第一章　MySQL 数据库的数据管理

1.1　数据库基础

数据库概述

在企业级开发中，数据都需要保存在专业的软件中，这些软件就是数据库软件。数据库软件不需要我们去研发，由专门的数据库厂商提供。

常用的数据库软件有：

Oracle：Oracle 数据库。甲骨文公司，为专门数据库厂商，收购了 BEA、SUN、MySQL。此数据库为大型收费数据库，可用于任何系统，任何平台。

IBM：BD2 数据库。IBM 数据库产品，为大型收费数据库，与 WebSphere 服务器一起使用。

Mysql 数据库：早期由瑞典一个公司开发，后期被 SUN 公司收购，随着 SUN 公司被 Oracle 收购，Mysql 也成了 Oracle 公司的产品。

Microsoft：Sqlserver 数据库。微软公司数据库产品，中等规模收费数据库，操作系统要求 Windows 结合. NET 一起使用。

Sybase：Sybase 数据库，中等规模数据库。

MySQL：是最流行的开放源码 SQL 数据库管理系统，它是由 MySQL AB 公司开发、发布并支持的。MySQL AB 是由多名 MySQL 开发人员创办的一家商业公司。它是一家第二代开放源码公司，结合了开放源码价值取向、方法和成功的商业模型。

在 MySQL 的网站（http://www.mysql.com/）上，给出了关于 MySQL 的最新信息。MySQL 是一种数据库管理系统。数据库是数据的结构化集合。它可以是任何东西，从简单的购物清单到画展，或企业网络中的海量信息。要想将数据添加到数据库，或访问、处理计算机数据库中保存的数据，需要使用数据库管理系统，如 MySQL 服务器。计算机是处理大量数据的理想工具，因此，数据库管理系统在计算方面扮演着关键的中心角色，或是作为独立的实用工具，或是作为其他应用程序的组成部分。

（1）MySQL 是一种关联数据库管理系统。

关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大的仓库内。这样

就增加了速度并提高了灵活性。MySQL的SQL指的是"结构化查询语言"。SQL是用于访问数据库的最常用标准化语言,它是由ANSI/ISO SQL标准定义的。SQL标准自1986年以来不断演化发展,有数种版本。在本手册中,"SQL-92"指的是1992年发布的标准,"SQL:1999"指的是1999年发布的标准,"SQL:2003"指的是标准的当前版本。我们采用术语"SQL标准"标示SQL标准的当前版本。

(2)MySQL软件是一种开放源码软件。

"开放源码"意味着任何人都能使用和改变软件。任何人都能从Internet下载MySQL软件,而无须支付任何费用。如果愿意,你可以研究源码并进行恰当地更改,以满足你自己的需求。

MySQL数据库服务器具有快速、可靠和易于使用的特点,采用了GPL(GNU通用公共许可证,http://www.mysql.com/company/legal/licensing/)。

1.1.1 存储引擎

MySQL支持多种类型的数据库引擎,可分别根据各个引擎的功能和特性为不同的数据库处理任务提供各自不同的适应性和灵活性。在MySQL中,可以利用SHOW ENGINES语句来显示可用的数据库引擎和默认引擎。

MySQL提供了多个不同的存储引擎,包括处理事务安全表的引擎和处理非事务安全表的引擎。在MySQL中,不需要在整个服务器中使用同一种存储引擎,针对具体的要求,可以对每一个表使用不同的存储引擎。

MySQL 5.7支持的存储引擎有InnoDB、MyISAM、Memory、Merge、Archive、Federated、CSV、BLACKHOLE等。

1.存储引擎

实例1:创建成绩表。

(1)创建数据表:

①表名:score

②字段:

A. sno:外键,参照student表的sno字段。

B. cno:外键,参照course表的cno字段。

C. degree

③存储引擎:InnoDB。

④默认字符编码:utf8。

(2)任务要求:

①判断score表是否已经存在,如果存在则先将原来的表删除。

②创建名称为score的表。

③创建字段学生编号sno,指定为varchar类型长度20,非空约束,外键参照student表

的 sno 字段。

④创建字段课程编号 cno，字符类型，长度为 20，非空约束，外键参照 course 表的 cno 字段。

⑤创建字段 degree，decimal 类型。

⑥在表的字段创建结束后指定存储引擎为 InnoDB。

⑦最后指定默认字符编码为 utf8。

(3)SQL 语句：

```
DROP TABLE IF EXISTS score;

CREATE TABLE score
(
  sno VARCHAR(20) NOT NULL ,
  FOREIGN KEY(sno) REFERENCES student(sno),
  cno VARCHAR(20) NOT NULL,
  FOREIGN KEY(cno) REFERENCES course(cno),
  degreeDECIMAL

)
ENGINE=INNODB DEFAULT CHARSET=utf8;

SHOW TABLES;
```

实例 2：创建课程表。

(1)创建数据表：

①表名：course

②字段：

A. cno：主键

B. cname

C. tno：外键参照 teacher 表的 tno 字段。

③存储引擎：InnoDB。

④默认字符编码：utf8。

(2)任务要求：

①判断 course 表是否已经存在，如果存在则先将原来的表删除。

②创建名称为 course 的表。

③创建字段编号 cno，指定为 varchar 类型长度 20，并实现此字段为主键。

④创建字段名称 cname，字符类型，长度为 20，非空约束。

⑤创建字段教师编号 tno，字符类型，长度 20，非空约束，外键参照 teacher 表的 tno 字段。

⑥在表的字段创建结束后指定存储引擎为 InnoDB。

⑦最后指定默认字符编码为 utf8。

(3)SQL语句：

```
DROP TABLE IF EXISTS course;
CREATE TABLE course
(
  cno VARCHAR(20) NOT NULL PRIMARY KEY,
  cname VARCHAR(20) NOT NULL,
  tno VARCHAR(20) NOT NULL,
  FOREIGN KEY(tno) REFERENCES teacher(tno)

)
ENGINE=INNODB DEFAULT CHARSET=utf8;

SHOW TABLES;
```

实例3：创建教师表。

(1)创建数据表：

①表名：teacher

②字段：

A. tno：主键

B. tname

C. tsex

D. tbirthday

E. prof

F. depart

③存储引擎：InnoDB。

④默认字符编码：utf8。

(2)任务要求：

①判断teacher表是否已经存在，如果存在则先将原来的表删除。

②创建名称为teacher的表。

③创建字段编号tno，指定为varchar类型长度20，并实现此字段为主键。

④创建字段名称tname，字符类型，长度为20，非空约束。

⑤创建字段性别tsex，字符类型，长度20，非空约束。

⑥创建字段生日tbirthday，datetime类型。

⑦创建字段prof，字符类型，长度20。

⑧创建字段depart，字符类型，长度20，非空约束。

⑨在表的字段创建结束后指定存储引擎为InnoDB。

⑩最后指定默认字符编码为utf8。

(3)SQL 语句：

```
DROP TABLE IF EXISTS teacher;
CREATE TABLE teacher
(
  tno VARCHAR(20) NOT NULL PRIMARY KEY,
  tname VARCHAR(20) NOT NULL,
  tsex VARCHAR(20) NOT NULL,
  tbirthdayDATETIME,
  prof VARCHAR(20),
  depart VARCHAR(20) NOT NULL
)
ENGINE=INNODB DEFAULT CHARSET=utf8;

SHOW TABLES;
```

1.1.2　MySQL 字符集

字符集是一套符号和编码，校验规则(collation)是在字符集内用于比较字符的一套规则，即字符集的排序规则。MySQL 可以使用多种字符集和检验规则来组织字符。

MySQL 服务器可以支持多种字符集，在同一台服务器，同一个数据库，甚至同一个表的不同字段都可以指定使用不同的字符集，相比 Oracle 等其他数据库管理系统，在同一个数据库只能使用相同的字符集，MySQL 明显存在更大的灵活性。

每种字符集都可能有多种校对规则，并且都有一个默认的校对规则，每个校对规则只是针对某个字符集，和其他的字符集没有关系。

在 MySQL 中，字符集的概念和编码方案被看作是同义词，一个字符集是一个转换表和一个编码方案的组合。

Unicode(Universal Code)是一种在计算机上使用的字符编码。Unicode 是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。Unicode 存在不同的编码方案，包括 Utf-8，Utf-16 和 Utf-32。Utf 表示 Unicode Transformation Format。

查看 MySQL 服务器支持的字符集：

```
mysql> show character set;
mysql> select * from information_schema.character_sets;
mysql> select character_set_name, default_collate_name, description, maxlen from
information_schema.character_sets;
```

1. MySQL 字符集

实例 1：创建学生信息表。

(1)创建数据表：

①表名：student

②字段：

A. sno：主键

B. sname

C. ssex

D. sbirthday

E. class

③存储引擎：InnoDB。

④默认字符编码：utf8。

(2)任务要求：

①判断 student 表是否已经存在，如果存在则先将原来的表删除。

②创建名称为 student 的表。

③创建字段学生编号 sno，指定为 varchar 类型长度 20，并实现此字段为主键。

④创建字段学生名称 sname，字符类型，长度为 20，非空约束。

⑤创建字段性别 ssex，字符类型，长度 20，非空约束。

⑥创建字段生日 sbirthday，datetime 类型。

⑦创建字段班级 class，字符类型，长度 20。

⑧在表的字段创建结束后指定存储引擎为 InnoDB。

⑨最后指定默认字符编码为 utf8。

(3)SQL 语句：

```
USE test;

DROP TABLE IF EXISTS dept ;
CREATE TABLE dept(
  deptno INT PRIMARY KEY AUTO_INCREMENT, -- 部门编号
  dname VARCHAR(14) , -- 部门名字
  loc VARCHAR(13) -- 地址
  )ENGINE=INNODB DEFAULT CHARSET=utf8;

SHOW TABLES;
```

实例 2：创建用户考试信息表。

(1)创建数据表：

①表名：user_test

②字段：

A. id：主键，自增

B. user_id

C. test_date

D. course_id

E. math

F. grade_id

③存储引擎:InnoDB。

④默认字符编码:utf8。

⑤AUTO_INCREMENT 会在新记录插入表中时生成一个唯一的数字。希望在每次插入新记录时,自动地创建主键字段的值,可以在表中创建一个 auto-increment 字段。

(2)任务要求:

①判断 emp 表是否已经存在,如果存在则先将原来的表删除。

②创建名称为 emp 的表。

③创建字段员工编号 empno,指定为 int 类型,并实现此 id 字段为主键自动增长。

④创建字段员工姓名 ename,字符类型,长度为 10。

⑤创建字段岗位 job,字符类型,长度 9。

⑥创建字段直接领导编号 mgr,int 类型,通过此字段实现表的自身关联。

⑦创建字段入职日期 hiredate,date 类型。

⑧创建字段薪水 sal,double 类型。

⑨创建字段提成 comm,double 类型。

⑩创建字段部门编号 deptno,int 类型,非空约束,此字段表示员工所属部门编号,作为 dept 表 deptno 字段的外键。

⑪创建 deptno 外键,参照 dept 表的 deptno 主键字段。

⑫在表的字段创建结束后指定存储引擎为 InnoDB。

⑬最后指定默认字符编码为 utf8。

(3)SQL 语句:

```
USE test;

DROP TABLE IF EXISTS emp;
CREATE TABLE emp(
  empno INT PRIMARY KEY AUTO_INCREMENT,-- 员工编号
  ename VARCHAR(10), -- 员工姓名 -
  job VARCHAR(9), -- 岗位
  mgr INT, -- 直接领导编号
  hiredate DATE, -- 雇佣日期,入职日期
  sal DOUBLE,-- 薪水
  comm DOUBLE,-- 提成
  deptno INT NOT NULL -- 部门编号
)ENGINE=INNODB DEFAULT CHARSET=utf8;

SHOW TABLES;
```

实例 3:创建 test_message 库。

(1)创建数据库:

①库名:test_message

A. 如果数据库不存在则创建,存在则不创建。

B. 创建 test_message 数据库,并设定编码集为 utf8。

(2)任务要求：

①判断 test_message 数据库是否已经存在，如果不存在则创建数据库。

②创建名称为 test_message 的数据库，字符编码为 utf8。

③切换到刚刚创建好的 test_message 数据库上。

(3)SQL 语句：

```
CREATE DATABASE IF NOT EXISTS test_message CHARACTER SET utf8;
USE test_message;

SHOW DATABASES;
```

1.2 数据库和数据表管理

数据库(database)的核心就是存放数据，而在关系型数据中，存放数据的核心组件就是表。

表的定义：列名和对应的列应该采用的数据类型。

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] 'DB_NAME';
CHARACTER SET 'character set name'COLLATE 'collate name
```

使用什么字符集和排序规则一般写至配置文件中，不会轻易修改，如果涉及修改库的字符集，需要修改以前各个表和各表中的各个字段，比较繁琐。

目前一般使用的字符集为 utf8、utf8mb4。

1.2.1 创建表

1. 建表语句语法

实例 1：创建部门表。

(1)创建数据表：

①表名：dept

②字段：

A. deptno：主键，自增。

B. dname

C. loc

③存储引擎：InnoDB。

④默认字符编码：utf8。

⑤DEFAULT CHARSET=utf8 数据库默认编码为 utf-8。

⑥AUTO_INCREMENT=1 自增键的起始序号为 1。

(2)任务要求：

①判断 dept 表是否已经存在，如果存在则先将原来的表删除。

②创建名称为 dept 的表。

③创建字段部门编号 deptno，指定为 int 类型，并实现此 id 字段为主键自动增长。
④创建字段部门名称 dname，字符类型，长度为 14。
⑤创建字段地址 loc，字符类型，长度 13。
⑥在表的字段创建结束后指定存储引擎为 InnoDB。
⑦最后指定默认字符编码为 utf8。
(3)SQL 语句：

```
DROP TABLE IF EXISTS dept ;
CREATE TABLE dept(
  deptno INT PRIMARY KEY AUTO_INCREMENT, -- 部门编号
  dname VARCHAR(14) , -- 部门名字
  loc VARCHAR(13) -- 地址
  )ENGINE=INNODB DEFAULT CHARSET=utf8;

SHOW TABLES;
```

实例 2：创建考试信息表。
(1)创建数据表：
①表名：user_test
②字段。
A. id：主键，自增
B. user_id
C. test_date
D. course_id
E. math
F. grade_id
(2)任务要求：
①判断 user_test 表是否已经存在，如果存在则先将原来的表删除。
②创建名称为 user_test 的表。
③创建名称为 id 的字段，指定为 int 类型，长度为 5，并实现此 id 字段为主键自动增长。
④创建名称为 user_id 的字段，字符类型，长度为 8。
⑤创建 test_date 字段，类型为 date。
⑥创建 course_id 字段，类型为 int 长度为 5。
⑦创建 math 字段，类型为 float(5,2)。
⑧创建 grade_id 字段，类型为 int，长度为 5。
⑨在表的字段创建结束后指定存储引擎为 InnoDB。
⑩最后指定默认字符编码为 utf8。
(3)任务目标：
①产生一个名称为 test 的表。
②具备 id 和 name 两个字段。
③并且是 InnoDB 类型的表空间。

(4)SQL语句：

```
use test;

DROP TABLE IF EXISTS user_test;
CREATE TABLE user_test(
  id INT(5) PRIMARY KEY AUTO_INCREMENT,
  user_id VARCHAR(8),
  test_date DATE,
  course_id INT(5),
  math FLOAT(5,2),
  grade_id INT(5)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

select * from user_test;
```

实例3：创建用户信息表。

(1)创建数据表：

①表名：user_message

②字段：

A. id：主键，varchar(8)。

B. user_name

C. user_birthday

D. zone

③存储引擎：InnoDB。

④默认字符编码：utf8。

(2)任务要求：

①判断user_message表是否已经存在，如果存在则先将原来的表删除。

②创建名称为user_message的表。

③创建名称为id的字段，指定为字符类型，长度为8，并实现此id字段为主键。

④创建名称为user_name的字段，字符类型，长度为10。

⑤创建名称为user_birthday的字段，类型为date。

⑥创建名称为zone的字段，字符类型，长度为20。

⑦在表的字段创建结束后指定存储引擎为InnoDB。

⑧最后指定默认字符编码为utf8。

(3)任务目标：

①产生一个名称为test的表。

②具备id和name两个字段。

③并且是InnoDB类型的表空间。

(4)SQL 语句：

```
use test;
DROP TABLE IF EXISTS user_message;
CREATE TABLE user_message(
  id VARCHAR(8) PRIMARY KEY ,
  user_name VARCHAR(10),
  user_birthday DATE,
  zone VARCHAR(20)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

select * from user_message;
```

1.2.2　删除表

1. 删除表语法

在日常使用数据库的时候，经常会需要将单个或多个表删除掉，如何有效地删除表，是掌握 MySQL 必知必会的知识。

(1)任务要求：

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③选择将要删除的表，用 drop table 命令进行删除。

④最后再次使用 show tables 命令查看删除后的结果。

⑤结果是，当前数据库中已经删除的表查询不到，表已经不存在。

(2)SQL 语句：

```
use test;
drop table if exists test1;
show tables;
```

1.2.3　修改表

1. 添加新列

如果想在一个已经建好的表中添加一列，可以用诸如：

```
alter table TABLE_NAME add column NEW_COLUMN_NAME varchar(20) not null;
```

这条语句会向已有的表中加入新的一列，这一列在表的最后一列位置。如果希望添加在指定的一列，可以用：

```
alter table TABLE_NAME add column NEW_COLUMN_NAME varchar(20) not null after COLUMN_NAME;
```

注意：上面这个命令的意思是说添加新列到某一列后面。如果想添加到第一列的话，可以用：

```
alter table TABLE_NAME add column NEW_COLUMN_NAME varchar(20) not null first;
```

(1)任务要求:

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③用 desc TABLE_NAME;来查看原来的表结构。

④选择将要添加新列的表,用 alter table TABLE_NAME 命令指定要调整的表名。

⑤在后面用 add column NEW_COLUMN_NAME varchar(20) not null; 命令来指定增加的新列名,类型(长度),是否为空。

例如:add 新列名 类型(长度) not null;

⑥用 desc TABLE_NAME;来查看新列是否加入到了当前表结构中。

⑦结果是,当前的表结构发生了改变,新的列加入到了当前表结构中。

(2)SQL 语句:

```
use test;
show tables;

desc test4;
ALTER TABLE test4 ADD deptnum int(20) not null;
desc test4;

desc test3;
ALTER TABLE test3 ADD slocation varchar(20) not null;
desc test3;

desc test2;
ALTER TABLE test2 ADD smatu VARCHAR(20) not null;
desc test2;
```

2. 修改已存在的列

实例 1

使用 MySQL ALTER TABLE 语句来设置列的属性,subject 列不再是原来的 varchar 类型,而是 char 类型,长度也发生了变化。那么可以使用 ALTER TABLE 语句将 subject 列的属性进行修改。

(1)任务要求:

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③用 desc TABLE_NAME;来查看原来的表结构。

④选择将要修改的列,用 ALTER TABLE test3 CHANGE COLUMN 命令指定要调整的表名。

A. 例如 CHANGE COLUMN subject subject char(24) NOT NULL;

a)subject 表示原列名;

b)subject char(24) NOT NULL;

B. 表示修改后列的属性，将 subject 由原来的 varchar 修改为 char，长度修改为 24。

⑤用 desc TABLE_NAME；来查看已经修改的列的类型和长度均发生了变化。

(2)SQL 语句：

```
use test;
show tables;
desc test3;
ALTER TABLE test3
CHANGE COLUMN subject subject char(24) NOT NULL;
desc test3;
```

实例 2

使用 MySQL ALTER TABLE 语句来设置列的属性，假设您希望在任务表中插入新行时，start_date 列不再是日期类型，而是字符串类型，那么可以使用 ALTER TABLE 语句将 start_date 列的属性进行修改。

(1)任务要求：

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③用 desc TABLE_NAME；来查看原来的表结构。

④选择将要修改的列，用 ALTER TABLE test1 CHANGE COLUMN 命令指定要调整的表名。

例如 ALTER TABLE test1 CHANGE COLUMN task_id task_id INT(11) NOT NULL AUTO_INCREMENT；

a)task_id 表示原列名；

b)task_id int(11) not null auto_increment 表示修改后列的属性。

⑤用 desc TABLE_NAME；来查看已经修改的列在 extra 项中出现 auto_increment 自动增长的属性。

⑥结果是，当前的表结构发生了改变，task_id 列在 extra 项中，多出 auto_increment 属性，显示在当前表结构中。

(2)SQL 语句：

```
use test;
show tables;
desc test2;
ALTER TABLE test2
CHANGE COLUMN start_date start_date varchar(20) NOT NULL;
desc test2;
```

实例 3

使用 MySQL ALTER TABLE 语句来设置列的自动递增属性，假设希望在任务表中插入新行时，task_id 列的值会自动增加 1，那么可以使用 ALTER TABLE 语句将 task_id 列的属性设置为 AUTO_INCREMENT。

(1)SQL语句：

```
use test;
show tables;
desc test1;
ALTER TABLE test1
CHANGE COLUMN task_id task_idINT(11) NOT NULL AUTO_INCREMENT;
desc test1;
```

3. 删除列

实例1

想要更改表的内容，有的列不想要了，就需要使用到删除列命令。用drop column可以将某列从表中删除。

(1)任务要求：

①用use test命令来使用test数据库。

②用show tables命令来查看当前数据库中有哪些表。

③用desc TABLE_NAME;来查看原来的表结构。

④用ALTER TABLE 表名 DROP COLUMN 列名。

⑤将原来的表名为test4的表中的subject列删除掉。

⑥用show tables来查看更改后的表结构，看到被删除的列已经没有了。

(2)SQL语句：

```
use test;
show tables;
desc test4;
ALTER TABLE test4 DROP COLUMN subject;
desc test4;
show tables;
```

实例2

练习掌握在已经存在的数据表中删除某列的语法，可以灵活地对原有数据表进行结构修改与调整。

(1)任务要求：

①用use test命令来使用test数据库。

②用show tables命令来查看当前数据库中有哪些表。

③用desc TABLE_NAME;来查看原来的表结构。

④用AALTER TABLE 表名 DROP COLUMN 列名。

⑤将原来的表名为test3的表中的start_date列删除掉。

⑥用show tables来查看更改后的表结构，看到被删除的列已经没有了。

(2)SQL语句：

```
use test;
show tables;
desc test3;
ALTER TABLE test3 DROP COLUMN start_date;
desc test3;
show tables;
```

实例 3

(1)任务要求：

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③用 desc TABLE_NAME;来查看原来的表结构。

④用 AALTER TABLE 表名 DROP COLUMN 列名。

⑤将原来的表名为 test2 的表中的 end_date 列删除掉。

⑥用 show tables 来查看更改后的表结构，看到被删除的列已经没有了。

(2)SQL 语句：

```
use test;
show tables;
desc test2;
ALTER TABLE test2 DROP COLUMN end_date;
desc test2;
show tables;
```

1.3　数据操作

1.3.1　数据操作语言

数据操纵语言(Data Manipulation Language, DML)是 SQL 语言中，负责对数据库对象运行数据访问工作的指令集，以 INSERT、UPDATE、DELETE 三种指令为核心，分别代表插入、更新与删除，是开发以数据为中心的应用程序必定会使用到的指令，因此有很多开发人员都把加上 SQL 的 SELECT 语句的四大指令以“CRUD”来定义。

1. 数据操作语言

实例 1

向学生表中增加一条学生记录，学生姓名为 z3，学生所属系为 net，学生性别为 m，学生出生日期 1988-05-22。记录增加成功后可以通过 select 命令查看到这条数据记录。

(1)任务要求：

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③网络系(插入值是“net”)新进一名学生，名叫“z3”，性别值为“m”，出生于“1988-05-22”，现需要将此学生的信息加入学生表中。

A. 例如：用 insert into 语句执行插入数据，后面接上表名 student，再后面接上 values 作为值关键字，后面跟上一对大括号。

B. 括号中按顺序填入值，如下：('学号','姓名','姓名','院系')；分号结束。

C. 命令如下：insert into 表名 values('学号','姓名','姓名','院系')。

④用 select 命令可以查看插入的数据结果。

⑤注意:学号不能重复,因为具备主键约束,当再次插入数据时,学号需要保持唯一性。

(2)SQL 语句:

```
use test;
show tables;
INSERT INTO student VALUES('200515026','z3','m','1988-05-27','net');
select * from student;
```

实例 2

向表中增加数据是操作数据的基本操作之一,使用 insert 语句可以一次性地向表批量插入多条记录。

(1)任务要求:

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③采用一次性向表中插入多条数据,要插入的数据信息如表 1-1 所示。

表 1-1 插入数据表

学号	姓名	性别	出生日期	院系
200515027	L4	m	1987/6/27	JAVA
200515028	z6	f	1988/3/20	JAVA
200515029	q7	f	1988/6/18	JAVA
200515030	y8	m	1989/8/20	JAVA

A. 语法:insert into 表名 values (值 1,值 2,值 3……),(值 1,值 2,值 3……),(值 1,值 2,值 3……),……;

B. 每一个括号中:

a)值 1 是学号字段对应的值;

b)值 2 是姓名字段对应的值;

c)值 3 是性别字段对应的值;

d)值 4 是出生日期对应的值;

e)值 5 是院系对应的值,如还有字段依次类推。

C. 有多少行数据,对应的就会有多少个括号与之对应。

④用 select 命令可以查看插入的数据结果。

⑤注意:学号不能重复,因为具备主键约束,当再次插入数据时,学号需要保持唯一性。

(2)SQL 语句:

```
use test;
show TABLEs;
insert into student values
('200515027', 'L4', 'm', '1987-06-27','JAVA'),
('200515028', 'z6', 'f', '1988-03-20','JAVA'),
('200515029', 'q7', 'f', '1988-06-18','JAVA'),
('200515030', 'y8', 'm', '1989-08-20','JAVA');
select * from student;
```

实例 3

向 student 学生表中添加指定字段的值，而不是所有字段的值，因为有些字段可以有默认值或者为空值，其中如果有非空字段，要求非空字段必须给值。学生表中只需要填写学生编号、姓名、院系的值。

(1)任务要求：

①用 use test 命令来使用 test 数据库。

②用 show tables 命令来查看当前数据库中有哪些表。

③向 student 学生表中添加指定字段的值，而不是所有字段的值，因为有些字段可以有默认值或者为空值，其中如果有非空字段，要求非空字段必须给值。

A. 学生表中只需要填写学生编号、姓名、院系的值。

B. 语法：insert into 表名（字段名 1，字段名 2，字段名 3……）values(值 1，值 2，值 3……)。

C. 第一个括号内全部是指定字段的名称，逗号分隔。

D. 第二个括号内全部是指定字段对应的值，逗号分隔。

④用 select 命令可以查看插入的数据结果。

⑤注意：学号不能重复，因为具备主键约束，当再次插入数据时，学号需要保持唯一性。

(2)SQL 语句：

```
use test;
show tables;
insert into student (sno,sname,sdept)values ('200515032','ww','JAVA');
select * from student;
```

1.3.2　插入数据

1. INSERT 语法结构

实例 1

向 student 表中插入 6 条数据，向 teacher 表中插入 4 条数据，记录增加成功后可以通过 select 命令查看数据记录。

(1)任务要求：

①使用 insert 语句向 student 表中插入 6 条数据。

②使用 insert 语句向 teacher 表中插入 4 条数据。

③使用 insert 语句向 course 表中插入 4 条数据。

④使用 insert 语句向 score 表中插入 12 条数据。

⑤数据插入成功后，使用 select 语句分别查询 4 张表中插入的记录。

⑥使用 delete 删除 2 张表数据，防止下次运行主键重复。

(2)SQL语句：

```
INSERT INTO student VALUES('108','tom','male','1977-09-01','95033');
INSERT INTO student VALUES('105','jerry','male','1975-10-02','95031');
INSERT INTO student VALUES('107','mike','female','1976-01-23','95033');
INSERT INTO student VALUES('101','smith','male','1976-02-20','95033');
INSERT INTO student VALUES('109','rose','female','1975-02-10','95031');
INSERT INTO student VALUES('103','scott','male','1974-06-03','95031');

INSERT INTO teacher VALUES('804','allen','male','1958-12-02','professor','Computer');
INSERT INTO teacher VALUES('856','jones','male','1969-03-12','lecturer','Electronic');
INSERT INTO teacher VALUES('825','blake','female','1972-05-05','tutor','Computer');
INSERT INTO teacher VALUES('831','ward','female','1977-08-14','tutor','Electronic');

SELECT * FROM student;
SELECT * FROM teacher;

delete from student;
delete from teacher;
```

实例2

向dept表中插入4条数据，向emp表中插入14条数据。记录增加成功后可以通过select命令查看数据记录。

INSERT语句是最常见的SQL语句之一，但是MySQL中INSERT语句的用法和标准用法不尽相同。

第一种方法将列名和列值分开了，在使用时，列名必须和列值的数一致；第二种方法允许列名和列值成对出现和使用。

如果字段上使用了自增值，字段列表和子列表中可以省略此字段。

(1)任务要求：

①使用insert语句向dept表中插入4条数据。

②使用insert语句向emp表中插入14条数据。

③数据插入成功后，使用select语句分别查询两张表中插入的记录。

(2)SQL语句：

```
use test;
insert into 'dept'('deptno','dname','loc') values (10,'ACCOUNTING','NEW YORK'),(20,'RESEARCH','
DALLAS'),(30,'SALES','CHICAGO'),(40,'OPERATIONS','BOSTON');

insert into 'emp'('empno','ename','job','mgr','hiredate','sal','comm','deptno') values (7369,'SMITH','
CLERK',7902,'1980-12-17',800.00,NULL,20),(7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600.00,300.
00,30),(7521,'WARD','SALESMAN',7698,'1981-02-22',1250.00,500.00,30),(7566,'JONES','MANAGER',7839,'
1981-04-02',2975.00,NULL,20),(7654,'MARTIN','SALESMAN',7698,'1981-09-28',1250.00,1400.00,30),
(7698,'BLAKE','MANAGER',7839,'1981-05-01',2850.00,NULL,30),(7782,'CLARK','MANAGER',7839,'1981-06-09
',2450.00,NULL,10),(7788,'SCOTT','ANALYST',7566,'1987-04-19',3000.00,NULL,20),(7839,'KING','
PRESIDENT',NULL,'1981-11-17',5000.00,NULL,10),(7844,'TURNER','SALESMAN',7698,'1981-09-08',1500.
00,0.00,30),(7876,'ADAMS','CLERK',7788,'1987-05-23',1100.00,NULL,20),(7900,'JAMES','CLERK',7698,'
1981-12-03',950.00,NULL,30),(7902,'FORD','ANALYST',7566,'1981-12-03',3000.00,NULL,20),(7934,'
MILLER','CLERK',7782,'1982-01-23',1300.00,NULL,10);

select * from dept;
select * from emp;
```

实例 3

向用户考试信息表 user_test 中插入多条数据，记录增加成功后可以通过 select 命令查看数据记录。

(1)任务要求：向用户考试信息表 user_test 中插入多条数据。

①使用一条 insert 语句向用户考试信息表 user_test 中插入多条数据。

②表中每个字段都要有对应类型的值。

③其中主键 id 为字符类型，需要手动分配值。

(2)SQL 语句：

```
use test;
show tables;
INSERT INTO user_test( user_id, test_date, course_id, math ) VALUES( 'U001', '2014-01-01', '1', '80.00'),( 'U001', '2014-03-01', '2', '90'),( 'U001', '2014-04-04', '3', '85'),( 'U002', '2014-01-01', '1', '67'),( 'U002', '2014-04-04', '3', '90'),( 'U002', '2014-05-04', '2', '80'),( 'U003', '2014-06-04', '1', '80'),( 'U001', '2014-01-01', '2', '90'),( 'U001', '2014-04-04', '3', '96');
select * from user_test;
```

1.3.3　修改数据

update 语句可以完成对表中数据的修改。

语法：update 表名 set 字段名 1＝值 1，字段名 2＝值 2where 条件；

其中 set 子句指出要修改的列和它们给定的值，where 子句是可选的，如果给出条件，将按照指定记录更新数据，否则，所有记录都会被更新。

1. UPDATE 语法结构

实例 1

将 emp 表职位 job 是“staff”的员工奖金 comm 改为 100，薪资 sal 增加 500 元。

(1)任务要求：

①update 表名 set 字段名 1＝值 1，字段名 2＝值 2，字段名 3＝值 3 where 条件；

②update 表示修改数据。

③set 字段名 1＝值 1，字段名 2＝值 2，字段名 3＝值 3 将某几个字段的值修改。

④where 条件表示连接限定条件。有条件要求就写，没有条件要求就不写。

(2)SQL 语句：

```
use test;
update emp set comm = 100,sal=sal+100 where job='staff';
```

实例 2

将 emp 表中奖金 comm 改为 1000。

(1)任务要求：

①update 表名 set 字段名 1＝值 1，字段名 2＝值 2，字段名 3＝值 3 where 条件。

②update 表示修改数据。

③set 字段名 1＝值 1，字段名 2＝值 2，字段名 3＝值 3 将某几个字段的值修改。

④where 条件表示连接限定条件。有条件要求就写,没有条件要求就不写。

(2)SQL 语句:

```
use test;
update emp set comm = 1000
```

实例 3

修改 user_message 表中"U001","U002"用户的 grade_id 字段值为 2。

(1)任务要求:

①使用 update 语句修改 user_message 表 grade_id 字段的值为 2。

②使用 where 条件及 in 运算符,只修改 id 为"U001","U002"的学生。

(2)SQL 语句:

```
use test;
show tables;
UPDATE user_message SET grade_id=2 WHERE id IN ('U001','U002');
select * from user_message;
```

2. 修改部分行记录

实例 1

将 student 学生表中,学生编号 sno 为"200515026",姓名 sname 为"w5"的学生所属院系"net"改为"java",修改完成后使用 select 语句查看数据是否被修改。

(1)任务要求:

①用 use test 命令来使用 test 数据库。

②用 show tables 查看当前库所有数据表。

③将 student 学生表中,学生编号 sno 为"200515026"、姓名 sname 为"w5"的学生所属院系"net"改为"java"。

语法:update 表名 set 字段名= 值 where 条件字段=值 and 条件字段=值;

④用 select 命令查看已经修改的数据结果。

⑤注意:尽量要带条件修改数据,否则会出现整张表的指定字段数据都被修改的情况。

(2)SQL 语句:

```
use test;
show tables;
update student set sdept='java' where sno = '200515026' and sname = 'w5';
select * from student;
```

实例 2

将 student 学生表中学生编号 sno 为"200515026"的学生姓名 sname 改为"w5",修改完成后使用 select 语句查看数据是否被修改。

(1)任务要求:

①用 use test 命令来使用 test 数据库。

②用 show tables 查看当前库所有数据表。

③将 student 学生表中学生编号 sno 为“200515026”的学生姓名 sname 改为“w5”。

语法:update 表名 set 字段名= 值 where 条件字段=值;

④用 select 命令查看已经修改的数据结果。

⑤注意:尽量要带条件修改数据,否则会出现整张表的指定字段数据都被修改的情况。

(2)SQL 语句:

```
use test;
show tables;
update student set sname = 'w5' where sno = '200515026';
select * from student;
```

实例 3

将 student 学生表中,学生编号 sno 为“200515026”的学生性别改为“f”,修改结束后使用 select 语句查看数据是否被修改。

(1)任务要求:

①用 use test 命令来使用 test 数据库。

②用 show tables 查看当前库所有数据表。

③将 student 学生表中,学生编号 sno 为“200515026”的学生性别改为“f”。

语法:update 表名 set 字段名= 值 where 条件字段=值;

④用 select 命令查看已经修改的数据结果。

⑤注意:尽量要带条件修改数据,否则会出现整张表的指定字段数据都被修改的情况。

(2)SQL 语句:

```
use test;
show tables;
update student set ssex='f' where sno = '200515026';
select * from student;
```

1.3.4　删除数据

1. 删除选中记录

在数据库中,有些数据已经失去意义或者错误时,就需要将它们删除,此时可以使用 delete 语句删除。

语法:delete from 表名 where 条件;

在语句的执行过程中,如果没有指定 where 条件,将删除所有的记录;如果指定了 where 条件,将按照指定条件删除行数据。

实例 1

删除 user_message 表中 id 为“U004”和“U005”的用户记录。

(1)任务要求:

①使用 delete 语句删除 user_message 表中记录。

②通过 where 条件删除 id 为“U004”和“U005”的学生记录。

③使用 or 运算符连接两个条件。

(2)SQL语句：

```
use test;
DELETE FROM user_message WHERE id='U004' OR id='U005';

select * from user_message;
```

实例2

删除emp表中员工eid为3的员工信息。

(1)任务要求：

①delete from 表名 where 条件。

②delete 表示删除数据。

③from 表名 表示从哪张表删除。

④where 条件表示连接限定条件。有条件要求就写，没有条件要求就不写。

(2)SQL语句：

```
use test;
delete from emp where eid = 3;
```

实例3

删除emp表中员工薪资sal小于1000的员工信息。

(1)任务要求：

```
use test;
delete from emp where sal<1000;
```

2. 删除全部记录

TRUNCATE和DELETE都可以清空表中数据。它们的区别是TRUNCATE是DDL，只能删除表中所有记录，释放存储空间，使用ROLLBACK不可以回滚。

参考语法：

(1)truncate table 表名。

(2)truncate table 表示删除数据。

(3)表名：表示指定的表。

实例1

删除emp表中所有员工信息。

(1)SQL语句：

```
use test;
delete from emp ;
```

实例2

删除部门表dept中所有员工信息。

(1)SQL语句：

```
use test;
delete from dept ;
```

实例 3

删除部门表 dept 中所有员工信息(使用 TRUNCATE)。

(1)SQL 语句:

```
use test;
TRUNCATE TABLE  dept ;
```

3. 事务概念及特征

在 MySQL 中只有使用了 Innodb 数据库引擎的数据库或表才支持事务。事务处理可以用来维护数据库的完整性,保证成批的 SQL 语句要么全部执行,要么全部不执行。事务用来管理 insert,update,delete 语句。

一般来说,事务是必须满足 4 个条件(ACID):原子性(Atomicity,或称不可分割性)、一致性(Consistency)、隔离性(Isolation,又称独立性)、持久性(Durability)。

实例 1

删除 user_message,user_test 表中所有记录,为删除语句添加事务,删除成功后回滚事务,并查询两张表中记录。

(1)任务要求:

①开启事务,MySQL 默认自动提交事务,需要通过 begin 开启事务。

②删除 user_message 表中所有记录。

③删除 user_test 表中所有记录。

④回滚事务。

⑤查询回滚后两张表的记录是否存在。

(2)SQL 语句:

```
use test;
BEGIN;
DELETE FROM user_message;
DELETE FROM user_test;
ROLLBACK;
SELECT * FROM user_message;
SELECT * FROM user_test;
```

实例 2

(1)任务要求:

①开启事务,MySQL 默认自动提交事务,需要通过 start TRANSACTION;开启事务。

②插入一条员工信息。

③再插入一条员工信息。

④回滚事务。

⑤查询回滚后插入的数据是否存在。

(2)SQL语句：

```
use test;
start TRANSACTION;
INSERT INTO emp VALUES ( '9','zzz', 'BM', null, '1980-02-01', '6000', null, '1');;
INSERT INTO emp VALUES ( '10','mmm', 'BM', null, '1980-02-01', '5000', null, '1');
ROLLBACK;
SELECT * FROM emp;
```

实例3

(1)任务要求：

①开启事务，MySQL默认自动提交事务，需要通过start TRANSACTION;开启事务。

②删除emp员工表中eid大于3的员工信息。

③插入一条员工信息。

④回滚事务。

⑤查询回滚后表中eid大于3的员工信息是否存在，插入的数据是否存在。

(2)SQL语句：

```
use test;
start TRANSACTION;
DELETE FROM emp where eid > 3;
INSERT INTO 'emp' VALUES ( '9','zzz', 'BM', null, '1980-02-01', '6000', null, '1');
ROLLBACK;
SELECT * FROM emp;
```

1.4 数据查询

1.4.1 基本SELECT语句

1.基本SELECT语句作用

数据查询语言(DQL:Data Query Language)中，SELECT语句是最常用的语句，它的使用方式有些复杂，但是功能很强大，用于从表中检索数据。在查询表中指定列数据时，多列之间，列名用英文逗号分隔。

实例1

在emp员工表中查询员工编号eid、员工姓名ename、员工薪资的信息sal。

(1)语法：select 字段名1，字段名2……from 表名；

①select表示开始检索某些列数据。

②字段名1，字段名2……表示要检索的字段的名称。

③from 表名表示从哪张表检索数据。

④分号表示该语句结束。

(2)SQL 语句：

```
use test;
select eid, ename,sal from emp;
```

实例 2

在 emp 员工表中查询部门编号为 1 的员工姓名和编号。

(1)语法：select 字段名 1，字段名 2……from 表名 where 条件；

①select 表示开始检索某些列数据。

②字段名 1，字段名 2……表示要检索的字段的名称。

③from 表名表示从哪张表检索数据。

④where 表示限定条件，条件的内容可以是：字段名＞值，字段名＜值，字段名＝值。

⑤分号表示该语句结束。

(2)SQL 语句：

```
use test;
select ename, eid from emp where dept_id = 1;
```

实例 3

在 dept 部门表中查询部门的名称 dname 和工作地点 loc。

(1)语法：select 字段名 1，字段名 2…… from 表名；

①select 表示开始检索某些列数据。

②字段名 1，字段名 2……表示要检索的字段的名称。

③from 表名表示从哪张表检索数据。

(2)SQL 语句：

```
use test;
select dname, dloc from dept;
```

1.4.2　选择列

1. 选择所有列

实例 1

查询 dept 员工表中的所有列信息。

(1)语法：select * from 表名；

①使用 select 关键字进行查询。

②使用 * 指定表中所有列数据。

③使用 from 指定表名。

④使用 dept 作为表名。

(2)SQL 语句：

```
use test;
select * from dept;
```

实例2

查询dept部门表中部门编号id大于3的部门所有信息。

(1)语法:select * from 表名 where 条件;

①select表示开始检索某些列数据。

②*表示所有列信息。

③from 表名表示从哪张表检索数据。

④where后跟限定条件,条件内容可以是:字段名 = 值、字段名>值、字段名<值等。

(2)SQL语句:

```
use test;
select * from dept where id > 3;
```

实例3

查询emp员工表中薪资sal小于等于3000的员工信息。

SQL语句:

```
use test;
select * from emp where sal <= 3000;
```

2. 选择特定列

实例1

在dept部门表中查询部门的名称。

(1)语法:select 字段名1,字段名2,字段名3…… from 表名;

①select表示开始检索某些列数据。

②字段名1,字段名2,字段名3……表示要检索的字段的名称。

③from 表名表示从哪张表检索数据。

(2)SQL语句:

```
use test;
select dname from dept;
```

实例2

在dept部门表中查询部门的名称dname和工作地点loc。

SQL语句:

```
use test;
select dname, dloc from dept;
```

实例3

在emp员工表中查询部门编号为1的员工姓名和编号。

(1)语法:select 字段名1,字段名2……from 表名 where 条件;

①select表示开始检索某些列数据。

②字段名1,字段名2……表示要检索的字段的名称。

③from 表名,表示从哪张表检索数据。

④where表示限定条件,条件的内容可以是:字段名>值,字段名<值,字段名=值。

⑤分号表示该语句结束。

(2)SQL 语句：

```
use test;
select ename, eid from emp where dept_id = 1;
```

1.4.3　算术运算符

1. 算术运算符

数据库中的表结构确定后，表中的数据代表的意义就已经确定了。通过 MySQL 运算符进行运算，就可以获取结构以外的另一种数据。

算术运算符是 MySQL 中最常用的一类运算符，其中支持的算术运算符包括加、减、乘、除、求余数。

实例 1

查看 emp 员工表中，员工的薪资和薪资的百分之五的奖励。

(1)语法：select 字段 1，字段 2，字段 3 * 值 from 表名；

①select 表示要检索数据了。

②字段 1，字段 2，字段 3 * 值表示要检索的字段名称，其中“字段 3 * 值”表示字段 3 的值乘以一个数。

③from 表名；表示从哪个表中检索数据。

(2)SQL 语句：

```
use test;
Select sal , sal * 0.05 From emp ;
```

实例 2

查看 emp 员工表中员工的薪资和薪资的百分之五的奖励以及两者的和。

(1)语法：select 字段 1＋值，字段 2－值，字段 3 * 值 from 表名；

①select 表示要检索数据了。

②字段 1＋值，字段 2－值，字段 3 * 值，字段 1、字段 2、字段 3 表示要检索的字段名称。

③其中字段 1＋值表示字段 1 的值加上一个数，字段 2－值表示字段 2 减去一个数，字段 3 * 值表示字段 3 的值乘以一个数。

④from 表名；表示从哪个表中检索数据。

⑤指定表名。

(2)SQL 语句：

```
use test;
Select sal , sal * 0.05,sal+(sal * 0.05) From emp ;
```

实例 3

查看 emp 表中员工一年的薪资是多少(不含奖励)，一年 12 个月。

提示：将薪资乘以 12 就是一年薪资。

SQL语句：

```
use test;
select sal * 12 from emp;
```

1.4.4 空值NULL

1. 空值NULL

空值也是数据的一种，空值是指一种无效的、未赋值、未知的或不可用的值，空值不同于零或者空格。

实例1

查看emp表中职员姓名ename，职员的工作job，职员的薪资sal，职员的奖金comm，观察comm一列值。

(1)语法：select 字段名1，字段名2 from 表名；

①select表示要检索数据了。

②字段名1，字段名2表示要检索的字段名。

③from 表名；表示从哪个表中检索数据。

(2)SQL语句：

```
use test;
SELECT ename, job, sal, comm  FROM   emp;
```

实例2

查看emp表中职员姓名ename，薪资sal，奖金comm，职员的年薪(提示：年薪=12 * 薪资+comm)，观察年薪一列值的结果。

(1)语法：select 字段名1，字段名2 from 表名；

①select表示要检索数据了。

②字段名1，字段名2表示要检索的字段名，如果有运算符，直接拼接即可，比如：字段名1 * 10。

③from 表名；表示从哪个表中检索数据。

(2)SQL语句：

```
use test;
select ename, sal, comm, 12 * sal+comm from emp;
```

实例3

查看emp表中薪资大于3000的职员姓名ename，薪资sal，奖金comm。

(1)语法：select 字段名1，字段名2 from 表名 where 条件；

①select表示要检索数据了。

②字段名1，字段名2表示要检索的字段名。

③from 表名表示从哪个表中检索数据。

④where后加限定条件，限定条件通常是字段名称>值，字段名<值，字段名=值等。

(2)SQL 语句：

```
use test;
SELECT ename, sal, comm  FROM   emp where sal > 3000;
```

1.4.5　列别名

1. 列别名

在查询时，可以为表和字段取一个别名，这个别名可以代替其指定的表和字段。为字段和表取别名，可以使查询更加方便。

使用别名还可以使查询结果以更加合理的方式显示。

(1)给表或者字段起别名有两种方式：

①带有 as 关键字，select 列名 1 as 列别名，列名 2 as 列别名，列名 3 as 列别名 …… from 表名 as 表别名。

②不带有 as 关键字的，select 列名 1 列别名，列名 2 列别名，列名 3 列别名 …… from 表名 表别名。

(2)语法：

①select 表示开始查询数据。

②列名 1 as 列别名，其中列别名用来重新命名列名的显示标题，比如：ename as '员工姓名'。

或列名 1　列别名，其中列别名用来重新命名列名的显示标题，比如：ename '员工姓名'。

③from 表名；表示要检索数据的表。

④表名 as 表别名，其中表别名用来重定义名表的名字。

或表名　表别名；其中表别名用来重新定义表的名字。

⑤where 表示限定条件。

⑥限定的条件中如果涉及有别名的字段，条件中要使用别名；限定条件中如果涉及有别名的表，条件中使用表的别名。

实例 1

使用列别名的方式(带关键字 as)查询 emp 员工表中员工姓名 ename，员工编号 eid，员工薪资 sal。查询的 ename、eid、sal 三列信息的显示标题分别是“员工姓名”，“员工编号”，“员工薪资”。emp 表的别名定义为 e。

SQL 语句：

```
use test;
select ename as '员工姓名',  eid  as '员工编号', sal as '员工薪资' from emp as e;
```

实例 2

使用列别名的方式(不带关键字 as)查询 emp 员工表。

SQL 语句：

```
use test;
select ename '员工姓名',  eid '员工编号', sal '员工薪资' from emp e;
```

实例3

使用别名的方式(带as关键字)查询emp员工表中职位job为BM的员工的姓名ename,员工的工作job,员工的薪资sal。查询ename、job、sal三列信息显示的标题分别是"姓名","职位","薪资"。emp表的别名为e。

SQL语句:

```
use test;
select ename as '员工姓名',job as '员工职位' ,sal as '员工薪资'
from emp as e where e.job = 'BM';
```

1.4.6 消除重复行

1. 使用DISTINCT关键字

select语句可以实现查询功能,当查询的数据出现重复的时候,可以使用DISTINCT关键字去除重复记录。或者在单表中,可能会包含重复值。在查询此表时也可以使用DISTINCT关键字去除重复的值。

注意:distinct只能作用于select后的第一个字段名前。

(1)语法:select distinct 字段名1,字段名2……from 表名;

①select 表示开始查询数据。

②distinct 表示去除重复字段。

③字段名1,字段名2……表示要查询的字段。

④from 表名; 表示查询哪张表。

实例1

查询dept部门表中部门名称dname,要求dname不能重复。

SQL语句:

```
use test;
select distinct dname from dept;
```

实例2

查询dept部门表中部门名称dloc,要求dloc不能重复。

SQL语句:

```
use test;
select distinct dloc from dept;
```

实例3

查询emp员工表中员工所在的部门编号dept_id,要求编号dept_id不重复。

SQL语句:

```
use test;
select DISTINCT dept_id from emp;
```

1.4.7　使用 WHERE 语句

在使用 select 语句查询时，可以使用 WHERE 子句返回限定的数据行，或者在 WHERE 子句中使用连接运算符来确定表之间的联系，然后根据这个条件返回查询结果。

where 后面通常格式为：列名　比较操作符　要比较的值；

(1)语法：select 字段名 1，字段名 2……from　表名　where　字段名　比较操作符　要比较的值；

①select 表示开始查询数据。

②字段名 1，字段名 2，字段名 3……表示要查询的字段。

③from 表名；表示查询哪张表。

④where 表示连接限定条件。

⑤字段名　比较操作符　要比较的值；表示限定条件。例如：字段名 = 值，字段名＞值等。

1. 比较数据

实例 1

查询 emp 表中员工编号 eid 大于 3 的员工姓名。

SQL 语句：

```
use test;
select ename from emp where eid > 3;
```

实例 2

查询 emp 表中员工薪资 sal 在 3000 到 5000 之间的员工姓名 ename 和员工的薪资 sal。

SQL 语句：

```
use test;
select ename, sal from emp where sal >3000 and sal < 5000;
```

实例 3

查询 emp 员工表中入职日期 hiredate 为“1980-02-12”的员工姓名 ename，员工工作 job，员工入职日期 hiredate。

SQL 语句：

```
use test;
select ename, job, hiredate from emp where hiredate ='1980-02-12';
```

1.4.8　特殊比较运算符

1. BETWEEN…AND…运算符

between and 关键字可以判断某个字段的值是否在指定的范围内。如果字段的值在指定的范围内，则满足查询条件；如果不在指定范围内，不满足查询条件。

(1)语法：select 字段名 1，字段名 2，字段名 3……from 表名　where 条件[not] between 取值 1 and 取值 2；

①select 表示开始查询数据。

②字段名1,字段名2,字段名3……表示要查询的字段。

③from 表名表示查询哪张表。

④where 表示连接限定条件。

⑤[not] between 取值1 and 取值2。

(2)between 取值1 and 取值2;表示条件是在某个范围内。

(3)not between 取值1 and 取值2;表示条件是不在某个范围内。

实例1

查看emp员工表薪资sal在2000到3000之间的员工信息。

SQL语句:

```
use test;
select * from emp where sal between 2000 and 3000;
```

实例2

查看emp员工表中入职日hiredate在1980-01-01到1990-01-01之间的员工姓名ename,部门编号dept_id。

SQL语句:

```
use test;
select ename,dept_id from emp where hiredate  BETWEEN '1980-01-01' and '1991-01-01';
```

实例3

查看emp员工表中入职日hiredate不在1980-01-01到1985-01-01之间的员工姓名ename,部门编号dept_id,入职日期hiredate。

SQL语句:

```
use test;
select ename,dept_id,hiredate from emp where hiredate not BETWEEN '1980-01-01' and '1985-01-01';
```

2. IN运算符

IN关键字可以判断某个字段的值是否在指定的集合中。如果字段的值在集合中,则满足条件;该记录将被查询出来;如果字段的值不在集合中,则不满足查询条件。

(1)语法:select 字段名1,字段名2,字段名3……from 表名 where 字段名[not] in(值1,值2,值3……);

①select 表示开始查询数据。

②字段名1,字段名2,字段名3……表示要查询的字段。

③from 表名 表示查询哪张表。

④where 表示连接限定条件。

⑤字段名[not] in(值1,值2,值3……)。

(2)字段名 in(值1,值2,值3……);表示字段名对应的值在指定的集合中。

(3)字段名 not in(值1,值2,值3……);表示字段名对应的值不在指定的集合中。

实例 1

查询 emp 表中入职日期 hiredate 在“1980-02-01”,“1984-02-01”这两天的员工的姓名 ename 和入职日期 hiredate。

SQL 语句：

```
use test;
select ename,hiredate from emp where hiredate in ('1980-02-01','1984-02-01') ;
```

实例 2

查询 Score 表中成绩为 85,86 或 88 的记录。

(1)任务要求：

①查询 score 表。

②使用 where 子句添加过滤条件,degree 等于 85,86,88 其中的任意一个,使用 in。

(2)SQL 语句：

```
use test;
SELECT *   FROM score WHERE Degree IN(85,86,88);
```

实例 3

查询 95033 班和 95031 班全体学生的记录。

(1)任务要求：

①查询 student 表。

②使用 where 子句添加过滤条件,class 等于“95033”, “95031”其中的一个,使用 in。

(2)SQL 语句：

```
use test;
SELECT * FROM  student WHERE  class IN ('95033','95031');
```

3. LIKE 运算符

如果在使用 like 操作符时,后面的没有使用通用匹配符,效果是和＝一致的。

(1)百分号通配符(%)：表示任何字符出现任意次数(可以是 0 次)。

(2)下划线通配符(_):表示只能匹配单个字符,不能多也不能少,就是一个字符。

实例 1

查询家庭住址在“g”的学生信息。

(1)任务要求：

①查询 user_message 表中记录。

②使用 where 添加过滤条件。

③使用模糊查询,地址为“g”的学生信息。

(2)SQL 语句：

```
use test;
SELECT * FROM user_message WHERE zone LIKE 'g%';
```

实例2

查询出名字中有"m"字符,并且薪水在1000以上(不包括1000)的所有员工信息。

(1)任务要求:

①查询emp表。

②使用where子句添加过滤条件,条件为名字中有"m"字符。

③使用like运算符,%匹配符表示匹配0或任意多个字符。

④添加过滤条件工资大于1000。

⑤使用and运算符连接两个过滤条件。

(2)SQL语句:

```
use test;
SELECT * FROM emp WHERE ename LIKE '%m%' AND sal>1000 ;
```

实例3

查询出名字第三个汉字是"a"的所有员工信息。

用like运算符,_匹配符表示匹配一个字符,第三个字符,需要在"a"前面使用两个。"a"后使用%匹配。

(1)任务要求:

①查询emp表。

②使用where子句添加过滤条件,条件为名字中第三个字符为"a"。

③使用like运算符,_匹配符表示匹配1个字符。第三个字符,需要在"a"前面使用两个。

④"a"后使用%匹配。

(2)SQL语句:

```
use test;
SELECT * FROM emp WHERE ename LIKE '__a%';
```

4. IS NULL运算符

IS NULL关键字可以用来判断字段的值是否为空值(NULL)。如果字段的值是空值,则满足条件,该记录将被查询出来;如果字段的值不是空值,则不满足查询条件。

(1)语法:select 字段名1,字段名2,字段名3……from 表名 where 字段名 is [not] null;

①select表示开始查询数据。

②字段名1,字段名2,字段名3……表示要查询的字段。

③from 表名 表示查询哪张表。

④where表示连接限定条件。

⑤字段名 is [not] null。

(2)字段名 is null;表示字段的值为空满足条件。

(3)字段名 is not null;表示字段的值不为空满足条件。

实例1

查看emp员工信息表中入职日期hiredate为空的员工姓名ename。

SQL 语句：

```
use test;
select ename from emp where hiredate is null;
```

实例 2

使用 ISNULL 关键字判断某个字段的值是否是空值(null)。

(1)任务要求：

①掌握 is null 的用法。

②掌握 is not null 的用法，加上 not 表示字段不是空值时满足条件。

(2)SQL 语句：

```
use test;
select ename, sal from emp where mgr is null;
```

实例 3

查看 dept 部门表中部门地址 dloc 为空的部门所有信息。

SQL 语句：

```
use test;
select * from dept where dloc is null;
```

1.4.9　逻辑运算符

1. 逻辑运算符操作

当需要和多个条件表达式进行比较时，需要使用逻辑运算符把多个表达式连接起来。逻辑运算符包括 AND、OR、NOT，逻辑表达式的结果为 TRUE，FALSE，NULL。

(1)语法：

①select 字段名 1，字段名 2，字段名 3……from 表名　where 条件 1 and 条件 2。

②select 字段名 1，字段名 2，字段名 3……from 表名　where 条件 1 or 条件 2。

③select 字段名 1，字段名 2，字段名 3……from 表名　where 字段 not 条件表达式。

A. select 表示开始查询数据。

B. 字段名 1，字段名 2，字段名 3……表示要查询的字段。

C. from 表名 表示查询哪张表。

D. where 表示连接限定条件。

E. 条件 1 and 条件 2；表示满足条件 1 同时满足条件 2。

(2)条件 1 or 条件 2；表示满足条件 1 或者满足条件 2 即可。

(3)not 条件表达式；表示否定条件表达式可以和 BETWEEN…AND、LIKE、IS NULL 一起使用。

实例 1

查询 emp 员工信息表中职位 mgr 为 BM 而且薪资大于等于 6000 的员工信息。

SQL 语句：

```
use test;
select * from emp where job='BM' and sal >= 6000;
```

实例 2

查看 emp 员工表职位 job 是 staff 或者薪资 sal 小于 3000 的员工信息。

SQL 语句：

```
use test;
select * from emp where job = 'staff' or sal<3000;
```

实例 3

查看 emp 员工表职位 job 不是 BM 并且薪资 sal 大于等于 3000 的员工信息。

SQL 语句：

```
use test;
select * from emp where not job='BM' and sal >= 3000;
```

1.4.10 ORDER BY 子句

使用 ORDER BY 子句能对查询结果集进行排序，语法结构如下：

```
SELECT [DISTINCT] { * | 列名 |表达式 [别名][,...]}
FROM   表名
[WHERE 条件]
[ORDER BY {列名|表达式|列别名|列序号} [ASC|DESC],…];
```

其中：

可以按照列名、表达式、列别名、结果集的列序号排序。

ASC：升序，默认值；

DESC：降序；

ORDER BY 子句必须写在 SELECT 语句的最后。

1. ORDER BY 子句语法

实例 1

查询 dept 表所有数据，并按 id 降序排列。

SQL 语句：

```
use test;
select * from dept ORDER BY id desc;
```

实例 2

查询 emp 表所有数据，并按 sal 升序排列。

SQL 语句：

```
use test;
select * from emp ORDER BY sal ASC;
```

实例 3

查询 emp 表中员工所有信息，要求按照员工 eid 降序排序。

SQL 语句：

```
use test;
select * from emp ORDER BY eid desc;
```

2. 排序规则

实例 1

列出所有员工的年工资，按年薪从低到高排序。

(1)任务要求：

①查询 emp 表。

②使用算数表达式计算员工年工资，并为年工资起别名。

③使用 order by 对年工资排序，默认为升序，也可以使用 asc 关键字。

④排序可以使用列别名。

⑤查询列表显示员工信息及年工资。

(2)SQL 语句：

```
use test;
SELECT emp.*,(sal+IFNULL(comm,0))*12 nianxin FROM emp ORDER BY nianxin ASC ;
```

实例 2

查询出 emp 表中部门编号为 20，薪水在 2000 以上(不包括 2000)的所有员工，显示他们的员工号、姓名以及薪水，以如下列名显示：员工编号 员工名字 薪水

(1)任务要求：

①查询 emp 表。

②使用 where 子句添加过滤条件，工资大于 2000 及部门编号为 20。

③使用 and 运算符连接两个过滤条件，要求两个条件同时成立。

④查询列表显示员工编号，员工姓名，工资，并为查询字段起别名。

(2)SQL 语句：

```
use test;
SELECT empno AS no, ename AS name, sal AS salary
  FROM emp WHERE deptno=20 AND sal>2000;
```

实例 3

将所有员工按薪水升序排序，薪水相同的按照入职时间降序排序。

(1)任务要求：

①查询 emp 表。

②使用 order by 子句添加排序条件。

③第一个排序条件为 sal，使用 asc 实现升序排序。

④第二个排序条件为 hiredate，使用 desc 实现降序排序。

⑤两个排序条件使用逗号分隔。

(2)SQL 语句：

```
use test;
SELECT * FROM emp ORDER BY sal ASC, hiredate DESC;
```

3. 按列名升序排序

实例 1

以 Class 降序查询 Student 表的所有记录。

(1)任务要求:

①查询 student 表。

②使用 order by 子句添加排序条件。

③排序条件为 class,使用 desc 实现降序排序。

(2)SQL 语句:

```
use test;
SELECT *   FROM student ORDER BY class DESC;
```

实例 2

以 Cno 升序,Degree 降序查询 Score 表的所有记录。

(1)任务要求:

①查询 score 表。

②使用 order by 子句添加排序条件。

③第一个排序条件为 cno,使用 asc 实现升序排序。

④第二个排序条件为 degree,使用 desc 实现降序排序。

⑤两个排序条件使用逗号分隔。

(2)SQL 语句:

```
use test;
SELECT *   FROM score ORDER BY cno ASC,degree DESC;
```

实例 3

查询 emp 表中员工的所有信息,要求按照入职日期 hiredate 升序排序。

SQL 语句:

```
use test;
select * from emp group by hiredate asc;
```

1.4.11 限制记录行数

1. 限制记录行数

LIMIT 子句可以被用于强制 SELECT 语句返回指定的记录数。LIMIT 接受一个或两个数字参数。参数必须是一个整数常量。如果给定两个参数,那么第一个参数指定第一个返回记录行的偏移量,第二个参数指定返回记录行的最大数目。初始记录行的偏移量是 0,而不是 1。

(1)语法:select 字段名 1,字段名 2,字段名 3……from 表名 where 条件 limit (当前页数-1) * 每页显示记录数,每页显示记录数;

①select 表示开始查询数据。

②字段名 1,字段名 2,字段名 3……表示要查询的字段。

③from 表名 表示查询哪张表。

④where 表示连接限定条件。

⑤条件表示对查询的内容做限定。

⑥ limit(当前页数－1) * 每页显示记录数，每页显示记录数;表示要从哪页检索数据，每页显示几条。

实例 1

查询 user_test 表中记录,要求每页显示 2 条记录,查询出第三页的记录。

(1)任务要求：

①查询 user_test 表中所有记录。

②计算第三页起始记录的索引号,为（当前页数－1) * 每页显示记录数。

③使用 limit 关键字实现分页查询。

(2)SQL 语句：

```
use test;
SELECT * FROM user_test LIMIT 4,2;
```

实例 2

查询 emp 表中员工的所有信息,要求每页显示 3 条,查询第一页的记录。

SQL 语句：

```
use test;
select * from emp limit 0,3;
```

实例 3

查询 emp 表中员工的所有信息,要求每页显示 2 条,查询第二页的记录。

SQL 语句：

```
use test;
select * from emp limit 2,2;
```

1.5　常用函数

MySQL 函数是 MySQL 数据库提供的内置函数,这些函数可以帮助用户更加方便地处理表中的数据。

1.5.1　数学函数

数学函数是 MySQL 中常用的一类函数,其主要用于处理数字,包括整型和浮点型等。常见的数学函数有绝对值函数、正弦函数、余弦函数和获取随机函数等。

1. 绝对值、平方、取余函数

实例 1

查询数值 6,－6 的绝对值。

SQL语句：

```
use test;
select abs(6),abs(-6);
```

实例2

查询4,-4的平方根。

(1)语法:select sqrt(值1),sqrt(值2);

①select表示开始查询数据。

②sqrt(值1),sqrt(值2);表示查询值1的平方根,查询值2的平方根。

(2)SQL语句：

```
use test;
select sqrt(4),sqrt(-4);
```

实例3

查询10除以3的余数。

(1)语法:select mod(值1,值2);

①select表示开始查询数据。

②mod(值1,值2);表示求值1除以值2的余数。

(2)SQL语句：

```
use test;
select mod(10,3);
```

1.5.2 字符串函数

字符串函数是MySQL中最常用的一类函数。字符串函数主要用于处理表中的字符串。

(1)repeat(s,n)函数将字符串s重复n次。

(2)subString(s,n,len)函数获取字符串s中的第n个位置开始,长度为len的字符串。

(3)reverse(s)函数将字符串s的顺序反过来。

(4)left(s,n)函数返回字符串s开始的前n个字符。

1.大小写转换函数

实例1

使用upper函数将"neUEdu"转成大写。

(1)语法:select upper(值);

①select表示开始查询数据。

②upper(值);表示将值中每一个字母转成大写字母。

(2)SQL语句：

```
use test;
select upper('neUEdu');
```