

主题

▼ 基础

▼ 类型

▼ 数值类型

数值类型

数值类型	大小	signed范围	unsigned范围
tinyint	1 byte	(-128,127)	
smallint	2 byte	(-32768,32767)	
Mediumint	3 byte	(-8388608,8388607)	
int/integer	4 byte	()	
bigint	8 byte		
float	4 byte		
double	8 byte		
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	

▼ 字符串类型

字符串类型

类型	大小
char	0-255 bytes (定长)
varchar	0-65535 bytes (变长)
Tinyblob	0-255 bytes (二进制数据)
Tinytext	0-255 bytes (文本)
Blob	0-65535 bytes (二进制数据)
text	0-65535 bytes (文本)
mediumblob	0-16777215 bytes (二进制数据)
mediumtext	0-16777215 bytes (文本)
longblob	0-4294967295 bytes (二进制数据)
Longtext	0-4294967295 bytes (文本)

▼ 日期时间类型

日期时间类型

类型	大小	格式
Date	3	Yyyy-mm-dd
time	3	Hh:mm:ss
year	1	Yyyy
datetime	8	Yyyy-mm-dd Hh:mm:ss
timestamp (2038.1.19 03:14:07) 时间戳	4	Yyyy-mm-dd Hh:mm:ss

▼ DDL

▼ 查询

- 查询所有数据库
show databases;
- 查询表结构
desc 表名
- 查询建表语句
show create table 表名
- 查询当前数据库
select databases();

▼ 增删改

▼ 数据库

- 创建
create database [if not exists] 名[default charset 字符集][collate 排序规则];
- 删除
drop database [if exists] 数据库名
- 使用
use 数据库名;

▼ 表

- 创建
CREATE TABLE 表名 (
字段1 字段1类型 [COKMENT 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] ;

- 添加字段

alter table 表名 *add* 字段名 类型 [注释]/[约束];

- 修改数据类型

alter table 表名 *modify* 字段名 新类型;

- 修改字段名和类型

alter table 表名 *change* 旧字段名 新字段名 类型 [注释]/[约束];

- 删除字段

alter table 表名 *drop* 字段名;

- 修改表名

alter table tablename *rename to* new name;

- 删除表

drop table [if exists] 表名;

- 删除指定表并重新创建

truncate table 表名;

▼ DML

- 给指定字段添加数据

insert into 表名 (字段1,字段2) *values* (值1,值2);

- 给全部字段添加数据

insert into 表名 *values* (值1,值2);

- 批量

insert into 表名 (字段1,字段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);

- 修改数据

upadte 表名 *set* 字段1=值1,字段2=值2 [where 条件];

主题

- 删除数据

`delete from 表名 [where 条件];`

`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。
`DELETE` 语句不能删除某一个字段的值（可以使用`UPDATE`）。

▼ DQL

- 查询多个字段

`select 字段1,字段2,字段3, from 表名;`

`select *, from 表名;`

- 设置别名

`select 字段1[as 别名1],字段2[as 别名2],字段3[as 别名3], from 表名;`

- 去除重复记录

`select distinct 字段列表 from 表名;`

- 条件

(同Java)

`between ...and...` 在某个范围

`in(.....,.....)` 在`in`后列表中值多选一

`like` 占位符 (模糊匹配 _匹配当个字符 %匹配任意字符)

- 条件查询

`select 字段1,字段2,字段3, from 表名 where 条件`

▼ 聚合函数

`select 聚合函数(字段名) from 表名;`

- count

统计数量

- max

最大值

- min

最小值

- avg

平均值

主题

- **sum**

求和

▼ 分组查询

`select` 字段列表 `from` 表名 [`where` 条件] `group by` 分组字段名 [分组后过滤条件]

▼ **where与having区别**

- 执行顺序：*where* > 聚合函数 > *having*。
- 分组之后，查询的字段一般为聚合函数和分组字段，查询其他字段无任何意义。

- **执行时机不同**

*where*是分组之前进行过滤，不满足*where*条件，不参与分组；而*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。

▼ 排序

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

- **ASC 升序(默认)**

- **desc 降序**

- **分页查询**

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

起始索引从0开始，起始索引=（查询页码-1）每页显示记录数。*

分页查询是数据库的方言，不同的数据库有不同的实现，MySQL中是LIMIT。

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*

编写顺序: `select` `from` `where` `group by` `having` `order by` `limit`

执行顺序: `from` `where` `group by` `having` `select` `order by` `limit`

▼ DCL

- **查询用户**

`USE mysql;`

`SELECT * FROM user;`

- **创建用户**

`create user` '用户名'@'主机名' `identified by` '密码'

主题

- 修改密码

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'

(主机名可以使用%通配)

▼ 权限控制

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。

▼ 图片

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

- 查询权限

SHOW GRANTS FOR '用户名'@'主机名';

- 授予权限

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

- 撤销权限

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

▼ 函数

▼ 字符串函数

CONCAT(S1,52,...Sn)	字符串拼接，将S1， S2， ..Sn拼接成一个字符串
LOWER(9\$r)	将字符串str全部转为小写
UPPER(str)	将字符串str全部转为大写
LPAD(str,n,pad)	左填充，用字符串pad对str的左边进行填充，达到n个字符串长度
RPAD(str,n,pad)	右填充，用字符串pad对str的右边进行填充，达到n个字符串长度
TRIM(str)	去掉字符串头部和尾部的空格
SUBSTRING(str,start,len)	返回从字符串str从start位置起的len个长度的字符串

•

▼ 数值函数

- CEIL(x)
向上取整
- FLOOR(x)
向下取整
- MOD(x,y)
取模
- RAND()
返回0~1内的随机数
- ROUND(x,y)
求参数x的四舍五入的值，保留y位小数

▼ 日期函数

CURDATE)	返回当前日期
CURTIME	返回当前时间
Now()	返回当前日期和时间
YEAR(date)	获取指定date的年份
MONTH(date)	获取指定date的月份
DAY(date)	获取指定date的日期
DATE_ADD(date, INTERVAL expr type)	返回一个日期/时间值加上一个时间间隔expr后的时间值
DATEDIFF(date1, date2)	返回起始时间date1 和 结束时间date2之间的天数

•

▼ 流程函数

主题

IF(value, t, f)	如果value为true，则返回t，否则返回f
IFNULL(value1, value2)	如果value1不为空，返回value1，否则返回value2
CASE WHEN [vall] THEN [res1] ... ELSE [default] END	如果vall为true，返回res1，…香则返回default默认值
CASE [expr] WHEN [vall] THEN [res1] ... ELSE [default] END	如果expr的值等于vall，返回res1，…否则返回default默认值

▼ 约束

▼ 图片

非空约束	限制该字段的数据不能为null	NOT NULL
唯一约束	保证该字段的所有数据都是唯一、不重复的	UNIQUE
主键约束	主键是一行数据的唯一标识，要求非空且唯一	PRIMARY KEY
默认约束	保存数据时，如果未指定该字段的值，则采用默认值	DEFAULT
检查约束（8.0.16版本之后）	保证字段值满足某一个条件	CHECK
外键约束 ↓	用来让两张表的数据之间建立连接，保证数据的一致性和完整性	FOREIGN KEY

• 外键约束

- ALTER TABLE 表名(字段名 数据类型,

.....

ADD CONSTRAINT 外键名称 FOREIGN KEY（外键字段名）REFERENCES 主表（主表列名）
);

- ALTER TABLE 表名 ADD CONSTRAINT 外键名称 FOREIGN KEY（外键字段名）REFERENCES 主表（主表列名）；

▼ 删除/更新行为

(on update on delete)

ALTER TABLE 表名 ADD CONSTRAINT 外键名称 FOREIGN KEY（外键字段）REFERENCES 主表名（主表字段名）ON UPDATE CASCADE ON DELETE CASCADE；

NO ACTION	当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除/更新。（与 RESTRICT 一致）
RESTRICT	当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除/更新。（与 NO ACTION 一致）
CASCADE	当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有，则也删除/更新外键在子表中的记录
SET NULL	当在父表中删除对应记录时，首先检查该记录是否有对应外键，如果有则设置子表中该外键值 null（这就要求该外键允许取null）。
SET DEFAULT	父表有变更时，子表将外键列设置成一个默认的值（Innodb不支持）

▼ 多表查询

▼ 多表关系

• 一对多(多对一)

一个部门对应多个员工，一个员工对应一个部门

实现：在多的一方建立外键，指向一的一方的主键

• 多对多

关系：一个学生可以选修多门课程，一门课程也可以供多个学生选择

实现：建立第三张中间表，中间表至少包含两个外键，分别关联两方主键

▼ 一 对一

一对一关系，多用于单表拆分，将一张表的基础字段放在一张表中，其他详情字段放在另一张表中，以提升操作效率

实现：在任意一方加入外键，关联另外一方的主键，并且设置外键为唯一的（*UNIQUE*）

id	name	age	gender	phone
1	黄渤	45	1	18800001111
2	冰冰	35	2	18800002222
3	码云	55	1	18800008888
4	李彦宏	50	1	18800009999

用户基本信息表(tb_user)

id	degree	major	primaryschool	middleschool	university	userid
1	本科	舞蹈	静安区第一小学	静安区第一中学	北京舞蹈学院	1
2	硕士	表演	朝阳区第一小学	朝阳区第一中学	北京电影学院	2
3	本科	英语	杭州市第一小学	杭州市第一中学	杭州师范大学	3
4	本科	应用数学	阳泉第一小学	阳泉区第一中学	清华大学	4

用户教育信息表(tb_user_edu)

▼ 多表查询

笛卡尔积：笛卡尔乘积是指在数学中，两个集合A集合和B集合的所有组合情况。（在多表查询时，需要消除无效的笛卡尔积）

▼ 连接查询

- 内连接

相当于查询A、B交集部分数据

```
`select * from emp, dept where emp.dept_id = dept.id;
```

- ▼ 外连接

- 左外连接

查询左表所有数据，以及两张表交集部分数据

SELECT 字段列表 FROM 表1 LEFT [OUTER] JOIN 表2 ON 条件..;

- 右外连接

查询右表所有数据，以及两张表交集部分数据

SELECT 字段列表 FROM 表1 RIGHT [OUTER] JOIN表? ON条件...;

- 自连接

当前表与自身的连接查询，**自连接必须使用表别名**

SELECT 字段列表 FROM 表A 别名A JOIN 表A 别名B ON 条件..;

- 联合查询

把多次查询的结果合并起来，形成一个新的查询结果集

SELECT 字段列表 FROM 表A..... UNION [ALL] SELECT 字段列表 FROM 表B:

//union:直接合并 union all:带去重

- ▼ 子查询

SQL语句中嵌套SELECT语句，称为嵌套查询，又称子查询

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

- 标量子查询

e.g. 查询销售部所有员工信息

第一步 查询"销售部\"部门ID

第二步 根据销售部部门ID，查询员工信息

```
select * from emp where dept_id = (select id from dept where name = '销售部')
```

- ▼ 列子查询

▼ 常用的操作符

e.g. 查询“销售部”和“市场部”的所有员工信息

第一步 查询“销售部”和“市场部”的部门ID

第二步 根据部门ID，查询员工信息

```
select * from emp where dept_id in (select id from dept where name = '销售部' or name = '市场部')
```

e.g. 查询比 财务部 所有人工资都高的员工信息

第一步 查询所有 财务部 人员工资

第二步 比财务部 所有人工资都高的员工信息

```
select * from emp where salary > all (select salary from emp where dept_id =(select id from dept where name='财务部'));
```

e.g. 查询比 比研发部其中任意一人工资高的员工信息

```
select * from emp where salary > any (select salary from emp where dept_id = (select id from dept where name = '研发部'));
```

- IN

在指定的集合范围之内，多选一

- NOT IN

不在指定的集合范围之内

- ANY

子查询返回列表中，有任意一个满足即可

- SOME

与ANY等同，使用SOME的地方都可以使用ANY

- ALL

子查询返回列表的所有值都必须满足

▼ 行子查询

子查询返回的结果是一行（可以是多列），这种子查询称为行子查询

e.g 查询与“张无忌”的薪资及直属领导相同的员工信息；

```
select * from emp where (salary, managerid) = (Select salary, managerid from emp where name = '张无忌');
```

▼ 常用的操作符

- =

- <>

- IN

- NOT IN

▼ 表子查询

子查询返回的结果是多行多列，这种子查询称为表子查询

e.g 查询与“鹿杖客”，“宋运桥”的职位和薪资相同的员工信息

```
select * from emp where (job,salary) in (select job,salary from emp where name = '鹿杖客' or name = '宋运桥');
```

e.g 查询入职日期是“2006-01-01”之后的员工信息，及其部门信息

```
select e.*,d.* from (select * from emp where entrydate > '2006-01-01') e Left join dept d on e.dept_id = d.id;
```

- 常用的操作符：IN

▼ 事务

是一组操作的集合，它是一个不可分割的工作单位，事务会把所有的操作作为一个整体一起向系统提交或撤销操作请求，

即这些操作要么同时成功，要么同时失败。

(默认MySQL的事务是自动提交的，也就是说，当执行一条DML语句，MySQL会立即隐式的提交事务。)

- 查看/设置事务提交方式

```
SELECT @@autocommit;  
SET @@autocommit = 0;
```

- 提交事务

```
COMMIT;
```

- 回滚事务

```
ROLLBACK;
```

- 开启事务

```
START TRANSACTION = BEGIN;
```

▼ 事务四大特性 (ACID)

原子性 (Atomicity)：事务是不可分割的最小操作单元，要么全部成功，要么全部失败。

一致性 (Consistency)：事务完成时，必须使所有的数据都保持一致状态。

隔离性 (Isolation)：数据库系统提供的隔离机制，保证事务在不受外部并发操作影响的独立环境下运行。

持久性 (Durability)：事务一旦提交或回滚，它对数据库中的数据的改变就是永久的。

▼ 并发事务问题

脏读	一个事务读到另外一个事务还没有提交的数据
不可重复读	一个事务先后读取同一条记录，但两次读取的数据不同，称之为不可重复读
幻读	一个事务按照条件查询数据时,没有对应的数据行，但是在插入数据时，又发现这行数据已经存在，好像出现了"幻影"

▼ 事务隔离级别

▼ 图片

隔离级别	脏读	不可重复读	幻读
Read uncommitted	√	√	√
Read committed	×	√	√
Repeatable Read(默认)	×	×	√
Serializable	×	×	×

- 查看事务隔离级别

```
SELECT COTRANSACTION_ISOLATION;
```

- 设置事务隔离级别

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL ( READ  
UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE )
```