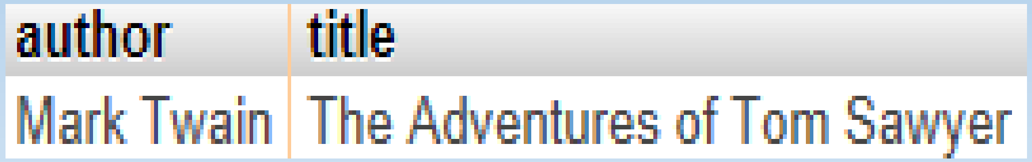
  This select will return NOTHING, since ‘and’ is stopword

- SELECT author,title FROM classics WHERE MATCH(author,title) AGAINST('curiosity shop');



author	title
Charles Dickens	The Old Curiosity Shop

- SELECT author,title FROM classics WHERE MATCH(author,title) AGAINST('tom sawyer');

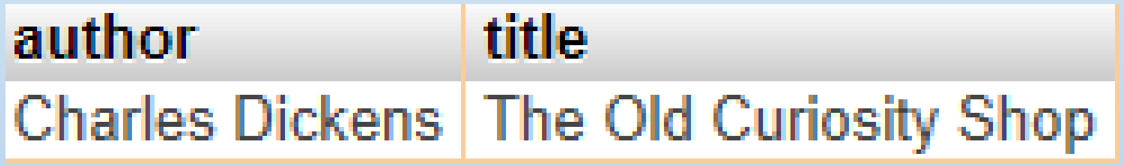


author	title
Mark Twain	The Adventures of Tom Sawyer

MATCH...AGAINST...in Boolean Mode

- Boolean mode also allows you to preface search words with a + or - sign to indicate whether they must be included or excluded.

```
SELECT author,title FROM classics WHERE MATCH(author,title)
AGAINST('+charles -species' IN BOOLEAN MODE);
```

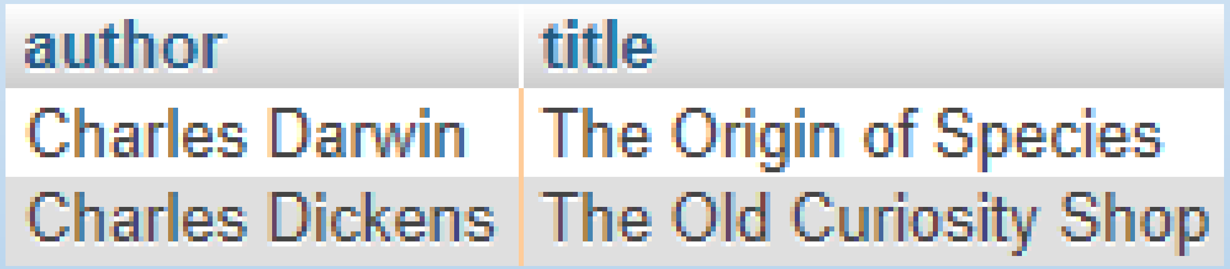


author	title
Charles Dickens	The Old Curiosity Shop

MATCH…AGAINST…in Boolean Mode

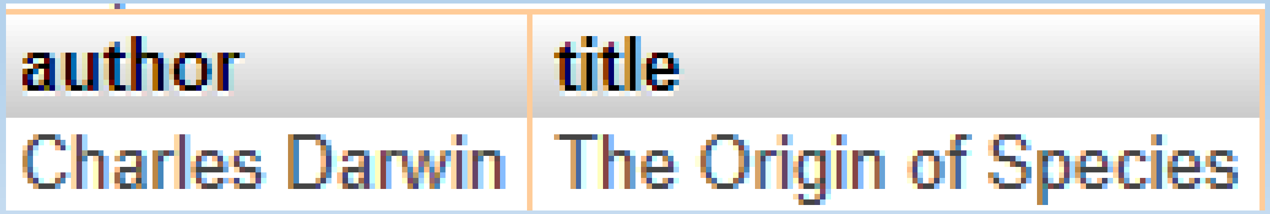
- To understand the diference between with/without IN BOOLEAN + -

  SELECT author,title FROM classics WHERE MATCH(author,title) AGAINST('charles species'); ….(ANY)



author	title
Charles Darwin	The Origin of Species
Charles Dickens	The Old Curiosity Shop

- SELECT author,title FROM classics WHERE MATCH(author,title) AGAINST('"origin of"' IN BOOLEAN MODE);



author	title
Charles Darwin	The Origin of Species

ORDER BY

- ORDER BY sorts returned results by one or more columns in ascending or descending order
  - SELECT author,title FROM classics ORDER BY author;

  It returns the publications by author in ascending alphabetical order (the default)

  SELECT author,title FROM classics ORDER BY title DESC;

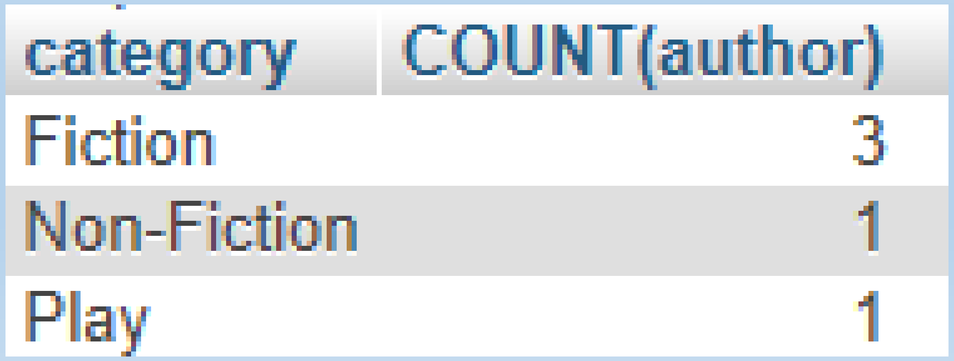
  It returns them by title in descending order

  SELECT author,title,year FROM classics ORDER BY author,year DESC;

GROUP BY

- you can group results returned from queries using GROUP BY, which is good for retrieving information about a group of data.
- if you want to know how many publications there are of each category in the classics table, you can issue the following query:

```
SELECT category,COUNT(author) FROM classics GROUP BY category;
```



category	COUNT(author)
Fiction	3
Non-Fiction	1
Play	1

References

- Learning PHP, MySQL & JavaScript, 4th Edition. With jQuery, CSS & HTML5, by Robin Nixon, O'Reilly Media, 2014, 978-1-4919-1866-1, chapter 7