# Installing MySQL

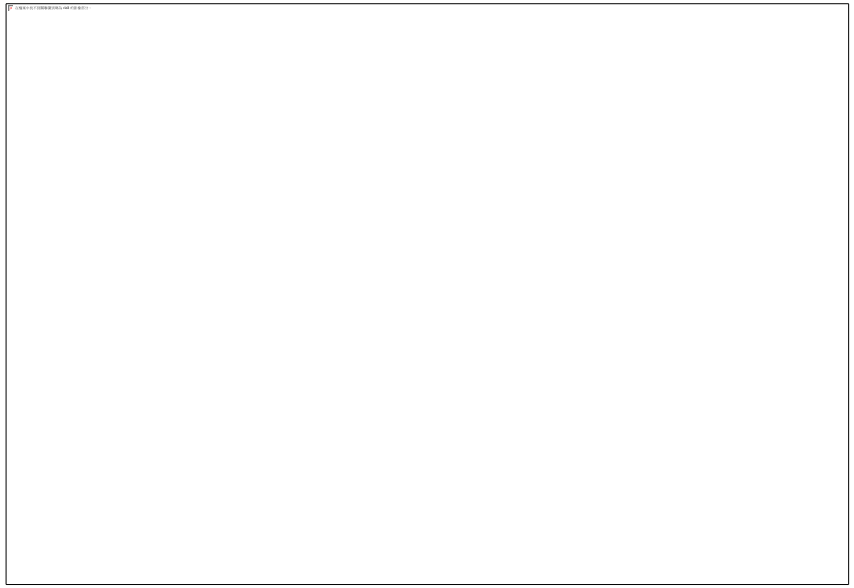
*Module 01*

Estimated time:

40 min.

# 學習目標

## Installing MySQL



# Checking Your System Works

- **mysql -u root**

- **Welcome to the MySQL monitor.  Commands end with ; or \g.**
- **Your MySQL connection id is 4 to server version: 4.1.0-alpha-max-debug-log**
- **Type 'help;' or '\h' for help. Type '\c' to clear the buffer.**
- **mysql>**

- **\q**

# Setting the Root Password

- **set password for root@localhost=password('your password');**
- **mysql -u root -p**

# Deleting Anonymous Accounts

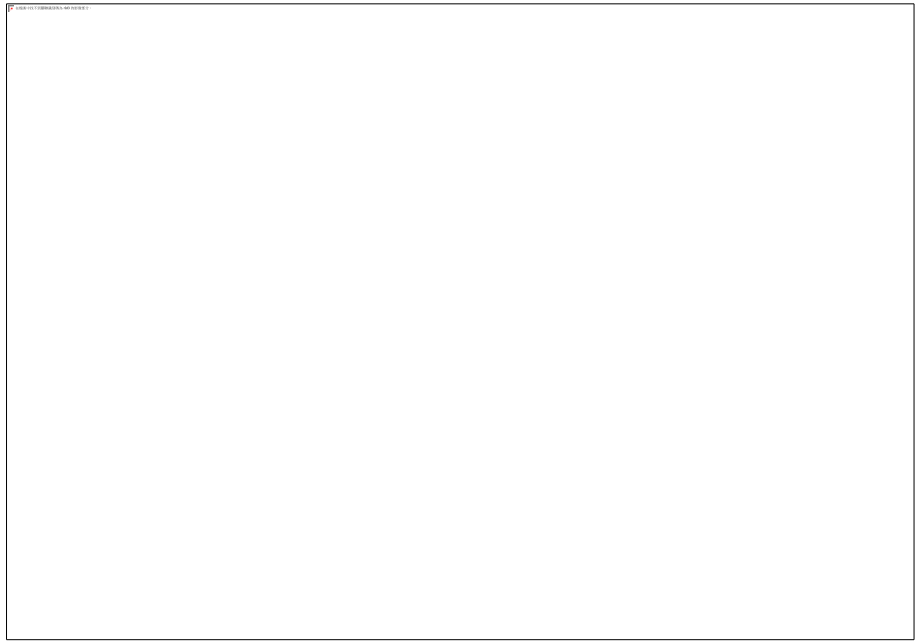
- use mysql;
- delete from user where User='';
- delete from db where User='';
- flush privileges;

# Creating an Account for Basic Use

- grant create, create temporary tables, delete, execute, index, insert,
- lock tables, select, show databases, update
- on *.*
- to username identified by 'password';

# 本章重點精華回顧

- **Installing MySQL**




THANK YOU

# Quick Tour

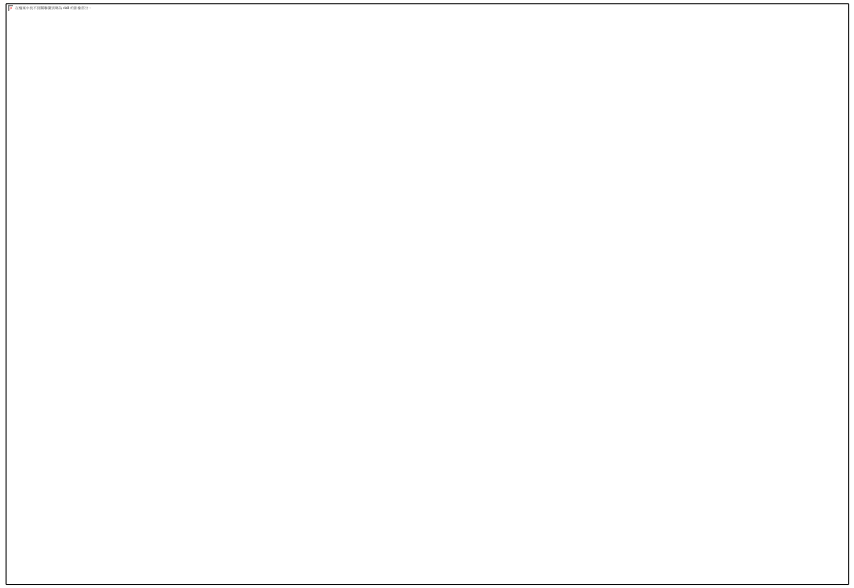
*Module 02*

Estimated time:

40 min.

# 學習目標

## Quick Tour



# MySQL Directory Structure

- **bin: This directory contains the MySQL server and client programs and several other useful compiled programs. The contents of this directory are covered in the next section of this chapter.**
- **scripts: This directory contains a set of Perl scripts that perform useful tasks. We will look at these in the next section of this chapter.**
- **data: This is where your actual database data resides.**

# Overview of Executables

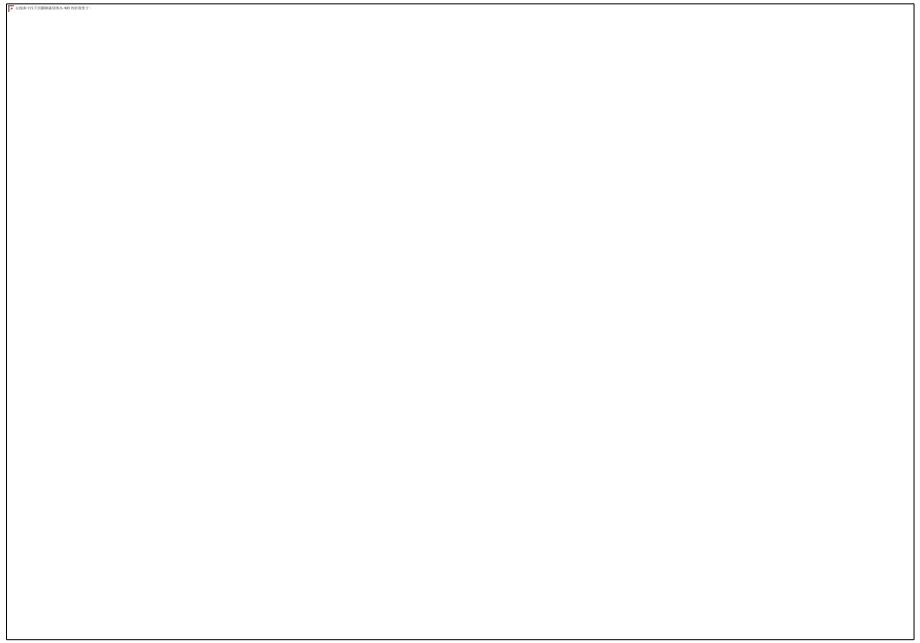
- **mysqladmin: Used to perform many administrative functions.**
- **myisamchk: Used to check and repair damaged MyISAM tables.**
- **mysqldump: Used to back up your databases.**
- **mysqlbinlog: Used to read the contents of the binary log, essential for disaster recovery.**
- **mysqlshow: Used to get information about databases and tables.**

# Introduction to the MySQL Monitor

- mysql -u username -p
- mysql -h hostname -u username -p
- show databases
- use databasename;
- show tables;
- describe tablename;
- \h
- (The h is for help.)
- source filename
- mysql -u username -p < filename

# 本章重點精華回顧

- **Quick Tour**




THANK YOU

# Database Design Crash Course

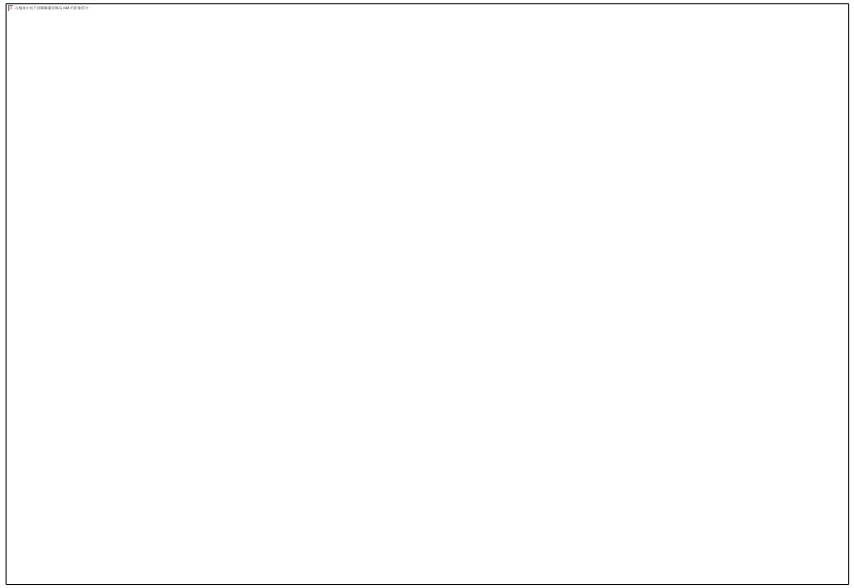
*Module 03*

Estimated time:

40 min.

# 學習目標

## Database Design Crash Course



# Database Design Principles

- Anomalies
- Insertion Anomalies
- Deletion Anomalies
- Update Anomalies

# Normalization

- First Normal Form
- 去除重複
- Second Normal Form
- 去除部分相依
- Third Normal Form
- 去除遞移相依

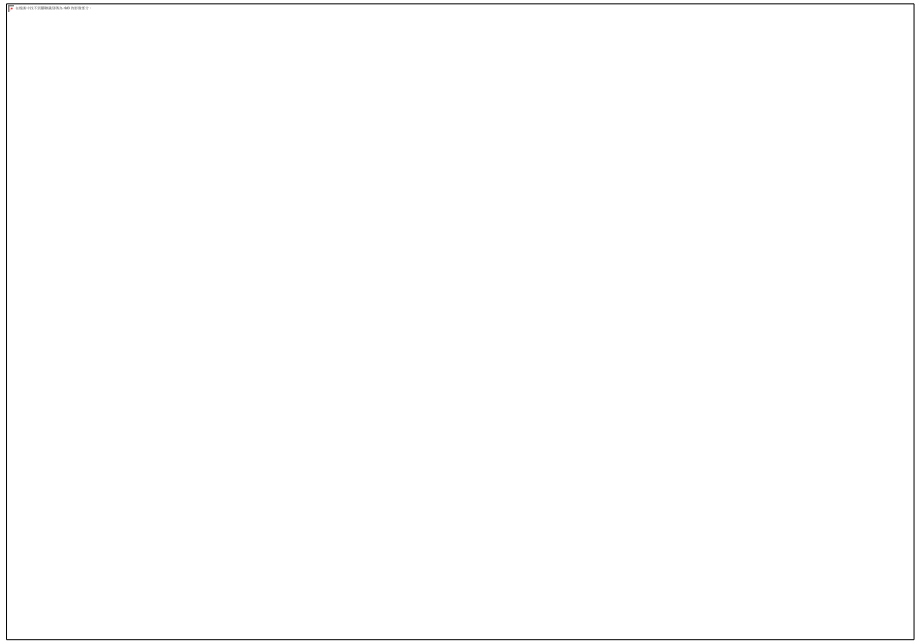
# Normalization

# Normalization

- **employee(employeeID, name, job, departmentID, departmentName, skill)**
- **departmentID-> departmentName 部分相依**
- **employeeID->name,job,departmentID 部分相依**
- **department(departmentID, departmentName)**
- **employee(employeeID, name, job, departmentID)**
- **employeeSkills(employeeID, skill)**

# 本章重點精華回顧

- **Normalization**




THANK YOU

# Creating Databases, Tables

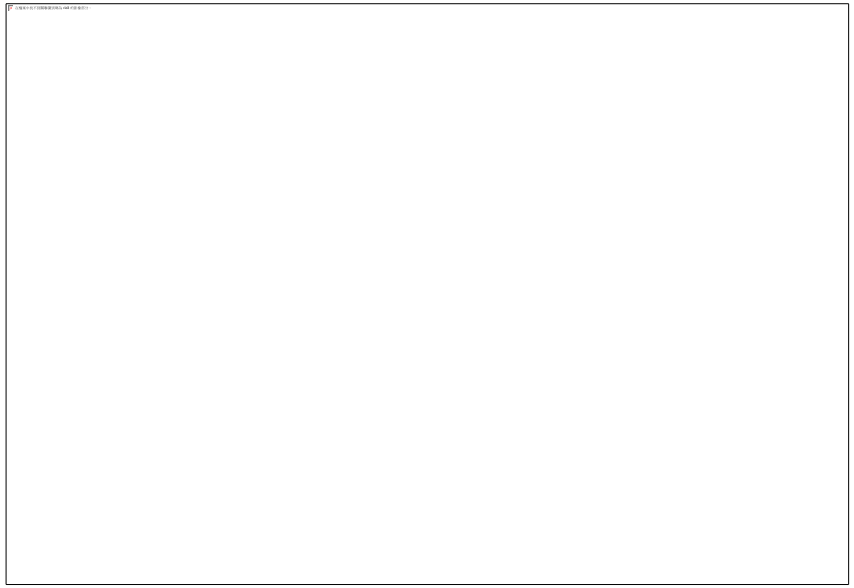
*Module 04*

Estimated time:

40 min.

# 學習目標

## Creating Databases, Tables, and Indexes



# Case Sensitivity

- **SQL keywords are not case sensitive**

# Identifiers in MySQL

- They can't contain quote characters, ACSII(0) and ASCII(255).
- Database names can contain any characters that are allowed in a directory name, but not the characters that have special meaning in a directory name (/, \, and .) for obvious reasons.
- Table names can contain any characters that are allowed in filenames, except for . and /.
-

# Creating a Database

- **create database employee;**
- **show databases;**

# Selecting a Database

- **use employee;**

# Creating Tables

- **create table tablename ( table definition ) [ENGINE=table_ENGINE];**

# Table Creation Example

```
drop database if exists employee;
create database employee;
use employee;
create table department
(
  departmentID int not null auto_increment primary key,
  name varchar(30)
) ENGINE=InnoDB;
```

# Table Creation Example

```
create table employee
(
  employeeID int not null auto_increment primary key,
  name varchar(80),
  job varchar(30),
  departmentID  int not null ,
  FOREIGN KEY (departmentID) references department(departmentID)
) ENGINE=InnoDB;
create table employeeSkills
(
  employeeID int not null,
  FOREIGN KEY (employeeID)  references employee(employeeID),
  skill varchar(15) not null,
  primary key (employeeID, skill)
) ENGINE=InnoDB;
```

# Table Creation Example

```
create table client
(
  clientID int not null auto_increment primary key,
  name varchar(40),
  address varchar(100),
  contactPerson varchar(80),
  contactNumber char(12)
) ENGINE=InnoDB;
```

# Table Creation Example

```
create table assignment
(
  clientID int not null ,
  FOREIGN KEY (clientID) references client(clientID),
  employeeID int not null ,
  FOREIGN KEY (employeeID)references employee(employeeID),
  workdate date not null,
  hours float,
  primary key (clientID, employeeID, workdate)
) ENGINE=InnoDB;
```

# Table Creation Example

- **show tables;**
- **describe department;**

# Creating Indexes

- **create index name on employee(name);**
- **create index part_name on employee(name(5));**

# Deleting Databases, Tables

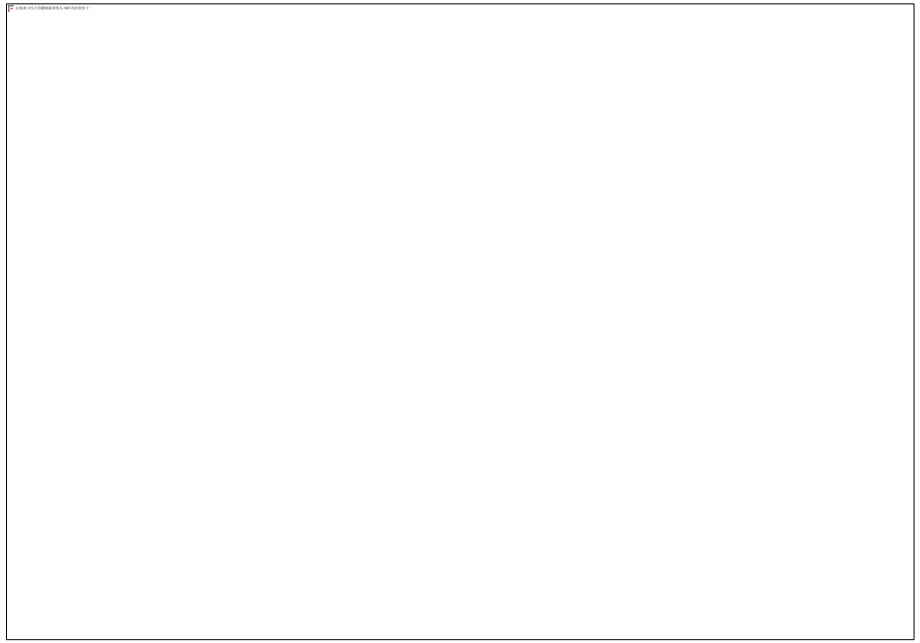
- **drop database employee;**
- **drop table assignment;**
- **DROP [TEMPORARY] TABLE [IF EXISTS] tbl_name [, tbl_name,..**
- **drop index part_name on employee;**

# Altering Existing Table Structures

- **alter table employee**
- **add index name (name);**

# 本章重點精華回顧

## Creating Databases, Tables, and Indexes




THANK YOU

# Inserting, Deleting, and Update

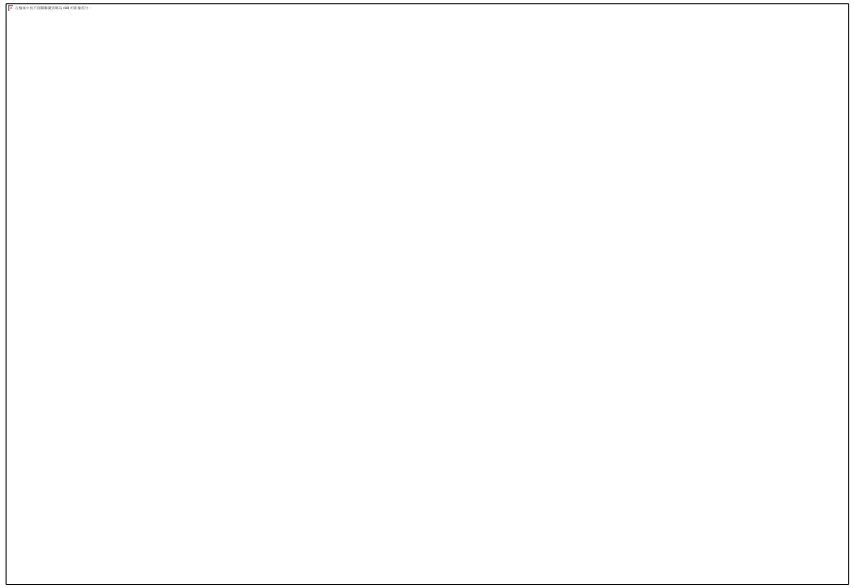
*Module 05*

Estimated time:

40 min.

# 學習目標

## Inserting, Deleting, and Updating Data



# Using INSERT

- use employee;
- delete from department;
- insert into department values
- (42, 'Finance'),
- (128, 'Research and Development'),
- (NULL, 'Human Resources'),
- (NULL, 'Marketing');
- delete from employee;
- insert into employee values
- (7513,'Nora Edwards','Programmer',128),
- (9842, 'Ben Smith', 'DBA', 42),
- (6651, 'Ajay Patel', 'Programmer', 128),
- (9006, 'Candy Burnett', 'Systems Administrator', 128);

# Using INSERT

- delete from employeeSkills;
- insert into employeeSkills values
- (7513, 'C'),
- (7513, 'Perl'),
- (7513, 'Java'),
- (9842, 'DB2'),
- (6651, 'VB'),
- (6651, 'Java'),
- (9006, 'NT'),
- (9006, 'Linux');
- delete from client;
- insert into client values
- (NULL, 'Telco Inc', '1 Collins St Melbourne',  'Fred Smith', '95551234'),
- (NULL, 'The Bank', '100 Bourke St Melbourne',  'Jan Tristan', '95559876');

# Using INSERT

- delete from assignment;
- insert into assignment values
- (1, 7513, '2003-01-20', 8.5);
- 
- select * from department;

# Using INSERT

```
insert into department
set name='Asset Management';
create table warning
(
  employeeID int primary key not null references employee(employeeID),
  count int default 1
) ENGINE =InnoDB;

insert into warning (employeeID)
  values (6651)
  on duplicate key update count=count+1;
```

# Using DELETE

- delete from department;
- delete from department where name='Asset Management';

# Using DELETE

- delete employee, employeeSkills
- from employee, employeeSkills, department
- where employee.employeeID = employeeSkills.employeeID
- and employee.departmentID = department.departmentID
- and department.name='Finance';
-
- delete from employee, employeeSkills
- using employee, employeeSkills, department
- where employee.employeeID = employeeSkills.employeeID
- and employee.departmentID = department.departmentID
- and department.name='Finance';

# Using UPDATE

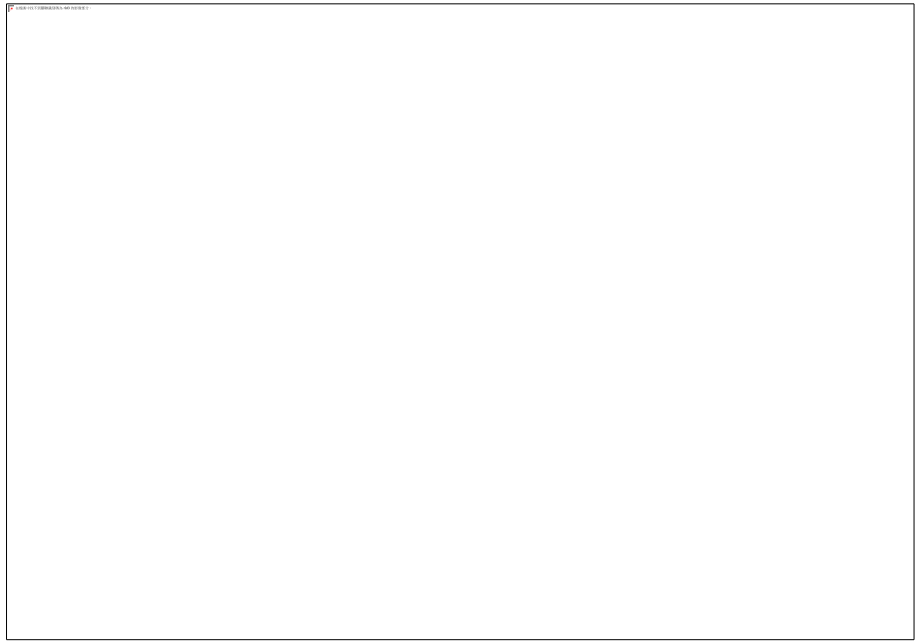
- update employee
- set job='DBA'
- where employeeID='6651';

# Uploading Data with LOAD DATA

- **SELECT * INTO OUTFILE 'department_infile.txt' FROM department;**
- **load data infile 'department_infile.txt' into table department;**

# 本章重點精華回顧

## Inserting, Deleting, and Updating Dataes




THANK YOU

# Querying MySQL

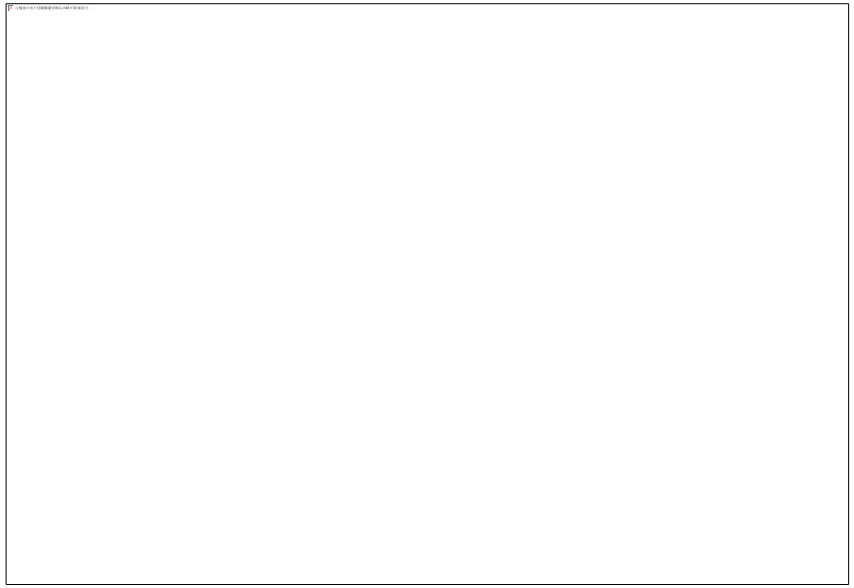
*Module 06*

Estimated time:

40 min.

# 學習目標

## Querying MySQL



# Overview of SELECT

- SELECT columns
- FROM tables
- [WHERE conditions]
- [GROUP BY group
- [HAVING group_conditions]]
- [ORDER BY sort_columns]
- [LIMIT limits];

# Simple Queries

- **select * from department;**

# Selecting Particular Columns

- **select name, employeeID from employee;**

# Specifying Absolute

- **select employee.name**
- **from employee;**

- **select name**
- **from employee.employee;**

- **select employee.employee.name**
- **from employee;**

# Aliases

- **select name as employeeName**
- **from employee;**

- **select e.name**
- **from employee as e;**
- 
- **select name employeeName**
- **from employee;**
- 
- **select e.name**
- **from employee e;**

# Using the WHERE Clause to Select

- select employeeID, name
- from employee
- where job='Programmer';

- select count(*) from employee;
-
- select * from assignment
- where employeeID=6651 and hours > 8;

# Removing Duplicates with DISTINCT

- select job
- from employee;

- select distinct job
- from employee;

- select count(job) from employee;
- select count(distinct job) from employee;

# Using the GROUP BY Clause

- select count(*), job
- from employee
- group by job;

- select count(*), job
- from employee
- group by job desc;

# Particular Groups with HAVING

- select count(*), job
- from employee
- group by job
- having count(*)=1;

# Sorting Search with ORDER BY

- **select ***
- **from employee**
- **order by job asc, name desc;**

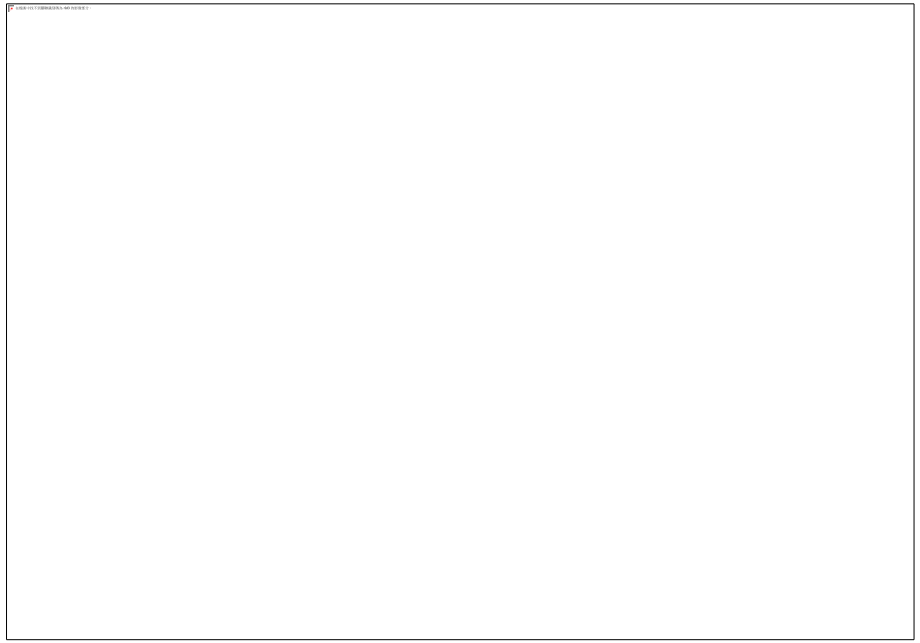
# Limiting Search Results with LIMIT

- select *
- from employeeSkills
- limit 5;

- select *
- from employeeSkills
- limit 5, 3;

# 本章重點精華回顧

## Querying MySQL




THANK YOU

# Advanced Queries

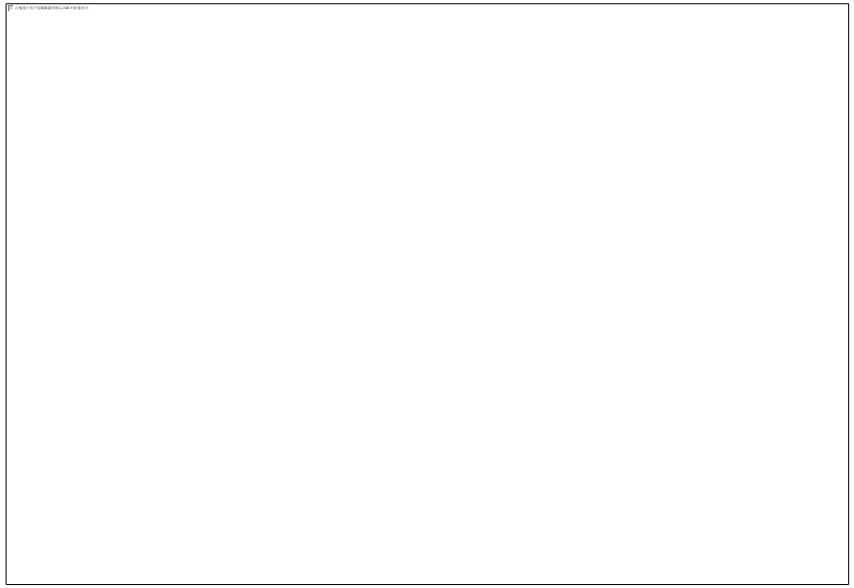
*Module 07*

Estimated time:

40 min.

# 學習目標

## Advanced Queries



# Using Joins to Run Queries

- **select employee.name, department.name**
  **from employee, department**
  **where employee.departmentID = department.departmentID;**

- **select employee.name, department.name**
- **from employee, department;**
- **select employee.name as employeeName, department.name as departmentName**
- **from employee, department**
- **where employee.departmentID = department.departmentID;**

# Joining Multiple Tables

- select department.name
- from client, assignment, employee, department
- where client.name='Telco Inc'
- and client.clientID = assignment.clientID
- and assignment.employeeID = employee.employeeID
- and employee.departmentID = department.departmentID;

# Joining a Table to Itself—Self Joins

- **select e2.name**
  **from employee e1, employee e2**
  **where e1.name = 'Nora Edwards'**
  **and e1.departmentID = e2.departmentID;**

- **select e2.name**
- **from employee e1, employee e2**
- **where e1.name = 'Nora Edwards'**
- **and e1.departmentID = e2.departmentID**
- **and e2.name != 'Nora Edwards';**

# Understanding the Basic Join

- **select employee.name, department.name**
  **from employee, department**
  **where employee.departmentID = department.departmentID;**
-
- **select employee.name, department.name**
- **from employee join department**
- **where employee.departmentID = department.departmentID;**

# Writing Subqueries

- select employeeID, name from employee where job='Programmer';
- 
- select programmer.name
- from (select employeeID, name from employee where job='Programmer')
- as programmer,
- assignment
- where programmer.employeeID = assignment.employeeID;
- 

# Using Single-Value Subqueries

- select max(hours) from assignment;
-
- select e.employeeID, e.name
- from employee e, assignment a
- where e.employeeID = a.employeeID
- and a.hours = (select max(hours) from assignment);

# Using Boolean Expression

- select name
- from employee
- where employeeID not in
- (select employeeID
- from assignment);

# Using Boolean Expression

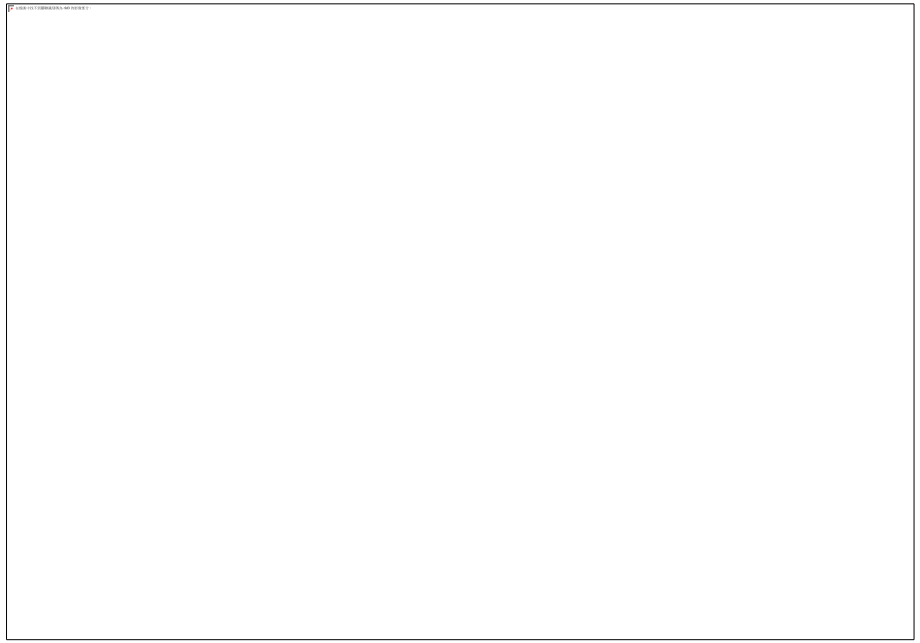
```
select name
from employee
where employeeID not in (6651, 1234);

select e.name, e.employeeID
from employee e
where not exists
        (select *
         from assignment
         where employeeID = e.employeeID);
```

# Using Boolean Expression

```
select e.name
from employee e, assignment a
where e.employeeID = a.employeeID
and a.hours > all
        (select a.hours
        from assignment a, employee e
        where e.employeeID = a.employeeID
        and e.job='Programmer');
```

# 本章重點精華回顧

## Advanced Queries




THANK YOU

# Using MySQL Built-In Function

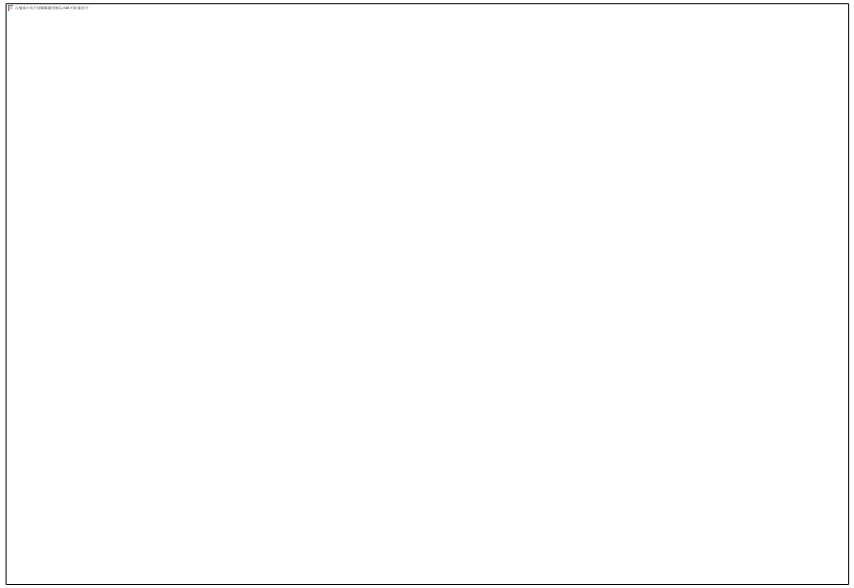
*Module 08*

Estimated time:

40 min.

# 學習目標

## Using MySQL Built-In Functions with SELECT



# Using MySQL Built-In Function

- **select 2+2;**

# Comparison Operators

- **select NULL=NULL;**
- **select NULL IS NULL;**
- **select * from department where name='marketing';**
- **select * from department where name = binary 'marketing';**

# Comparison Operators

| Table 8.1. Comparison Operators | |
|---|---|
| **Operator** | Meaning |
| **=** | Equality |
| **!= or <>** | Inequality |
| **<** | Less than |
| **<=** | Less than or equal to |
| **>** | Greater than |
| **>=** | Greater than or equal to |
| **n BETWEEN min AND max** | Range testing |
| **n IN (set)** | Set membership. Can be used with a list of literal values or expressions or with a subquery as the set. An example of a set is (apple, orange, pear) |
| **<=>** | NULL safe equal. This will return 1 (true) if we compare two NULL values |
| **n IS NULL** | Use to test for a NULL value in n |
| **ISNULL(n)** | Use to test for a NULL value in n |

# Logical Operators

| Table 8.2. Logical Operators | | |
|---|---|---|
| **Operator** | Example | Meaning |
| **AND or &&** | n && m | Logical AND. Here is the truth table:<br>true&&true = true<br>false&&anything = false<br>All other expressions evaluate to NULL. |
| **OR or \|\|** | n \|\| m | Logical OR. Here is the truth table:<br>true\|\|anything = true<br>NULL\|\|false = NULL<br>NULL\|\|NULL = NULL<br>false\|\|false = false |
| **NOT or !** | NOT n | Logical NOT. Here is the truth table:<br>!true = false<br>!false = true<br>!NULL = NULL |
| **XOR** | n XOR m | Logical exclusive OR. Here is the truth table:<br>true XOR true = false<br>true XOR false = true<br>false XOR true = true<br>NULL XOR n = NULL<br>n XOR NULL = NULL |

# Control Flow Functions

- IF (e1, e2, e3)
- select name, if(job='Programmer', "nerd", "not a nerd")
- from employee;
- select workdate, case
- when workdate < 2000-01-01 then "archived"
- when workdate < 2003-01-01 then "old"
- else "current"
- end
- from assignment;

# String Processing Functions

| Table 8.3. String Processing Functions | |
|---|---|
| **Function** | **Purpose** |
| concat(s1, s2, ...) | Concatenate the strings in s1, s2, .... |
| conv (n, original_base, new_base) | Convert the number n from original_base to new_base. (It may surprise you to see this as a string function, but some bases use letters in their notations, such as hexadecimal.) |
| length(s) | Returns the length in characters of the string s. |
| load_file(filename) | Returns the contents of the file stored at filename as a string. |
| locate(needle, haystack, position) | Returns the starting position of the needle string in the haystack string. The search will start from position. |
| lower(s) and upper(s) | Convert the string s to lowercase or uppercase. |

# String Processing Functions

| | |
|---|---|
| **quote(s)** | **Escapes a string s so that it is suitable for insertion into the database. This involves putting the string between single quotes and inserting a backslash.** |
| **replace(target, find, replace)** | Returns a string based on target with all incidences of find replaced with replace. |
| **soundex(s)** | Returns a soundex string corresponding to s. A soundex string represents how the string sounds when pronounced. It can be easier to match soundex strings of names than names themselves, for example. |
| **substring (s, position, length)** | Returns length characters from s starting at position. |
| **trim(s)** | Removes leading and trailing whitespace from s. (You can also use ltrim() to just remove whitespace from the left or rtrim() for the right.) |

# Using LIKE for Wildcard Matching

- **select ***
- **from department**
- **where name like '%research%';**

# Using STRCMP()

- STRCMP(s1, s2)
- and returns the following values:
- 0 if the strings are equal
- -1 if s1 is less than s2— that is, if it comes before s2 in the sort order
- 1 if s1 is greater than s2— that is, if it comes after s2 in the sort order
-
- select strcmp('cat', 'cat');
- select strcmp('cat', 'dog');

# Numeric Functions

| Table 8.4. Numeric Functions | |
|---|---|
| **Function** | **Purpose** |
| abs(n) | Returns the absolute value of n—that is, the value without a sign in front of it. |
| ceiling(n) | Returns the value of n rounded up to the nearest integer. |
| floor(n) | Returns the value of n rounded down to the nearest integer. |
| mod(n,m) and div | These two functions divide n by m. div returns the integral quotient, and mod() returns the integral remainder. |
| power(n,m) | Returns n to the power of m. |
| rand(n) | Returns a random number between 0 and 1. The parameter n is optional, but if supplied, it is used as a seed for the pseudorandom number generation. (Giving the same n to rand will produce the same pseudorandom number.) |
| round(n[,d]) | Returns n rounded to the nearest integer. If you supply d, n will be rounded to d decimal places. |
| sqrt(n) | Returns the square root of n. |

# Numeric Functions

- **select 9 mod 2;**
- **select 9 div 2;**

# Date and Time Functions

| Table 8.5. Date and Time Functions | |
|---|---|
| **Function** | **Purpose** |
| adddate(date, INTERVAL n type) and subdate(date, INTERVAL n type) | These functions are used to add and subtract dates. Both start from the date supplied in date and add or subtract the period specified after the keyword INTERVAL. You need to specify both a quantity n and the type of that quantity.<br>The type can be SECOND, MINUTE, HOUR, DAY, MONTH, YEAR, MINUTE:SECOND (the format of n should be 'm:s'), HOUR:MINUTE ('h:m'), DAY_HOUR ('d h'), YEAR_MONTH ('y-m'), HOUR_SECOND ('h:m:s'), DAY_MINUTE ('d h:m'), DAY_SECOND ('d h:m:s').<br>These functions are really useful, but remembering the data formats is virtually impossible (because they are all different), so you will usually have to look them up. |
| curdate(), curtime(), now() | These return the current date, the current time, and the current date and time, respectively. |

# Date and Time Functions

| | |
|---|---|
| date_format(date, format) and time_format(time, format) | These are used to reformat dates and times to pretty much any format you like. You do this by supplying a format string, such as date_format(workdate, '%W %D of %M, %Y'). (This gives, for example, 'Monday 16th of June, 2003'). There is a massive list of formats, so consult the manual for details. |
| dayname(date) | This returns the name of the day in date (for example, 'Monday'). |
| extract(type FROM date) | This returns the value of type in date. For example, if you specify YEAR, it will return the year from date. The types are the same as in adddate() and subdate(). |
| unix_timestamp([date]) | This returns the current Unix timestamp. (That's the number of seconds since the first of January 1970.) If called with a date, this returns the timestamp corresponding to that date. |

# Date and Time Functions

- **select adddate("1999-01-01", INTERVAL "1-6" YEAR_MONTH);**
- **select unix_timestamp(adddate("1999-01-01", INTERVAL "1-6" YEAR_MONTH));**

# Functions for Use with GROUP BY

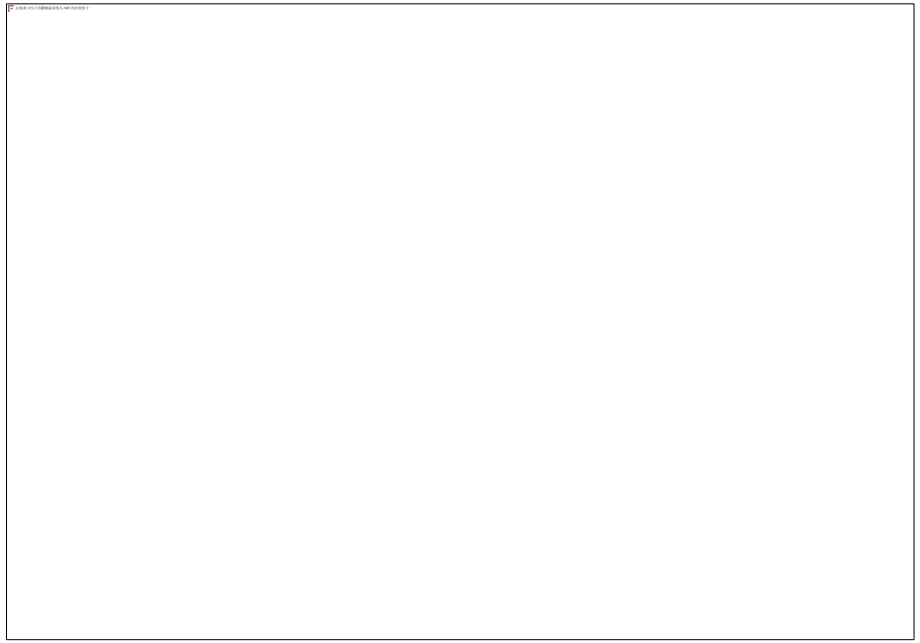
- select count(*)
- from employee;
- select job, count(job)
- from employee
- group by job;

# Functions for Use with GROUP BY

| Table 8.7. Grouping Functions | |
|---|---|
| **Function** | **Purpose** |
| avg(column) | Returns the average value in column. |
| count(column) | Returns the number of values in column. |
| min(column) | Returns the smallest value in column. |
| max(column) | Returns the largest value in column. |
| std(column) | Returns the standard deviation of the values in column. |
| sum(column) | Returns the sum of values in column |

# 本章重點精華回顧

## Using MySQL Built-In Functions with SELECT




THANK YOU

# Understanding MySQL's Table ENGINEs

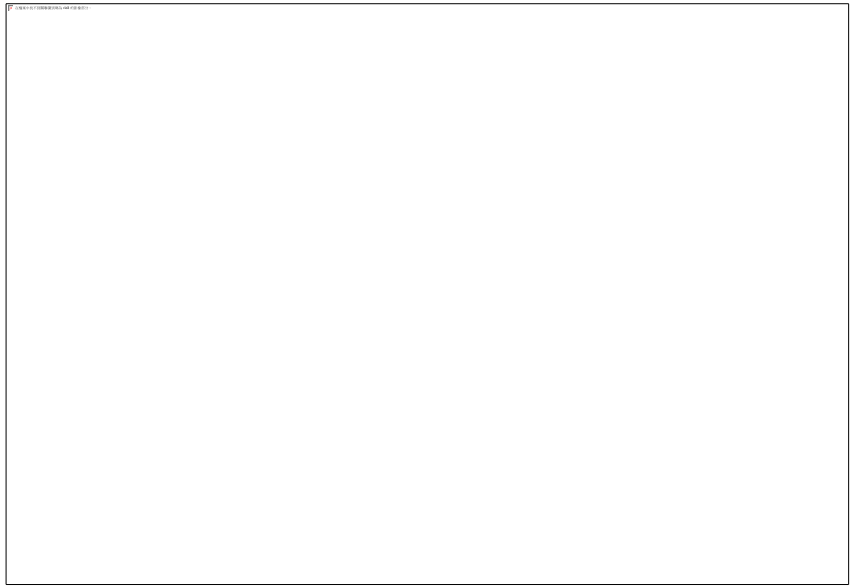
*Module 09*

Estimated time:

40 min.

# 學習目標

## Understanding MySQL's Table ENGINEs



# Full-Text Searching on MyISAM Table

```
create table article (
  articleID int not null auto_increment primary key,
  title varchar(255),
  body text,
  fulltext (title,body)
) ENGINE=MyISAM;
```

# Full-Text Searching on MyISAM Table

- select title
- from article
- where match (title,body) against ('merger');
-
- select title from article
- where match (title,body) against ('merge acquisition acquire takeover');

# Boolean Full-Text Search

- select title
- from article
- where match (title,body)
- against ('+linux +"Open Source" -desktop Java ~Oracle' IN BOOLEAN MODE);

# Boolean Full-Text Search

Table 9.1. Boolean Mode Search Operators

| Operator | Meaning |
|---|---|
| + | This word is compulsory. |
| - | This word must not appear. |
| < | This word is less important. |
| > | This word is more important. |
| ( ) | Group words together as a subexpression. |
| ~ | This word may appear, but it has a negative effect on ranking. |
| * | Wildcard suffix. For example, merge will not match merger, but merge* will match both merge and merger. May be used only at the end of a word. |
| " " | This is a phrase. Matches only exactly the same content in the same order. |

# InnoDB Tables

- **Transactions.**

# MERGE Tables

- create database logs;
- use logs;

- create table log2003Jan
- (logid int auto_increment primary key,
- logts datetime,
- entry char(255)) ENGINE=MyISAM;

- insert into log2003Jan values
- (NULL, '2003-01-01', 'first jan entry');
-

# MERGE Tables

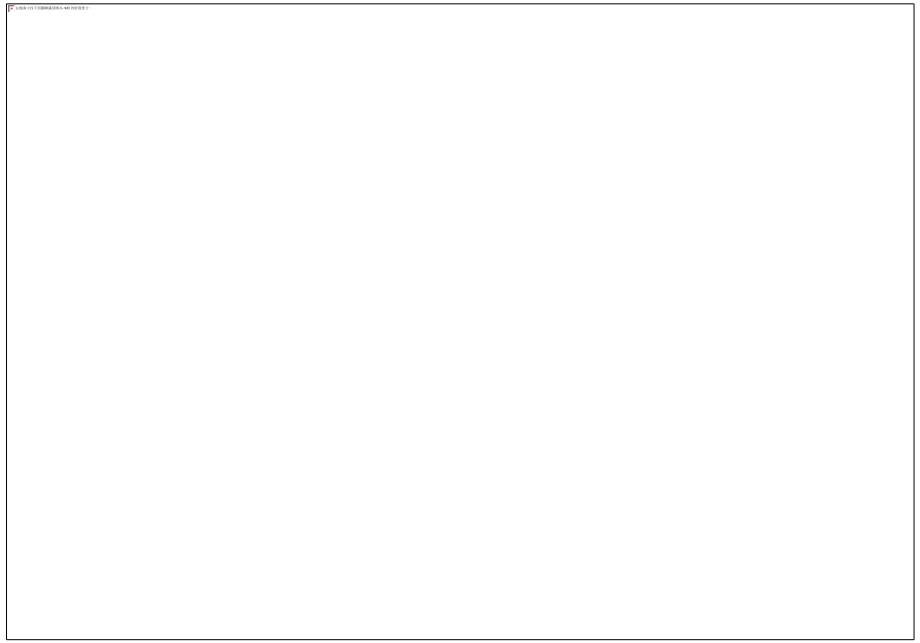
- create table log2003Feb
- (logid int auto_increment primary key,
- logts datetime,
- entry char(255)) ENGINE=MyISAM;
- insert into log2003Feb values
- (NULL, '2003-02-01', 'first feb entry');
- create table log2003Mar
- (logid int auto_increment primary key,
- logts datetime,
- entry char(255)) ENGINE=MyISAM;
- insert into log2003Mar values
- (NULL, '2003-03-01', 'first mar entry');

# MERGE Tables

```
create table logs
(logid int auto_increment primary key,
logts datetime,
entry char(255))
ENGINE = merge
union = (log2003Jan, log2003Feb, log2003Mar)
insert_method = last;

select * from logs;
```

# 本章重點精華回顧

## Understanding MySQL's Table ENGINEs




THANK YOU

# Transactions InnoDB Tables

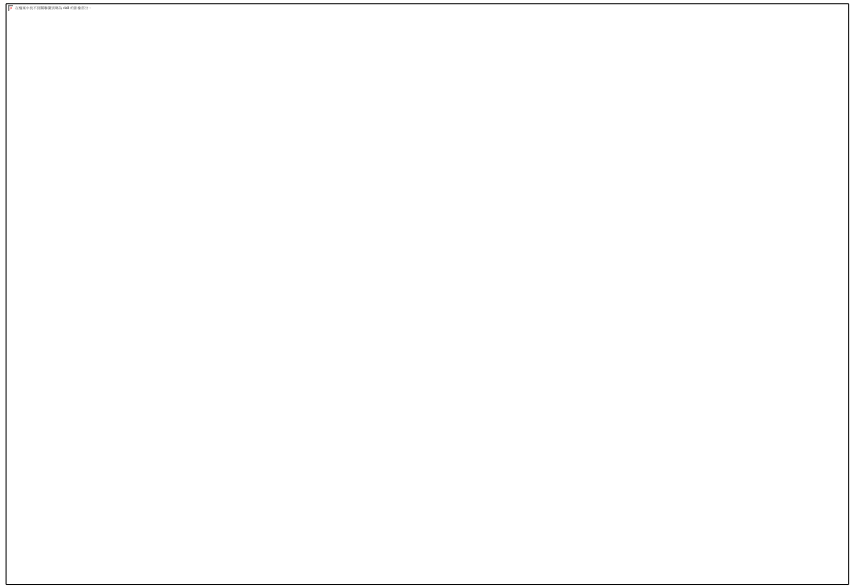
*Module 10*

Estimated time:

40 min.

# 學習目標

## Using Transactions with InnoDB Tables



# What Are Transactions?

- create table account
- (
- number int not null auto_increment primary key,
- balance float
- ) ENGINE = InnoDB;
- insert into account (balance) values (0.0);
- insert into account (balance) values (1000.0);
- insert into account (balance) values (2000.0);

# What Are Transactions?

- start transaction;
- update account set balance = balance - 1000 where number = 2;
- update account set balance = balance + 1000 where number = 1;
- commit;
- start transaction;
- update account set balance = balance - 1000 where number = 2;
- update account set balance = balance + 1000 where number = 1;
- select balance from account where number = 2;
- # select tells us that account #2 has a negative balance!
- # we'd better abort
- rollback;

# Using Transactions in MySQL

- start transaction;
- update account set balance = balance - 1000 where number = 2;
- commit;

- start transaction;
- update account set balance = balance + 1000 where number = 1;
- commit;

# Using Transactions in MySQL

- You can disable the autocommit behavior using the SET command as follows:
- set autocommit=0;
- As you would probably guess, the following command will put MySQL back into autocommit mode:
- set autocommit=1;

# Using Locks

- lock tables account write;
- select balance from account where number = 2;
- update account set balance = 1500 where number = 2;
- unlock tables;
-
- lock tables account write, account as a read, othertable low_priority write;

# Transaction Isolation

- Serializable
- Repeatable read
- Read committed
- Read uncommitted
- set transaction isolation level serializable;
- select * from account where number=1;
- select * from account where balance>1000;

- set transaction isolation level read committed;
- set transaction isolation level read uncommitted;

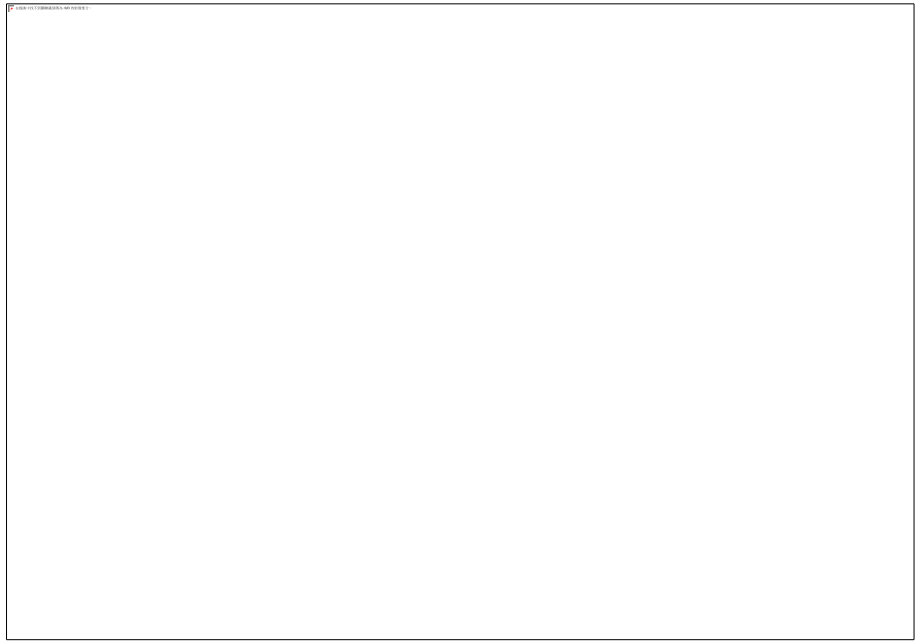
# Transaction Isolation

Table 10.1. Transaction Isolation Level Characteristics

| | Dirty Read | Nonrepeatable Read | Phantom Read |
|---|---|---|---|
| Read Uncommitted | Possible | Possible | Possible |
| Read Committed | Not possible | Possible | Possible |
| Repeatable Read | Not possible | Not possible | Possible (but unlikely) |
| Serializable | Not possible | Not possible | Not possible |

# 本章重點精華回顧

## Using Transactions with InnoDB Tables




THANK YOU

# Managing User Privileges

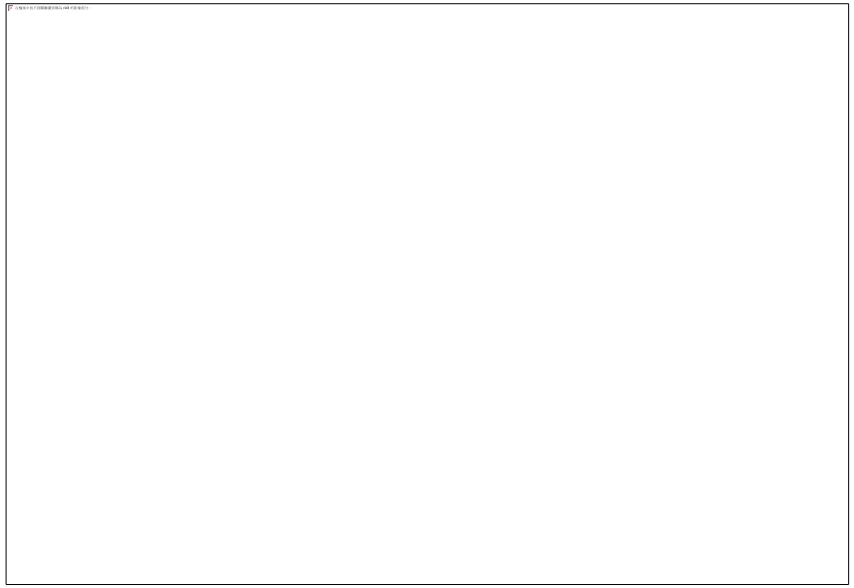
*Module 11*

Estimated time:

40 min.

# 學習目標

## Managing User Privileges



# Granting Privileges

- grant usage
- on *
- to fred@localhost identified by 'password';

# User-Level Privileges

| Table 11.1. User-Level Privileges | |
|---|---|
| **Privilege** | **Meaning** |
| CREATE | User can create tables. |
| CREATE TEMPORARY TABLES | User can create temporary tables. |
| DELETE | User can delete rows. |
| EXECUTE | User can execute procedures. |
| INDEX | User can create indexes. |
| INSERT | User can insert rows. |
| LOCK TABLES | User can lock tables. |
| SELECT | User can select rows. |
| SHOW DATABASES | User can execute a SHOW DATABASES command to retrieve the list of available databases. |
| UPDATE | User can update rows. |
| USAGE | User can log in, but cannot do anything else. |

# Administrator-Level Privileges

| Table 11.2. Administrator-Level Privileges | |
|---|---|
| **Privilege** | **Meaning** |
| ALL | User has all the privileges except WITH GRANT OPTION. |
| ALTER | User can alter tables. You may give this to some power users, but proceed with caution because it may be used to change the privilege tables. |
| DROP | User can drop tables. You may give this to trusted users. |
| FILE | User can load data from a file. Again, you may give this to trusted users. Beware of users trying to load arbitrary files, such as /etc/passwd or similar files! |
| PROCESS | User can show full process list—that is, see all the processes that MySQL is executing. |
| RELOAD | User can use the FLUSH statement. This has various purposes. We will look at FLUSH PRIVILEGES later in this chapter |
| REPLICATION CLIENT | User can check where the masters and slaves are. |
| REPLICATION SLAVE | Special privilege designed for the special replication user on the slave. |
| SHUTDOWN | User can run mysqladmin shutdown. |
| SUPER | User can connect even if MySQL has its maximum number of connections and can execute the commands CHANGE MASTER, KILL (thread), mysqladmin debug, PURGE MASTER LOGS, and SET GLOBAL. |
| WITH GRANT OPTION | User can pass on any privileges he has |

# Evaluating Privileges

- grant all on *.* to fred@localhost;
- grant all on employee.* to fred@localhost;
- grant select on department to fred@localhost;
- grant select (employeeID) on employee to fred@localhost;

# Using the REVOKE Statement

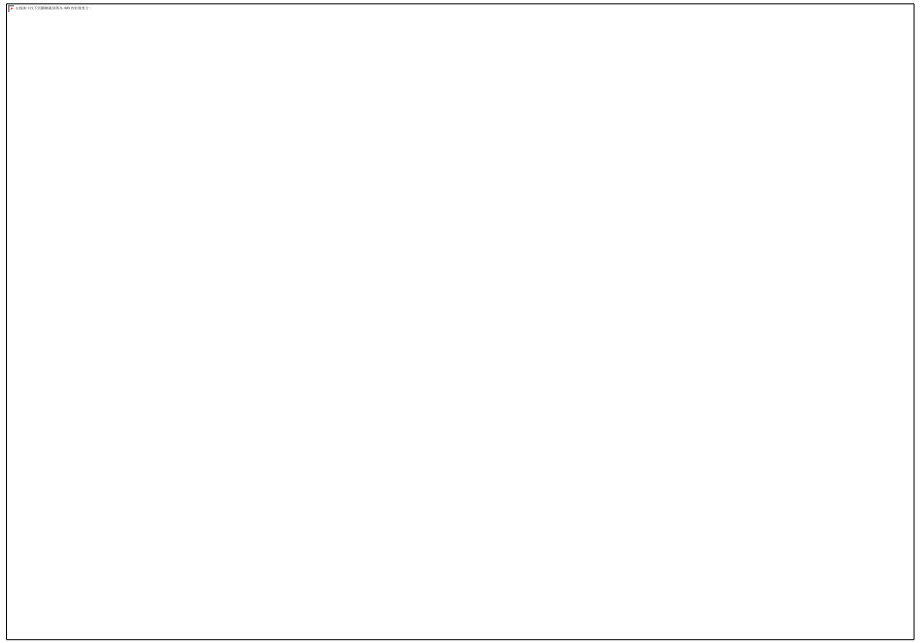
- **revoke all on employee.* from fred@localhost;**

# Understanding the Privilege Tables

- user
- db
- host
- tables_priv
- columns_priv
- func

# 本章重點精華回顧

## Managing User Privileges




THANK YOU

# Configuring MySQL

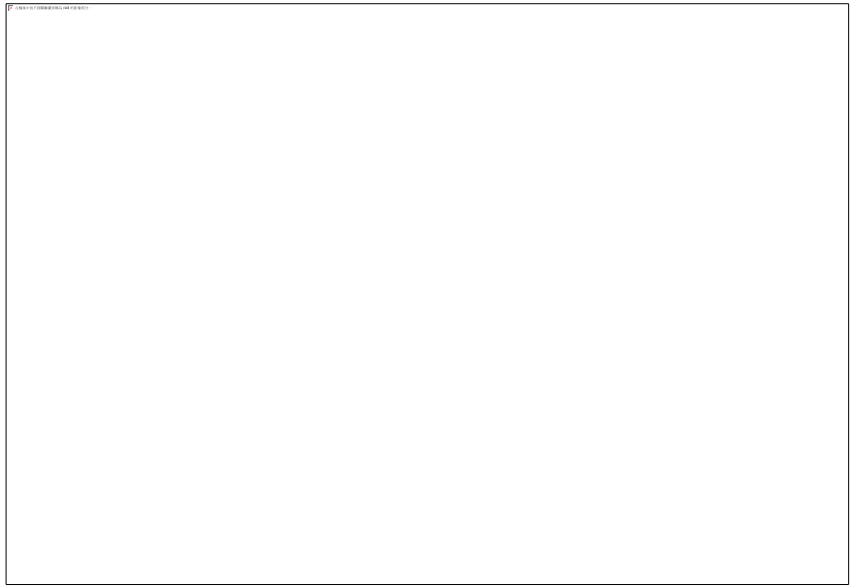
*Module 12*

Estimated time:

40 min.

# 學習目標

## Configuring MySQL



# Setting MySQL Configuration Options

- **Listing 12.1 Sample my.cnf File**
- **[mysqld]**
- **# turn on binary logging and slow query logging**
- **log-bin**
- **log-slow-queries**
-

# Setting MySQL Configuration Options

- # InnoDB config
- # This is the basic config as suggested in the manual
- # Datafile(s) must be able to
- # hold your data and indexes.
- # Make sure you have enough
- # free disk space.
- innodb_data_file_path = ibdata1:10M:autoextend
- # Set buffer pool size to
- # 50 - 80 % of your computer's
- # memory
- set-variable = innodb_buffer_pool_size=70M

# Setting MySQL Configuration Options

- set-variable = innodb_additional_mem_pool_size=10M
- # Set the log file size to about
- # 25 % of the buffer pool size
- set-variable = innodb_log_file_size=20M
- set-variable = innodb_log_buffer_size=8M
- # Set ..flush_log_at_trx_commit
- # to 0 if you can afford losing
- # some last transactions
- innodb_flush_log_at_trx_commit=1

# Setting Options for mysqld

- ansi: Run the server in ANSI compatibility mode. This makes MySQL use ANSI-99 SQL.
- basedir: Set the base directory of your installation if you want to put it in a nonstandard location.
- datadir: The same thing as basedir, but for the data directory.
- log-bin: Turn on binary logging. You can specify a filename for the location of the log.
- log-error: Turn on error logging. Again, you can specify the location of the log.
- log-slow-queries: Turn on slow query logging.
- port: Specify the port that the server should listen on. The default is 3306.
- user: Specify the user that the MySQL server should run as.

# Setting InnoDB Configuration Options

- innodb_data_file_path = ibdata1:10M:autoextend
- This option tells MySQL where to store InnoDB data.
- The general format of this option is
-
- filename:filesize[;filename:filesize;...][:autoextend[:max:size]]
- The autoextend option allows the tablespace to grow. The max option allows you to set a maximum size to which it can grow.

# Setting InnoDB Configuration Options

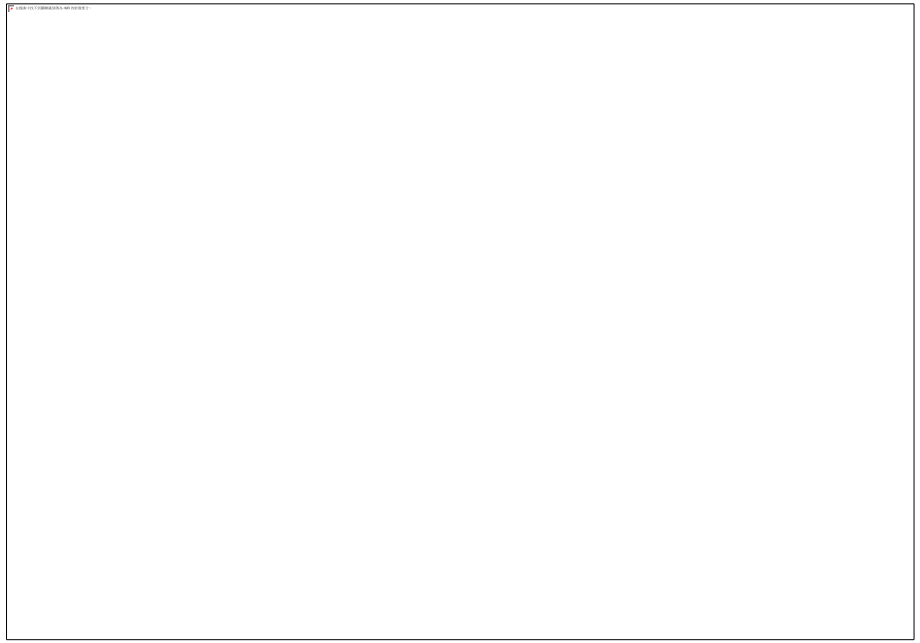
- **innodb_buffer_pool_size=70M**
- **This option sets the size of the buffer used to cache InnoDB table data and indexes.**
- **innodb_additional_mem_pool_size=10M**
- **This option sets aside memory to store internal MySQL data structures.**
- **innodb_log_file_size=20M**
- **This option sets the size of each log file.**

# Setting InnoDB Configuration Options

- **innodb_log_buffer_size=8M**
- **This option sets the size of the buffer in which logs are stored before they are written to disk.**
- **innodb_flush_log_at_trx_commit=1**
- **Setting this option to 1 means that every time a transaction is committed the log will be flushed to disk.**

# 本章重點精華回顧

## Configuring MySQL




THANK YOU

# Administering Your Database

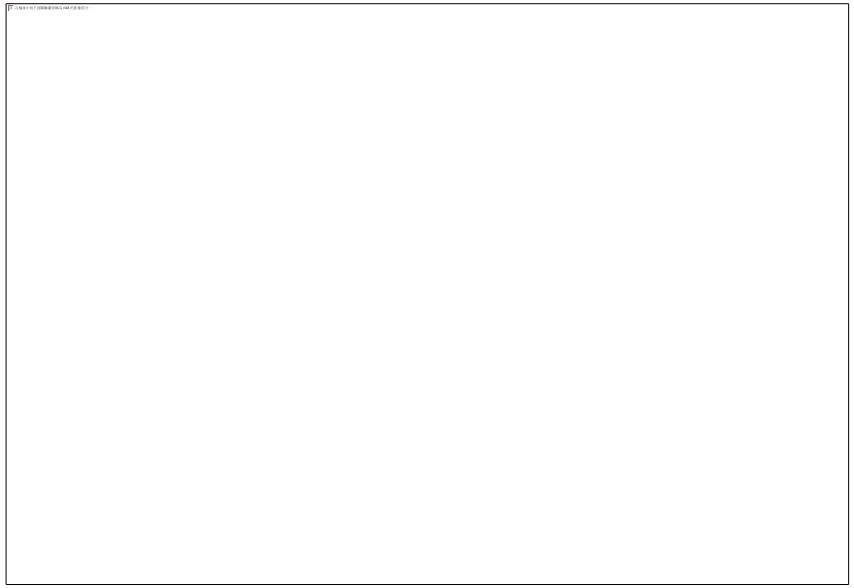
*Module 13*

Estimated time:

40 min.

# 學習目標

## Administering Your Database



# Retrieving Database Information

- **mysqlshow**
- **show databases;**
- **show databases;**
- **show tables;**
- **show columns from tablename;**
- **show table status**
-

# Viewing Server Status and Variables

- **SHOW STATUS**
- **mysqladmin –u username –p –extended-status**
- **show variables;**
- **mysqladmin –u username –p variables**

# Viewing Process Information

- **show processlist;**
- **mysqladmin –u username –p showprocesslist**

# Viewing Grant and Privilege

- **show grants for root@localhost;**
- **show privileges;**

# Viewing Reference Information

- show table types;
- show create table tablename;
- show create table department;
- CREATE TABLE 'department' (
- 'departmentID' int(11) NOT NULL auto_increment,
- 'name' varchar(30) default NULL,
- PRIMARY KEY ('departmentID')
- ) TYPE=InnoDB CHARSET=latin1

# Setting Variables

- **set variable=value;**
- **set sql_safe_updates=1;**
- **This turns on safe updates (as we can at the command line with -–i-am-a-dummy).**

# Clearing Caches

- **flush privileges;**
- **flush query cache;**
- **reset query cache;**

# Understanding the Log Files

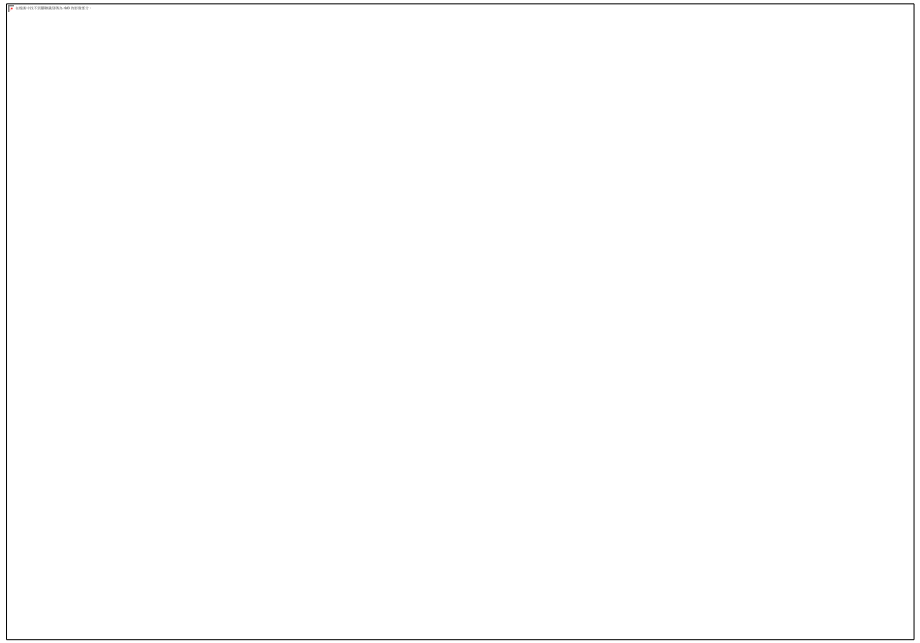
- **mysqlbinlog logfile**
- 
- **mysqladmin flush-logs**

# mysqladmin Option Summary

- mysqladmin create databasename
-
- mysqladmin drop databasename
- mysqladmin ping
- mysqladmin version
- mysqladmin status
- mysqladmin extended-status
- mysqladmin processlist
- mysqladmin kill id1,id2,id3...
- mysqladmin variables

# 本章重點精華回顧

## Administering Your Database




THANK YOU

# Backup and Disaster Recovery

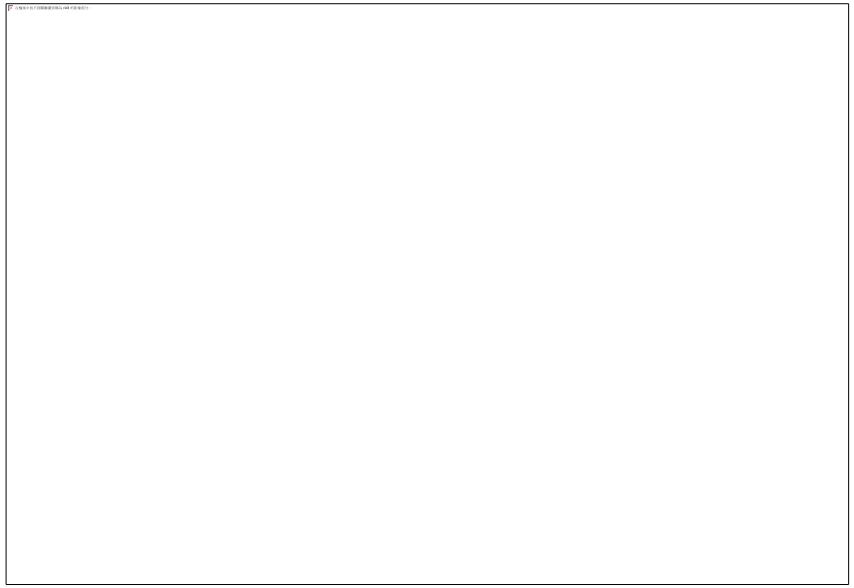
*Module 14*

Estimated time:

40 min.

# 學習目標

## Backup and Disaster Recovery



# Backing Up and Restoring with

- **mysqldump --opt –u username –p password employee > backup.sql**
- **mysql –u username –p  employee < backup.sql**

# Backing Up and Restoring Manually

- lock tables
- employee read,
- department read,
- client read,
- assignment read,
- employeeSkills read;
- flush tables;
- flush tables with read lock;

# Restoring from the Binary Log

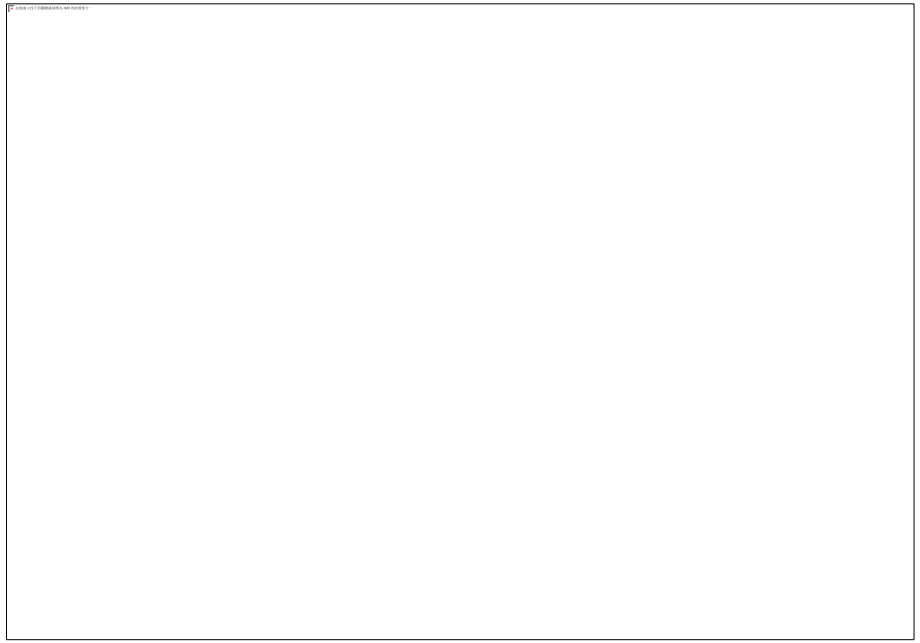
- **mysqlbinlog logfile > updates.sql**

# Checking and Repairing Tables

- **check table department;**
- **repair table t1;**

# 本章重點精華回顧

## Backup and Disaster Recovery




THANK YOU

# Securing Your MySQL

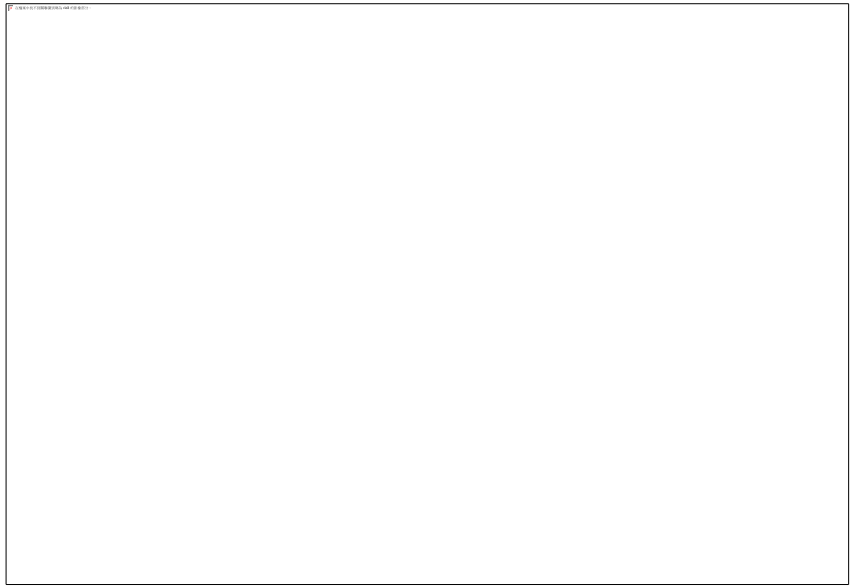
*Module 15*

Estimated time:

40 min.

# 學習目標

## Securing Your MySQL Installation



# Deleting Anonymous Accounts

- delete from user where User='';
- delete from db where User='';
- FLUSH PRIVILEGES

# Dangerous Privileges

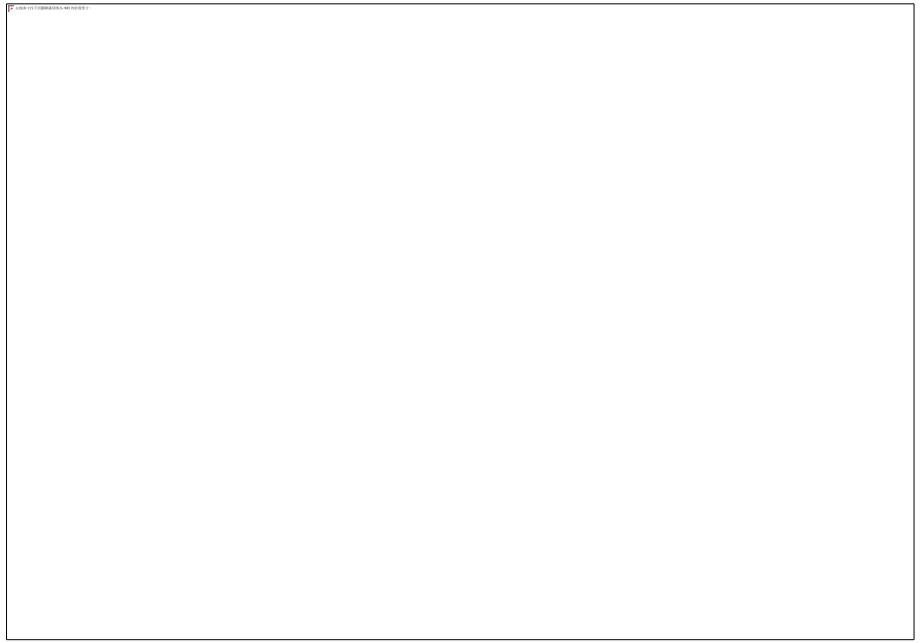
- careful of are FILE, PROCESS, and WITH GRANT OPTION.
- The FILE privilege allows users to LOAD DATA INFILE
- The PROCESS privilege allows users to SHOW PROCESSLIST
- The WITH GRANT OPTION privilege allows a user to share his privileges with others

# Passwords and Encryption

- other than the PASSWORD() function to encrypt them. We recommend use of MD5() or ENCRYPT() instead

# 本章重點精華回顧

## Securing Your MySQL Installation




THANK YOU

# Optimizing Your Database

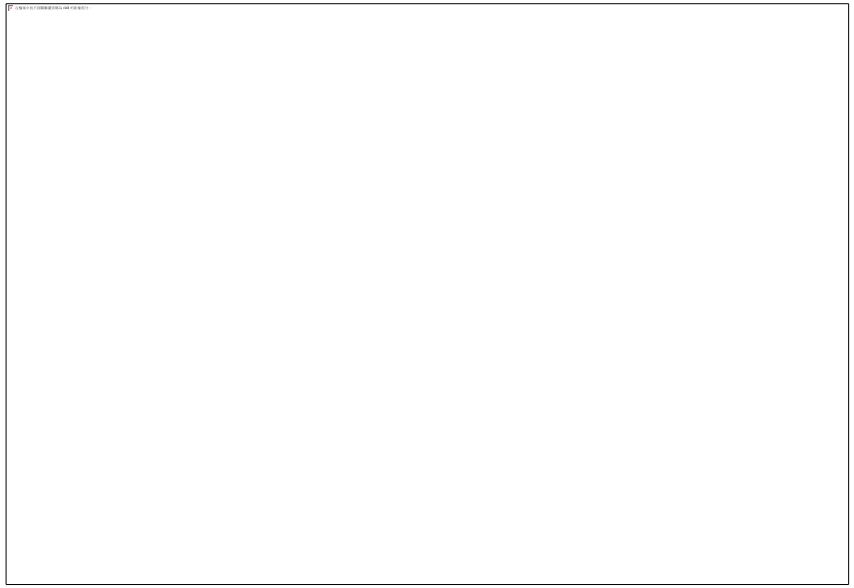
*Module 16*

Estimated time:

40 min.

# 學習目標

## Optimizing Your Database



# What's Slow in MySQL Databases?

- **Not using enough indexes.**
- **Using too many indexes.**
- **Using table- and column-level privileges.**
- **Making the wrong database design choices.**

# Making the Right Design Choices

- Use the smallest type that data will fit in.
- Use fixed-length records where possible.
- Declare as many columns NOT NULL as possible.
- Choose the table type on a table-by-table basis.
- Choose appropriate indexes.
- In extreme cases, you may even consider denormalization of tables to reduce the number of joins made for common queries.

# Indexing for Optimization

- A single column that has a single-column index
- A set of columns that forms a multicolumn index
- A column or set of columns that forms a subset of a multicolumn index, as long as there is a leftmost prefix of the index columns—for example, with the assignment table as before, with an index on (clientID, employeeID, workdate), indexes would be used for these types of queries:
- 
- SELECT...WHERE clientID=x
- SELECT...WHERE clientID=x AND employeeID=y
- But, they would not be used for this type:
- 
- SELECT...WHERE employeeID=y AND workdate=z

# ANALYZE TABLE

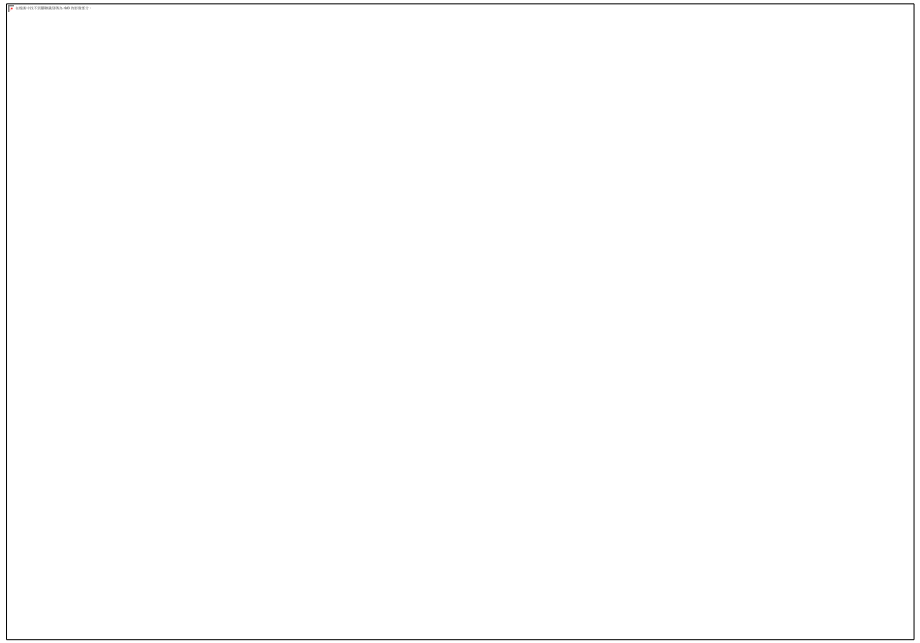
- **analyze table tablename;**

# Using OPTIMIZE TABLE

- **OPTIMIZE TABLE tablename;**

# 本章重點精華回顧

## Optimizing Your Database




THANK YOU

# Optimizing Your Queries

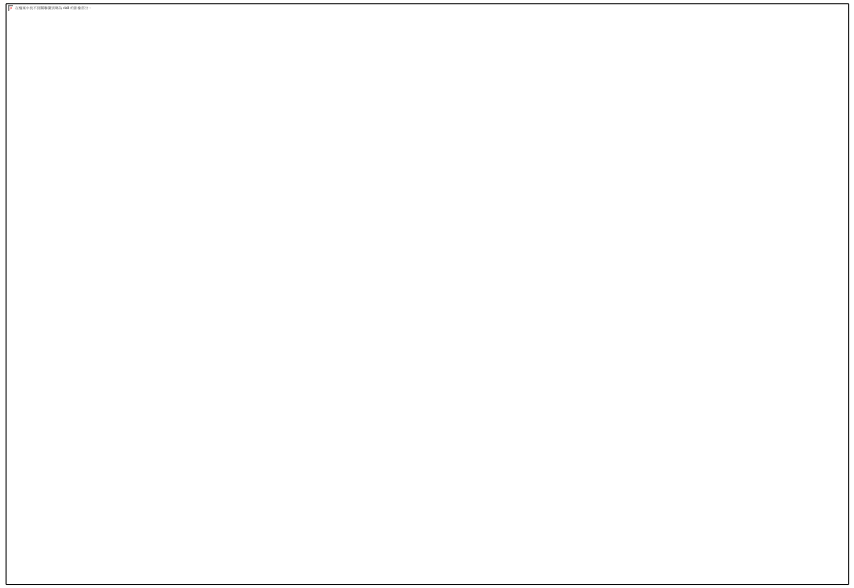
*Module 17*

Estimated time:

40 min.

# 學習目標

## Optimizing Your Queries



# Benchmarking Your Queries

- select benchmark(1000000, 6*9);
- select benchmark(10000000,
  'select employee.name, department.name
  from employee, department
  where employee.departmentID=department.departmentID');

# Using the Slow Query Log

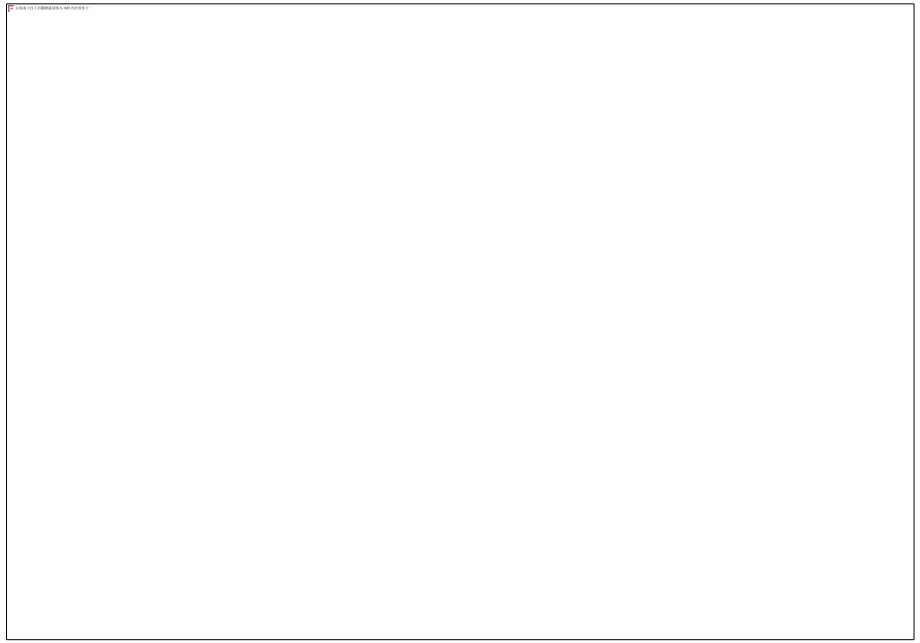
- turn on slow query logging with the --log-slow-queries=filename option when starting the MySQL server or in your configuration file
- turn on the --log-long-format option, all queries that run without using an index are also logged

# Using EXPLAIN to See How Queries

- explain
- select e.name, d.name
- from employee e, department d
- where e.departmentID = d.departmentID;
- create index ename_did on employee(name, departmentID);
- explain
  select e.name, d.name
  from employee e, department d
  where e.departmentID = d.departmentID;

# 本章重點精華回顧

## Optimizing Your Queries




THANK YOU

# Views,Stored Procedures and Functions,Triggers

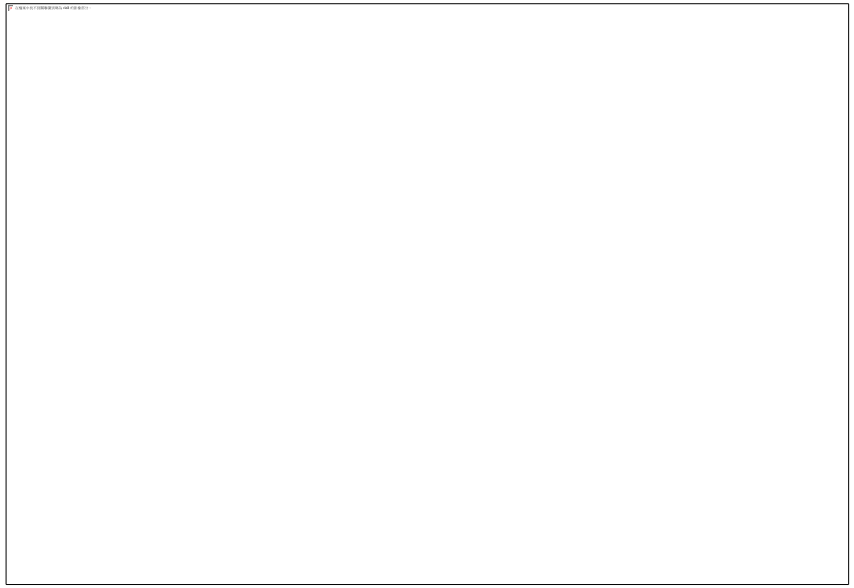
*Module 18*

Estimated time:

40 min.

# 學習目標

## Views,Stored Procedures and Functions,Triggers



# Using Views

- # Create president table for U.S. Historical League

```
DROP TABLE IF EXISTS president;
#@ _CREATE_TABLE_
CREATE TABLE president
(
last_name VARCHAR(15) NOT NULL,
first_name VARCHAR(15) NOT NULL,
suffix VARCHAR(5) NULL,
city VARCHAR(20) NOT NULL,
state VARCHAR(2) NOT NULL,
birth DATE NOT NULL,
death DATE NULL
);
#@ _CREATE_TABLE_
```

# Using Views

- **DELETE FROM president;**
  **INSERT INTO president VALUES ('Washington','George',NULL,'Wakefield','VA','1732-02-22','1799-12-14');**
  **INSERT INTO president VALUES ('Adams','John',NULL,'Braintree','MA','1735-10-30','1826-07-04');**
  **INSERT INTO president VALUES ('Jefferson','Thomas',NULL,'Albemarle County','VA','1743-04-13','1826-07-04');**
  **INSERT INTO president VALUES ('Madison','James',NULL,'Port Conway','VA','1751-03-16','1836-06-28');**
  **INSERT INTO president VALUES ('Monroe','James',NULL,'Westmoreland County','VA','1758-04-28','1831-07-04');**
  **INSERT INTO president VALUES ('Adams','John Quincy',NULL,'Braintree','MA','1767-07-11','1848-02-23');**
  **INSERT INTO president VALUES ('Jackson','Andrew',NULL,'Waxhaw settlement','SC','1767-03-15','1845-06-08');**
  **INSERT INTO president VALUES ('Van Buren','Martin',NULL,'Kinderhook','NY','1782-12-05','1862-07-24');**

# Using Views

- INSERT INTO president VALUES ('Harrison','William H.',NULL,'Berkeley','VA','1773-02-09','1841-04-04');
  INSERT INTO president VALUES ('Tyler','John',NULL,'Greenway','VA','1790-03-29','1862-01-18');
  INSERT INTO president VALUES ('Polk','James K.',NULL,'Pineville','NC','1795-11-02','1849-06-15');
  INSERT INTO president VALUES ('Taylor','Zachary',NULL,'Orange County','VA','1784-11-24','1850-07-09');
  INSERT INTO president VALUES ('Fillmore','Millard',NULL,'Locke','NY','1800-01-07','1874-03-08');
  INSERT INTO president VALUES ('Pierce','Franklin',NULL,'Hillsboro','NH','1804-11-23','1869-10-08');
  INSERT INTO president VALUES ('Buchanan','James',NULL,'Mercersburg','PA','1791-04-23','1868-06-01');

# Using Views

- INSERT INTO president VALUES ('Lincoln','Abraham',NULL,'Hodgenville','KY','1809-02-12','1865-04-15');
- INSERT INTO president VALUES ('Johnson','Andrew',NULL,'Raleigh','NC','1808-12-29','1875-07-31');
- INSERT INTO president VALUES ('Grant','Ulysses S.',NULL,'Point Pleasant','OH','1822-04-27','1885-07-23');
- INSERT INTO president VALUES ('Hayes','Rutherford B.',NULL,'Delaware','OH','1822-10-04','1893-01-17');
- INSERT INTO president VALUES ('Garfield','James A.',NULL,'Orange','OH','1831-11-19','1881-09-19');
- INSERT INTO president VALUES ('Arthur','Chester A.',NULL,'Fairfield','VT','1829-10-05','1886-11-18');
- INSERT INTO president VALUES ('Cleveland','Grover',NULL,'Caldwell','NJ','1837-03-18','1908-06-24');
- INSERT INTO president VALUES ('Harrison','Benjamin',NULL,'North Bend','OH','1833-08-20','1901-03-13');
- INSERT INTO president VALUES ('McKinley','William',NULL,'Niles','OH','1843-01-29','1901-09-14');
- INSERT INTO president VALUES ('Roosevelt','Theodore',NULL,'New York','NY','1858-10-27','1919-01-06');
- INSERT INTO president VALUES ('Taft','William H.',NULL,'Cincinnati','OH','1857-09-15','1930-03-08');
-

# Using Views

- INSERT INTO president VALUES ('Wilson','Woodrow',NULL,'Staunton','VA','1856-12-19','1924-02-03');
INSERT INTO president VALUES ('Harding','Warren G.',NULL,'Blooming Grove','OH','1865-11-02','1923-08-02');
INSERT INTO president VALUES ('Coolidge','Calvin',NULL,'Plymouth Notch','VT','1872-07-04','1933-01-05');
INSERT INTO president VALUES ('Hoover','Herbert C.',NULL,'West Branch','IA','1874-08-10','1964-10-20');
INSERT INTO president VALUES ('Roosevelt','Franklin D.',NULL,'Hyde Park','NY','1882-01-30','1945-04-12');
INSERT INTO president VALUES ('Truman','Harry S',NULL,'Lamar','MO','1884-05-08','1972-12-26');
INSERT INTO president VALUES ('Eisenhower','Dwight D.',NULL,'Denison','TX','1890-10-14','1969-03-28');
INSERT INTO president VALUES ('Kennedy','John F',NULL,'Brookline','MA','1917-05-29','1963-11-22');
INSERT INTO president VALUES ('Johnson','Lyndon B.',NULL,'Stonewall','TX','1908-08-27','1973-01-22');
INSERT INTO president VALUES ('Nixon','Richard M',NULL,'Yorba Linda','CA','1913-01-09','1994-04-22');
INSERT INTO president VALUES ('Ford','Gerald R',NULL,'Omaha','NE','1913-07-14',NULL);

# Using Views

- **INSERT INTO president VALUES ('Carter','James E.','Jr.','Plains','GA','1924-10-01',NULL);**
  **INSERT INTO president VALUES ('Reagan','Ronald W.',NULL,'Tampico','IL','1911-02-06','2004-06-05');**
  **INSERT INTO president VALUES ('Bush','George H.W.',NULL,'Milton','MA','1924-06-12',NULL);**
  **INSERT INTO president VALUES ('Clinton','William J.',NULL,'Hope','AR','1946-08-19',NULL);**
  **INSERT INTO president VALUES ('Bush','George W.',NULL,'New Haven','CT','1946-07-06',NULL);**
-

# Using Views

- CREATE VIEW vpres AS
- SELECT last_name, first_name, city, state FROM president;
- 
- SELECT * FROM vpres;
- SELECT * FROM vpres WHERE last_name = 'Adams';

# Using Views

- CREATE VIEW vpres2 (ln, fn) AS
- SELECT last_name, first_name FROM president;
- 
- 
- mysql> SELECT last_name, first_name FROM vpres2;
- ERROR 1054 (42S22) at line 1: Unknown column 'last_name' in 'field list'
- mysql> SELECT ln, fn FROM vpres2;
- 

# Using Views

```
CREATE VIEW pres_age AS
    SELECT last_name, first_name, birth, death,
    (YEAR(death) - YEAR(birth))
    - IF(RIGHT(death,5) < RIGHT(birth,5),1,0)
    AS age
    FROM president;

mysql> SELECT * FROM pres_age;
```

# Using Views

- CREATE TABLE t (i INT);
- INSERT INTO t (i) VALUES(1),(2),(3);
- CREATE VIEW v AS SELECT i FROM t;
- mysql> SELECT i FROM v;

- INSERT INTO v (i) VALUES(4);
- DELETE FROM v WHERE i < 3;
- mysql> SELECT i FROM v;

- mysql> UPDATE v SET i = i + 1;
- mysql> SELECT i FROM v;

# Using Stored Procedures

```
DROP PROCEDURE IF EXISTS born_in_year;
CREATE PROCEDURE born_in_year (year_of_birth INT)
   SELECT first_name, last_name, birth, death
   FROM president
   WHERE YEAR(birth) = year_of_birth;

mysql> CALL born_in_year(1908);
```

# Using Stored Procedures

```
delimiter $
CREATE PROCEDURE count_born_in_year
  (year_of_birth INT, OUT how_many INT)
  BEGIN
   DECLARE c CURSOR FOR
   SELECT COUNT(*) FROM president WHERE YEAR(birth) = year_of_birth;
    OPEN c;
    FETCH c INTO how_many;
    CLOSE c;
  END$
delimiter ;
mysql> CALL count_born_in_year(1908, @count);
mysql> SELECT @count;
```

# Using Stored Procedures

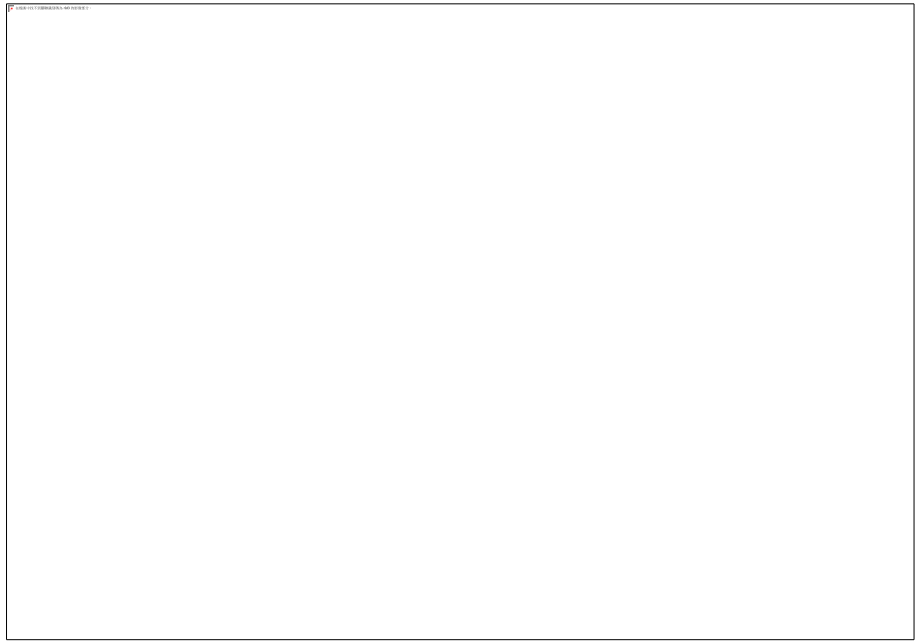
- **mysql> CALL count_born_in_year(1913, @count);**
- **mysql> SELECT @count;**

# Using Triggers

```
CREATE TABLE t2 (i INT, dt DATETIME);
delimiter $
CREATE TRIGGER t_ins BEFORE INSERT ON t2
    FOR EACH ROW BEGIN
     SET NEW.dt = CURRENT_TIMESTAMP;
     IF NEW.i < 0 THEN SET NEW.i = 0; END IF;
    END$
delimiter ;
mysql> INSERT INTO t2 (i) VALUES(-2),(0),(2);
mysql> SELECT * FROM t2;
```

# 本章重點精華回顧

## Views,Stored Procedures and Functions,Triggers




THANK YOU