



西安交通大学  
XI'AN JIAOTONG UNIVERSITY



# 数据库建模实验指导

## 白秀秀



第二次实验课

## 第二次实验课内容

- **5. 数据入库操作：**对自己建的表增添数据，每个表不少于 **5**条数据。
- **6. 建立并测试视图：**建立至少**2**个视图，编写建立视图的语句及执行结果，并对视图进行测试。
- **7. 针对自己的需求，设计增删改查操作，**每项操作不少于**2**个，写增删改查的**sql**语句，截图展示每条**sql**语句测试的执行结果。
- **8. 建立并测试触发器：**针对自己的需求，建立至少**2**个触发器。编写建立触发器的语句，并对每个触发器进行测试，展示测试结果。

# 目录

1. 数据入库操作
2. 建立并测试视图
3. 增删改查操作测试
4. 建立并测试触发器





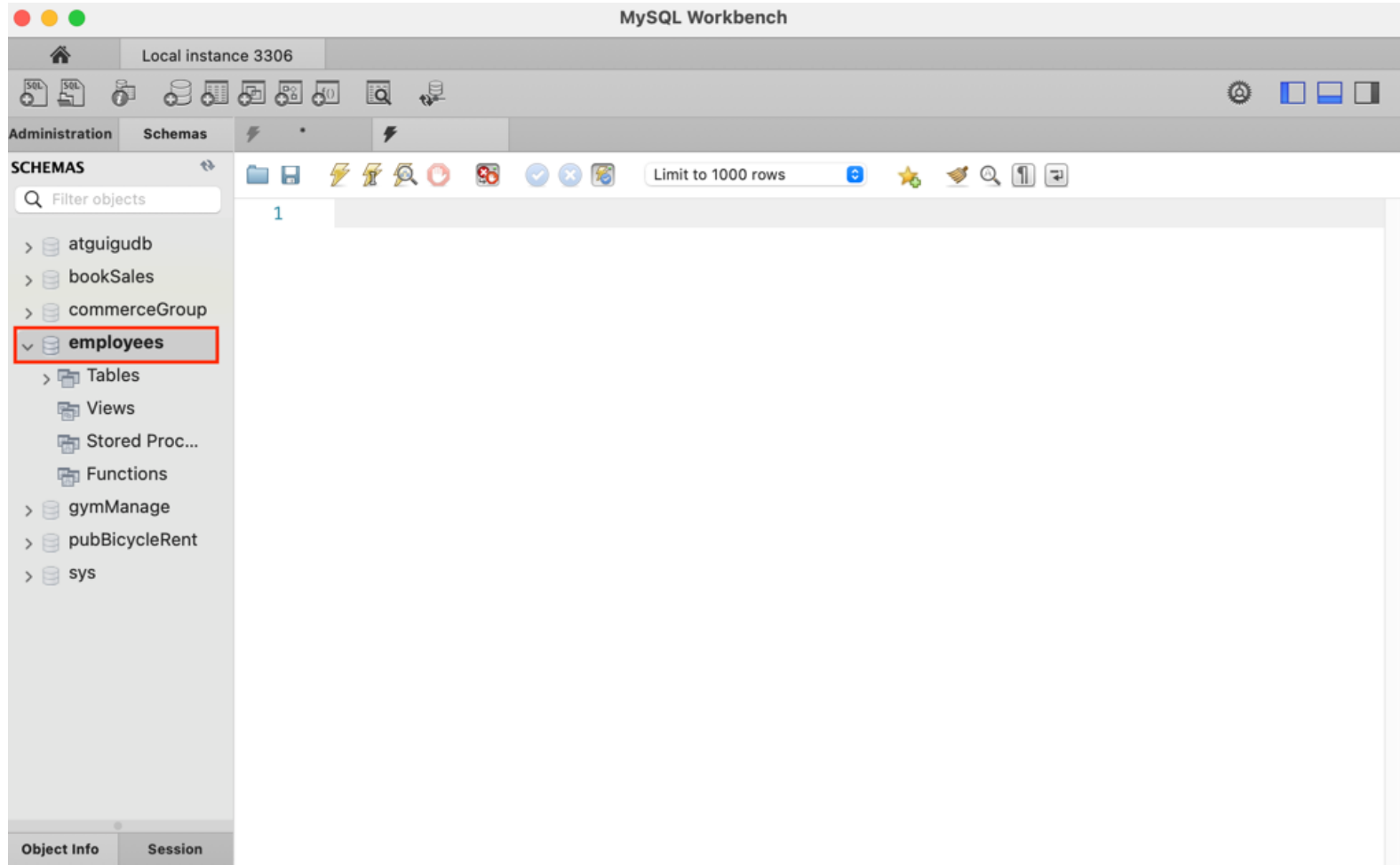
Part1

# 数据入库操作



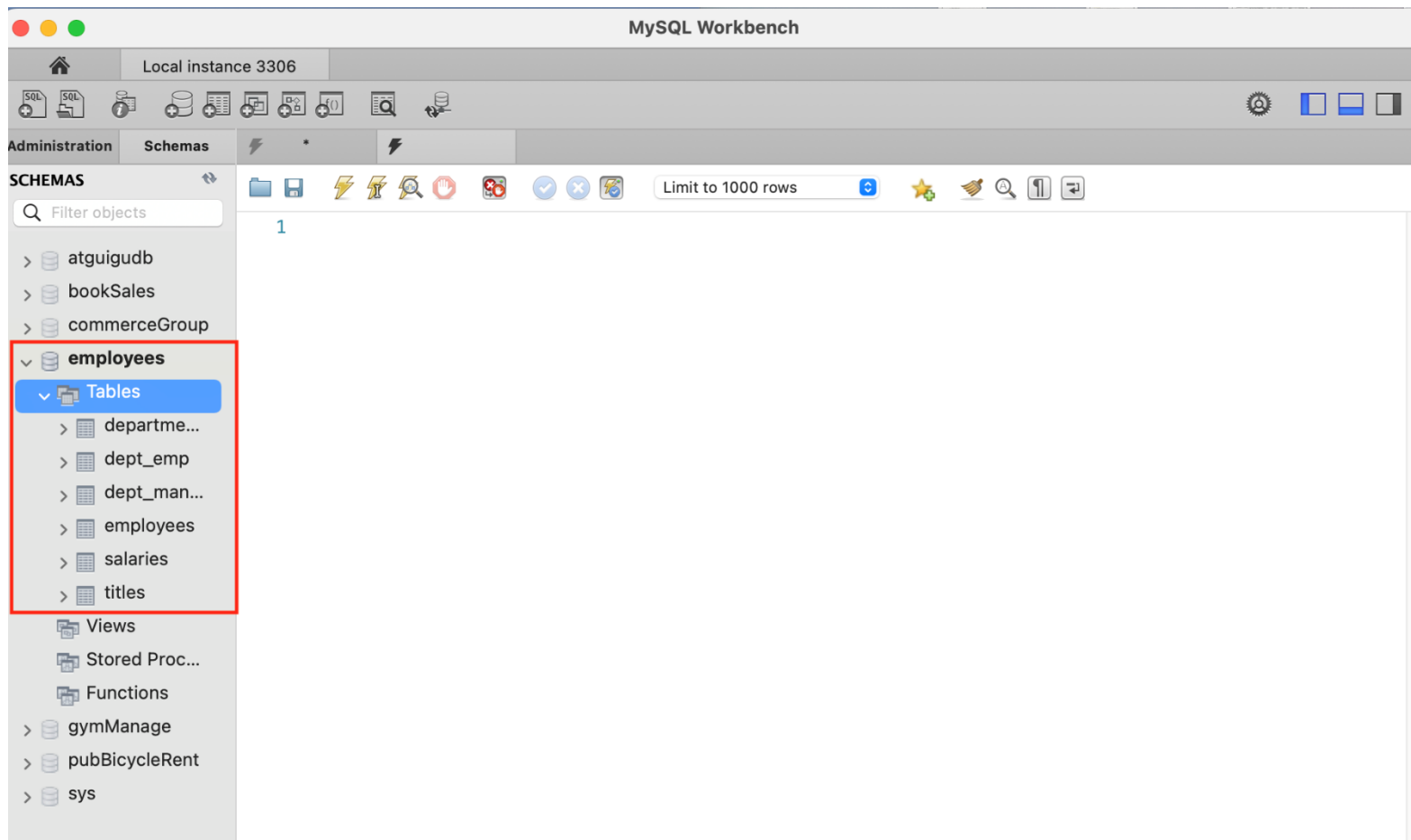
# 1. 数据入库操作

1.1 在Workbench 的左侧栏：**SCHEMAS** 中找到刚刚创建的数据库，  
双击激活数据库。



# 1. 数据入库操作

## 1.2 在Workbench 的左侧栏：点击 **Tables**，查看数据库中已经创建好的表



# 1. 数据入库操作

## 1.3 数据入库方式 1：直接在表中填数据

① 点击 **表格样** 按钮，点击**新增一行**，点击**Edit**，在表格中**编辑数据**，最后点击**Apply**

1. Clicking the 'Table' icon in the sidebar.

2. Clicking the 'Add Row' icon in the toolbar.

3. Clicking the 'Edit' icon in the toolbar.

4. Clicking the 'Apply' button in the bottom toolbar.

dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales
	NULL

# 1. 数据入库操作

## 1.3 数据入库方式 1：直接在表中填数据

② workbench 会**自动生成插入语句**，点击 Apply，会自动执行语句，最后点击finish。

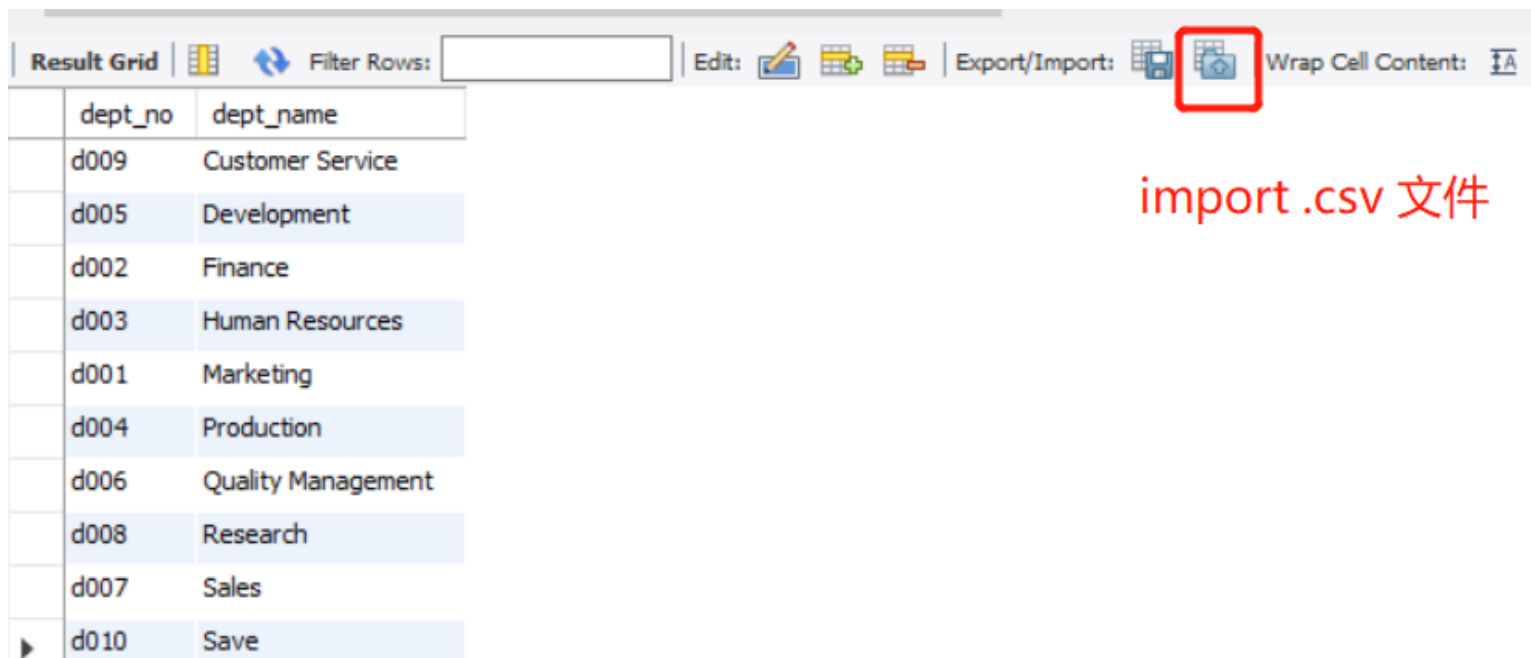




# 1. 数据入库操作

## 1.3 数据入库方式 2：导入Excel 的 .CSV 格式数据文件

将需要入表的数据填入 Excel 中，并另存为 .CSV 格式的数据文件，之后点击 import ，选择此 CSV 文件，导入数据。



The screenshot shows a software interface with a table of department data and a toolbar. The table has two columns: 'dept\_no' and 'dept\_name'. The data rows are as follows:

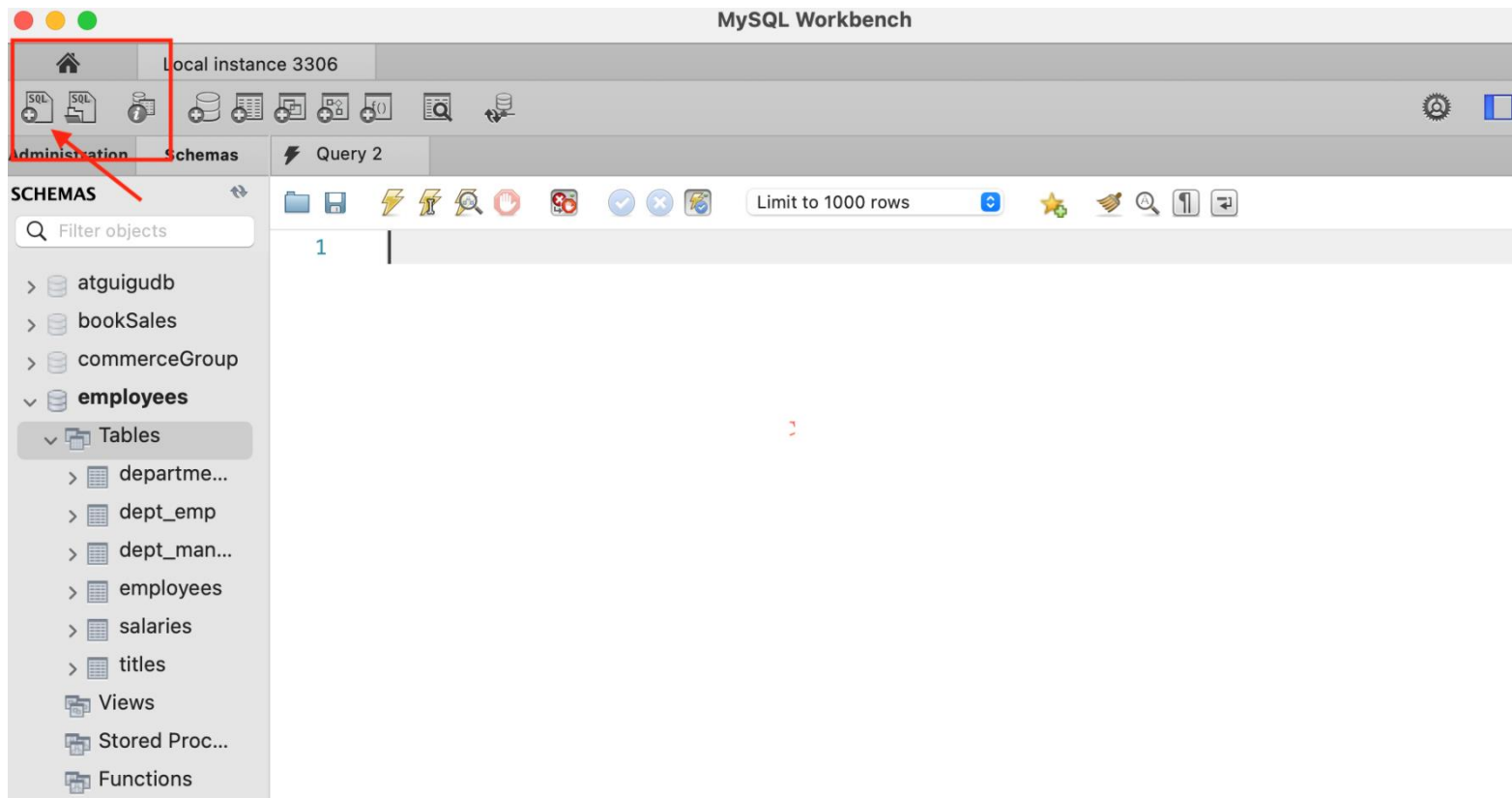
dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales
d010	Save

The toolbar includes buttons for 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The 'Export/Import' button, which features a document icon with a plus sign, is highlighted with a red box. Below the toolbar, the text 'import .csv 文件' is written in red.

# 1. 数据入库操作

## 1.3 数据入库方式 3：自己写插入语句

① 在Workbench 的左侧栏：点击带加号的SQL图标，新建 Query 文件



# 1. 数据入库操作

## 1.3 数据入库方式 3：自己写插入语句

② (可选) 数据入库操作前，如果遗忘表格式，键入DESC 表名语句，查看表格式。

The screenshot shows the MySQL Workbench interface. The 'Query 2' tab is active, and the SQL editor contains the command `DESC departments;`. A red box highlights this command, with a red arrow pointing to the lightning bolt icon in the toolbar, labeled '2: 点击闪电图标, 执行sql语句'. The 'SCHEMAS' sidebar on the left shows the 'employees' database selected, with the 'Tables' folder expanded. Below the SQL editor, the 'Result Grid' is displayed, showing the table structure for 'departments'. A red box highlights the first two columns of the result grid: 'Field' and 'Type'.

2: 点击闪电图标, 执行sql语句

1 • `DESC departments;`

3: 查看结果, 第一列为属性名, 第二列为属性类型

Field	Type	Null	Key	Default	Extra
dept_no	char(4)	NO	PRI	HULL	
dept_name	varchar(40)	NO	UNI	HULL	

# 1. 数据入库操作

## 1.3 数据入库方式 3：自己写插入语句

③ 在 Query 文件中，执行 INSERT 语句，进行数据入库操作。

The screenshot shows a database query editor interface. At the top, a toolbar includes icons for saving, running, and other actions, along with a 'Limit to 1000 rows' dropdown. Below the toolbar, a query editor window displays the following SQL statement:

```
1 • INSERT INTO departments VALUES('d010','Software Engineering');
```

Below the query editor, a red label indicates the next step: "1: 执行INSERT语句".

At the bottom of the interface, the 'Action Output' section shows a table of execution results. A red label indicates the next step: "2: 检查INSERT语句是否成功执行，看到绿色小对号代表数据入库成功". A red arrow points from this label to the green checkmark in the 'Action Output' table.

	Time	Action	Response	Duration / Fetch Time
✓	16:36:49	select * from departments LIMIT 0, 1000	9 row(s) returned	0.034 sec / 0.00029...
✓ 2	16:37:08	DESC departments	2 row(s) returned	0.0050 sec / 0.00001...
✓ 3	16:41:56	DESC departments	2 row(s) returned	0.030 sec / 0.000025...
✓ 4	16:49:15	SELECT * FROM departments LIMIT 0, 1000	9 row(s) returned	0.027 sec / 0.00045...
✓ 5	16:51:29	INSERT INTO departments VALUES('d010','Softw...	1 row(s) affected	0.012 sec



Part2

**建立并测试视图**



## 2. 建立并测试视图

### 2.1 视图相关语法

- **Why?** 把要保密的数据对无权存取这些数据用户隐藏起来，对数据提供一定程度的安全保护。

- 创建视图：

**CREATE VIEW** 视图名称

**AS** 查询语句;

- 查看视图：

**SHOW TABLES;**

- 查看视图详细定义信息：

**SHOW CREATE VIEW** 视图名称;

- 删除视图：

**DROP VIEW IF EXISTS** 视图名称;

## 2. 建立并测试视图

### 2.2 创建视图操作示意

举例：在表 employees 上建立视图 emp\_name ( emp\_no, first\_name, last\_name )

The screenshot shows a database query editor window titled "Query 2". The query text is as follows:

```
1 CREATE VIEW emp_name
2 AS
3 SELECT emp_no, first_name, last_name
4 FROM employees;
5
```

A red box highlights the query text, and a red arrow points to the "Execute" button (lightning bolt icon) in the toolbar. To the right of the query, red text reads: "1. 写入语句，选中并点击闪电执行".

Below the query editor, the "Action Output" tab is selected, displaying a table of execution results:

	Time	Action	Response	Duration / Fetch Time
3	16:41:56	DESC departments	2 row(s) returned	0.030 sec / 0.000025...
4	16:49:15	SELECT * FROM departments LIMIT 0, 1000	9 row(s) returned	0.027 sec / 0.00045...
5	16:51:29	INSERT INTO departments VALUES('d010','So...	1 row(s) affected	0.012 sec
6	17:20:39	SELECT * FROM dept_emp LIMIT 0, 1000	1000 row(s) returned	0.049 sec / 0.0011 sec
7	17:21:43	SELECT * FROM employees LIMIT 0, 1000	1000 row(s) returned	0.012 sec / 0.0016 sec
8	17:28:11	CREATE VIEW emp_name AS SELECT emp_no...	0 row(s) affected	0.014 sec

A red box highlights the last row of the "Action Output" table, which corresponds to the successful execution of the CREATE VIEW statement.

2. 检查创建视图语句是否成功

## 2. 建立并测试视图

### 2.3 在创建的视图上进行检查

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' sidebar is expanded to show the 'employees' database, with the 'Views' folder selected and 'emp\_name' view highlighted. In the center, the 'Query 2' editor contains the SQL statement: `SELECT * FROM emp_name;`. On the right, the 'Result Grid' displays the query results in a table format.

emp_no	first_name	last_name
10001	Georgi	Facello
10002	Bezael	Simmel
10003	Parto	Bamford
10004	Chirstian	Koblick
10005	Kyoichi	Maliniak
10006	Anneke	Preusig
10007	Tzvetan	Zielinski
10008	Saniya	Kalloufi
10009	Sumant	Peac
10010	Duangkaew	Piveteau

2. 执行select语句，在视图上进行查询操作

3. 在结果区查看在视图上查询的结果

1. 左侧点击views可以看到创建的视图



## 2. 建立并测试视图

在视图上进一步定义存取权限

```
GRANT SELECT    /*王平只能检索学生的信息 */  
ON SE_Student  
TO 王平;
```

```
GRANT ALL PRIVILIGES /*张明具有检索、增删改的所有权限 */  
ON SE_Student  
TO 张明;
```



## Part3

### 增删改查操作测试



### 3.1 插入操作

#### 3.1.1 插入数据语法

- 方式一：每次插入一条记录，操作步骤参考数据入库部分
- 方式二：同时插入多条记录：

INSERT INTO table\_name

VALUES

(value1 [,value2, ..., valuen]),

(value1 [,value2, ..., valuen]),

.....

(value1 [,value2, ..., valuen]);

## 3. 增删改查操作测试

### 3.1 插入操作

#### 3.1.2 插入数据举例

方式一：见数据入库部分

方式二：

The screenshot shows a database query tool interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL query is entered in a text area, highlighted with a red box. The query is:

```
1 • INSERT INTO departments(dept_no, dept_name)
2 VALUES ('d011','BigData'),
3 ('d012','UI'),
4 ('d013','Law');
```

Below the query, there's a red label: "1.一次性插入多条记录sql语句".

At the bottom of the interface, there's a section labeled "Action Output". A red arrow points to a row in this section, which is also highlighted with a red box. The row contains the following information:

	Time	Action	Response	Duration / Fetch Time
1	19:00:55	INSERT INTO departments(dept_no, dept_name)...	3 row(s) affected Records: 3 Duplicates: 0 Warnings...	0.011 sec

Below the "Action Output" section, there's a red label: "2.查看数据成功入库".

### 3.2 更新操作

#### 3.2.1 更新数据语法

➤ 方式：UPDATE语句

UPDATE table\_name

SET column1=value1, column2=value2, ... , column=value

[WHERE condition];

- 使用WHERE子句指定需要更新的数据，如果省略WHERE子句，表中所有的数据都将被更新

## 3. 增删改查操作测试

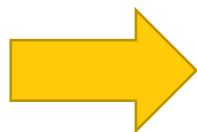
### 3.2 更新操作

#### 3.2.2 更新数据示例

- 需求：将部门号为d005的部门名称改为'Develop'

原数据

	dept_no	dept_name
	d011	BigData
	d009	Customer Service
▶	d005	Development
	d002	Finance
	d003	Human Resources
	d013	Law



	dept_no	dept_name
▶	d011	BigData
	d009	Customer Service
	d005	Develop
	d002	Finance
	d003	Human Resources
	d013	Law

SQL语句：

UPDATE departments

SET dept\_name = 'Develop'

WHERE dept\_no = 'd005';

## 3. 增删改查操作测试

### 3.3 删除操作

#### 3.3.1 删除数据语法

**DELETE FROM** table\_name [**WHERE** <condition>];

table\_name指定要执行删除操作的表；“[WHERE]”为可选参数，指定删除条件，如果没有WHERE子句，DELETE语句将删除表中的所有记录

- 需求：删除大数据部门这条记录

**DELETE FROM** departments **WHERE** dept\_name = 'Bigdata' ;

删除前：

	dept_no	dept_name
▶	d011	BigData
	d009	Customer Service
	d005	Develop
	d002	Finance
	d003	Human Resources
	d013	Law
	d001	Marketing
	d004	Production



删除后：

	dept_no	dept_name
▶	d009	Customer Service
	d005	Develop
	d002	Finance
	d003	Human Resources
	d013	Law
	d001	Marketing

## 3. 增删改查操作测试

### 3.4 查询操作

#### 3.4.1 查询数据基本语法

**SELECT** 字段1, 字段2

**FROM** 表名

**WHERE** 过滤条件

- 需求：查询员工id为10001在公司任职期间，工资变动情况

**SELECT** salary

**FROM** salaries

**WHERE** emp\_no = '10001';

查询结果展示：

salary
60117
62102
66074
66596
66961
71046
74333
75286
75994
76884
80013
81025
81097





## Part4

### 建立并测试触发器



## 4. 建立并测试触发器

- 触发器 (Trigger) 是用户定义在关系表上的一类由事件驱动的特殊过程。
  - 任何用户对表的增、删、改操作均由服务器自动激活相应的触发器。
- 优点：可以实施更为复杂的检查和操作，具有更精细和更强大的数据控制能力。
- 缺点：触发器是很强大的工具，但在使用时要慎重。因为在每次访问一个表时，都可能触发一个触发器，这样会影响系统的性能。
- 触发器又叫做事件-条件-动作 (event-condition-action) 规则。

## 4. 建立并测试触发器

### 4.1 触发器建立语法

**CREATE TRIGGER** 触发器名称

**{BEFORE|AFTER} {INSERT|UPDATE|DELETE} ON 表名**

**FOR EACH ROW**

触发器执行的语句块;

- 表名：触发器监控对象
- 触发器执行的语句块：可以是单条SQL，也可以是由BEGIN...END结构组成的复合语句块。
- **BEFORE | AFTER：触发时间（执行前触发 or 执行后触发）**
  - INSERT：插入记录时触发
  - UPDATE：更新记录时触发
  - DELETE：删除记录时触发

## 4. 建立并测试触发器

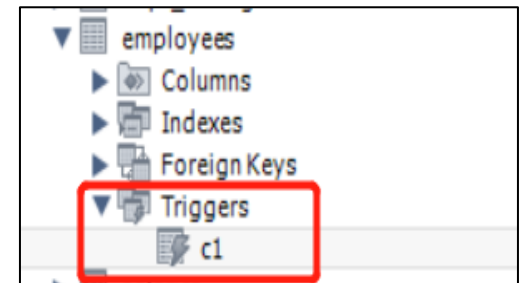
### 4.2 实际应用 1: Delete Trigger

#### ① 需求分析:

- 员工编号 emp\_no=1 的员工离职了,
- 我们在 员工信息表 employees 中删除该员工信息前, 还需要将部门员工表 dept\_emp 中删除该员工的工作信息

➤ 基于上述需求, 我们可以在 employees 表上建立一个 名为c1的 前触发: 写相应创建触发器的 sql 语句并执行, 在employees下的triggers中可查看该触发器 c1

```
DELIMITER //  
create trigger c1  
  before delete on employees  
  for each row  
begin  
  delete from dept_emp where dept_emp.emp_no=OLD.emp_no;  
END  
//DELIMITER ;
```



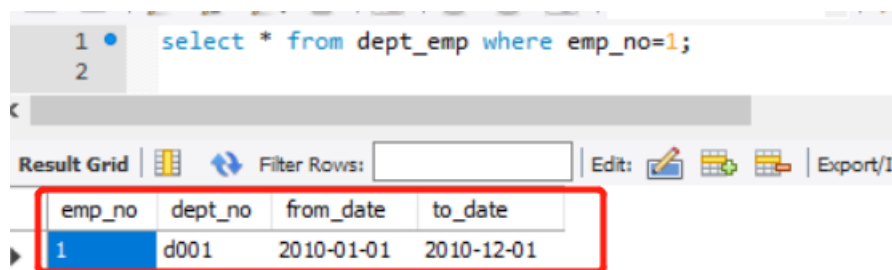
## 4. 建立并测试触发器

### 4.2 实际应用 1: Delete Trigger

#### ② 测试触发器c1:

- 在 employees 表中删除 emp\_no=1 前，在 emp\_dept 中查询其信息，能够查询到其记录：

```
1 select * from dept_emp where emp_no=1;
```

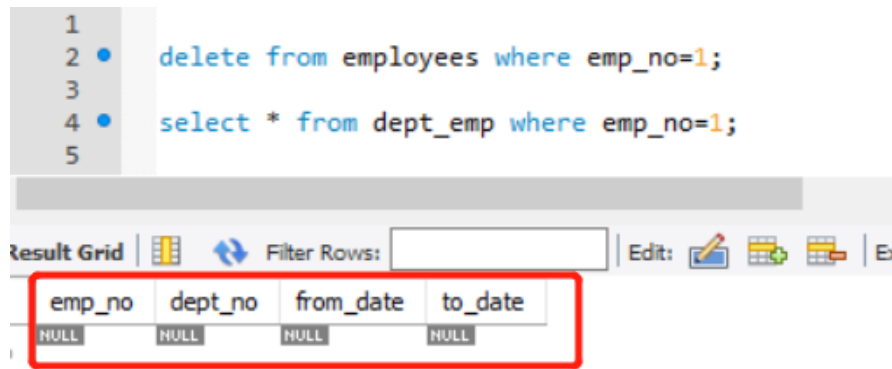


The screenshot shows a database query result grid. The query is 'select \* from dept\_emp where emp\_no=1;'. The result grid has columns: emp\_no, dept\_no, from\_date, and to\_date. The first row contains the values 1, d001, 2010-01-01, and 2010-12-01. The first row is highlighted with a blue background, and the entire grid is enclosed in a red border.

emp_no	dept_no	from_date	to_date
1	d001	2010-01-01	2010-12-01

- 在 employees 表中删除 emp\_no=1 后，在 emp\_dept 中查询其信，发现其信息也被删除了：

```
1  
2 delete from employees where emp_no=1;  
3  
4 select * from dept_emp where emp_no=1;  
5
```



The screenshot shows a database query result grid. The query is 'select \* from dept\_emp where emp\_no=1;'. The result grid has columns: emp\_no, dept\_no, from\_date, and to\_date. The first row contains the values NULL, NULL, NULL, and NULL. The first row is highlighted with a blue background, and the entire grid is enclosed in a red border.

emp_no	dept_no	from_date	to_date
NULL	NULL	NULL	NULL

## 4. 建立并测试触发器

### 4.3 实际应用 2: Insert Trigger

#### ① 需求分析:

- 有如下两张表: test\_trigger 和 test\_trigger\_log:

1) test\_trigger 数据表:

```
CREATE TABLE test_trigger(  
id INT PRIMARY KEY AUTO_INCREMENT,  
t_note VARCHAR(30)  
);
```

2) test\_trigger\_log 数据表:

```
CREATE TABLE test_trigger_log(  
id INT PRIMARY KEY AUTO_INCREMENT,  
t_log VARCHAR(30)  
);
```

- 向 test\_trigger 数据表插入数据之前, 需要向 test\_trigger\_log 数据表中插入 before\_insert 的日志信息。

## 4. 建立并测试触发器

### 4.3 实际应用 2: Insert Trigger

② 创建触发器：创建名称为 before\_insert 的触发器，向 test\_trigger 数据表插入数据之前，向 test\_trigger\_log 数据表中插入 before\_insert 的日志信息。

```
DELIMITER //
```

```
CREATE TRIGGER before_insert
```

```
BEFORE INSERT ON test_trigger
```

```
FOR EACH ROW
```

```
BEGIN
```

```
INSERT INTO test_trigger_log (t_log)
```

```
VALUES('before_insert');
```

```
END //
```

```
DELIMITER ;
```

**DELIMITER**，即改变输入结束符。默认情况下，delimiter是分号“;”。  
在命令行客户端中，如果有一行命令以分号结束，那么回车后，mysql将会执行该命令。  
但有时候，不希望MySQL这么做。  
因为可能输入较多的语句，且语句中包含有分号。  
这种情况下，就可以使用delimiter，把delimiter后面换成其它符号，如//

workbench界面展示：

```
12 DELIMITER //
```

```
13 • CREATE TRIGGER before_insert
```

```
14 BEFORE INSERT ON test_trigger
```

```
15 FOR EACH ROW
```

```
16 ⊖ BEGIN
```

```
17   INSERT INTO test_trigger_log (t_log)
```

```
18   VALUES('before_insert');
```

```
19   END //
```

```
20 DELIMITER ;
```

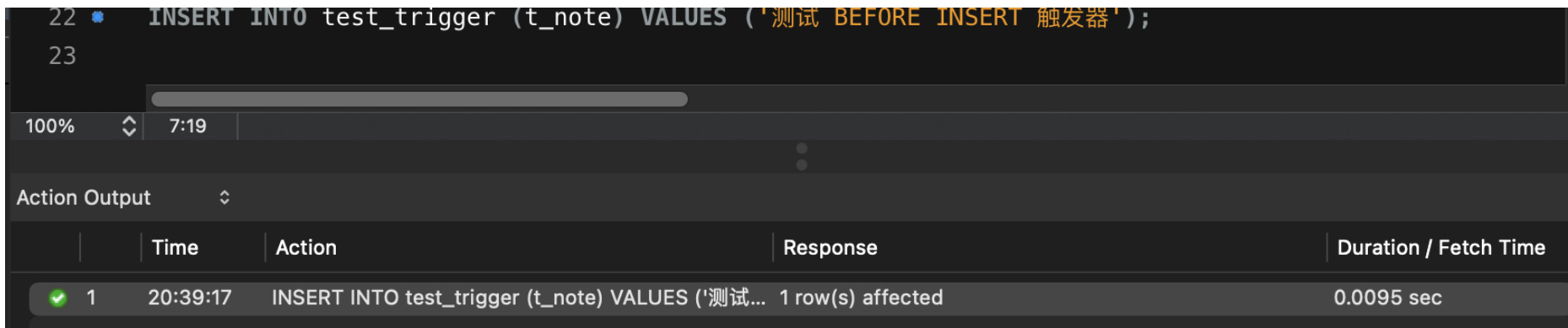
## 4. 建立并测试触发器

### 4.3 实际应用 2: Insert Trigger

③ 向test\_trigger数据表中插入数据:

```
INSERT INTO test_trigger (t_note) VALUES ('测试 BEFORE INSERT 触发器');
```

workbench界面展示:



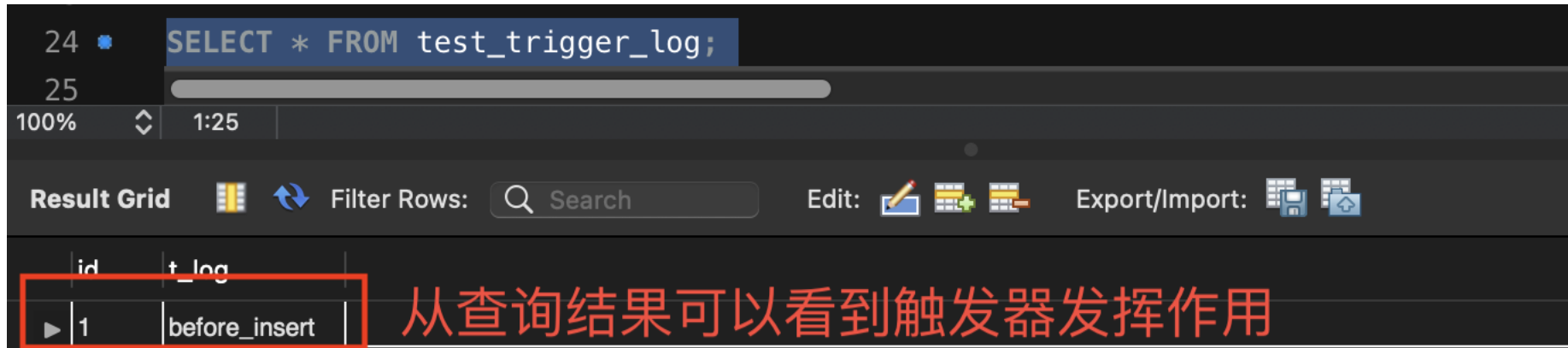


## 4. 建立并测试触发器

### 4.3 实际应用 2: Insert Trigger

④ 查看test\_trigger\_log数据表中的数据:

```
SELECT * FROM test_trigger_log;
```



24 `SELECT * FROM test_trigger_log;`

25

100% 1:25

Result Grid Filter Rows: Search Edit: Export/Import:

	id	t_log
▶	1	before_insert

从查询结果可以看到触发器发挥作用



Thank you!