MySQL手册

参考https://blog.csdn.net/weixin\_45851945/article/details/114287877

YFY 著

SQL

Oracle Mysql

CSDN

全文参考[MySQL 有这一篇就够（呕心狂敲37k字，只为博君一点赞！！！）\_mysql有这一篇幅就够了-CSDN博客](https://blog.csdn.net/weixin_45851945/article/details/114287877?ops_request_misc=%7B%22request%5Fid%22%3A%22171964820416800184134114%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=171964820416800184134114&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-2-114287877-null-null.142^v100^pc_search_result_base3&utm_term=MySQL&spm=1018.2226.3001.4187)

（https://blog.csdn.net/weixin\_45851945/article/details/114287877）

目录

[1. 数据库的数据类型 5](#_Toc23099)

[1.1 整数类型 5](#_Toc13913)

[1.2 浮点数类型和定点数类型 5](#_Toc13599)

[1.3 字符串类型 6](#_Toc18588)

[1.4 字符串类型 6](#_Toc17192)

[1.5 日期与时间类型 6](#_Toc18214)

[1.5.1 YEAR类型 7](#_Toc26680)

[1.5.2 TIME类型 7](#_Toc9743)

[1.5.3 DATETIME类型 7](#_Toc20695)

[1.5.4 TIMESTAMP类型 8](#_Toc3395)

[1.6 二进制类型 8](#_Toc9380)

[2.全局 8](#_Toc12230)

[3.数据库 8](#_Toc13571)

[查看该数据库基本信息 9](#_Toc8720)

[4.数据表 9](#_Toc21136)

[5.表的约束 10](#_Toc20553)

[5.1 主键约束 10](#_Toc1764)

[5.2非空约束 10](#_Toc31600)

[5.3 默认值约束 11](#_Toc7587)

[5.4 唯一性约束 11](#_Toc7347)

[5.5 外键约束 11](#_Toc32504)

[6. 数据一致性概念 12](#_Toc18557)

[7. 数据表插入数据 12](#_Toc12002)

[7.1 为表中所有字段插入数据 13](#_Toc14088)

[7.2 为表中指定字段插入数据 13](#_Toc8195)

[7.3 同时插入多条记录 13](#_Toc1658)

[8. 更新数据 13](#_Toc15825)

[8.1 UPDATE基本语法 13](#_Toc10953)

[8.2 UPDATE更新部分数据 14](#_Toc29013)

[8.3 UPDATE更新全部数据 14](#_Toc25096)

[9. 删除数据 14](#_Toc15579)

[9.1 DELETE基本语法 14](#_Toc30662)

[9.2 DELETE删除部分数据 14](#_Toc11851)

[9.3 DELETE删除全部数据 15](#_Toc14493)

[9.4 TRUNCATE和DETELE的区别 15](#_Toc9460)

[10. 数据表简单查询 15](#_Toc16180)

[10.1 简单查询概述 15](#_Toc3864)

[10.2 查询所有字段（方法不唯一只是举例） 16](#_Toc20019)

[10.3 查询指定字段（sid、sname） 16](#_Toc9341)

[10.4 常数的查询 16](#_Toc18175)

[10.5 从查询结果中过滤重复数据 16](#_Toc3781)

[10.6 算术运算符（举例加运算符） 16](#_Toc2877)

[11. 函数 17](#_Toc13672)

[11.1 聚合函数 17](#_Toc18274)

[11.1.1 count() 18](#_Toc21111)

[11.1.2 max() 18](#_Toc22239)

[11.1.3 min() 18](#_Toc24220)

[11.1.4 sum() 18](#_Toc14004)

[11.1.5 avg() 18](#_Toc5617)

[11.2 其他常用函数 18](#_Toc24926)

[11.2.1 时间函数 18](#_Toc9371)

[11.2.2 字符串函数 19](#_Toc1925)

[11.2.3 数学函数 19](#_Toc16729)

[12. 条件查询 20](#_Toc29398)

[12.1 使用关系运算符查询 20](#_Toc28002)

[12.2 使用IN关键字查询 21](#_Toc3305)

[12.3.使用BETWEEN AND关键字查询 21](#_Toc3135)

[12.4 使用空值查询 21](#_Toc4651)

[12.5 使用AND关键字查询 21](#_Toc23441)

[12.6 使用OR关键字查询 21](#_Toc30996)

[12.7 使用LIKE关键字查询 22](#_Toc20955)

[12.7.1 普通字符串 22](#_Toc6209)

[12.7.2 含有%通配的字符串 22](#_Toc25803)

[12.7.3 含有\_通配的字符串 22](#_Toc10791)

[12.8 使用LIMIT限制查询结果的数量 22](#_Toc31382)

[12.9 使用GROUP BY进行分组查询 23](#_Toc880)

[12.9.1 GROUP BY和聚合函数一起使用 23](#_Toc10080)

[12.9.2 GROUP BY和聚合函数以及HAVING一起使用 23](#_Toc14666)

[12.10 使用ORDER BY对查询结果排序 24](#_Toc7885)

[13. 别名设置 24](#_Toc7986)

[13.1 为表取别名 24](#_Toc17193)

[13.2 为字段取别名 24](#_Toc32763)

[14. 表的关联关系 25](#_Toc18129)

[14.1 关联查询 26](#_Toc21535)

[14.2 关于关联关系的删除数据 26](#_Toc14772)

[15. 多表连接查询 26](#_Toc1957)

[15.1 交叉连接查询 26](#_Toc27698)

[15.2 内连接查询 26](#_Toc13340)

[15.3 外连接查询 27](#_Toc6561)

[15.3.1 左（外）连接查询 29](#_Toc6005)

[15.3.2 右（外）连接查询 29](#_Toc8253)

[16. 子查询 29](#_Toc5839)

[16.1 带比较运算符的子查询 30](#_Toc31052)

[16.2 带EXISTS关键字的子查询 30](#_Toc21609)

[16.3 带ANY关键字的子查询 31](#_Toc31683)

[16.4 带ALL关键字的子查询 31](#_Toc23126)

[17. 总结 31](#_Toc1197)

[18. SQL简述 31](#_Toc10154)

[18.1 SQL的概述 31](#_Toc15388)

[18.2 SQL的优点 31](#_Toc7224)

[18.3 SQL的分类 32](#_Toc8828)

# 1. 数据库的数据类型

使用MySQL数据库存储数据时，不同的数据类型决定了 MySQL存储数据方式的不同。为此，MySQL数据库提供了多种数据类型，其中包括整数类型、浮点数类型、定点 数类型、日期和时间类型、字符串类型、二进制等等数据类型。

## 1.1 整数类型

根据数值取值范围的不同MySQL 中的整数类型可分为5种，分别是TINYINT、SMALUNT、MEDIUMINT、INT和 BIGINT。下图列举了 MySQL不同整数类型所对应的字节大小和取值范围而最常用的为INT类型的，

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 字节数 | 无符号数的取值范围 | 有符号数的取值范围 |
| TINYINT | 1 | 0~255 | -128~127 |
| SMALLINT | 2 | 0~65535 | -32768~32768 |
| MEDIUMINT | 3 | 0~16777215 | -8388608~8388608 |
| INT | 4 | 0~4294967295 | -2147483648~ 2147483648 |
| BIGINT | 8 | 0~18446744073709551615 | -9223372036854775808~9223372036854775808 |

## 1.2 浮点数类型和定点数类型

在MySQL数据库中使用浮点数和定点数来存储小数。浮点数的类型有两种：单精度浮点数类型（FLOAT）和双精度浮点数类型（DOUBLE）。而定点数类型只有一种即DECIMAL类型。下图列举了 MySQL中浮点数和定点数类型所对应的字节大小及其取值范围：

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 字节数 | 有符号的取值范围 | 无符号的取值范围 |
| FLOAT | 4 | -3.402823466E+38~-1.175494351E-38 | 0和1.175494351E-38~3.402823466E+38 |
| DOUBLE | 8 | -1.7976931348623157E+308~2.2250738585072014E-308 | 0和2.2250738585072014E-308~1.7976931348623157E+308 |
| DECIMAL（M,D） | M+2 | -1.7976931348623157E+308~2.2250738585072014E-308 | 0和2.2250738585072014E-308~1.7976931348623157E+308 |

从上图中可以看出：DECIMAL类型的取值范围与DOUBLE类型相同。但是，请注意：DECIMAL类型的有效取值范围是由M和D决定的。其中，M表示的是数据的长 度，D表示的是小数点后的长度。比如，将数据类型为DECIMAL(6,2)的数据6.5243 插入数据库后显示的结果为6.52

## 1.3 字符串类型

在MySQL中常用CHAR 和 VARCHAR 表示字符串。两者不同的是：VARCHAR存储可变长度的字符串。

当数据为CHAR(M)类型时，不管插入值的长度是实际是多少它所占用的存储空间都是M个字节；而VARCHAR(M)所对应的数据所占用的字节数为实际长度加1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 插入值 | CHAR(3) | 存储需求（字节） | VARCHAR(3) | 存储需求（字节） |
| ‘’ | ‘’ | 3 | ‘’ | 1 |
| ‘a’ | ‘a’ | 3 | ‘a’ | 2 |
| ‘ab’ | ‘ab’ | 3 | ‘ab’ | 3 |
| ‘abc’ | ‘ab’ | 3 | ‘abc’ | 4 |
| ‘abcd’ | ‘ab’ | 3 | ‘abc’ | 4 |

## 1.4 字符串类型

文本类型用于表示大文本数据，例如，文章内容、评论、详情等，它的类型分为如下4种：

|  |  |
| --- | --- |
| 数据类型 | 储存范围 |
| TINYTEXT | 0~255字节 |
| TEXT | 0~65535字节 |
| MEDIUMTEXT | 0~16777215字节 |
| LONGTEXT | 0~4294967295字节 |

## 1.5 日期与时间类型

MySQL提供的表示日期和时间的数据类型分别是 ：YEAR、DATE、TIME、DATETIME 和 TIMESTAMP。下图列举了日期和时间数据类型所对应的字节数、取值范围、日期格式以及零值：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据类型 | 字节数 | 取值范围 | 日期格式 | 零值 |
| YEAR | 1 | 1901~2155 | YYYY | 0000 |
| DATE | 4 | 1000-01-01~9999-12-31 | YYYY-MM-DD | 0000-00-00 |
| TIME | 3 | -838：59：59~ 838：59：59 | HH:MM:SS | 00:00:00 |
| DATETIME | 8 | 1000-01-01 00:00:00~9999-12-31 23:59:59 | YYYY-MM-DD | 0000-00-00 00:00:00 |
| TIMESTAMP | 4 | 1970-01-01 00:00:01~2038-01-19 03:14:07 | YYYY-MM-DD | 0000-00-00 00:00:00 |

### 1.5.1 YEAR类型

YEAR类型用于表示年份，在MySQL中，可以使用以下三种格式指定YEAR类型 的值。

1、使用4位字符串或数字表示，范围为’1901’—'2155’或1901—2155。例如，输入 ‘2019’或2019插入到数据库中的值均为2019。

2、使用两位字符串表示，范围为’00’—‘99’。其中，‘00’—'69’范围的值会被转换为 2000—2069范围的YEAR值，‘70’—'99’范围的值会被转换为1970—1999范围的YEAR 值。例如，输入’19’插入到数据库中的值为2019。

3、使用两位数字表示，范围为1—99。其中，1—69范围的值会被转换为2001— 2069范围的YEAR值，70—99范围的值会被转换为1970—1999范围的YEAR值。例 如，输入19插入到数据库中的值为2019。

请注意：当使用YEAR类型时，一定要区分’0’和0。因为字符串格式的’0’表示的YEAR值是2000而数字格式的0表示的YEAR值是0000。

### 1.5.2 TIME类型

TIME类型用于表示时间值，它的显示形式一般为HH:MM:SS，其中，HH表示小时， MM表示分,SS表示秒。在MySQL中，可以使用以下3种格式指定TIME类型的值。

1、以’D HH:MM:SS’字符串格式表示。其中，D表示日可取0—34之间的值, 插入数据时，小时的值等于(DX24+HH)。例如，输入’2 11:30:50’插入数据库中的日期为59:30:50。

2、以’HHMMSS’字符串格式或者HHMMSS数字格式表示。 例如，输入’115454’或115454,插入数据库中的日期为11:54:54

3、使用CURRENT\_TIME或NOW()输入当前系统时间。

### 1.5.3 DATETIME类型

DATETIME类型用于表示日期和时间，它的显示形式为’YYYY-MM-DD HH: MM:SS’，其中，YYYY表示年，MM表示月，DD表示日，HH表示小时，MM表示分，SS 表示秒。在MySQL中，可以使用以下4种格式指定DATETIME类型的值。

以’YYYY-MM-DD HH:MM:SS’或者’YYYYMMDDHHMMSS’字符串格式表示的日期和时间，取值范围为’1000-01-01 00:00:00’—‘9999-12-3 23:59:59’。例如，输入’2019-01-22 09:01:23’或 ‘20140122\_0\_90123’插入数据库中的 DATETIME 值都为 2019-01-22 09:01:23。

1、以’YY-MM-DD HH:MM:SS’或者’YYMMDDHHMMSS’字符串格式表示的日期和时间，其中YY表示年，取值范围为’00’—‘99’。与DATE类型中的YY相同，‘00’— '69’范围的值会被转换为2000—2069范围的值，‘70’—'99’范围的值会被转换为1970—1999范围的值。

2、以YYYYMMDDHHMMSS或者YYMMDDHHMMSS数字格式表示的日期 和时间。例如，插入20190122090123或者190122090123,插入数据库中的DATETIME值都 为 2019-01-22 09:01:23。3、使用NOW来输入当前系统的日期和时间。

### 1.5.4 TIMESTAMP类型

TIMESTAMP类型用于表示日期和时间，它的显示形式与DATETIME相同但取值范围比DATETIME小。在此，介绍几种TIMESTAMP类型与DATATIME类型不同的形式：

1、使用CURRENT\_TIMESTAMP输入系统当前日期和时间。

2、输入NULL时系统会输入系统当前日期和时间。

3、无任何输入时系统会输入系统当前日期和时间。

## 1.6 二进制类型

在MySQL中常用BLOB存储二进制类型的数据，例如：图片、PDF文档等。BLOB类型分为如下四种：

|  |  |
| --- | --- |
| 数据类型 | 储存范围 |
| TINYBLOB | 0~255字节 |
| BLOB | 0~65535字节 |
| MEDIUMBLOB | 0~16777215字节 |
| LONGBLOB | 0~4294967295字节 |

# 2.全局

查看所有数据库

show databases;

# 3.数据库

**创建新的数据库**

create database 数据库名;

**使用和切换数据库**

use 数据库名;

**查看该数据库基本信息**

show create database 数据库名;

**删除数据库**

drop database 数据库名;

**查询出MySQL中所有的数据库**

show databases;

**将数据库的字符集修改为gbk**

alter database 数据库名 character set gbk;

**查看当前使用的数据库**

select database();

# 数据表

**创建数据表**

create table 表名(

字段1 字段类型,

字段2 字段类型,

…

字段n 字段类型

);

-- 学生表

create table student(

id int,

name varchar(20),

gender varchar(10),

birthday date

);

**查看当前数据库中所有表**

show tables;

**查看表的基本信息**

show create table 表名;

**查看表的字段信息**

desc 表名;

**修改表名**

alter table 原表名 rename to 修改后的表名;

**修改字段名**

alter table 表名 change 字段名 修改后的字段名 varchar(10);

**修改字段数据类型**

alter table 表名 modify 字段名 字段类型;

**增加字段**

alter table 表名 add 字段名 字段类型;

**删除字段**

alter table 表名 drop 字段名;

**删除数据表**

drop table 表名;

**输出数据表**

SELECT \* FROM 表名;

# 5.表的约束

为防止错误的数据被插入到数据表，MySQL中定义了一些维护数据库完整性的规则；这些规则常称为表的约束。常见约束如下：

|  |  |
| --- | --- |
| 约束条件 | 说明 |
| PRIMAPY KEY | 主键约束，用于唯一标识对应的记录 |
| FOREIGN KEY | 外键约束 |
| NOT NULL | 非空约束 |
| UNIQUE | 唯一性约束 |
| DEFAULT | 默认值约束，用于设置字段的默认值 |

## 5.1 主键约束

主键约束即primary key用于唯一的标识表中的每一行。被标识为主键的数据在表中是唯一的且其值不能为空。这点类似于我们每个人都有一个身份证号，并且这个身份证号是唯一的。

**主键约束基本语法：**

字段名 数据类型 primary key;

**示例：**

第1种方式

create table student(

id int primary key,

name varchar(20)

);

第2种方式

create table student01(

id int

name varchar(20),

primary key(id)

);

## 5.2非空约束

非空约束即 NOT NULL指的是字段的值不能为空.

**非空约束基本语法：**

字段名 数据类型 NOT NULL;

**示例：**

create table student02(

id int

name varchar(20) not null

);

## 5.3 默认值约束

默认值约束即DEFAULT用于给数据表中的字段指定默认值，即当在表中插入一条新记录时若未给该字段赋值，那么，数据库系统会自动为这个字段插入默认值；

**默认值约束基本语法：**

字段名 数据类型 DEFAULT 默认值；

**示例：**

create table student03(

id int,

name varchar(20),

gender varchar(10) default 'male'

);

## 5.4 唯一性约束

唯一性约束即UNIQUE用于保证数据表中字段的唯一性，即表中字段的值不能重复出现。

**唯一性约束基本语法：**

字段名 数据类型 UNIQUE;

**示例：**

create table student04(

id int,

name varchar(20) unique

);

## 5.5 外键约束

外键约束即FOREIGN KEY常用于多张表之间的约束。

**外键约束基本语法：**

-- 在创建数据表时语法如下：

CONSTRAINT 外键名 FOREIGN KEY (从表外键字段) REFERENCES 主表 (主键字段)

-- 将创建数据表创建后语法如下：

ALTER TABLE 从表名 ADD CONSTRAINT 外键名 FOREIGN KEY (从表外键字段) REFERENCES 主表 (主键字段);

**示例：创建一个学生表**

create table student05(

id int primary key,

name varchar(20)

);

**示例：创建一个班级表**

create table class(

classid int primary key,

studentid int

);

**示例：学生表作为主表，班级表作为副表设置外键**

alter table class add constraint fk\_class\_studentid foreign key(studentid) references student05(id);

# 数据一致性概念

大家知道：建立外键是为了保证数据的完整和统一性。但是，如果主表中的数据被删除或修改从表中对应的数据该怎么办呢？很明显，从表中对应的数据也应该被删除，否则数据库中会存在很多无意义的垃圾数据。

**删除外键**

**语法如下：**

alter table 从表名 drop foreign key 外键名;

**示例：删除外键**

alter table class drop foreign key fk\_class\_studentid;

**关于外键约束需要注意的细节**

1、从表里的外键通常为主表的主键  
2、从表里外键的数据类型必须与主表中主键的数据类型一致  
3、主表发生变化时应注意主表与从表的数据一致性问题

# 7. 数据表插入数据

在MySQL通过INSERT语句向数据表中插入数据。

**先准备一张学生表，代码如下：**

create table student(

id int,

name varchar(30),

age int,

gender varchar(30)

);

## 7.1 为表中所有字段插入数据

每个字段与其值是严格一一对应的。也就是说：每个值、值的顺序、值的类型必须与对应的字段相匹配。但是，各字段也无须与其在表中定义的顺序一致，它们只要与 VALUES中值的顺序一致即可。  
**语法如下：**

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

**示例：向学生表中插入一条学生信息**

insert into student (id,name,age,gender) values (1,'bob',16,'male');

## 7.2 为表中指定字段插入数据

**语法如下：**

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

-- 插入数据的方法基本和为表中所有字段插入数据，一样，只是需要插入的字段由你自己指定

## 7.3 同时插入多条记录

**语法如下：**

INSERT INTO 表名 [(字段名1,字段名2,...)]VALUES (值 1,值 2,…),(值 1,值 2,…),...;

-- 在该方式中：(字段名1,字段名2,…)是可选的，它用于指定插入的字段名；(值 1,值 2,…),(值 1,值 2,…)表示要插入的记录，该记录可有多条并且每条记录之间用逗号隔开。

# 更新数据

在MySQL通过UPDATE语句更新数据表中的数据。在此，我们将用六、数据表插入数据中的student学生表。

## 8.1 UPDATE基本语法

UPDATE 表名 SET 字段名1=值1[,字段名2 =值2,…] [WHERE 条件表达式];

-- 在该语法中：字段名1、字段名2…用于指定要更新的字段名称；值1、值 2…用于表示字段的新数据；WHERE 条件表达式 是可选的，它用于指定更新数据需要满足的条件

## 8.2 UPDATE更新部分数据

**示例：将name为tom的记录的age设置为20并将其gender设置为female**

update student set age=20,gender='female' where name='tom';

## 8.3 UPDATE更新全部数据

**示例：将所有记录的age设置为18**

update student set age=18;

# 删除数据

在MySQL通过DELETE语句删除数据表中的数据。在此，我们先准备一张数据表，代码如下：

-- 创建学生表

create table student(

id int,

name varchar(30),

age int,

gender varchar(30)

);

-- 插入数据

insert into student (id,name,age,gender) values (2,'lucy',17,'female'),(3,'jack',19,'male'),(4,'tom',18,'male'),(5,'sal',19,'female'),(6,'sun',20,'male')

,(7,'sad',13,'female'),(8,'sam',14,'male');

## DELETE基本语法

在该语法中：表名用于指定要执行删除操作的表；[WHERE 条件表达式]为可选参数用于指定删除的条件。

DELETE FROM 表名 [WHERE 条件表达式];

## DELETE删除部分数据

示例：删除age等于14的所有记录 MySQL命令：

delete from student where age=14;

## DELETE删除全部数据

示例：删除student表中的所有记录 MySQL命令：

delete from student;

## TRUNCATE和DETELE的区别

9.4.1TRUNCATE和DETELE都能实现删除表中的所有数据的功能，但两者也是有区别的：

9.4.2 DELETE语句后可跟WHERE子句，可通过指定WHERE子句中的条件表达式只删除满足条件的部分记录；但是，TRUNCATE语句只能用于删除表中的所有记录。

9.4.3 使用TRUNCATE语句删除表中的数据后，再次向表中添加记录时自动增加字段的默认初始值重新由1开始；使用DELETE语句删除表中所有记录后，再次向表中添加记录时自动增加字段的值为删除时该字段的最大值加1

9.4.4 DELETE语句是DML语句，TRUNCATE语句通常被认为是DDL语句

# 数据表简单查询

## 10.1 简单查询概述

简单查询即不含where的select语句。在此，我们讲解简单查询中最常用的两种查询：查询所有字段和查询指定字段。

在此，先准备测试数据，代码如下：

-- 创建数据库

DROP DATABASE IF EXISTS mydb;

CREATE DATABASE mydb;

USE mydb;

-- 创建student表

CREATE TABLE student (

sid CHAR(6),

sname VARCHAR(50),

age INT,

gender VARCHAR(50) DEFAULT 'male'

);

-- 向student表插入数据

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1001', 'lili', 14, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1002', 'wang', 15, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1003', 'tywd', 16, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1004', 'hfgs', 17, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1005', 'qwer', 18, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1006', 'zxsd', 19, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1007', 'hjop', 16, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1008', 'tyop', 15, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1009', 'nhmk', 13, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1010', 'xdfv', 17, 'female');

## 10.2 查询所有字段（方法不唯一只是举例）

查询所有字段 MySQL命令：

select \* from student;

## 10.3 查询指定字段（sid、sname）

查询指定字段（sid、sname） MySQL命令：

select sid,sname from student;

## 10.4 常数的查询

在SELECT中除了书写列名，还可以书写常数。可以用于标记

常数的查询日期标记 MySQL命令：

select sid,sname,'2021-03-02' from student;

## 10.5 从查询结果中过滤重复数据

在使用DISTINCT 时需要注意：  
在SELECT查询语句中DISTINCT关键字只能用在第一个所查列名之前。  
MySQL命令：

select distinct gender from student;

## 10.6 算术运算符（举例加运算符）

在SELECT查询语句中还可以使用加减乘除运算符。

查询学生10年后的年龄 MySQL命令：

select sname,age+10 from student;

# 函数

在此，先准备测试数据，代码如下：

-- 创建数据库

DROP DATABASE IF EXISTS mydb;

CREATE DATABASE mydb;

USE mydb;

-- 创建student表

CREATE TABLE student (

sid CHAR(6),

sname VARCHAR(50),

age INT,

gender VARCHAR(50) DEFAULT 'male'

);

-- 向student表插入数据

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1001', 'lili', 14, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1002', 'wang', 15, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1003', 'tywd', 16, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1004', 'hfgs', 17, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1005', 'qwer', 18, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1006', 'zxsd', 19, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1007', 'hjop', 16, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1008', 'tyop', 15, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1009', 'nhmk', 13, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1010', 'xdfv', 17, 'female');

## 11.1 聚合函数

在开发中，我们常常有类似的需求：统计某个字段的最大值、最小值、 平均值等等。为此，MySQL中提供了聚合函数来实现这些功能。所谓聚合，就是将多行汇总成一行；其实，所有的聚合函数均如此——输入多行，输出一行。聚合函数具有自动滤空的功能，若某一个值为NULL，那么会自动将其过滤使其不参与运算。

聚合函数使用规则：

只有SELECT子句和HAVING子句、ORDER BY子句中能够使用聚合函数。例如，在WHERE子句中使用聚合函数是错误的。

接下来，我们学习常用聚合函数。

### 11.1.1 count()

统计表中数据的行数或者统计指定列其值不为NULL的数据个数

查询有多少该表中有多少人

MySQL命令：

select count(\*) from student;

### 11.1.2 max()

计算指定列的最大值，如果指定列是字符串类型则使用字符串排序运算

查询该学生表中年纪最大的学生

MySQL命令：

select max(age) from student;

### 11.1.3 min()

计算指定列的最小值，如果指定列是字符串类型则使用字符串排序运算

查询该学生表中年纪最小的学生 MySQL命令：

select sname,min(age) from student;

### 11.1.4 sum()

计算指定列的数值和，如果指定列类型不是数值类型则计算结果为0

查询该学生表中年纪的总和 MySQL命令：

select sum(age) from student;

### 11.1.5 avg()

计算指定列的平均值，如果指定列类型不是数值类型则计算结果为

查询该学生表中年纪的平均数 MySQL命令：

select avg(age) from student;

## 11.2 其他常用函数

这里我就不一一举例了，基本混个眼熟，以后用到再细说。

### 11.2.1 时间函数

SELECT NOW();

SELECT DAY (NOW());

SELECT DATE (NOW());

SELECT TIME (NOW());

SELECT YEAR (NOW());

SELECT MONTH (NOW());

SELECT CURRENT\_DATE();

SELECT CURRENT\_TIME();

SELECT CURRENT\_TIMESTAMP();

SELECT ADDTIME('14:23:12','01:02:01');

SELECT DATE\_ADD(NOW(),INTERVAL 1 DAY);

SELECT DATE\_ADD(NOW(),INTERVAL 1 MONTH);

SELECT DATE\_SUB(NOW(),INTERVAL 1 DAY);

SELECT DATE\_SUB(NOW(),INTERVAL 1 MONTH);

SELECT DATEDIFF('2019-07-22','2019-05-05');

### 11.2.2 字符串函数

--连接函数

SELECT CONCAT ()

--

SELECT INSTR ();

--统计长度

SELECT LENGTH();

### 11.2.3 数学函数

-- 绝对值

SELECT ABS(-136);

-- 向下取整

SELECT FLOOR(3.14);

-- 向上取整

SELECT CEILING(3.14);

# 条件查询

数据库中存有大量数据，我们可根据需求获取指定的数据。此时，我们可在查询语句中通过WHERE子句指定查询条件对查询结果进行过滤。  
 在开始学习条件查询之前，我们先准备测试数据，代码如下：

-- 创建数据库

DROP DATABASE IF EXISTS mydb;

CREATE DATABASE mydb;

USE mydb;

-- 创建student表

CREATE TABLE student (

sid CHAR(6),

sname VARCHAR(50),

age INT,

gender VARCHAR(50) DEFAULT 'male'

);

-- 向student表插入数据

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1001', 'lili', 14, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1002', 'wang', 15, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1003', 'tywd', 16, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1004', 'hfgs', 17, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1005', 'qwer', 18, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1006', 'zxsd', 19, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1007', 'hjop', 16, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1008', 'tyop', 15, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1009', 'nhmk', 13, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1010', 'xdfv', 17, 'female');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1012', 'lili', 14, 'male');

INSERT INTO student (sid,sname,age,gender) VALUES ('S\_1013', 'wang', 15, 'female');

## 12.1 使用关系运算符查询

在WHERE中可使用关系运算符进行条件查询，常用的关系运算符如下所示：

|  |  |
| --- | --- |
| 关系运算符 | 说明 |
| = | 等于 |
| <> | 不等于 |
| != | 不等于 |
| < | 小于 |
| <= | 小于等于 |
| > | 大于 |
| >= | 大于等于 |

查询年龄等于或大于17的学生的信息 MySQL命令：

select \* from student where age>=17;

## 12.2 使用IN关键字查询

IN关键字用于判断某个字段的值是否在指定集合中。如果字段的值恰好在指定的集合中，则将字段所在的记录将査询出来。

查询sid为S\_1002和S\_1003的学生信息 MySQL命令：

select \* from student where sid in ('S\_1002','S\_1003');

查询sid为S\_1001以外的学生的信息 MySQL命令：

select \* from student where sid not in ('S\_1001');

## 12.3.使用BETWEEN AND关键字查询

BETWEEN AND用于判断某个字段的值是否在指定的范围之内。如果字段的值在指定范围内，则将所在的记录将查询出来

查询15到18岁的学生信息 MySQL命令：

select \* from student where age between 15 and 18;

查询不是15到18岁的学生信息 MySQL命令：

select \* from student where age not between 15 and 18;

## 12.4 使用空值查询

在MySQL中，使用 IS NULL关键字判断字段的值是否为空值。请注意：空值NULL不同于0，也不同于空字符串

由于student表没有空值就不演示查询空值的了

查询sname不为空值的学生信息 MySQL命令：

select \* from student where sname is not null;

## 12.5 使用AND关键字查询

在MySQL中可使用AND关键字可以连接两个或者多个查询条件。

查询年纪大于15且性别为male的学生信息 MySQL命令：

select \* from student where age>15 and gender='male';

## 12.6 使用OR关键字查询

在使用SELECT语句查询数据时可使用OR关键字连接多个査询条件。在使用OR关键字时，只要记录满足其中任意一个条件就会被查询出来

查询年纪大于15或者性别为male的学生信息 MySQL命令：

select \* from student where age>15 or gender='male';

## 12.7 使用LIKE关键字查询

MySQL中可使用LIKE关键字可以判断两个字符串是否相匹配

### 12.7.1 普通字符串

查询sname中与wang匹配的学生信息 MySQL命令：

select \* from student where sname like 'wang';

### 12.7.2 含有%通配的字符串

%用于匹配任意长度的字符串。例如，字符串“a%”匹配以字符a开始任意长度的字符串

查询学生姓名以li开始的记录 MySQL命令：

select \* from student where sname like 'li%';

查询学生姓名以g结尾的记录 MySQL命令：

select \* from student where sname like '%g';

查询学生姓名包含s的记录 MySQL命令：

select \* from student where sname like '%s%';

### 12.7.3 含有\_通配的字符串

下划线通配符只匹配单个字符，如果要匹配多个字符，需要连续使用多个下划线通配符。例如，字符串“ab\_”匹配以字符串“ab”开始长度为3的字符串，如abc、abp等等；字符串“a\_\_d”匹配在字符“a”和“d”之间包含两个字符的字符串，如"abcd"、"atud"等等。

查询学生姓名以zx开头且长度为4的记录 MySQL命令：

select \* from student where sname like 'zx\_\_';

查询学生姓名以g结尾且长度为4的记录 MySQL命令：

select \* from student where sname like '\_\_\_g';

## 12.8 使用LIMIT限制查询结果的数量

当执行查询数据时可能会返回很多条记录，而用户需要的数据可能只是其中的一条或者几条

查询学生表中年纪最小的3位同学 MySQL命令：

select \* from student order by age asc limit 3;

## 12.9 使用GROUP BY进行分组查询

GROUP BY 子句可像切蛋糕一样将表中的数据进行分组，再进行查询等操作。换言之，可通俗地理解为：通过GROUP BY将原来的表拆分成了几张小表。

接下来，我们通过一个例子开始学习GROUP BY，代码如下：

-- 创建数据库

DROP DATABASE IF EXISTS mydb;

CREATE DATABASE mydb;

USE mydb;

-- 创建员工表

CREATE TABLE employee (

id int,

name varchar(50),

salary int,

departmentnumber int

);

-- 向员工表中插入数据

INSERT INTO employee values(1,'tome',2000,1001);

INSERT INTO employee values(2,'lucy',9000,1002);

INSERT INTO employee values(3,'joke',5000,1003);

INSERT INTO employee values(4,'wang',3000,1004);

INSERT INTO employee values(5,'chen',3000,1001);

INSERT INTO employee values(6,'yukt',7000,1002);

INSERT INTO employee values(7,'rett',6000,1003);

INSERT INTO employee values(8,'mujk',4000,1004);

INSERT INTO employee values(9,'poik',3000,1001);

### 12.9.1 GROUP BY和聚合函数一起使用

统计各部门员工个数 MySQL命令：

select count(\*), departmentnumber from employee group by departmentnumber;

统计部门编号大于1001的各部门员工个数 MySQL命令：

select count(\*), departmentnumber from employee where departmentnumber>1001 group by departmentnumber;

### 12.9.2 GROUP BY和聚合函数以及HAVING一起使用

统计工资总和大于8000的部门 MySQL命令：

select sum(salary),departmentnumber from employee group by departmentnumber having sum(salary)>8000;

## 12.10 使用ORDER BY对查询结果排序

从表中査询出来的数据可能是无序的或者其排列顺序不是我们期望的。为此，我们可以使用ORDER BY对查询结果进行排序

其语法格式如下所示：

SELECT 字段名1,字段名2,…

FROM 表名

ORDER BY 字段名1 [ASC 丨 DESC],字段名2 [ASC | DESC];

在该语法中：字段名1、字段名2是查询结果排序的依据；参数 ASC表示按照升序排序，DESC表示按照降序排序；默认情况下，按照ASC方式排序。通常情况下，ORDER BY子句位于整个SELECT语句的末尾。

查询所有学生并按照年纪大小升序排列 MySQL命令：

select \* from student order by age asc;

查询所有学生并按照年纪大小降序排列 MySQL命令：

select \* from student order by age desc;

# 别名设置

在査询数据时可为表和字段取別名，该别名代替表和字段的原名参与查询操作。

操作的表事先已准备。

## 13.1 为表取别名

在查询操作时，假若表名很长使用起来就不太方便，此时可为表取一个別名，用该别名来代替表的名称。语法格式如下所示：

SELECT \* FROM 表名 [AS] 表的别名 WHERE .... ;

将student改为stu查询整表 MySQL命令：

select \* from student as stu;

## 13.2 为字段取别名

在查询操作时，假若字段名很长使用起来就不太方便，此时可该字段取一个別名，用该别名来代替字段的名称。语法格式如下所示：

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

将student中的name取别名为“姓名” 查询整表 MySQL命令：

select name as '姓名',id from student;

# 表的关联关系

在实际开发中数据表之间存在着各种关联关系。在此，介绍MySQL中数据表的三种关联关系。

**多对一**

多对一(亦称为一对多)是数据表中最常见的一种关系。例如：员工与部门之间的关系，一个部门可以有多个员工；而一个员工不能属于多个部门只属于某个部门。在多对一的表关系 中，应将外键建在多的一方否则会造成数据的冗余。

**多对多**

多对多是数据表中常见的一种关系。例如：学生与老师之间的关系，一个学生可以有多个老师而且一个老师有多个学生。通常情况下，为了实现这种关系需要定义一张中间表(亦称为连接表)该表会存在两个外键分别参照老师表和学生表。

**一对一**

在开发过程中，一对一的关联关系在数据库中并不常见；因为以这种方式存储的信息通常会放在同一张表中。

接下来，我们来学习在一对多的关联关系中如果添加和删除数据。先准备一些测试数据，代码如下：

DROP TABLE IF EXISTS student;

DROP TABLE IF EXISTS class;

-- 创建班级表

CREATE TABLE class(

cid int(4) NOT NULL PRIMARY KEY,

cname varchar(30)

);

-- 创建学生表

CREATE TABLE student(

sid int(8) NOT NULL PRIMARY KEY,

sname varchar(30),

classid int(8) NOT NULL

);

-- 为学生表添加外键约束

ALTER TABLE student ADD CONSTRAINT fk\_student\_classid FOREIGN KEY(classid) REFERENCES class(cid);

-- 向班级表插入数据

INSERT INTO class(cid,cname)VALUES(1,'Java');

INSERT INTO class(cid,cname)VALUES(2,'Python');

-- 向学生表插入数据

INSERT INTO student(sid,sname,classid)VALUES(1,'tome',1);

INSERT INTO student(sid,sname,classid)VALUES(2,'lucy',1);

INSERT INTO student(sid,sname,classid)VALUES(3,'lili',2);

INSERT INTO student(sid,sname,classid)VALUES(4,'domi',2);

## 14.1 关联查询

查询Java班的所有学生 MySQL命令：

select \* from student where classid=(select cid from class where cname='Java');

## 14.2 关于关联关系的删除数据

请从班级表中删除Java班级。在此，请注意：班级表和学生表之间存在关联关系；要删除Java班级，应该先删除学生表中与该班相关联的学生。否则，假若先删除Java班那么学生表中的cid就失去了关联

删除Java班 MySQL命令：

delete from student where classid=(select cid from class where cname='Java');

delete from class where cname='Java';

# 多表连接查询

## 15.1 交叉连接查询

交叉连接返回的结果是被连接的两个表中所有数据行的笛卡儿积；比如：集合A={a,b}，集合B={0,1,2}，则集合A和B的笛卡尔积为{(a,0),(a,1),(a,2),(b,0),(b,1),(b,2)}。所以，交叉连接也被称为笛卡尔连接，其语法格式如下：

SELECT \* FROM 表1 CROSS JOIN 表2;

在该语法中：CROSS JOIN用于连接两个要查询的表，通过该语句可以查询两个表中所有的数据组合。

**由于这个交叉连接查询在实际运用中没有任何意义，所以只做为了解即可。**

## 15.2 内连接查询

内连接(Inner Join)又称简单连接或自然连接，是一种非常常见的连接查询。内连接使用比较运算符对两个表中的数据进行比较并列出与连接条件匹配的数据行，组合成新的 记录。也就是说在内连接查询中只有满足条件的记录才能出现在查询结果中。其语法格式如下：

SELECT 查询字段1,查询字段2, ... FROM 表1 [INNER] JOIN 表2 ON 表1.关系字段=表2.关系字段

在该语法中：INNER JOIN用于连接两个表，ON来指定连接条件；其中INNER可以省略。

准备数据，代码如下：

-- 若存在数据库mydb则删除

DROP DATABASE IF EXISTS mydb;

-- 创建数据库mydb

CREATE DATABASE mydb;

-- 选择数据库mydb

USE mydb;

-- 创建部门表

CREATE TABLE department(

did int (4) NOT NULL PRIMARY KEY,

dname varchar(20)

);

-- 创建员工表

CREATE TABLE employee (

eid int (4) NOT NULL PRIMARY KEY,

ename varchar (20),

eage int (2),

departmentid int (4) NOT NULL

);

-- 向部门表插入数据

INSERT INTO department VALUES(1001,'财务部');

INSERT INTO department VALUES(1002,'技术部');

INSERT INTO department VALUES(1003,'行政部');

INSERT INTO department VALUES(1004,'生活部');

-- 向员工表插入数据

INSERT INTO employee VALUES(1,'张三',19,1003);

INSERT INTO employee VALUES(2,'李四',18,1002);

INSERT INTO employee VALUES(3,'王五',20,1001);

INSERT INTO employee VALUES(4,'赵六',20,1004);

查询员工姓名及其所属部门名称 MySQL命令：

select employee.ename,department.dname from department inner join employee on department.did=employee.departmentid;

## 15.3 外连接查询

在使用内连接查询时我们发现：返回的结果只包含符合查询条件和连接条件的数据。但是，有时还需要在返回查询结果中不仅包含符合条件的数据，而且还包括左表、右表或两个表中的所有数据，此时我们就需要使用外连接查询。外连接又分为左(外)连接和右(外)连接。其语法格式如下：

SELECT 查询字段1,查询字段2, ... FROM 表1 LEFT | RIGHT [OUTER] JOIN 表2 ON 表1.关系字段=表2.关系字段 WHERE 条件

由此可见，外连接的语法格式和内连接非常相似，只不过使用的是LEFT [OUTER] JOIN、RIGHT [OUTER] JOIN关键字。其中，关键字左边的表被称为左表，关键字右边的表被称为右表；OUTER可以省略。

在使用左(外)连接和右(外)连接查询时，查询结果是不一致的，具体如下：

LEFT [OUTER] JOIN 左(外)连接：返回包括左表中的所有记录和右表中符合连接条件的记录。RIGHT [OUTER] JOIN 右(外)连接：返回包括右表中的所有记录和左表中符合连接条件的记录。

先准备数据，代码如下：

-- 若存在数据库mydb则删除

DROP DATABASE IF EXISTS mydb;

-- 创建数据库mydb

CREATE DATABASE mydb;

-- 选择数据库mydb

USE mydb;

-- 创建班级表

CREATE TABLE class(

cid int (4) NOT NULL PRIMARY KEY,

cname varchar(20)

);

-- 创建学生表

CREATE TABLE student (

sid int (4) NOT NULL PRIMARY KEY,

sname varchar (20),

sage int (2),

classid int (4) NOT NULL

);

-- 向班级表插入数据

INSERT INTO class VALUES(1001,'Java');

INSERT INTO class VALUES(1002,'C++');

INSERT INTO class VALUES(1003,'Python');

INSERT INTO class VALUES(1004,'PHP');

-- 向学生表插入数据

INSERT INTO student VALUES(1,'张三',20,1001);

INSERT INTO student VALUES(2,'李四',21,1002);

INSERT INTO student VALUES(3,'王五',24,1002);

INSERT INTO student VALUES(4,'赵六',23,1003);

INSERT INTO student VALUES(5,'Jack',22,1009);

准备这组数据有一定的特点，为的是让大家直观的看出左连接与右连接的不同之处：班级编号为1004的PHP班级没有学生。学号为5的学生Jack班级编号为1009，该班级编号并不在班级表中。

### 15.3.1 左（外）连接查询

左(外)连接的结果包括LEFT JOIN子句中指定的左表的所有记录，以及所有满足连接条件的记录。如果左表的某条记录在右表中不存在则在右表中显示为空。

查询每个班的班级ID、班级名称及该班的所有学生的名字 MySQL命令：

select class.cid,class.cname,student.sname from class left outer join student on class.cid=student.classid;

展示结果分析：分别找出Java班、C++班、Python班的学生。右表的Jack不满足查询条件故其没有出现在查询结果中。虽然左表的PHP班没有学生，但是任然显示了PHP的信息；但是，它对应的学生名字为NULL。

### 15.3.2 右（外）连接查询

右(外)连接的结果包括RIGHT JOIN子句中指定的右表的所有记录，以及所有满足连接条件的记录。如果右表的某条记录在左表中没有匹配，则左表将返回空值。

查询每个班的班级ID、班级名称及该班的所有学生的名字 MySQL命令：

select class.cid,class.cname,student.sname from class right outer join student on class.cid=student.classid;

展示结果分析：分别找出Java班、C++班、Python班的学生。左表的PHP班不满足查询条件故其没有出现在查询结果中。虽然右表的jack没有对应班级，但是任然显示王跃跃的信息；但是，它对应的班级以及班级编号均为NULL。

# 子查询

子查询是指一个查询语句嵌套在另一个查询语句内部的查询；该查询语句可以嵌套在一个 SELECT、SELECT…INTO、INSERT…INTO等语句中。在执行查询时，首先会执行子查询中的语句，再将返回的结果作为外层查询的过滤条件。在子査询中通常可以使用比较运算符和IN、EXISTS、ANY、ALL等关键字。

准备数据，代码如下：

DROP TABLE IF EXISTS student;

DROP TABLE IF EXISTS class;

-- 创建班级表

CREATE TABLE class(

cid int (4) NOT NULL PRIMARY KEY,

cname varchar(20)

);

-- 创建学生表

CREATE TABLE student (

sid int (4) NOT NULL PRIMARY KEY,

sname varchar (20),

sage int (2),

classid int (4) NOT NULL

);

-- 向班级表插入数据

INSERT INTO class VALUES(1001,'Java');

INSERT INTO class VALUES(1002,'C++');

INSERT INTO class VALUES(1003,'Python');

INSERT INTO class VALUES(1004,'PHP');

INSERT INTO class VALUES(1005,'Android');

-- 向学生表插入数据

INSERT INTO student VALUES(1,'张三',20,1001);

INSERT INTO student VALUES(2,'李四',21,1002);

INSERT INTO student VALUES(3,'王五',24,1003);

INSERT INTO student VALUES(4,'赵六',23,1004);

INSERT INTO student VALUES(5,'小明',21,1001);

INSERT INTO student VALUES(6,'小红',26,1001);

INSERT INTO student VALUES(7,'小亮',27,1002);

## 16.1 带比较运算符的子查询

比较运算符前面我们提到过得，就是>、<、=、>=、<=、!=等

查询张三同学所在班级的信息 MySQL命令：

select \* from class where cid=(select classid from student where sname='张三');

查询比张三同学所在班级编号还大的班级的信息 MySQL命令：

select \* from class where cid>(select classid from student where sname='张三');

## 16.2 带EXISTS关键字的子查询

EXISTS关键字后面的参数可以是任意一个子查询， 它不产生任何数据只返回TRUE或FALSE。当返回值为TRUE时外层查询才会 执行

假如王五同学在学生表中则从班级表查询所有班级信息 MySQL命令：

select \* from class where exists (select \* from student where sname='王五');

## 16.3 带ANY关键字的子查询

ANY关键字表示满足其中任意一个条件就返回一个结果作为外层查询条件。

查询比任一学生所属班级号还大的班级编号 MySQL命令：

select \* from class where cid > any (select classid from student);

## 16.4 带ALL关键字的子查询

ALL关键字与ANY有点类似，只不过带ALL关键字的子査询返回的结果需同时满足所有内层査询条件。

查询比所有学生所属班级号还大的班级编号 MySQL命令：

select \* from class where cid > all (select classid from student);

# 总结

重要（从关键字分析）：

查询语句的书写顺序和执行顺序

select ===> from ===> where ===> group by ===> having ===> order by ===> limit

查询语句的执行顺序

from ===> where ===> group by ===> having ===> select ===> order by ===> limi

# SQL简述

## 18.1 SQL的概述

Structure Query Language(结构化查询语言)简称SQL，它被美国国家标准局(ANSI)确定为关系型数据库语言的美国标准，后被国际化标准组织(ISO)采纳为关系数据库语言的国际标准。数据库管理系统可以通过SQL管理数据库；定义和操作数据，维护数据的完整性和安全性。

## 18.2 SQL的优点

1、简单易学，具有很强的操作性

2、绝大多数重要的数据库管理系统均支持SQL

3、高度非过程化；用SQL操作数据库时大部分的工作由DBMS自动完成

## 18.3 SQL的分类

1、DDL(Data Definition Language) 数据定义语言，用来操作数据库、表、列等； 常用语句：CREATE、 ALTER、DROP

2、DML(Data Manipulation Language) 数据操作语言，用来操作数据库中表里的数据；常用语句：INSERT、 UPDATE、 DELETE

3、DCL(Data Control Language) 数据控制语言，用来操作访问权限和安全级别； 常用语句：GRANT、DENY

4、DQL(Data Query Language) 数据查询语言，用来查询数据 常用语句：SELECT

————————————————

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

原文链接：https://blog.csdn.net/weixin\_45851945/article/details/114287877