

0x01 数据库的区别

参考链接

0x02 关系型数据库学习

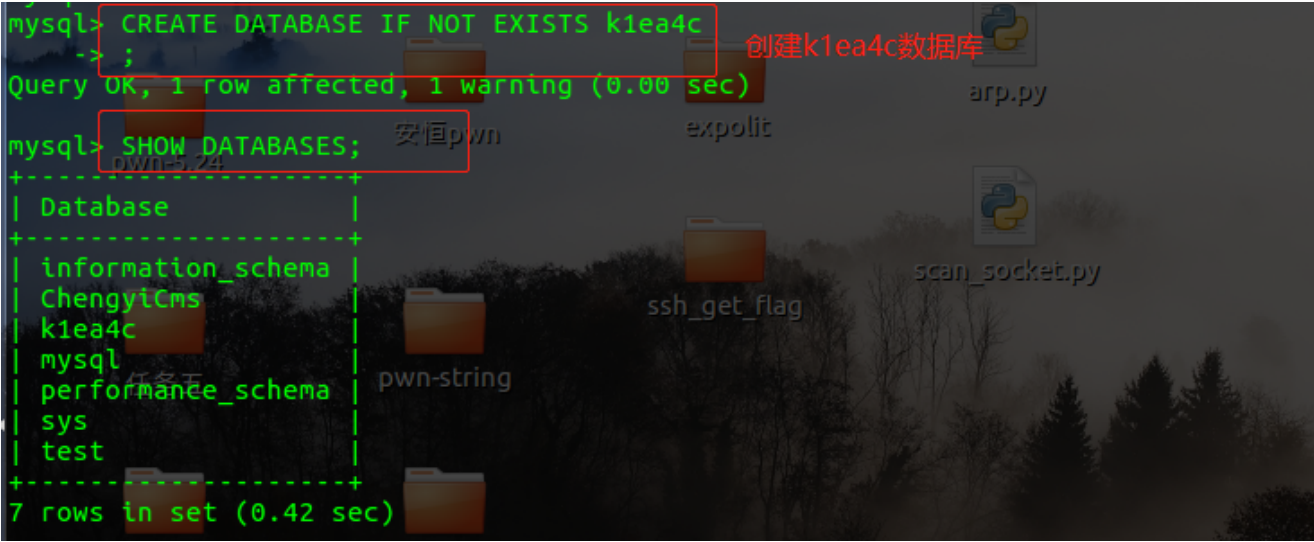
DB, DBMS（数据库管理系统：MySQL, Oracle（贵！性能高）, DB2（适合处理海量数据）, SqlServer（只适合windows系统）等），SQL（结构化查询语句）

MySQL基本操作

DDL（数据定义语言）

- 数据库管理

SQL语句	功能
CREATE DATABASE 数据库名;	创建数据库
USE 数据库名;	选择使用的数据库
SHOW DATABASES;	查看服务器中有哪些数据库
DROP DATABASE 数据库名	删除数据库



- 数据表管理

SQL语句	功能
CREATE TABLE 表名 (字段设定列表);	在当前数据库中创建数据表
SHOW TABLES;	显示当前数据库中有哪些数据表
DESCRIBE (数据库名.) 表名;	显示当前或指定数据库中指定数据表的结构(字段)信息
DROP TABLE (数据库名.) 表名;	删除当前或指定数据库中指定数据表
ALTER TABLE 表名 MODIFY 字段名 字段类型;	修改字段类型
ALTER TABLE 表名 ADD 字段名 字段类型;	增加表字段
ALTER TABLE 表名 DROP 字段名;	删除表字段
ALTER TABLE 表名 CHANGE 旧字段名 新字段名 字段类型	字段改名
ALTER TABLE 表名 RENAME 新表名	更改表名

```

mysql> use k1ea4c;
Database changed
mysql> CREATE TABLE classes (
->     id BIGINT NOT NULL AUTO_INCREMENT,
->     name VARCHAR(100) NOT NULL,
->     PRIMARY KEY (id)
-> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.12 sec)

mysql> CREATE TABLE students (
->     id BIGINT NOT NULL AUTO_INCREMENT,
->     class_id BIGINT NOT NULL,
->     name VARCHAR(100) NOT NULL,
->     gender VARCHAR(1) NOT NULL,
->     score INT NOT NULL,
->     PRIMARY KEY (id)
-> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.05 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_k1ea4c |
+-----+
| classes          |
| students         |
+-----+
2 rows in set (0.00 sec)

```

创建classes和students数据库

```
mysql> DESCRIBE classes;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra           |
+-----+-----+-----+-----+-----+-----+
| id     | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| name   | varchar(100)  | NO   |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.03 sec)
```

表结构

```
mysql> DESCRIBE students;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra           |
+-----+-----+-----+-----+-----+-----+
| id     | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| class_id | bigint(20)    | NO   |     | NULL    |                 |
| name   | varchar(100)  | NO   |     | NULL    |                 |
| gender | varchar(1)    | NO   |     | NULL    |                 |
| score  | int(11)       | NO   |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

DML（数据操作语句）

- 修改数据
 - 插入记录

```
INSERT INTO 表名(field1,field2..) VALUES(value1,value2..);
```

```
INSERT INTO 表名 VALUES(value1,value2..);
```

插入多条记录

```
INSERT INTO 表名 (field1,field2..)
VALUES
(record1_value1,record1_value2...record1_valuen),
...
(recordn_value1,recordn_value2...recordn_valuen)
;
```

```
mysql> INSERT INTO classes(id, name) VALUES (1, '一班');
Query OK, 1 row affected (0.46 sec)
```

```
mysql> INSERT INTO classes VALUES (2, '二班');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from classes;
```

```
+-----+-----+
| id | name |
+-----+-----+
| 1  | 一班 |
| 2  | 二班 |
+-----+-----+
```

```
2 rows in set (0.00 sec)
```

单条数据插入

```
mysql> INSERT INTO classes(id,name)
VALUES
-> (3,'三班'),
-> (4,'四班')
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> select * from classes;
+----+-----+
| id | name |
+----+-----+
| 1  | 一班 |
| 2  | 二班 |
| 3  | 三班 |
| 4  | 四班 |
+----+-----+
4 rows in set (0.00 sec)
```

多条记录插入

更新记录

UPDATE 表名 **SET** field1=values,field2=value2..fieldn=valuen[**WHERE** **CONDITION**]

```
mysql> INSERT INTO classes(id,name) VALUE(5,'六班');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM classes WHERE id=5;
+----+-----+
| id | name |
+----+-----+
| 5  | 六班 |
+----+-----+
1 row in set (0.00 sec)

mysql> UPDATE classes SET name='五班' WHERE id=5;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT * FROM classes WHERE id=5;
+----+-----+
| id | name |
+----+-----+
| 5  | 五班 |
+----+-----+
1 row in set (0.00 sec)
```

删除记录

DELETE FROM 表名 [**WHERE** **CONDITION**]

查询数据

基础查询

SQL语句	功能
SELECT 字段名 FROM 表名;	查询列表
SELECT * FROM 表名;	查询所有
SELECT 字段名 (AS) 别名 FROM 表名	起别名
SELECT DISTINCT 字段名 FROM 表名	去重
SELECT CONCAT(多个查询列表) FROM 表名	拼接

○ 条件查询

SELECT 查询列表 FROM 表名 WHERE 筛选条件

简单条件运算符: > < = != <> >= <=

逻辑运算符: &&(and) ||(or) !(not)

like

SELECT 查询列表 FROM 表名 LIKE '_1e%'

% 任意多个字符, 包含0个字符

_ 任意单个字符

between and (包含临界值)

SELECT 查询列表 FROM 表名 BETWEEN 1 AND 9

in

SELECT 查询列表 FROM 表名 IN ('hello' , 'k1ea4c')

is null (等于号不能判断null值)

SELECT 查询列表 FROM 表名 查询列表 IS NULL

○ 排序查询

SQL语句	功能
SELECT 查询列表 FROM 表名 WHERE 筛选条件 ORDER BY 排序列表 ASC	升序(默认)
SELECT 查询列表 FROM 表名 WHERE 筛选条件 ORDER BY 排序列表 DESC	降序

```
mysql> SELECT * FROM students ORDER BY score ASC;
```

id	class_id	name	gender	score
6	2	小兵	M	55
4	1	小米	F	73
5	2	小白	F	81
7	2	小林	M	85
10	3	小丽	F	85
3	1	小军	M	88
9	3	小王	M	89
1	1	小明	M	90
8	3	小新	F	91
2	1	小红	F	95

```
10 rows in set (0.00 sec)
```

分页查询

```
SELECT * FROM userinfo LIMIT 3 OFFSET 0;
```

结果集分页，每页3条记录。要获取第1页的记录

```
SELECT * FROM userinfo LIMIT 0,3;
```

```
mysql> SELECT * FROM students ORDER BY score LIMIT 3 OFFSET 0;
```

id	class_id	name	gender	score
6	2	小兵	M	55
4	1	小米	F	73
5	2	小白	F	81

```
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM students ORDER BY score LIMIT 0,3;
```

id	class_id	name	gender	score
6	2	小兵	M	55
4	1	小米	F	73
5	2	小白	F	81

```
3 rows in set (0.00 sec)
```

聚合查询

```
SELECT COUNT(*) num FROM userinfo;
```

聚合函数

SUM 计算某一列的合计值，该列必须为数值类型
AVG 计算某一列的平均值，该列必须为数值类型
MAX 计算某一列的最大值
MIN 计算某一列的最小值

聚合查询的WHERE条件没有匹配到任何行，COUNT()会返回0，而SUM()、AVG()、MAX()和MIN()会返回NULL


```
mysql> SELECT SUM(score) 总成绩 FROM students;
+-----+
| 总成绩 |
+-----+
|      832 |
+-----+
1 row in set (0.00 sec)
```

○ 分组查询

SELECT 查询列表, **COUNT**(*) num **FROM** 表名 **GROUP BY** 查询列表;

```
mysql> SELECT gender, COUNT(*) FROM students GROUP BY gender;
+-----+-----+
| gender | COUNT(*) |
+-----+-----+
| F      | 5        |
| M      | 5        |
+-----+-----+
2 rows in set (0.00 sec)
```

获取男女的人数

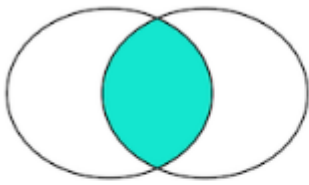
○ 连接查询

INNER JOIN (内连接, 或等值连接): 获取两个表中字段匹配关系的记录。

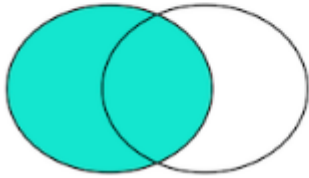
LEFT JOIN (左连接): 获取左表所有记录, 即使右表没有对应匹配的记录。

RIGHT JOIN (右连接): 与 **LEFT JOIN** 相反, 用于获取右表所有记录, 即使左表没有对应匹配的记录。

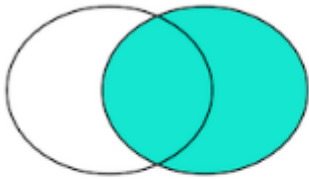
我们把tableA看作左表，把tableB看成右表，那么INNER JOIN是选出两张表都存在的记录：



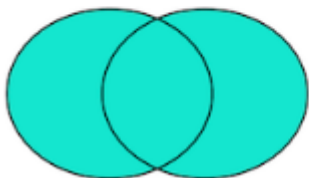
LEFT OUTER JOIN是选出左表存在的记录：



RIGHT OUTER JOIN是选出右表存在的记录：



FULL OUTER JOIN则是选出左右表都存在的记录：



```
mysql> SELECT s.id, s.name, s.class_id, c.name class_name, s.gender, s.score FROM students s LEFT JOIN classes c ON s.class_id = c.id;
```

id	name	class_id	class_name	gender	score
1	小明	1	一班	M	90
2	小红	1	一班	F	95
3	小军	1	一班	M	88
4	小米	1	一班	F	73
5	小白	2	二班	F	81
6	小兵	2	二班	M	55
7	小林	2	二班	M	85
8	小新	3	三班	F	91
9	小王	3	三班	M	89
10	小丽	3	三班	F	85

10 rows in set (0.01 sec)

```
mysql> SELECT s.id, s.name, s.class_id, c.name class_name, s.gender, s.score FROM students s RIGHT JOIN classes c ON s.class_id = c.id;
```

id	name	class_id	class_name	gender	score
1	小明	1	一班	M	90
2	小红	1	一班	F	95
3	小军	1	一班	M	88
4	小米	1	一班	F	73
5	小白	2	二班	F	81
6	小兵	2	二班	M	55
7	小林	2	二班	M	85
8	小新	3	三班	F	91
9	小王	3	三班	M	89
10	小丽	3	三班	F	85
NULL	NULL	NULL	四班	NULL	NULL

11 rows in set (0.00 sec)

○ UNION合并查询

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2
```

DCL（数据库控制）

GRANT命令说明：

ALL PRIVILEGES 是表示所有权限，你也可以使用select、update等权限。

ON 用来指定权限针对哪些库和表。

.* 中前面的*号用来指定数据库名，后面的*号用来指定表名。

TO 表示将权限赋予某个用户。

test@'localhost' 表示test用户，@后面接限制的主机，可以是IP、IP段、域名以及%，%表示任何地方。

IDENTIFIED BY 指定用户的登录密码。

WITH GRANT OPTION 这个选项表示该用户可以将自己拥有的权限授权给别人。注意：经常有人在创建操作用户的时候不指定WITH GRANT OPTION选项导致后来该用户不能使用GRANT命令创建用户或者给其他用户授权。

ALTER:	修改表和索引
CREATE:	创建数据库和表
DROP:	抛弃（删除）数据库和表
INDEX:	创建或抛弃索引
INSERT:	向表中插入新行
SELECT:	检索表中的记录
UPDATE:	修改现存表记录
FILE:	读或写服务器上的文件
PROCESS:	查看服务器中执行的线程信息或杀死线程
RELOAD:	重载授权表或清空日志。主机缓存或表缓存
SHUTDOWN:	关闭服务器
ALL:	所有权限，ALL PRIVILEGES同义词
USAGE:	特殊的“无权限”权限

```
mysql> SELECT user,host FROM user;
```

```
+-----+-----+
| user          | host          |
+-----+-----+
| root          | %             |
| debian-sys-maint | localhost    |
| mysql.session | localhost    |
| mysql.sys     | localhost    |
| root          | localhost    |
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> CREATE USER 'k1ea4c'@'%' IDENTIFIED BY 'k1ea4c';
Query OK, 0 rows affected (0.14 sec)
```

```
mysql> RENAME USER 'k1ea4c'@'%' TO 'K1EA4C'@'%' ;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT user,host FROM user;
```

```
+-----+-----+
| user          | host          |
+-----+-----+
| K1EA4C        | %             |
| root          | % pwnable_kr |
| debian-sys-maint | localhost    |
| mysql.session | localhost    |
| mysql.sys     | localhost    |
| root          | localhost    |
+-----+-----+
6 rows in set (0.00 sec)
```

创建K1EA4C新用户

```
mysql> GRANT select,delete,insert on *.* to 'K1EA4C'@'%';
Query OK, 0 rows affected (0.00 sec)
```

增加权限

```
mysql> SHOW GRANTS FOR 'K1EA4C'@'%';
+-----+
| Grants for K1EA4C@% |
+-----+
| GRANT SELECT, INSERT, DELETE ON *.* TO 'K1EA4C'@'%' |
+-----+
1 row in set (0.00 sec)
```

查看权限

```
mysql> REVOKE delete ON *.* FROM 'K1EA4C'@'%';
Query OK, 0 rows affected (0.00 sec)
```

取消权限

```
mysql> SHOW GRANTS FOR 'K1EA4C'@'%';
+-----+
| Grants for K1EA4C@% |
+-----+
| GRANT SELECT, INSERT ON *.* TO 'K1EA4C'@'%' |
+-----+
1 row in set (0.00 sec)
```

利用元数据库进行SQL注入（information_schema）

目标：

```
mysql> select * from userinfo;
+----+-----+-----+-----+
| id | username | password | email |
+----+-----+-----+-----+
| 3 | k1ea4c | 123 | 123@qq.com |
| 4 | hello | 123 | 123 |
+----+-----+-----+-----+
2 rows in set (0.00 sec)
```

目标数据

注入三部曲

```
mysql> select table_name from information_schema.tables where table_schema=database();
+-----+
| table_name |
+-----+
| userinfo  |
+-----+
1 row in set (0.00 sec)

mysql> select column_name from information_schema.columns where table_name='userinfo';
+-----+
| column_name |
+-----+
| id          |
| username    |
| password    |
| email       |
+-----+
4 rows in set (0.00 sec)

mysql> select username,password from userinfo;
+-----+-----+
| username | password |
+-----+-----+
| k1ea4d   | 123      |
| hello    | 123      |
+-----+-----+
2 rows in set (0.00 sec)
```

0x03 学习资料

[SQL注入](#)

[MySQL提权](#)