

【MySQL】SQL入门(一)

🚗 MySQL学习·起始站~
📌 本文已收录至专栏：[MySQL通关路](#)
❤️ 每章节附章节思维导图，文末附全文思维导图，感谢各位点赞收藏支持~
★ 学习汇总贴，超详细思维导图：[【MySQL】学习汇总\(完整思维导图\)](#)

一.引入

(1) SQL分类

SQL语句，根据其功能，主要分为四类：`DDL`、`DML`、`DQL`、`DCL`。

分类	全称	说明
DDL	Data Definition Language	数据定义语言，用来 定义数据库对象(数据库，表，字段)
DML	Data Manipulation Language	数据操作语言，用来 对数据库表中的数据进行增删改
DQL	Data Query Language	数据查询语言，用来 查询数据库中表的记录
DCL	Data Control Language	数据控制语言，用来 创建数据库用户、控制数据库的访问权限

注：后续SQL示例中**[可选内容]**表示可选内容，即可写可不写。

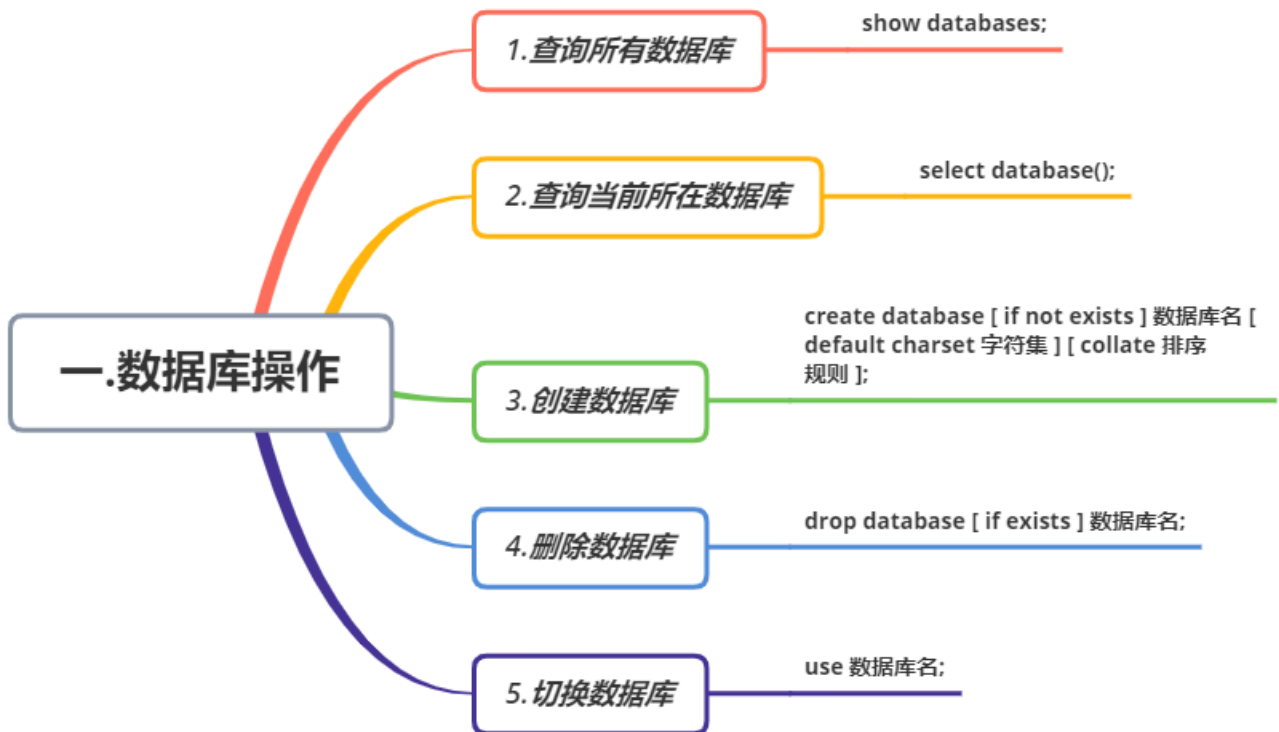
(2) 书写规则

- SQL语句可以**单行或多行书写，以分号结尾**。
- MySQL数据库的SQL语句**不区分大小写**，关键字建议使用大写。
- 在同一个数据库服务器中，**不能创建两个名称相同的数据库**，否则将会报错。
- 如果删除一个不存在的数据库，将会报错。
- 注释：
 - 单行注释**：`--` 注释内容 或 `#` 注释内容
 - 多行注释**：`/*` 注释内容 `*/`

二.DDL语句

Data Definition Language，数据定义语言，用来**定义数据库对象(数据库，表，字段)**

(1) 数据库操作



CSDN @观止study

- 查询所有数据库

```
show databases;
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| cloud_order |
| cloud_user |
| db1 |
| es-demo |
| hmdp |
| information_schema |
| my_db |
| my_db_1 |
| mybatis |
| mysql |
| performance_schema |
| practice |
| reggier |
| studentmessage |
| study |
| sys |
| web_test |
| yuapi |
| yupi |
+-----+
19 rows in set (0.06 sec)
```

- 查询当前所在数据库 (若之前未选择数据库则为null)

```
select database();
```

```
mysql> select database();
+-----+
| database() |
+-----+
| NULL      |
+-----+
1 row in set (0.00 sec)

mysql> use study;
Database changed
mysql> select database();
+-----+
| database() |
+-----+
| study      |
+-----+
1 row in set (0.00 sec)
```

- 创建数据库

```
create database [ if not exists ] 数据库名 [ default charset 字符集 ] [ collate 排序规则 ];
```

```
mysql> create database guanzhi;           创建名为guanzhi数据库
Query OK, 1 row affected (0.00 sec)

mysql> create database if not exists guanzhi;  如果名为guanzhi的数据库不存在则创建它
Query OK, 1 row affected, 1 warning (0.00 sec)  否则不创建

mysql> create database guanzhiA default charset utf8mb4;
Query OK, 1 row affected (0.00 sec)         创建字符集为utf8mb4名为guanzhiA数据库
```

- 删除数据库

```
drop database [ if exists ] 数据库名;
```

```
mysql> drop database guanzhi;           删除名为guanzhi的数据库
Query OK, 0 rows affected (0.00 sec)

mysql> drop database if exists guanzhiA;  如果名为guanzhi的数据库存在则删除
Query OK, 0 rows affected (0.00 sec)      CSDN @观止study
```

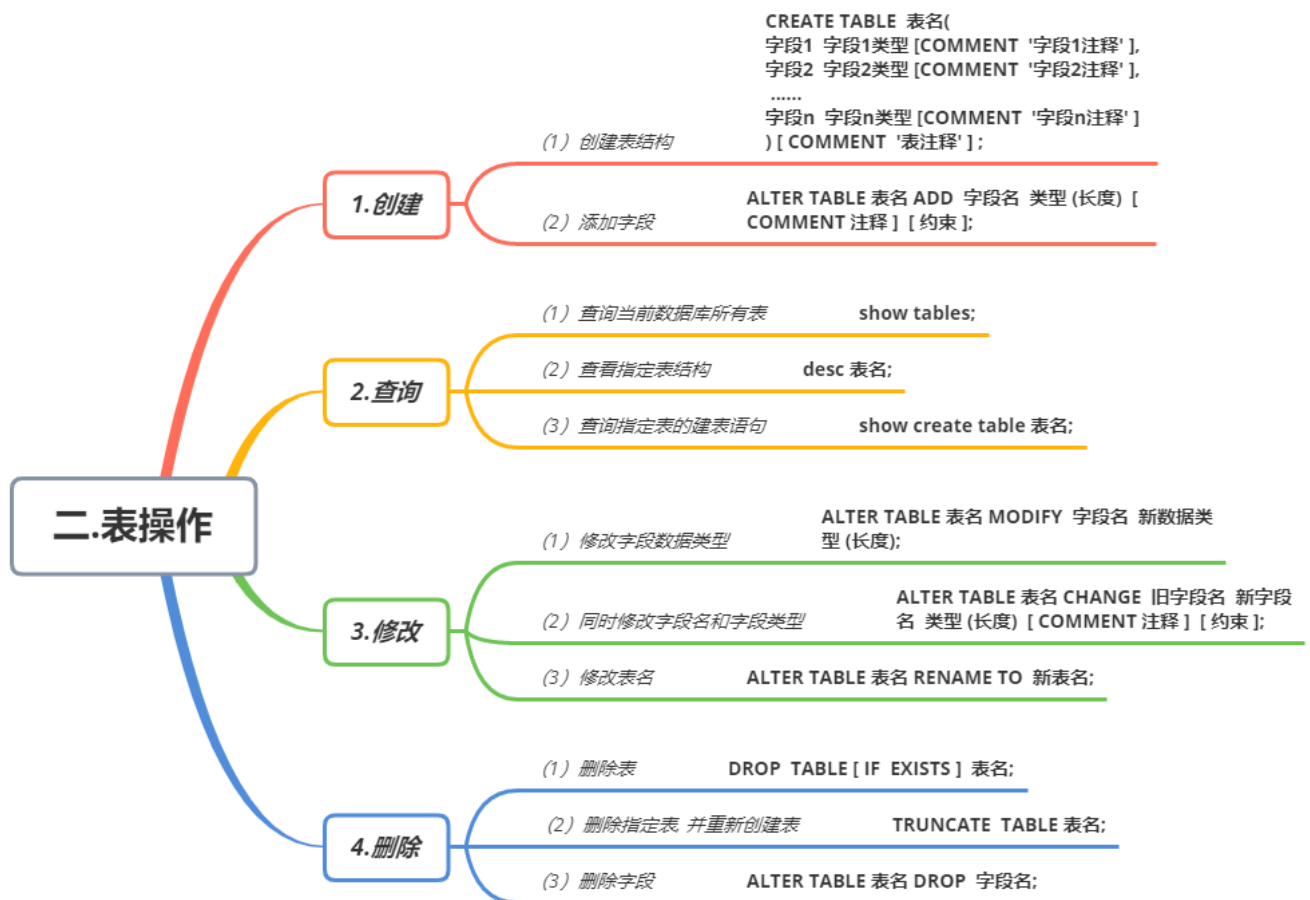
- 切换数据库（当我们要操作某一个数据库下的表时，就需要切换到对应的数据库下，否则是不能操作的）

```
use 数据库名;
```

```
mysql> select database();
+-----+
| database() |
+-----+
| NULL      |
+-----+
1 row in set (0.00 sec)

mysql> use study;
Database changed
mysql> select database();
+-----+
| database() |
+-----+
| study      |
+-----+
1 row in set (0.00 sec)
```

(2) 表操作



CSDN @观止study

(2.1) 创建

- 创建表结构

```
CREATE TABLE 表名(
  字段1 字段1类型 [COMMENT '字段1注释' ],
  字段2 字段2类型 [COMMENT '字段2注释' ],
  字段3 字段3类型 [COMMENT '字段3注释' ],
  .....
  字段n 字段n类型 [COMMENT '字段n注释' ]
) [ COMMENT '表注释' ] ;
# 注意:最后一个字段后面没有逗号
```

```
mysql> create table user(
  -> id int comment '编号',
  -> username varchar(50) comment '用户名',
  -> password varchar(50) comment '密码'
  -> ) comment '账号密码表';
Query OK, 0 rows affected (0.01 sec) CSDN @观止study
```

创建用户记录账号密码的表

- 添加字段

```
ALTER TABLE 表名 ADD 字段名 类型 (长度) [ COMMENT 注释 ] [ 约束 ] ;
```

```
mysql> desc user;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | YES  |     | NULL    |       |
| username | varchar(50) | YES  |     | NULL    |       |
| password | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> alter table user add vip int comment '是否为vip用户';
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0
为user表增加一个int类型的名为vip的字段

mysql> desc user;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | YES  |     | NULL    |       |
| username | varchar(50) | YES  |     | NULL    |       |
| password | varchar(50) | YES  |     | NULL    |       |
| vip   | int  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec) CSDN @观止study
```

(2.2) 查询

- 查询当前数据库所有表

```
show tables;
```

```
mysql> use study;
Database changed
mysql> show tables;
+-----+
| Tables_in_study |
+-----+
| tb_news          |
| tb_newstype      |
| tb_user          |
| user             |
+-----+
4 rows in set (0.00 sec)
```

切换到已创建数据库

查看数据库中所有表

- 查看指定表结构

`desc` 表名;

可以查看到指定表的字段，字段的类型、是否可以为NULL，是否存在默认值等信息。

```
mysql> desc user;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int           | YES  |     | NULL    |       |
| username | varchar(50)  | YES  |     | NULL    |       |
| password | varchar(50)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

CSDN @观止study

- 查询指定表的建表语句

`show create table` 表名;

可以用来查看建表语句的sql，其中部分参数我们在创建表的时候，并未指定也会查询到，因为这部分是数据库的默认值，如：存储引擎、字符集等。

```
mysql> show create table user;
+-----+-----+
| Table | Create Table
+-----+-----+
| user  | CREATE TABLE `user` (
  `id` int DEFAULT NULL COMMENT '编号',
  `username` varchar(50) DEFAULT NULL COMMENT '用户名',
  `password` varchar(50) DEFAULT NULL COMMENT '密码'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci COMMENT='账号密码表'
```

1 row in set (0.00 sec)

CSDN @观止study

(2.3) 修改

- 修改字段数据类型

```
ALTER TABLE 表名 MODIFY 字段名 新数据类型 (长度);
```

```
mysql> desc user;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | int           | YES  |     | NULL    |       |
| username   | varchar(50)   | YES  |     | NULL    |       |
| password   | varchar(50)   | YES  |     | NULL    |       |
| vip        | int          | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> alter table user modify vip varchar(20);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc user;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | int           | YES  |     | NULL    |       |
| username   | varchar(50)   | YES  |     | NULL    |       |
| password   | varchar(50)   | YES  |     | NULL    |       |
| vip        | varchar(20) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

将vip字段的类型改为varchar

CSDN @观止study

- 同时修改字段名和字段类型

```
ALTER TABLE 表名 CHANGE 旧字段名 新字段名 类型 (长度) [ COMMENT 注释 ] [ 约束 ];
```

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
username	varchar(50)	YES		NULL	
password	varchar(50)	YES		NULL	
<u>vip</u>	varchar(20)	YES		NULL	

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

```
mysql> alter table user change vip isVip int;
```

```
Query OK, 0 rows affected (0.03 sec)
```

```
Records: 0 Duplicates: 0 Warnings: 0
```

将字段vip名称改为isVip数据类型改为int

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
username	varchar(50)	YES		NULL	
password	varchar(50)	YES		NULL	
<u>isVip</u>	int	YES		NULL	

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

CSDN @观止study

- 修改表名

```
ALTER TABLE 表名 RENAME TO 新表名;
```



```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
username	varchar(50)	YES		NULL	
password	varchar(50)	YES		NULL	

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

```
mysql> alter table user rename to new_user;
```

Query OK, 0 rows affected (0.01 sec)

修改user表名为new_user

```
mysql> desc new_user;
```

Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
username	varchar(50)	YES		NULL	
password	varchar(50)	YES		NULL	

```
3 rows in set (0.01 sec)
```

CSDN @观止study

(2.4) 删除

注意: 在删除表的时候, 表中的全部数据也都会被删除。

- 删除字段

ALTER TABLE 表名 **DROP** 字段名;

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
username	varchar(50)	YES		NULL	
password	varchar(50)	YES		NULL	
isVip	int	YES		NULL	

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

```
mysql> alter table user drop isVip;
```

Query OK, 0 rows affected (0.02 sec)

Records: 0 Duplicates: 0 Warnings: 0

删除名为isVip的字段

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
username	varchar(50)	YES		NULL	
password	varchar(50)	YES		NULL	

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

CSDN @观止study

- 删除表

`DROP TABLE [IF EXISTS] 表名;`

```
mysql> drop table new_user;
Query OK, 0 rows affected (0.01 sec)

mysql> desc new_user;
ERROR 1146 (42S02): Table 'study.new_user' doesn't exist
mysql>
```

删除名为new_user的表

CSDN @观止study

- 删除指定表, 并重新创建表

`TRUNCATE TABLE 表名;`

```
mysql> select * from tb_user;
+----+-----+-----+
| 14 | 6     | 6     |
| 15 | 8     | 8     |
| 16 | 7     | 7     |
| 17 | 0     | 0     |
+----+-----+-----+
14 rows in set (0.00 sec)
```

之前表中存在14条数据

```
mysql> desc tb_user;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| userName | varchar(25) | YES |     | NULL    |
| userPassword | varchar(25) | YES |     | NULL    |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> truncate table tb_user;
Query OK, 0 rows affected (0.01 sec)
```

删除并重新创建该表

```
mysql> desc tb_user
-> ;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| userName | varchar(25) | YES |     | NULL    |
| userPassword | varchar(25) | YES |     | NULL    |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

```
mysql> select * from tb_user;
Empty set (0.00 sec)
```

删除重新创建后为empty表

CSDN @观止study

(2.5) 表字段数据类型

在上述的建表语句中, 我们在指定字段的数据类型时, 用到了int, varchar。除此之外,MySQL中还有许多的其他数据类型, 它们主要分为三类: **数值类型**、**字符串类型**、**日期时间类型**。

- 数值类型

类型	大小	有符号(SIGNED)范围	无符号(UNSIGNED)范围	描述
TINYINT	1byte	(-128, 127)	(0, 255)	小整数值
SMALLINT	2bytes	(-32768, 32767)	(0, 65535)	大整数值
MEDIUMINT	3bytes	(-8388608, 8388607)	(0, 16777215)	大整数值
INT/INTEGER	4bytes	(-2147483648, 2147483647)	(0, 4294967295)	大整数值
BIGINT	BIGINT	(-2^63, 2^63-1)	(0, 2^64-1)	极大整数值
FLOAT	4bytes	(-3.402823466 E+38, 3.402823466351 E+38)	0 和 (1.175494351 E-38, 3.402823466 E+38)	单精度浮点数值
DOUBLE	8bytes	(-1.7976931348623157 E+308, 1.7976931348623157 E+308)	0 和 (2.2250738585072014 E-308, 1.7976931348623157 E+308)	双精度浮点数值
DECIMAL		依赖于M(精度)和D(标度)的值	依赖于M(精度)和D(标度)的值	小数值(精确到小数点)

注：DECIMAL 中M(精度)表示整个数值长度，D(标度)表示小数位长度。例如：123.45。M=5,D=2。

```
# 使用示例
1) 年龄字段 -- 不会出现负数，而且人的年龄不会太大
age tinyint unsigned

2) 分数 -- 总分100分，最多出现一位小数
score double(4,1)
```

• 字符串类型

类型	大小	描述
CHAR	0-255 bytes	定长字符串(需要指定长度)
VARCHAR	0-65535 bytes	变长字符串(需要指定长度)
TINYBLOB	0-255 bytes	不超过255个字符的二进制数据
TINYTEXT	0-255 bytes	短文本字符串
BLOB	0-65 535 bytes	二进制形式的长文本数据
TEXT	0-65 535 bytes	长文本数据
MEDIUMBLOB	0-16 777 215 bytes	二进制形式的中等长度文本数据
MEDIUMTEXT	0-16 777 215 bytes	中等长度文本数据
LOBLOB	0-4 294 967 295 bytes	二进制形式的极大文本数据
LONGTEXT	0-4 294 967 295 bytes	极大文本数据

注：使用时通常会在类型后面加上()表示占用空间。

char 与 varchar 都可以描述字符串，char是定长字符串，指定长度多长，就占用多少个字符，和 字段值的长度无关。而 varchar是变长字符串，指定的长度为最大占用长度。相对来说，char的性能会更高些，而varchar相对更节省存储空间。

使用示例

1). 用户名 `username` -----> 长度不定，最长不会超过50
`username varchar(50)`

2). 性别 `gender` -----> 存储值，不是男,就是女
`gender char(1)`

3). 手机号 `phone` -----> 固定长度为11
`phone char(11)`

• 日期时间类型

类型	大小	范围	格式	描述
DATE	3	1000-01-01 至 9999-12-31	YYYY-MM-DD	日期值
TIME	3	-838:59:59 至 838:59:59	HH:MM:SS	时间值或持续 时间
YEAR	1	1901 至 2155	YYYY	年份值
DATETIME	8	1000-01-01 00:00:00 至 9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS	混合日期和时 间值
TIMESTAMP	4	1970-01-01 00:00:01 至 2038-01-19 03:14:07	YYYY-MM-DD HH:MM:SS	混合日期和时 间值，时 间戳

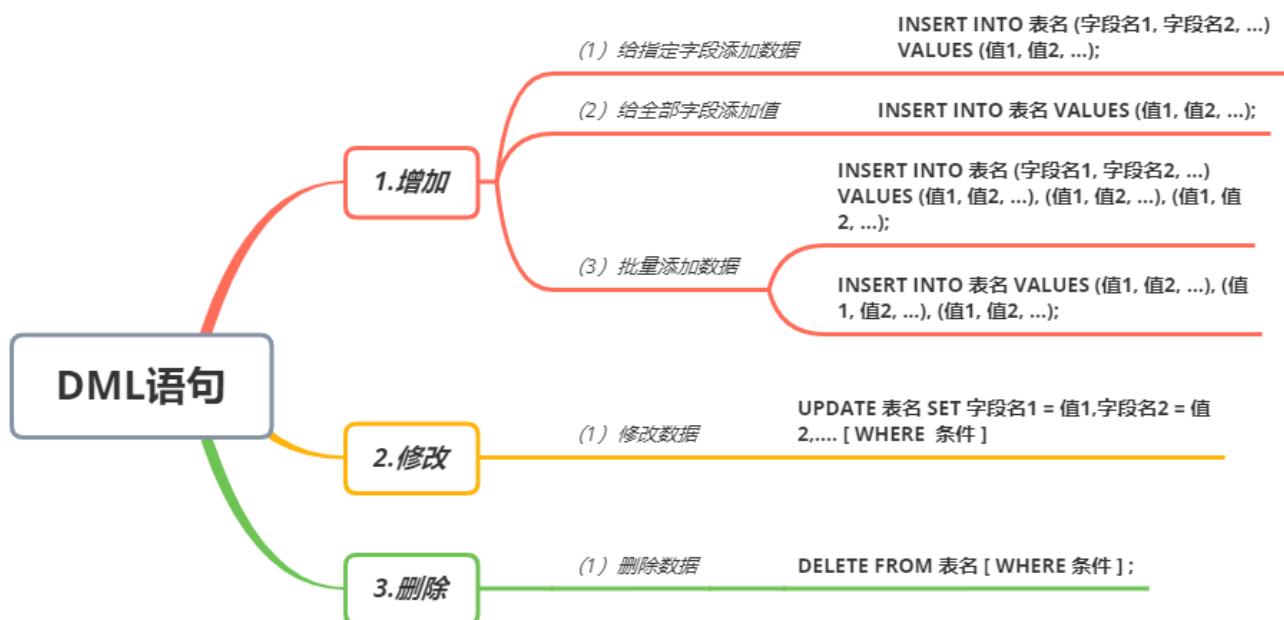
使用示例

1). 生日字段 `birthday`
`birthday date`

2). 创建时间 `createtime`
`createtime datetime`

三.DML语句

Data Manipulation Language，数据操作语言，用来**对数据库中表的数据记录进行增、删、改操作**



CSDN @观止study

(1) 增加

注意点:

1. 插入数据时, 指定的字段顺序需要与值的顺序是一一对应的
2. 字符串和日期型数据应该包含在引号中
3. 插入的数据大小, 应该在字段的规定范围内

- 给指定字段添加数据

```
INSERT INTO 表名 (字段名1, 字段名2, ...) VALUES (值1, 值2, ...);
```

```
mysql> insert into tb_user (id,userName) values(1,'test1');
Query OK, 1 row affected (0.00 sec)
为id, userName字段添加

mysql> insert into tb_user (id,userPassword) values(2,'test2');
Query OK, 1 row affected (0.00 sec)
为id, userPassword字段添加值

mysql> insert into tb_user (id,userName,userPassword) values(3,'test3','test3');
Query OK, 1 row affected (0.00 sec)
为id, userName,userPassword字段添加值

mysql> select * from tb_user;
+----+-----+-----+
| id | userName | userPassword |
+----+-----+-----+
| 1  | test1   | NULL        |
| 2  | NULL    | test2       |
| 3  | test3   | test3       |
+----+-----+-----+
3 rows in set (0.00 sec)
```

CSDN @观止study

- 给全部字段添加值

```
INSERT INTO 表名 VALUES (值1, 值2, ...);
```

```
mysql> insert into tb_user values(4,'test4','test4');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from tb_user;
```

简写，为全部字段添加值

id	userName	userPassword
1	test1	NULL
2	NULL	test2
3	test3	test3
4	test4	test4

4 rows in set (0.00 sec)

CSDN @观止study

- 批量添加数据

1. 指定字段批量添加

INSERT INTO 表名 (字段名1, 字段名2, ...) **VALUES** (值1, 值2, ...), (值1, 值2, ...), (值1, 值2, ...);

2. 全部字段批量添加

INSERT INTO 表名 **VALUES** (值1, 值2, ...), (值1, 值2, ...), (值1, 值2, ...);

```
mysql> insert into tb_user values(4,'test4','test4'),(5,'test5','test5');
ERROR 1062 (23000): Duplicate entry '4' for key 'tb_user.PRIMARY'
mysql> insert into tb_user values(5,'test5','test5'),(6,'test6','test6');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

```
mysql> select * from tb_user;
```

批量添加多条数据

id	userName	userPassword
1	test1	NULL
2	NULL	test2
3	test3	test3
4	test4	test4
5	test5	test5
6	test6	test6

6 rows in set (0.00 sec)

CSDN @观止study

(2) 修改

修改语句的where条件可以有，也可以没有，如果没有条件，则会修改整张表的所有数据。

- 修改字段值

UPDATE 表名 **SET** 字段名1 = 值1, 字段名2 = 值2, ... [**WHERE** 条件]

```
mysql> update tb_user set userName = 'guanzhi';
Query OK, 6 rows affected (0.00 sec)
Rows matched: 6  Changed: 6  Warnings: 0
```

**1.不指定条件修改该字段所属下所有值，
将表中所有userName值改为guanzhi**

```
mysql> update tb_user set userName = 'test666',userPassword = 'test666' where id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

**2.指定条件，修改符合条件字段值。
修改id为1的记录username值为test666
userPassword值为test666**

```
mysql> select * from tb_user;
+----+-----+-----+
| id | userName | userPassword |
+----+-----+-----+
| 1 | test666 | test666      |
| 2 | guanzhi | test2        |
| 3 | guanzhi | test3        |
| 4 | guanzhi | test4        |
| 5 | guanzhi | test5        |
| 6 | guanzhi | test6        |
+----+-----+-----+
6 rows in set (0.00 sec)
```

CSDN @观止study

(3) 删除

注意点:

1. DELETE 语句的条件可以有，也可以没有，如果没有条件，则会删除整张表的所有数据
2. DELETE 语句不能删除某一个字段的值(可以使用UPDATE，将该字段值置为NULL)

- 删除记录

```
DELETE FROM 表名 [ WHERE 条件 ] ;
```

```
mysql> delete from tb_user where id = 1;
Query OK, 1 row affected (0.00 sec)
```

删除id为1的记录

```
mysql> delete from tb_user;
Query OK, 5 rows affected (0.00 sec)
```

```
mysql> select * from tb_user;
```

删除所有记录

```
Empty set (0.00 sec)
```

CSDN @观止study

四.DQL语句

Data Query Language，数据查询语言，用来**查询数据库中表的记录**。



CSDN @观止study

(1) 基础查询

- 查询指定字段

```
SELECT 字段1, 字段2, 字段3 ... FROM 表名;
```



```
mysql> select id,userPassword from tb_user;
+----+-----+
| id | userPassword |
+----+-----+
| 5  | test5       |
| 6  | test6       |
+----+-----+
2 rows in set (0.00 sec)
```

查询指定字段

```
mysql> select id,userName,userPassword from tb_user;
+----+-----+-----+
| id | userName | userPassword |
+----+-----+-----+
| 5  | test5   | test5       |
| 6  | test6   | test6       |
+----+-----+-----+
2 rows in set (0.00 sec)
```

CSDN @观止study

- 查询全部字段

* 号代表所有字段

SELECT * FROM 表名;

```
mysql> select * from tb_user;
+----+-----+-----+
| id | userName | userPassword |
+----+-----+-----+
| 5  | test5   | test5       |
| 6  | test6   | test6       |
+----+-----+-----+
2 rows in set (0.00 sec)
```

查询全部字段

CSDN @观止study

- 查询时为字段设置别名

1. 可使用 as 设置别名

SELECT 字段1 [AS 别名1],字段2 [AS 别名2] ... FROM 表名;

2. 也可以直接空格设置别名

SELECT 字段1 [别名1],字段2 [别名2] ... FROM 表名;

```
mysql> select * from tb_user;
+----+-----+-----+
| id | userName | userPassword |
+----+-----+-----+
| 5  | test5    | test5        |
| 6  | test6    | test6        |
+----+-----+-----+
2 rows in set (0.00 sec)
```

不使用别名，默认字段名

```
mysql> select id,userName as 账号, userPassword as 密码 from tb_user;
+----+-----+-----+
| id | 账号  | 密码  |
+----+-----+-----+
| 5  | test5 | test5 |
| 6  | test6 | test6 |
+----+-----+-----+
2 rows in set (0.00 sec)
```

使用as 设置别名

```
mysql> select id,userName 账号, userPassword 密码 from tb_user;
+----+-----+-----+
| id | 账号  | 密码  |
+----+-----+-----+
| 5  | test5 | test5 |
| 6  | test6 | test6 |
+----+-----+-----+
2 rows in set (0.00 sec)
```

使用空格设置别名

CSDN @观止study

- 去除重复记录查询

```
SELECT DISTINCT 字段1,字段2... FROM 表名;
```

```
mysql> select * from tb_user;
+----+-----+-----+
| id | userName | userPassword |
+----+-----+-----+
| 3 | test6    | test6        |
| 4 | test5    | test5        |
| 5 | test5    | test5        |
| 6 | test6    | test6        |
+----+-----+-----+
4 rows in set (0.00 sec)
```

查看所有记录

```
mysql> select distinct userName,userPassword from tb_user;
+-----+-----+
| userName | userPassword |
+-----+-----+
| test6    | test6        |
| test5    | test5        |
+-----+-----+
2 rows in set (0.00 sec)
```

查看所有记录中不重复的
userName以及userPassword

```
mysql> select distinct userName from tb_user;
+-----+
| userName |
+-----+
| test6    |
| test5    |
+-----+
2 rows in set (0.00 sec)
```

查看所有记录中不重复的
userName

CSDN @观止study

(2) 条件查询

(2.1) 语法

SELECT 字段列表 **FROM** 表名 **WHERE** 条件列表;

(2.2) 条件

- 常用的**比较运算符**如下:

比较运算符	功能
>	大于
>=	大于等于
<	小于
<=	小于等于
=	等于
<> 或 !=	不等于
BETWEEN ... AND ...	在某个范围之内(含最小、最大值)
LIKE 占位符	模糊匹配(_ 匹配单个字符, % 匹配任意个字符)
IS NULL	是否为NULL
IN(...)	在(...)列表中的值, 多选一
ANY(...)	在(...)列表中的值, 有任意一个满足即可
SOME(...)	与ANY一致, 可相互替换
ALL(...)	必须满足(...)列表中的所有值

- 常用的**逻辑运算符**如下:

逻辑运算符	功能
AND 或 &&	并且 (多个条件同时成立)
OR 或	或者 (多个条件任意一个成立)
NOT 或 !	非, 不是

(2.3) 使用示例

- 查询年龄等于 88 的员工

```
select * from emp where age = 88;
```

mysql> select * from emp where age = 88;

id	workno	name	gender	age	idcard	workaddress	entrydate
13	00013	张三丰	男	88	123656789012345678	江苏	2020-11-01

1 row in set (0.00 sec)

CSDN @观止study

- 查询年龄不等于 88 的员工

```
select * from emp where age != 88;
select * from emp where age <> 88;
```

```
mysql> select * from emp where age != 88;
```

id	workno	name	gender	age	idcard	workaddress	entrydate
1	00001	柳岩666	女	20	123456789012345678	北京	2000-01-01
2	00002	张无忌	男	18	123456789012345670	北京	2005-09-01
3	00003	韦一笑	男	38	123456789712345670	上海	2005-08-01
4	00004	赵敏	女	18	123456757123845670	北京	2009-12-01
5	00005	小昭	女	16	123456769012345678	上海	2007-07-01
6	00006	杨逍	男	28	12345678931234567X	北京	2006-01-01
7	00007	范瑤	男	40	123456789212345670	北京	2005-05-01
8	00008	黛绮丝	女	38	123456157123645670	天津	2015-05-01
9	00009	范凉凉	女	45	123156789012345678	北京	2010-04-01
10	00010	陈友谅	男	53	123456789012345670	上海	2011-01-01
11	00011	张士诚	男	55	123567897123465670	江苏	2015-05-01
12	00012	常遇春	男	32	123446757152345670	北京	2004-02-01
14	00014	灭绝	女	65	123456719012345670	西安	2019-05-01
15	00015	胡青牛	男	70	12345674971234567X	西安	2018-04-01
16	00016	周芷若	女	18	NULL	北京	2012-06-01

- 查询没有身份证号员工信息

```
select * from emp where idcard is null;
```

```
mysql> select * from emp where idcard is null;
```

id	workno	name	gender	age	idcard	workaddress	entrydate
16	00016	周芷若	女	18	NULL	北京	2012-06-01

1 row in set (0.00 sec)

- 查询有身份证号员工信息

```
select * from emp where idcard is not null;
```

```
mysql> select * from emp where idcard is not null;
```

id	workno	name	gender	age	idcard	workaddress	entrydate
1	00001	柳岩666	女	20	123456789012345678	北京	2000-01-01
2	00002	张无忌	男	18	123456789012345670	北京	2005-09-01
3	00003	韦一笑	男	38	123456789712345670	上海	2005-08-01
4	00004	赵敏	女	18	123456757123845670	北京	2009-12-01
5	00005	小昭	女	16	123456769012345678	上海	2007-07-01
6	00006	杨逍	男	28	12345678931234567X	北京	2006-01-01
7	00007	范瑤	男	40	123456789212345670	北京	2005-05-01
8	00008	黛绮丝	女	38	123456157123645670	天津	2015-05-01
9	00009	范凉凉	女	45	123156789012345678	北京	2010-04-01
10	00010	陈友谅	男	53	123456789012345670	上海	2011-01-01
11	00011	张士诚	男	55	123567897123465670	江苏	2015-05-01
12	00012	常遇春	男	32	123446757152345670	北京	2004-02-01
13	00013	张三丰	男	88	123656789012345678	江苏	2020-11-01
14	00014	灭绝	女	65	123456719012345670	西安	2019-05-01
15	00015	胡青牛	男	70	12345674971234567X	西安	2018-04-01

- 查询年龄在15岁(包含)到20岁(包含)之间的员工信息

```
select * from emp where age >= 15 && age <= 20;
```

```
select * from emp where age >= 15 and age <= 20;
```

```
select * from emp where age between 15 and 20;
```

```
mysql> select * from emp where age between 15 and 20;
```

id	workno	name	gender	age	idcard	workaddress	entrydate
1	00001	柳岩666	女	20	123456789012345678	北京	2000-01-01
2	00002	张无忌	男	18	123456789012345670	北京	2005-09-01
4	00004	赵敏	女	18	123456757123845670	北京	2009-12-01
5	00005	小昭	女	16	123456769012345678	上海	2007-07-01
16	00016	周芷若	女	18	NULL	北京	2012-06-01

- 查询年龄等于18 或 20 或 40 的员工信息

```
select * from emp where age = 18 or age = 20 or age = 40;
```

```
select * from emp where age in(18,20,40);
```

```
mysql> select * from emp where age in(18,20,40);
```

id	workno	name	gender	age	idcard	workaddress	entrydate
1	00001	柳岩666	女	20	123456789012345678	北京	2000-01-01
2	00002	张无忌	男	18	123456789012345670	北京	2005-09-01
4	00004	赵敏	女	18	123456757123845670	北京	2009-12-01
7	00007	范瑶	男	40	123456789212345670	北京	2005-05-01
16	00016	周芷若	女	18	NULL	北京	2012-06-01

5 rows in set (0.00 sec)

- 查询姓范名字为两个字的员工信息

```
select * from emp where name like '范_'; # 名字只可以为范X
```

```
mysql> select * from emp where name like '范_';
```

id	workno	name	gender	age	idcard	workaddress	entrydate
7	00007	范瑶	男	40	123456789212345670	北京	2005-05-01

1 row in set (0.00 sec)

- 查询姓范员工信息

```
select * from emp where name like '范%'; # 名字可以为范X, 范XX, 范XXX等等
```

```
mysql> select * from emp where name like '范%';
```

id	workno	name	gender	age	idcard	workaddress	entrydate
7	00007	范瑶	男	40	123456789212345670	北京	2005-05-01
9	00009	范凉凉	女	45	123156789012345678	北京	2010-04-01

2 rows in set (0.00 sec)

(3) 聚合函数查询

将一列数据作为一个整体，进行纵向计算

(3.1) 语法

```
SELECT 聚合函数(字段列表) FROM 表名;
```

(3.2) 常见函数

函数	功能
<code>count</code>	统计数量
<code>max</code>	最大值
<code>min</code>	最小值
<code>avg</code>	平均值
<code>sum</code>	求和

(3.3) 使用示例

NULL值不参与所有聚合函数运算

- 统计该企业员工数量

```
select count(*) from emp; -- 统计的是总记录数
select count(idcard) from emp; -- 统计的是idcard字段不为null的记录数
```

```
mysql> select count(*) from emp;
+-----+
| count(*) |
+-----+
|      16 |
+-----+
1 row in set (0.00 sec)      CSDN @观止study
```

- 统计该企业员工的平均年龄

```
select avg(age) from emp;
```

```
mysql> select avg(age) from emp;
+-----+
| avg(age) |
+-----+
|  40.1250 |
+-----+
1 row in set (0.00 sec)      CSDN @观止study
```

- 统计该企业员工的最大年龄

```
select max(age) from emp;
```

```
mysql> select max(age) from emp;
+-----+
| max(age) |
+-----+
|      88 |
+-----+
1 row in set (0.00 sec)
```

- 统计该企业员工的最小年龄

```
select min(age) from emp;
```

```
mysql> select min(age) from emp;
+-----+
| min(age) |
+-----+
|      16 |
+-----+
1 row in set (0.00 sec)
```

- 统计西安地区员工的年龄之和

```
select sum(age) from emp where workaddress = '西安';
```

```
mysql> select sum(age) from emp where workaddress='西安';
+-----+
| sum(age) |
+-----+
|      135 |
+-----+
1 row in set (0.00 sec)
```

(4) 分组查询

(4.1) 语法

```
SELECT 字段列表 FROM 表名 [ WHERE 条件 ] GROUP BY 分组字段名 [ HAVING 分组后过滤条件 ];
```

注意事项:

- where** 与 **having** 区别
 - 执行时机不同**: where是**分组之前**进行过滤; having是**分组之后**对结果进行过滤。
 - 判断条件不同**: where不能对聚合函数进行判断, 而having可以。执行顺序: where > 聚合函数 > having。
- 支持多字段分组, 具体语法为: group by 分组字段名1, 分组字段名2

(4.2) 使用示例

- 根据性别分组, 统计男性员工 和 女性员工的数量

```
select gender, count(*) from emp group by gender;
```



```
mysql> select gender, count(*) from emp group by gender;
+-----+-----+
| gender | count(*) |
+-----+-----+
| 女     | 7        |
| 男     | 9        |
+-----+-----+
2 rows in set (0.00 sec)
```

CSDN @观止study

- 查询年龄小于45的员工，并根据工作地址分组，获取员工数量大于等于3的工作地址

```
select workaddress, count(*) address_count from emp where age < 45 group by
workaddress having address_count >= 3;
```

```
mysql> select workaddress, count(*) address_count from emp where age < 45 group by
-> workaddress having address_count >= 3;
+-----+-----+
| workaddress | address_count |
+-----+-----+
| 北京       | 7            |
+-----+-----+
1 row in set (0.00 sec)
```

CSDN @观止study

- 统计各个工作地址上班的男性及女性员工的数量

```
select workaddress, gender, count(*) '数量' from emp group by gender,workaddress;
```

```
mysql> select workaddress, gender, count(*) '数量' from emp group by gender,workaddress;
+-----+-----+-----+
| workaddress | gender | 数量 |
+-----+-----+-----+
| 北京       | 女     | 4    |
| 北京       | 男     | 4    |
| 上海       | 男     | 2    |
| 上海       | 女     | 1    |
| 天津       | 女     | 1    |
| 江苏       | 男     | 2    |
| 西安       | 女     | 1    |
| 西安       | 男     | 1    |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

CSDN @观止study

(5) 排序查询

(5.1) 语法

```
SELECT 字段列表 FROM 表名 ORDER BY 字段1 排序方式, 字段2 排序方式;
```

- 排序方式类别
 - ASC: 升序(默认值)
 - DESC: 降序
- 注意事项
 - 如果是升序, 可以不指定排序方式ASC;
 - 如果是多字段排序, 当第一个字段值相同时, 才会根据第二个字段进行排序

(5.2) 使用示例

- 根据年龄对公司的员工进行升序排序

```
select * from emp order by age asc;  
select * from emp order by age;
```

```
mysql> select * from emp order by age;
```

id	workno	name	gender	age	idcard	workaddress	entrydate
5	00005	小昭	女	16	123456769012345678	上海	2007-07-01
2	00002	张无忌	男	18	123456789012345670	北京	2005-09-01
4	00004	赵敏	女	18	123456757123845670	北京	2009-12-01
16	00016	周芷若	女	18	NULL	北京	2012-06-01
1	00001	柳岩666	女	20	123456789012345678	北京	2000-01-01
6	00006	杨逍	男	28	12345678931234567X	北京	2006-01-01
12	00012	常遇春	男	32	123446757152345670	北京	2004-02-01
3	00003	韦一笑	男	38	123456789712345670	上海	2005-08-01
8	00008	黛绮丝	女	38	123456157123645670	天津	2015-05-01
7	00007	范瑤	男	40	123456789212345670	北京	2005-05-01
9	00009	范凉凉	女	45	123156789012345678	北京	2010-04-01
10	00010	陈友谅	男	53	123456789012345670	上海	2011-01-01
11	00011	张士诚	男	55	123567897123465670	江苏	2015-05-01
14	00014	灭绝	女	65	123456719012345670	西安	2019-05-01
15	00015	胡青牛	男	70	12345674971234567X	西安	2018-04-01
13	00013	张三丰	男	88	123656789012345678	江苏	2020-11-01

- 根据年龄对公司的员工进行升序排序，年龄相同，再按照入职时间进行降序排序

```
select * from emp order by age asc , entrydate desc;
```

```
mysql> select * from emp order by age asc , entrydate desc;
```

id	workno	name	gender	age	idcard	workaddress	entrydate
5	00005	小昭	女	16	123456769012345678	上海	2007-07-01
16	00016	周芷若	女	18	NULL	北京	2012-06-01
4	00004	赵敏	女	18	123456757123845670	北京	2009-12-01
2	00002	张无忌	男	18	123456789012345670	北京	2005-09-01
1	00001	柳岩666	女	20	123456789012345678	北京	2000-01-01
6	00006	杨逍	男	28	12345678931234567X	北京	2006-01-01
12	00012	常遇春	男	32	123446757152345670	北京	2004-02-01
8	00008	黛绮丝	女	38	123456157123645670	天津	2015-05-01
3	00003	韦一笑	男	38	123456789712345670	上海	2005-08-01
7	00007	范瑤	男	40	123456789212345670	北京	2005-05-01
9	00009	范凉凉	女	45	123156789012345678	北京	2010-04-01
10	00010	陈友谅	男	53	123456789012345670	上海	2011-01-01
11	00011	张士诚	男	55	123567897123465670	江苏	2015-05-01
14	00014	灭绝	女	65	123456719012345670	西安	2019-05-01
15	00015	胡青牛	男	70	12345674971234567X	西安	2018-04-01
13	00013	张三丰	男	88	123656789012345678	江苏	2020-11-01

(6) 分页查询

(6.1) 语法

SELECT 字段列表 **FROM** 表名 **LIMIT** 起始索引, 查询记录数;

- 注意事项:
 - 起始索引从0开始, 计算规则为: (查询页码 - 1) * 每页显示记录数。
 - 如果查询的是第一页数据, 起始索引可以省略, 直接简写为 limit 10。

(6.2) 使用示例

- 查询第1页员工数据, 每页展示5条记录

```
select * from emp limit 0,5;  
select * from emp limit 5;
```

```
mysql> select * from emp limit 0,5;  
+----+-----+-----+-----+-----+-----+-----+-----+  
| id | workno | name   | gender | age | idcard                | workaddress | entrydate |  
+----+-----+-----+-----+-----+-----+-----+-----+  
| 1  | 00001  | 柳岩666 | 女     | 20  | 123456789012345678   | 北京       | 2000-01-01 |  
| 2  | 00002  | 张无忌  | 男     | 18  | 123456789012345670   | 北京       | 2005-09-01 |  
| 3  | 00003  | 韦一笑  | 男     | 38  | 123456789712345670   | 上海       | 2005-08-01 |  
| 4  | 00004  | 赵敏    | 女     | 18  | 123456757123845670   | 北京       | 2009-12-01 |  
| 5  | 00005  | 小昭    | 女     | 16  | 123456769012345678   | 上海       | 2007-07-01 |  
+----+-----+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

CSDN @观止study

- 查询第2页员工数据, 每页展示10条记录 -----> (页码-1)*页展示记录数

```
select * from emp limit 5,5;
```

```
mysql> select * from emp limit 5,5;  
+----+-----+-----+-----+-----+-----+-----+-----+  
| id | workno | name   | gender | age | idcard                | workaddress | entrydate |  
+----+-----+-----+-----+-----+-----+-----+-----+  
| 6  | 00006  | 杨逍    | 男     | 28  | 12345678931234567X   | 北京       | 2006-01-01 |  
| 7  | 00007  | 范瑤    | 男     | 40  | 123456789212345670   | 北京       | 2005-05-01 |  
| 8  | 00008  | 黛绮丝  | 女     | 38  | 123456157123645670   | 天津       | 2015-05-01 |  
| 9  | 00009  | 范凉凉  | 女     | 45  | 123156789012345678   | 北京       | 2010-04-01 |  
| 10 | 00010  | 陈友谅  | 男     | 53  | 123456789012345670   | 上海       | 2011-01-01 |  
+----+-----+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

CSDN @观止study

(7) 复合使用

(7.1) 书写顺序

同时使用上述多种查询时必须严格遵循以下顺序书写:

SELECT
字段列表

FROM
表名列表

WHERE
条件列表

GROUP BY
分组字段列表

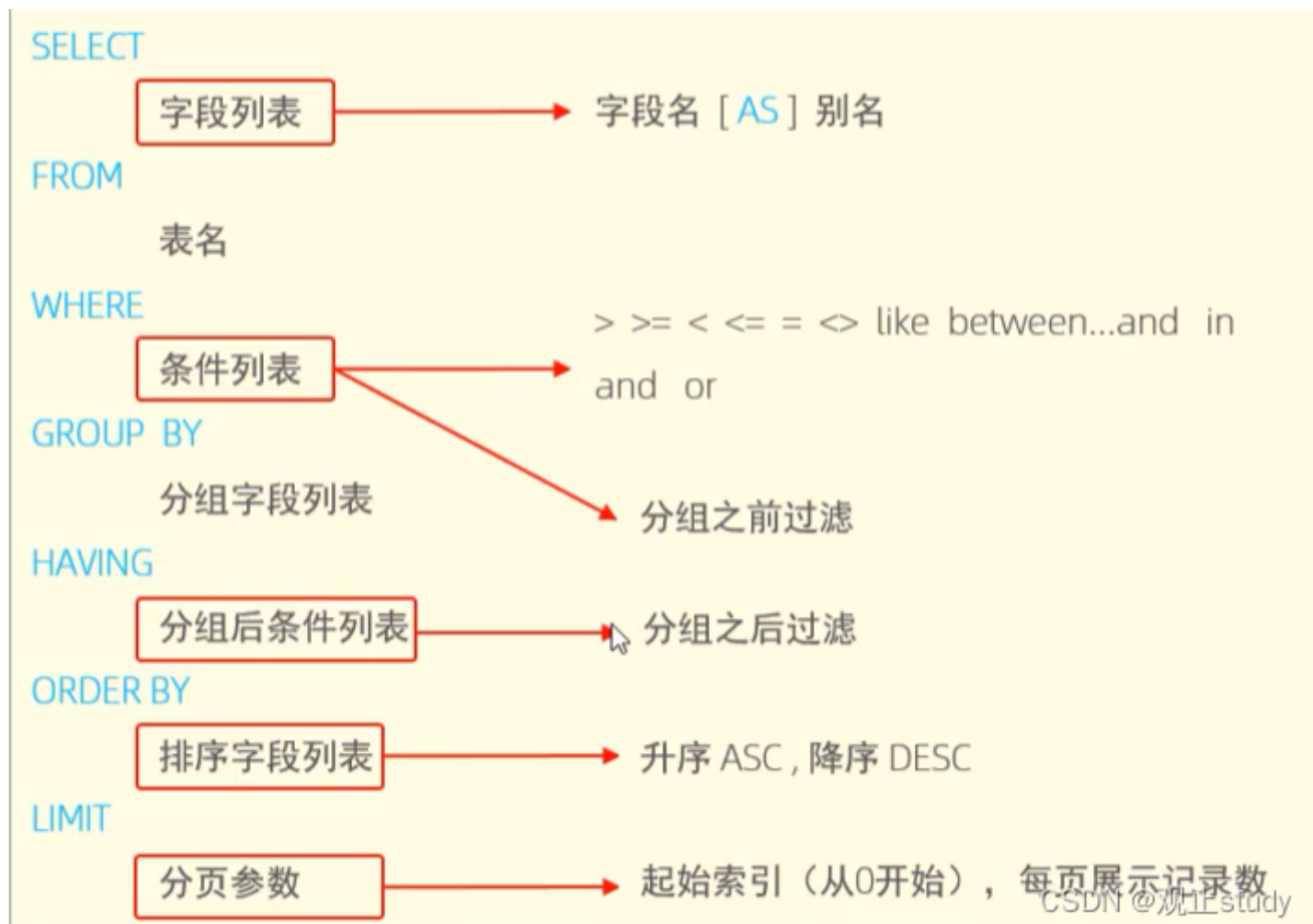
HAVING
分组后条件列表

ORDER BY

排序字段列表

LIMIT

分页参数



(7.2) 执行顺序

- DQL语句复合使用系统执行顺序为: `from ... where ... group by ... having ... select ... order by ... limit ...`

关于having、group by以及聚合函数的执行顺序问题，整个执行顺序可以这么理解：

1. 先执行from语句（表之间的笛卡尔积、交并差等），获得一个虚拟表
2. 如果where语句存在，从虚拟表中筛出符合where条件的数据，不满足的被剔除
3. 如果group by语句存在，则目前存活的数据分组；如果不存在group by，则将这些数据视为一个组
4. 如果存在having语句，则将满足having条件的组留下，不满足的组被剔除
5. 执行select语句：对存活下来的每个组分别执行聚合函数，形成查询结果
6. 执行order by 语句：对剩下的数据进行排序
7. 执行limit 语句：限制返回的数据条数

(7.3) 使用示例

- 统计员工表中, 年龄小于60岁的, 男性员工和女性员工的人数。

```
select gender, count(*) from emp where age < 60 group by gender;
```

```
mysql> select gender, count(*) from emp where age < 60 group by gender;
+-----+-----+
| gender | count(*) |
+-----+-----+
| 女     | 6        |
| 男     | 7        |
+-----+-----+
2 rows in set (0.00 sec)
```

CSDN @观止study

- 查询所有年龄小于等于35岁员工的姓名和年龄，并对查询结果按年龄升序排序，如果年龄相同按 入职时间降序排序

```
select name , age from emp where age <= 35 order by age asc,entrydate desc;
```

```
mysql> select name , age from emp where age <= 35 order by age asc,entrydate desc;
+-----+-----+
| name   | age |
+-----+-----+
| 小昭   | 16  |
| 周芷若 | 18  |
| 赵敏   | 18  |
| 张无忌 | 18  |
| 柳岩666 | 20  |
| 杨逍   | 28  |
| 常遇春 | 32  |
+-----+-----+
7 rows in set (0.00 sec)
```

CSDN @观止study

- 查询性别为男，且年龄在20-40 岁(含)以内的前5个员工信息，对查询的结果按年龄升序排序， 年龄相同按入职时间升序排序。

```
select * from emp where gender = '男' and age between 20 and 40 order by age asc ,entrydate asc
limit 5 ;
```

```
mysql> select * from emp where gender = '男' and age between 20 and 40 order by age asc ,entrydate asc limit 5 ;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | workno | name   | gender | age | idcard           | workaddress | entrydate |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 6  | 00006  | 杨逍   | 男     | 28  | 12345678931234567X | 北京       | 2006-01-01 |
| 12 | 00012  | 常遇春 | 男     | 32  | 123446757152345670 | 北京       | 2004-02-01 |
| 3  | 00003  | 韦一笑 | 男     | 38  | 123456789712345670 | 上海       | 2005-08-01 |
| 7  | 00007  | 范瑤   | 男     | 40  | 123456789212345670 | 北京       | 2005-05-01 |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

CSDN @观止study

五.DCL语句

Data Control Language，数据控制语言，**用来管理数据库用户、控制数据库的访问权限**



CSDN @观止study

(1) 管理用户

- 查询用户

```
select * from mysql.user
```

其中 Host代表当前用户访问的主机, 如果为localhost, 代表只能够在当前本机访问, 不可以远程访问的。User代表的是访问该数据库的用户名。在MySQL中需要**通过Host和User来唯一标识一个用户**。

```
2
3 select * from mysql.user;
4
```

信息	结果1	概况	状态
----	-----	----	----

Host	User	Select_priv	Insert_priv	Update_priv
localhost	mysql.infoschema	Y	N	N
localhost	mysql.session	N	N	N
localhost	mysql.sys	N	N	N
localhost	root	Y	Y	Y

CSDN @观止study

- 创建用户

```
CREATE USER '用户名'@'主机名' IDENTIFIED BY '密码';
```

```
1 CREATE USER 'guanzhi'@'localhost' IDENTIFIED BY 'guanzhi666';
```

信息	结果1	概况	状态		
Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv
localhost	guanzhi	N	N	N	N
localhost	mysql.infoschema	Y	N	N	N
localhost	mysql.session	N	N	N	N
localhost	mysql.sys	N	N	N	N
localhost	root	Y	Y	Y	Y

- 修改用户密码

```
ALTER USER '用户名'@'主机名' IDENTIFIED WITH mysql_native_password BY '新密码';
```

- 删除用户

```
DROP USER '用户名'@'主机名';
```

5						
6	DROP USER 'guanzhi'@'localhost';					
信息	结果1	概况	状态			
Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	
localhost	mysql.infoschema	Y	N	N	N	
localhost	mysql.session	N	N	N	N	
localhost	mysql.sys	N	N	N	N	
localhost	root	Y	Y	Y	Y	

(2) 权限控制

常见[权限列表](#):

权限	说明
ALL, ALL PRIVILEGES	所有权限
SELECT	查询数据
INSERT	插入数据
UPDATE	修改数据
DELETE	删除数据
ALTER	修改表
DROP	删除数据库/表/视图
CREATE	创建数据库/表

- 查询用户权限

```
SHOW GRANTS FOR '用户名'@'主机名' ;
```

3				
4	SHOW GRANTS FOR 'guanzhi'@'localhost';			
信息	结果1	概况	状态	
Grants for guanzhi@localhost				
	GRANT USAGE ON *.* TO `guanzhi`@`localhost`			


- 授予权限

```
GRANT 权限列表 ON 数据库名.表名 TO '用户名'@'主机名';
```

```
5
6 GRANT INSERT,SELECT ON study.emp TO 'guanzhi'@'localhost';
```

信息	结果1	概况	状态
----	-----	----	----

Grants for guanzhi@localhost			
▶	GRANT USAGE ON *.* TO `guanzhi`@`localhost`		
	GRANT SELECT, INSERT ON `study`.`emp` TO `gua		



CSDN @观止study


- 撤销权限

```
REVOKE 权限列表 ON 数据库名.表名 FROM '用户名'@'主机名';
```

```
/
8 REVOKE SELECT ON study.emp FROM 'guanzhi'@'localhost';
```

信息	结果1	概况	状态
----	-----	----	----

Grants for guanzhi@localhost			
▶	GRANT USAGE ON *.* TO `guanzhi`@`localhost`		
	GRANT INSERT ON `study`.`emp` TO `guanzhi`@`l		



CSDN @观止study

- 注意事项:
 - 多个权限之间，使用逗号分隔
 - 授权时，数据库名和表名可以使用 * 进行通配，代表所有

六.全文概览

SQL基础

DDL (数据定义语言)

一.数据库操作

- 1.查询所有数据库 `show databases;`
- 2.查询当前所在数据库 `select database();`
- 3.创建数据库 `create database [if not exists] 数据库名 [default charset 字符集] [collate 排序规则];`
- 4.删除数据库 `drop database [if exists] 数据库名;`
- 5.切换数据库 `use 数据库名;`

二.表操作

- 1.创建
 - (1) 创建表结构 `CREATE TABLE 表名(字段1 字段1类型 [COMMENT '字段1注释'], 字段2 字段2类型 [COMMENT '字段2注释'], 字段n 字段n类型 [COMMENT '字段n注释']) [COMMENT '表注释'];`
 - (2) 添加字段 `ALTER TABLE 表名 ADD 字段名 类型 (长度) [COMMENT 注释] [约束];`
- 2.查询
 - (1) 查询当前数据库所有表 `show tables;`
 - (2) 查看指定表结构 `desc 表名;`
 - (3) 查询指定表的建表语句 `show create table 表名;`
- 3.修改
 - (1) 修改字段数据类型 `ALTER TABLE 表名 MODIFY 字段名 新数据类型 (长度);`
 - (2) 同时修改字段名和字段类型 `ALTER TABLE 表名 CHANGE 旧字段名 新字段名 类型 (长度) [COMMENT 注释] [约束];`
 - (3) 修改表名 `ALTER TABLE 表名 RENAME TO 新表名;`
- 4.删除
 - (1) 删除表 `DROP TABLE [IF EXISTS] 表名;`
 - (2) 删除指定表, 并重新创建表 `TRUNCATE TABLE 表名;`
 - (3) 删除字段 `ALTER TABLE 表名 DROP 字段名;`

DML (数据操作语言)

1.增加

- (1) 给指定字段添加数据 `INSERT INTO 表名 (字段名1, 字段名2, ...) VALUES (值1, 值2, ...);`
- (2) 给全部字段添加值 `INSERT INTO 表名 VALUES (值1, 值2, ...);`
- (3) 批量添加数据 `INSERT INTO 表名 (字段名1, 字段名2, ...) VALUES (值1, 值2, ...), (值1, 值2, ...), (值1, 值2, ...);`
`INSERT INTO 表名 VALUES (值1, 值2, ...), (值1, 值2, ...), (值1, 值2, ...);`

2.修改

- (1) 修改数据 `UPDATE 表名 SET 字段名1 = 值1, 字段名2 = 值2, ... [WHERE 条件]`

3.删除

- (1) 删除数据 `DELETE FROM 表名 [WHERE 条件];`

DQL (数据查询语言)

1.基本查询 (无条件)

- (1) 查询指定字段 `SELECT 字段1, 字段2, 字段3 ... FROM 表名;`
- (2) 查询全部字段 `SELECT * FROM 表名;`
- (3) 查询时为字段设置别名 `SELECT 字段1 [AS 别名1], 字段2 [AS 别名2] ... FROM 表名;`
`SELECT 字段1 [别名1], 字段2 [别名2] ... FROM 表名;`
- (4) 去除重复记录查询 `SELECT DISTINCT 字段1, 字段2 ... FROM 表名;`

2.条件查询 (WHERE)

- `SELECT 字段列表 FROM 表名 WHERE 条件列表;`

3.聚合函数 (count, max, min, avg, sum)

- `SELECT 聚合函数(字段列表) FROM 表名;`

4.分组查询 (group by)

- `SELECT 字段列表 FROM 表名 [WHERE 条件] GROUP BY 分组字段名 [HAVING 分组后过滤条件];`

5.分页查询 (limit)

- `SELECT 字段列表 FROM 表名 LIMIT 起始索引, 查询记录数;`

6.排序查询 (order by)

- `SELECT 字段列表 FROM 表名 ORDER BY 字段1 排序方式, 字段2 排序方式;`

7.复合使用

- `SELECT`
字段列表
`FROM`
表名列表
`WHERE`
条件列表
`GROUP BY`
分组字段列表
`HAVING`
分组后条件列表
`ORDER BY`
排序字段列表
`LIMIT`
分页参数
- (1) 书写顺序
 - (2) 系统执行顺序
`from ... where ... group by ... having ... select ... order by ... limit ...`

