目录

2024年4月21日　　16:57

数据库简介

2024年5月23日 14:35

一、数据库简介

1.1 主流数据库

1、关系型数据库

（1）MySQL:开源免费的关系型数据库，易于使用和学习，支持大型企业级应用。其特点包括高性能、可靠性和可扩展性，支持多种编程语言和操作系统，拥有大量的社区支持和插件。
（2）SQLite:轻量级的嵌入式关系型数据库，支持跨平台开发和部署，易于使用和集成适用于小型应用和移动用。
（3）SQL Server:微软开发的关系型数据库管理系统，适用于中小型企业应用，拥有可靠的性能、安全性和易性。
（4）Oracle:全球领先的商业关系型数据库，拥有极强的稳定性、安全性和可扩展性，支持高可用、分布式架构，提供强大的数据分析和管理功能。

2、非关系型数据库

（1）MongoDB:面向文档的非关系型数据库，可扩展性强，可在多台机器上进行分布式部署。
（2）Redis:内存数据库，以键值对形式存储数据，支持多种数据结构，适用于高速读写和缓存等场景。

关系和非关系型数据库的区别：存储时是否使用了字段（列名），关系型数据库是按照字段存储，非关系型数据库是直接一条存起来的

1.2 数据库的基本概念

数据库管理系统(Database ManagementSystem，DBMS):数据库系统中对数据进行管理的软件系统。----> MySQL等软件
数据库(Database，DB):按照特定的数据结构来组织、存储和管理数据的仓库 文件 ----> 数据库
表(Table):某种特定类型数据的结构化清单。 工作表 ----> 表
列(Column)或字段:表由一个或多个列组成，每个列有对应的数据类型， 表头/列 ----> 字段/列
行(Row)或记录:表中的数据是按行存储的，每行存储一条数据(记录) 一行数据 ----> 一条记录/数据
主键(Primary Key):一列(或一组列)，它的值能够唯一区分表中每一行! 最主要的列 ----> 主键
SQL(Structured QueryLanguage):结构化查询语言，专门用于与数据库通信的语言 查询的语言 ----> SQL

1.3 MySQ启动

1、软件的安装需要注意的问题:
安装路径使用默认路径
用户名:root密码:123456
2、软件启动
（1）开始->MySQL->MySQL5.7->123456
net start mysql57(关闭命令为 net stop mysql57,需要管理员权限打开 cmd，以后如果无法连接数据库，先执行该命令) Windows+r打开cmd输入:mysql-uroot -p123456。

（2）welcome...安装成功
（3）mysql不是内部命令x
修改环境变量:电脑->属性->高级系统设置->高级->环境变量->系统变量(path)->编辑->新建->(默认安装路径:C\Program Files\MySQL\MySQL Server 5.7\bin)->所有窗口确定
（4）不能连接...x
服务开启:搜索服务->点击名称一列->按 M 键->找到MySQL->鼠标右键->启动
（5）闪退原因:
(1)服务没有开启
(2)密码错误
（6）命令行中退出
Exit
Quit

1.4 MySQ结构

1、Server层:
（1）连接器:负责和客户端建立连接，管理连接
（2）查询缓存:(如果开启了查询缓存功能)先到缓存中查询是否查询过这条语句，缓存的失效非常频繁，只要对表有更新，该表的查询缓存都会被清空，MVSQL8.0删除了该功能
（3）分析器:对 SQL语句进行词法分析和语法分析，判断语句是否合法
（4）优化器:对 SQL语句进行优化，选择索引
（5）执行器:调用存储引擎接口，返回结果
2、存储引擎层:
数据的存储和提取，默认的存储引擎是InnoDB，5.5 之前的默认是 MyISAM

基本命令

2024年5月23日　　15:21

二、基本命令

2.1 基本命令

1、显示数据库　show databases;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydb2311           |
| mysql              |
| performance_schema |
| sakila             |
| test               |
| world              |
+--------------------+
7 rows in set (0.00 sec)
```

2、创建数据库 create database [if not exists] mydb2311;

3、使用数据库 use mydb2311;

```
mysql> use mydb2311;
Database changed
mysql>
```

4、查询当前数据库信息

```
#查询当前连接的数据库
select database();
#查询当前的数据库版本
select version();
#查询当前的日期
select now();
#查询当前的用户
select user();
```

5、删除数据库：将已经存在的数据库清除，数据库中的数据也将被清除

drop database [if exists] mydb;

```
mysql> drop database mydb;
Query OK, 0 rows affected (0.00 sec)
```

6、导入数据库脚本(结尾不加分号，读取外部 SQL脚本，SQL脚本里的语句是以分号结尾

导入的时候没有分号，路径中不能含有中文

需要先创建数据库，然后 use 使用才能导入

source 脚本文件路径

7、查看数据库中的表 show tables;

```
mysql> show tables;
+--------------------+
| Tables_in_mydb2311 |
+--------------------+
| dept               |
| emp                |
| salgrade           |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

8、查看表的基本机构（描述表）：desc emp;（表名）

字段----------类型---是否能为空 -主键--默认值----备注

```
mysql> desc emp;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| EMPNO    | int(4)      | NO   | PRI | NULL    |       |
| ENAME    | varchar(10) | YES  |     | NULL    |       |
| JOB      | varchar(9)  | YES  |     | NULL    |       |
| MGR      | int(4)      | YES  |     | NULL    |       |
| HIREDATE | date        | YES  |     | NULL    |       |
| SAL      | double(7,2) | YES  |     | NULL    |       |
| COMM     | double(7,2) | YES  |     | NULL    |       |
| DEPTNO   | int(2)      | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
8 rows in set (0.01 sec)

mysql>
```

9、查看数据库\表的创建语句

查看数据库创建：show create database mydb2311;

```
mysql> show create database mydb2311;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| mydb2311 | CREATE DATABASE `mydb2311` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

查看表创建：show create table emp;

```
mysql> show create table dept;
+-------+-------------------------------------------
-----------------------------------------------------
| Table | Create Table

+-------+-------------------------------------------
-----------------------------------------------------
| dept  | CREATE TABLE `dept` (
  `DEPTNO` int(2) NOT NULL,
  `DNAME` varchar(14) DEFAULT NULL,
  `LOC` varchar(13) DEFAULT NULL,
  PRIMARY KEY (`DEPTNO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+-------------------------------------------
-----------------------------------------------------
1 row in set (0.00 sec)

mysql>
```

创建的语句和脚本语句不一定一致。这是因为数据库管理系统MySQL结构中带有优化器，会自动优化SQL语句
ENGINE（引擎）：默认的存储引擎是InnoDB，5.5 之前的默认是 MyISAM
CHARSET(字符集)： utf-8支持中文。

2.2 基本查询

1、指定字段查询：select empno,ename,sal from emp;（优点：可以指定字段顺序）
select 字段1,字段2,字段3 from 表名;

```
mysql> select empno,ename,sal from emp;
+-------+--------+---------+
| empno | ename  | sal     |
+-------+--------+---------+
|  7396 | SMITH  |  800.00 |
|  7499 | ALLEN  | 1600.00 |
|  7521 | WARD   | 1250.00 |
|  7566 | JONES  | 2975.00 |
|  7654 | MARTIN | 1250.00 |
|  7698 | BLAKE  | 2850.00 |
|  7782 | CLARK  | 2450.00 |
|  7788 | SCOTT  | 3000.00 |
|  7839 | KING   | 5000.00 |
|  7844 | TURNER | 1500.00 |
|  7876 | ADAMS  | 1100.00 |
|  7900 | JAMES  |  950.00 |
|  7902 | FORD   | 3000.00 |
|  7934 | MILLER | 1300.00 |
+-------+--------+---------+
14 rows in set (0.00 sec)

mysql>
```

2、查询全部字段：select * from emp; （优点：快速）

3、带条件的查询：select ename,job,sal from emp where sal >= 2000;

2.3 运算符

1、算术运算符

- 算术运算符：+ - * / %

```
# 员工年工资
select * from emp where sal * 12 > 20000;
```

2、比较运算符

运算符	作用
> <	大于 小于
>= <=	大于等于 小于等于
=	等于
<>/!=	不等于
is (not) NULL	判断一个值是否为空(或不为空)
between ... and	判断一个值是否在两个值之间
(not) in	判断一个值是否是(或不是) in 列表中的值
like	通配符匹配,支持%(通配符)或_(占位符)

模糊查询： select * from emp where ename like 'a%';（查看姓名首字母是a的员工）
select * from emp where ename like '_a%';（_ 占位符，查看姓名第二个字母是a的员工）

3、特殊
（1）查看某行为空值的数据：select ename,job,sal,comm from emp where comm is NULL;

表

2024年5月23日　　17:48

三、表

3.1 数据类型

1、整型

数据类型	存储范围	字节
TINYINT	有符号值：-128到127（-2^7到2^7-1） 无符号值：0到255（0到2^8-1）	1
SMALLINT	有符号值：-32768到32767（-2^{15}到$2^{15}-1$） 无符号值：0到65535（0到$2^{16}-1$）	2
MEDIUMINT	有符号值：-8388608到8388607（-2^{23}到$2^{23}-1$） 无符号值：0到16777215到(0到$2^{24}-1$)	3
INT	有符号值：-2147483648到2147483647（-2^{31}到$2^{31}-1$） 无符号值：0到4294967295到(0到$2^{32}-1$)	4
BIGINT	有符号值：-9223372036854775808到9223373036854775807（-2^{63}到$2^{63}-1$） 无符号值：0到18446744073709551615到(0到$2^{64}-1$)	8

2、浮点型和定点型

数据类型		存储范围
浮点数类型	FLOAT[(M,D)]	4个字节存储 范围允许的值是: -3.402823466E+38到-1.175494351E-38、0和1.175494351E-38到3.402823466E+38
	DOUBLE[(M,D)]	8个字节存储 范围允许的值是: -1.7976931348623157E+308到-2.2250738585072014E-308、0和2.2250738585072014E-308到 1.7976931348623157E+308。
定点数类型	DECIMAL[(M,D)]	M是 精度（=整数位数+小数位数），D是标度（小数点后的位数）。

3、日期时间类型

列类型	字节数	取值范围	表示形式
YEAR	1	1901 ~ 2155	YYYY
TIME	3	-838:59:59 ~ 838:59:59	HH:MM:SS
DATE	4	1000-01-01~9999-12-31	YYYY-MM-DD
DATETIME	8	1000-01-01 00:00:00~9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS
TIMESTAMP	4	19700101080001~20380119111407	YYYY-MM-DD HH:MM:SS

4、字符型

VARCHAR(M)：定长字符串

列类型	存储需求
CHAR(M)	M个字节，0<=M<=255
VARCHAR(M)	L+1个字节，其中L<=M且0<=M<=65535
TINYTEXT	0 ~ 255（2^8-1）
TEXT	0 ~ 65535（2^{16}-1）
MEDIUMTEXT	0 ~ 2^{24}-1
LONGTEXT	0 ~ 2^{32}-1
ENUM('value1','value2',...)	取决于枚举值的个数（最多65,535个值）例：性别 enum('男','女')
SET('value','value2',...)	取决于set成员的数目（最多64个成员）

3.2 DDL（数据定义语言）

Data Definition Language，用于定义和管理数据库中的对象和结构，如表、列、索引等。

3.2.1 创建表

```
create table [if not exists] 表名(
    字段1 字段类型 [列级约束条件],
    字段2 字段类型 [列级约束条件],
    ...
    [表级约束条件]
);
```

```
# 创建 users表
create table if not exists users(
  id int,
  name varchar(20)
);
```

练习

1.创建一个文章表 article，有如下字段

a. id:文章ID

b.title:标题

c.author:作者

d.content:文章内容

```
# 创建文章表article
create table if not exists article(
  id int,
  title varchar(30),
  author varchar(20),
  content text
);
```

2.创建一个订单表，有如下字段

a.id:订单 ID

b.customer id:顾客编号

c.date:订单日期

d.money:订单金额

```
# 创建订单表 orders
create table if not exists orders(
  id int,
  customer_id int,
  date datetime,
  money decimal(10,3) # 共10位，小数点后3位
);
```

3.创建一个产品表，产品有如下属性，请自己设计字段

a.产品名

b.价格

是否有库存

产品描述

```
# 创建产品表 product
create table if not exists product(
  id int,
  name varchar(50),
  price decimal(10,2),
  is_null boolean,
  pro_desc text
);
```

3.2.2 查看表

```
# 查看数据库中的表
show tables;

# 查看表的基本结构 desc 表名;
desc emp;

# 查看创建数据库/表 的创建语句
# show create database 数据库名;
show create database mydb2311;
# show create database 表名;
show create table emp;
```

3.2.3 修改表

理想状态下，当表中存储数据以后，该表就不应该再被更新。在表的设计过程中需要花费大量时间来考虑，以便后期不对该表进行大的改动。

添加字段

```
alter table 表名 add column 新列名 数据类型[约束条件][first | after 列名];

# 添加字段--在某个字段后添加新的字段
alter table product2 add column size int after price;
# 添加字段--在开始添加新的字段
alter table product2 add column size2 int first;
```

修改字段的类型

```
alter table 表名 modify column 列名 数据类型![约束条件];

# 修改字段类型
alter table product2 modify column size2 decimal(10,2);
```

修改字段的位置

```
alter table 表名 modify column 列名 数据类型 first 1 after 列名;

# 修改字段位置
alter table product2 modify column size2 decimal(10,2) after is_null;
```

修改字段名

```
alter table 表名 change column 旧列名 新列名 数据类型

# 修改字段名
alter table product2 change column size2 size3 decimal(10,2);
```

删除字段

```
alter table 表名 column 列名 drop
# 删除字段
alter table product2 drop column size3;
```

3.2.4 重命名表

```
alter table 表名 rename to 新表名
# 更改表名
alter table product rename to product2;
```

3.2.5 删除表

```
drop table [if exists]表1[, 表2, 表3...];
# 删除表
drop table if exists users;
```

3.2.6 练习

1.员工 emp 表

添加字段 emp_bonus 奖金数据类型为decimal(10,2)
emp_bonus 字段数据类型改为decimal(12,2)
修改 emp_bonus 字段名为 bonus
删除字段 bonus

```
alter table emp add column emp_donus decimal(10,2);
alter table emp modify column emp_donus decimal(12,2);
alter table emp change column emp_donus bonus decimal(12,2);
alter table emp drop column bonus;
```

2.练习学生信息表的创建与修改

建立学生信息表，字段包括:学号、姓名、年龄、出生日期
在姓名字段后加电话字段，字段名 tel no，类型为 varchar 类型
将电话字段修改为 char 类型
将出生日期字段放在学号字段之前
将电话字段名 tel no 修改为 phone_number
删除字段 phone_number
将表名修改为另外的名字
删除表

```
# 建立学生信息表，字段包括:学号、姓名、年龄、出生日期
create table if not exists StuInfo(
  id int,
  name varchar(20),
  age int,
  brithday date
);

# 在姓名字段后加电话字段，字段名 telno，类型为 varchar 类型
alter table StuInfo add column telno varchar(11) after name;
# 将电话字段修改为 char 类型
alter table stuinfo modify column telno char;
# 将出生日期字段放在学号字段之前
alter table stuinfo modify brithday date first;
# 将电话字段名 tel no 修改为 phone_number
alter table stuinfo change column telno phone_number varchar(11);
# 将表名修改为另外的名字
alter table stuinfo rename to student;
# 删除表
drop table if exists student;
```

3.3 DML（数据定义语言）

Data Manipulation Language，用于操作数据库中的实际数据，如插入、更新、删除和查询数据。

3.3.1 插入数据

插入完整的行，所有字段，每一个字段都必须提供一个值，如果某个字段没有值，应该使用 NULL，每个字段必须以他们在表中定义的顺序给出。这种语法简单，但是不安全，应该避免使用。

更安全的方法是，在表名后的括号中给出字段名，values 中的值的顺序与前面给出的字段的顺序相同，一一对应。不需要与表中定义的顺序相同，没有值的字段可以不提供。

```
insert into 表名(
字段 1,
字段 2,
)
values(
字段1 的值,
字段2的值,
...
 );
```

values 后面可以跟多个括号，括号间用逗号分隔，每个括号代表要插入的一条数据。

单条 insert 语句插入多条数据比多条 insert 语句快

```
# 插入一条数据
```

```
insert into users
values(100,"张三");

select * from users;

# 插入多条数据
insert into users
values
(101,"张三1"),
(102,"张三2"),
(103,"张三3"),
(104,"张三4");

# 在某个字段插入数据，其他字段为NULL
insert into users
values
(105,NULL);
```

3.3.2 更新数据

```
update 表名
set 字段1 =字段1的值
字段2=字段2的值
where 限制条件
```

记得加 where 限制条件，否则更新整个表中的每一条数据

```
# 更新数据-----必须加where限制，否则将更新整个表
update users
set
id = 110,
name = "张三5"
where id = 105;
```

3.3.3 删除数据

```
delete from 表名
where 限制条件;

delete from users where id = 106;
```

同样，不加限制条件会删除整个表中每一条数据。物理删除，无法恢复。
注意:在对 update 或 delete 使用 where 前，应该先用 select 语句进行测试，保证它的过滤结果是正确的。

一般不会使用delete进行删除，一般是添加一个 is_delete 的字段，
如果用户删除，会使用update将该条记录的 is_delete 的数值改为 1，
然后在查询部分 在where 部分添加 只能查看is_delete = 0 的限制，
这样的操作称之为软删除

3.3.4 练习

- 建立学生信息表，字段包括：学号、姓名、年龄、出生日期
- 为学生信息表添加三条记录（张三、李四、王五）

学号	姓名	年龄	出生日期
1	张三	20	2002/5/8
2	李四	21	2001/9/6
3	王五	19	2003/12/15

- 将张三的年龄改为 21，出生日期改为 2001/5/8
- 删除不到 21 岁的学生

```
# 创建表
create table if not exists student(
 id int,
 name varchar(20),
 age int,
 birthday date
);

# 插入数据
insert into student
values
(1,"张三",20,"2002-5-8"),
(2,"李四",21,"2001-9-6"),
(3,"王五",19,"2003-12-15");

select * from student;


# 将张三的年龄改为 21，出生日期改为 2001/5/8
update student
set age = 21, birthday = "2001-5-8" where id = 1;
# 删除不到 21 岁的学生
delete from student where age < 21;
```

3.4 约束

constraint

约束类型	非空约束	默认约束	唯一约束	主键约束	外键约束
关键字	not null	default	unique	primary key	foreign key

非空约束 not null
默认约束 default
唯一约束 unique
主键约束 primary key
外键约束 foreign key

创建表时添加约束

```
# 创建表时添加约束
create table users(
  id int not [约束条件1 约束条件2 ...],
  name varchar(20)
```

```
);
```

3.4.1 非空约束 not null

注意不要混淆 NULL 值和空串，NULL 值是没有值，空串是""两个单引号，中间没有字符

```
# 创建表时添加非空约束 not null
create table users(
  id int not null,
  name varchar(20)
);

insert into users(name) values("张三"); # 报错：id没有默认值（不能为空）

# 创建表后修改非空约束
alter table users modify column name varchar(20) not null;
insert into users(id) values(1); # 报错：name没有默认值（不能为空）
alter table users modify column name varchar(20) null; # null可以不写，默认可以为空
```

3.4.2 默认约束 default

DEFAULT，如果在插入行时没有给出值，通过 DEFAULT 指定此时使用的默认值

```
# 默认约束 - default
# 创建表时添加默认值
create table users(
  id int not null default 555,
  name varchar(20)
);

insert into users(name) values("张三"); # 不报错：有默认值（就不为空了）
```

3.4.3 唯一约束 unique

用于确保特定列或列组合的唯一性，被约束的列的值在整个表中是唯一的，唯一约束默认允许空值(nu)，因此多个空值不违反唯一约束。

```
# 唯一约束 --- unique
create table users(
  id int not null default 555,
  name varchar(20),
  # 可以给联合唯一约束起名字
  constraint id_name unique(id,name) # 俩个字段不同时重复（联合唯一约束）
);

insert into users(id,name) values(1,"张三");
insert into users(id,name) values(1,"李四");
insert into users(id,name) values(2,"张三");

insert into users(name) values("张三");
insert into users(name) values("张三"); # 报错 id 默认555，名字张三 都重复了

insert into users(id) values(1);
insert into users(id) values(1); # 不报错，name无默认值，默认为null，null可以重复
```

```
# 已有字段添加唯一约束
alter table users modify id int unique;
alter table users add constraint id_name unique(id,name);

# 删除唯一约束
# 唯一约束又是索引，因此可以安装删除索引的方式来删除
alter table users drop index id_name;
alter table users drop key id_name;
```

3.4.4 主键约束 primary key

唯一标识表中每行的这个列(或这组列)称为主键。主键用来表示一个特定的行。没有主键，更新或删除表中特定行很困难，因为没有安全的方法保证只涉及相关的行。

主键非空且唯一

因此:

一个表都应该定义主键
主键的值不应该修改
不使用可能会修改值的列作为主键(与业务无关，这通常使用 id 作为主键)

特点：

唯一性：主键要求每一行数据的主键值都必须是唯一的，不允许有重复值
非空性：主键要求主键列的值不能为空，即不能为 NULL
单一性：每个表只能有一个主键。主键可以由一个列或多个列组成，形成复合主键
列级：

```
create table 表名(
    字段名 字段类型 primary key,
    ...
);
```

```
# 创建表时，设置主键
create table if not exists users(
  id int primary key,
  name varchar(20) not null
);
# 插入数据
insert into users(id,name) values(1,"张三");
insert into users(id,name) values(1,"张三"); # 报错，id是主键，唯一
insert into users(id,name) values(2,"张三");

insert into users(name) values("李四"); # 报错，插入值时必须给主键赋值
```

- 表级，可以给约束起名，可以创建多列的联合主键

```
create table 表名(
    字段1 字段类型,
    字段2 字段类型,
    ...
    [constraint 约束名] primary key(字段1[, 字段2, ...])
);
```

```
# 删除主键
alter table users drop primary key;
```

3.4.5 自动递增 auto_increment

auto_increment:设置 auto increment的列，当每增加一行时自动增量。每个表只允许一个auto increment列。

自动递增一般配合主键使用

```
create table 表名(
    字段1 字段类型 auto_increment
);
```

```
# 创建表时，对主键添加 自动递增
create table if not exists users
(
  id int primary key auto_increment,
  name varchar(20) not null
);

# 插入数据
insert into users(name) values("张三");
insert into users(name) values("张三"); # 不报错，id从一开始自动递增

insert into users(id,name) values(7,"张三"); # 也可以自己设置id不存在的数据
insert into users(name) values("张三"); # 再次插入的时候，就从8开始
```

3.4.6 外键约束 foreign key

外键都是两个表之间的约束

外键为表中的某一字段，该字段是另一表的主键值，外键用来在两个表的数据之间建立联结，一个表中可以有一个或多个外键。外键的作用是保持数据的一致性、完整性

注意：

外键字段可以为 null，外键为空的数据也叫孤儿数据

有了外键引用之后，表分为父表和子表：

创建表时先创建父表，再创建子表

插入数据时，先插入父表数据，再插入子表数据

删除时先删除子表，再删除父表

子表外键类型要与父表外键类型一致

在子表创建外键

```
[constraint 外键名] foreign key (列名) references 主表名(主键);
```

```
# 创建父表--student
create table if not exists student
(
  id int primary key auto_increment,
  name varchar(20) not null
);
# 创建子表 -- score，并连接外键
create table if not exists score
(
  id int primary key auto_increment,
  student_id int not null,
```

```
  class_name varchar(20) not null,
  score int not null,
  #         外键名           子表的字段              父表的字段
  constraint fk1 foreign key (student_id) references student(id)
);

insert into score values(1,1,"数学",88); # 报错，主键中没有 id为1 的学生

insert into student(name) values("张三");
insert into score values(1,1,"数学",88); # 成功添加

# 可以不给 student_id 赋值，在父表里无对应数据，称之为孤儿数据
insert into score(id,class_name,score) values(2,"数学",88);

# 报错，子表外键还有链接的数据，得先删除子表中的数据，才能删除主表中的数据
delete from student where id =1;

delete from score where id =1; # 删除主表数据

delete from student where id =1; # 成功
```

高级查询

2024年6月25日 15:25

四、高级查询

4.1 查询处理

4.1.1 排序 order by

默认不是按照主键进行排序的

order by 子句:对查询结果按指定字段进行排序。也可以指定 select列表中列的序号进行排序

```
# 按照empno升序查询(asc),默认升序
select * from emp order by empno asc;
# 降序查询
select * from order by empno desc;

create table if not exists users
(
  id int primary key,
  name varchar(20)
);

insert into users values(1,"zhangsan"),(2,"lisi"),(3,"wangwu");
select * from users; # 默认不是按照主键排序的
alter table users add age int; # 添加一个字段
select * from users; # 排序方式会改变

# 按照多个字段排序
select * from emp order by deptno,sal desc; # 按照deptno升序，deptno相同的话，按照sal降序
```

4.1.2 限制数量 limit

在真实的业务中，数据的条数是很大的，对数据集进行排序后，不可能之间显示全部的数据，一般只是输出几条数据。这时就就要用到 limit 来限制显示数据条数

limit 子句:select 语句返回所有匹配的行，它们可能是指定表中的每个行。为了前几行或中间几行，可使用

limit 子句。使用limit 可以解决分页问题。

```
limit 行数 (从第一行开始)
limit  开始行 (从 0 开始) , 行数
```

```
# 显示几条数据
select * from emp order by empno asc limit 5;
# 从第3+1条开始，显示5条数据
select * from emp order by empno asc limit 3,5;
```

4.1.3 去重 distinct

distinct 关键字:用于返回唯一不同的值

```
# 查询 有几种工作
select distinct job from emp;
# 同时作用两列，不能查询目标列以外的列
select distinct job,mgr from emp; # 两个字段不能同时重复
```

4.1.4 组合查询 union

union 操作符:执行多个查询(多条 select语句)，将结果合并为单个结果集返回。

注意：查询时，字段数量一致，字段类型相似
union 会自动去重，不想去重 用 union all

```
select 字段1[,字段2,...] from 表1
union
select 字段1[,字段2,...] from 表2;
```

```
# 组合查询，按照行排列
select empno,ename from emp
union
select deptno,dname from dept;
```

4.2 函数

4.2.1 数值函数

```
# abs(x):返回x的绝对值
select abs(-20); # 20

# ceil(x):向上取整，返回大于等于x的最小整数值
select ceil(5.3); # 6

# floor(x):向下取整，返回小于等于x的最大整数值
select floor(5.99); # 5

# round(x,y=0):四舍五入，将x四舍五入y位小数，y不传返回整数，y为负数时，保留 x值到小数点左边 y位
select round(5.5);

# truncate(x,y):截断函数，返回被舍去至小数点后y位的数字x，y为负数时截断小数点左边y位
select truncate(5.23,1);

# mod(x,y):返回x除以y的余数
select mod(5,2);
```

```
# rand():生成 0-1 的随机数
select rand();

# rand():生成 1-10 的随机数
select truncate(rand()*10+1,0);
```

4.2.2 字符函数

(1) concat(s1,s2,...):字符串连接，如果任何一个参数为 nul，则返回值为 null
```
select concat("hello ","world");
```

(2) concat_ws(x,s1,s2,….):指定分隔符的字符连接函数，x是连接分隔符，如果分隔符为null，则结果为 null。
```
select concat_ws("xxx","hello ","world");
```

(3) lower(str):大写转小写
```
select lower("WeiHong");
```

(4) upper(str):小写转大写
```
select upper("weihong");
```
(5) length(str):字符串长度
(6) ltrim(str):删除字符串左侧空格
(7) rtrim(str):删除字符串右侧空格
(8) trim(str):删除字符串两侧空格
(9) substr(str,n,len):截取子字符串，字符串 str 从n的位置截取长度为 len 的字符串，如果 n为负数，则子字符串的位置起始于字符串结尾的n个字符
(10) left(str, n):返回字符串 str 的最左边n个字符
(11) right(str,n): 返回字符串 str 的最右边 n个字符
(12) replace(str,from str,to str):替换函数，字符串 str 中所有的字符串 from str均被 to str 替换，然后返回这个字符串
(13) format (x,n):将数字x格式化，并以四舍五入的方式保留小数点后n位，结果以字符串的形式返回。若n为0，则返回结果不含小数部分。

4.2.3 日期时间函数

(1) curdate()/current_date():获取当前日期，YYYY-MM-DD 格式
(2) curtime()/current_time():获取当前时间，HH:MM:SS 格式
(3) week(date):返回 date 为一年中的第几周
(4) now()/sysdate():获取当前日期和时间，YYYY-MM-DD HH:MM:SS 格式
(5) date_add(date,interval expr type):执行日期的加运算，date 是一个 datetime 或 date 值，指定起始时间。expr是时间间隔。type为关键词，如YEAR,MONTH,DAY,WEEK HOUR等。
(6) datediff(date1,date2):计算两个日期之间的间隔天数
(7) unix_timestamp(date):返回 date 的 UNIX 时间戳
(8) form_unixtime(unix):返回 unix 时间戳的日期值
(9) date_format(date,format):日期格式化，按 format 格式化 date 值
(10) str_to_date(date,format):将字符串转换成 date 类型

```
# (9) date_format(date,format):日期格式化，按 format 格式化 date 值
select date_format('2024/6/25','%Y年%m月%d日');

# (10) str_to_date(date,format):将字符串转换成 date 类型
select str_to_date('2024年6月+++++++++25日','%Y年%m月+++++++++%d日');
```
将 2024年6月+++++++++25日 转为 2024-06-25

4.2.4 聚合函数(分组函数)

聚合函数会改变数据的条数,查询后变为一条数据

avg(expression):返回某列的平均值
sum(expression):返回某列值的和
count(expression):返回某列的行数
max(expression):返回某列的最大值
min(expression):返回某列的最小值

注意:

聚合函数会自动的忽略空值，不需要手动增加条件排除 NULL

聚合函数不能作为 where 子句后的限制条件（因为在查的过程，聚合函数无法计算，得全部查完才能算）

```
# avg(expression):返回某列的平均值
select avg(sal) from emp;
# sum(expression):返回某列值的和
select sum(sal) from emp;
# count(expression):返回某列的行数
select count(sal) from emp;
# max(expression):返回某列的最大值
select max(sal) from emp;
# min(expression):返回某列的最小值
select min(sal) from emp;
```

4.2.5 流程函数

if(value,t,f):如果 value 为真返回 t，否则返回 f

ifnull(column, value):如果 column 为空返回 value，否则返回 column

在 SQL 语句当中若有 NULL 值参与数学运算，计算结果一定是 NULL，为了防止计算结果出现 NULL，建议先使用 ifnu 空值处理函数预先处理。

```
# 任何数 + null = null
select sal + comm from emp; # 这样的结果大多都是null

select sal + ifnull(comm,0) from emp;

select ename,empno,if(sal >3000,"sal_high","sal_low") from emp;
```

4.2 分组查询 group by

4.2.1 创建分组

group by 子句:根据一个或多个字段对结果集进行分组，在分组的字段上可以使用count、sum、avg等函数。

```
select 字段1[字段2, function(字段1), function(字段2)...]
from 表
group by 字段1;
```

按照分组的字段进行分组，分组之后，一组只有一条数据，因此，分组之后，只有分组的字段是有意义的。

```
# 查看公司有几个部门
select deptno from emp group by deptno;

# 添加部门编号 为 20 的员工人数
select count(*) from emp where deptno = 20;

# 查看每个部门的人数
select deptno,count(*) from emp group by deptno;
```

注意

如果分组列中具有 NULL 值，则 NULL 将作为一个分组返回。
如果列中有多行 NULL值，它们将分为一组。
group by 子句必须出现在 where 子句之后，order by 子句之前

因此使用where 只能先过滤，再分组

4.2.2 过滤分组

having 子句:having 非常类似于 where。唯一的差别是 where 过滤行，而 having 过滤分组。
having 必须和 group by 一起使用。

having 和 where 的区别也可以理解为，**where 是分组前过滤，having 是分组后过滤**。

```
# 查看每个部门的人数,并保留部门人数大于5的分组
select deptno,count(*) from emp group by deptno having count(*)>5;

# 查看每个部门工资大于1000的人数,并保留部门人数大于2的分组
select deptno,count(*) from emp where sal > 1000 group by deptno having count(*)>2;
```

 # 过滤分组查询练习

```
# 查询出该公司有哪几种岗位及每个岗位的人数
select job,count(job) from emp group by job;


# 计算每个工作岗位的最高薪水，并且由低到高进行排序
select job,max(sal) as max_sal from emp group by job order by max_sal;

# 计算每个部门的平均薪水
select deptno,avg(sal) from emp group by deptno having avg(sal);

# 计算出不同部门不同岗位的最高薪水
select deptno,sal from emp  group by deptno having max(sal);

# 找出每个工作岗位的最高薪水，除manager 之外
select job,sal from emp where job != "manager" group by job having max(sal);

# 找出每个工作岗位的平均薪水，显示平均薪水大于2000的
select job,avg(sal) from emp group by job having avg(saL) > 2000;
```

4.3 select执行顺序

5. select 字段名	对当前临时表进行整列读取
1. from 表名	将硬盘上的表文件加载到内存
2. where ...	将符合条件的数据行摘取生成一张新的临时表
3. group by ...	根据列中的数据种类，将当前临时表划分成若干个新的临时表
4. having ...	可以过滤掉 group by生成的不符合条件的临时表
6. order by ...	对 select 生成的临时表重新排序，生成新的临时表
7. limit ...	对最终生成的临时表数据行进行截取

以上关键字的顺序不能改变，严格遵守

4.4 正则表达式

regexp 操作符，regexp 操作符后面跟的就是正则表达式，正则表达式的作用是匹配文本，将一个模式(正则表达式)与一个文本串进行比较。

like 和 regexp 的区别

like 匹配整个列，如果被匹配的文本仅在列值中出现(没有配合其他通配符)，like将不会找到它。regexp 在列值内进行匹配，如果被匹配的文本在列值中出现，regexp将会找到它，相应的行将被返回。

4.4.1 匹配单个实例

(1)|:表示匹配其中之一,使用|从功能上类似 or
(2)[]:匹配字符之一，[]是另一种形式的 or 语句。[123]为 [1|2|3]的缩写。
(3)[-]:匹配范围，使用-来定义一个范围。例如:[1-3]、[a-z]。
(4)\\ :转义字符，多数正则表达式使用单个反斜杠作为转义字符，但 MySQL 要求两个反斜杠(MySQL 自己解释一 个，正则表达式库解释另一个)。
(5)匹配字符类:存在找出你自己经常使用的数字、所有字母字符或所有数字字母字符等的匹配。为更方便工作，可以使用预定义的字符集，称为字符类。

类	说明
[[:alnum:]]	任意字母和数字（同 [a-zA-Z0-9]）
[[:alpha:]]	任意字母（同 [a-zA-Z]）
[[:blank:]]	空格和制表（同 [\\t]）
[[:cntrl:]]	ASCII 控制字符（ASCII 0 到 31 和 127）
[[:digit:]]	任意数字（同 [0-9]）
[[:graph:]]	与 [:print:] 相同，但不包括空格
[[:lower:]]	任意小写字母（同 [a-z]）
[[:print:]]	任意可打印字符
[[:punct:]]	既不在 [[:alnum:]] 又不在 [[:cntrl:]] 中的任意字符
[[:space:]]	包括空格在内的任意空白字符（同 [\\f\\n\\r\\t\\v]）
[[:upper:]]	任意大写字母（同 [A-Z]）
[[:xdigit:]]	任意十六进制数字（同 [a-fA-F0-9]）

4.4.2 匹配多个实例

- 常用元字符

元字符	说明
.	匹配任意字符
^	匹配字符串的开始，^ 在 [] 中表示否定
$	匹配字符串的结束

- 重复元字符（修饰前一个字符）

元字符	说明
*	任意个匹配
+	一个或多个匹配（等于{1,}）
?	0 个或 1 个（等于{0,1}）
{n}	指定数目的匹配
{n,}	不少于指定数目的匹配
{n,m}	匹配数目的范围（m 不超过 255）

多表查询

2024年6月28日 16:46

五、多表查询

5.1 连接查询

5.1.1 内连接

多表连接查询就是从多个表中获取数据，若两张表进行连接查询的时候没有任何条件限制，最终的查询结果总数是两张表记录的乘积，该现象称为笛卡儿积现象。

根据连接条件从多个表中查询选择数据，显示这些表中与连接条件相匹配的行，组合成新的记录。

分类:

- 等值连接:连接条件为相等判断的
- 非等值连接:连接条件不为相等判断的
- 自连接:在一个连接查询中，涉及的两个表都是同一张表的查询，自连接是一种特殊的连接查询，它指相互连接的表在物理上为同一张表，在逻辑上分为两张表

```
select * from emp,dept; #56行
select * from emp,dept where emp.deptno = dept.deptno;

select * from emp join dept on emp.deptno = dept.deptno;


# 自连接
select emp.ename,emp_mgr.ename as boss from emp join emp as emp_mgr on emp.mgr = emp_mgr.empno;
```

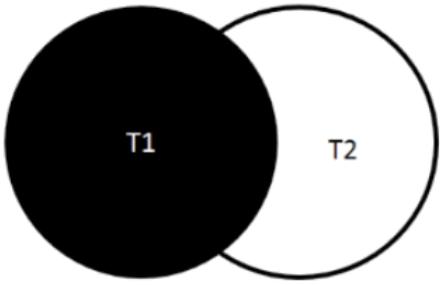
5.1.2 外连接

A 表和 B 表能够完全匹配的记录查询出来之外，将其中一张表的记录无条件的完全查询出来，对方表没有匹配的记录时，会自动模拟出 nu值与之匹配。

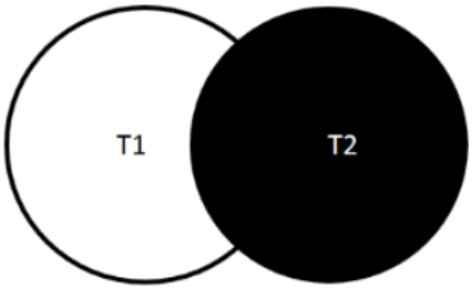
注意:外连接的查询结果条数 >=内连接的查询结果条数

分类:

- **左外连接** left [outer] join 显示左表全部记录，右表满足连接条件的记录



- **右外连接** right [outer] join 显示右表全部记录，左表 满足连接条件的记录




```
# 左外连接
select * from emp left join dept on emp.deptno = dept.deptno;
# 右外连接
select * from emp right join dept on emp.deptno = dept.deptno;
```

5.2 子查询

子查询，嵌套在其他 SQL语句内的查询语句，且必须出现在圆括号内(查询一般指select 语句)。

子查询的结果可以作为外层查询的过滤条件或计算字段。

5.2.1 使用子查询过滤

子查询一般与 [not] in 结合使用，也可也使用其他运算符：> < = !=

```
select 字段1[, 字段2...]
from 表1
where 字段1 [not] in (select 字段1 from 表2 where 条件);
```

```
#查找所在部门名包含 e 的员工信息
select *
from emp
where deptno in (select deptno from dept where dname like "%e%)");
```

all、any、some 是用于条件比较的关键字，用于比较一个值与子查询结果集中的多个值。

all：表示与子查询结果集中的所有值进行比较，需要满足条件的是所有值。

```
# 查询员工所在 部门的部门名称中 有 e 的员工信息
select * from emp where deptno  in (select deptno from dept where dname like "%e%");
```

any、some：any 和 some 含义相同，与子查询结果集中的值比较，有任意一个满足条件即可。

```
# 佣金比有的人工资还高的员工
select *
from emp
where ifnull(comm, 0) > any(select sal from emp);
```

5.2.2 子查询作为计算字段

```sql
select
    字段1,
    字段2,
    ...,
    (select 聚合函数 from 表2 [where 表2.字段1=表1.字段1])
from 表1 [where 条件];
```

```sql
#查询部门名和部门人数
select dname, (select count(*) from emp where
dept.deptno=emp.deptno) as emps
from dept;
```

查询每个部门对应的人数
select dname,(select count(*) from emp where dept.deptno = emp.deptno) from dept;

select完整格式

2024年6月28日　　17:54

7. select 字段名	对临时表进行整列读取
8. distinct 字段名	去除重复数据
1. from 表名	将硬盘上的表文件加载到内存
3. join 表名	连接外表
2. on 条件	对主表进行过滤
4. where ...	将符合条件的数据行摘取生成一张新的临时表
5. group by ...	根据列中的数据种类，将当前临时表划分成若干个新的临时表
6. having ...	可以过滤掉 group by 生成的不符合条件的临时表
9. order by ...	对 select 生成的临时表重新排序，生成新的临时表
10. limit ...	对最终生成的临时表数据行进行截取

以上关键字的顺序不能改变，严格遵守

存储过程和自定义函数

2024年6月28日　　17:52

六、存储过程和自定义函数

6.1 存储过程

存储过程，一组预编译的 SQL语句和流程控制语句，被命名并存储在数据库中。存储过程可以用来封装复杂的数据库操作逻辑，并在需要时进行调用。

6.1.1 使用存储过程

```
#创建存储过程
create procedure 存储过程名()
begin
    -- 存储过程的逻辑代码
    -- 可以包含 SQL 语句、控制结构和变量操作等
end;
#执行存储过程
call 存储过程名();
#删除存储过程
drop procedure [if exists] 存储过程名;
```

6.1.2 使用参数

```
create procedure 存储过程名(
    [in|out|inout] 参数名 1 参数的数据类型,
    [in|out|inout] 参数名 2 参数的数据类型,
        ...
)
begin
    -- 存储过程的逻辑代码
    -- 可以包含 SQL 语句、控制结构和变量操作等
end;
```

参数类型:

- in(默认):输入参数，存储过程的输入值，从外部传递给存储过程，存储过程内部是只读的，不能修改它的值
- out:输出参数，存储过程的返回值，存储过程可以修改它的值并将其返回
- inout:输入和输出参数既可以作为输入值传递给存储过程，也可以由存储过程修改返回

6.1.3 使用变量

声明语句必须放在最上面

```
#定义变量
declare 变量名 变量的数据类型 [default 默认值];
#变量赋值
set 变量名=要赋的值;
#通过查询将结果赋值给变量
select 字段名 into 变量名 from 表名...
```

6.1.4 逻辑语句

- 条件语句（if、case）

```
if condition then
    逻辑代码;
[elseif condition then
    逻辑代码;]
[else
    逻辑代码;]
end if;

case
    when condition1 then
        逻辑代码
    when condition2 then
        逻辑代码
    else
        逻辑代码
end case;
```

- 循环语句（while、repeat）

```
while 循环条件 do
    逻辑代码
end while;

repeat
    逻辑代码
until condition end repeat;
```

```
create procedure mypro(in a int, out b int, inout c int)
begin
  declare max_sal int default 0;
  select a;
  set a = 2;
  select a;

  select b;
  set b = 20;
  select b;

  select c;
  set c = 200;
  select c;

  select max(sal) into max_sal from emp;
  select max_sal;

  if max_sal > 3000 then
    select "high";
  else
    select "low";
  end if;
```

```
    end;

    set @a = 1;
    set @b = 10;
    set @c = 100;

    select @a;
    select @b;
    select @c;

    call mypro(@a,@b,@c);
```

练习：计算前N项和

```
# 前n项和
create procedure mypro1(in a int)
begin
  declare sum_n int default 0;
  declare n int default 1;
  while n <= a do
    set sum_n = sum_n+n;
    set n = n+1;
  end while;
  select sum_n;
end;

set @a = 5;
select @a;


call mypro1(@a);
```

6.1.5 特点

- 优点:
  - 代码复用:存储过程可以被多个应用程序或脚本调用，实现了代码的复用。
  - 提高性能:MySQL将编译后的存储过程放入缓存中。如果应用程序在单个连接中多次使用存储过程，直接使用编译版本。
  - 减少网络流量:存储过程可以一次执行多条 SQL 语句，减少了与数据库的交互次数。
  - 安全控制:存储过程可以对数据库中的数据进行严格的访问控制和权限管理。
  - 数据一致性:存储过程可以实现复杂的数据操作和事务处理，确保数据的一致性和完整性。
- 缺点:
  - 创建和维护成本高:SQL是一种结构化查询语言，难以处理复杂的业务逻辑。
  - 开发调试复杂:需要通过特定的工具和技术进行，不方便调式。
  - 可移植性差:存储过程通常依赖于特定的数据库平台和版本，不同的数据库系统之间。
  - 存储过程的语法和特性可能有差异，导致存储过程的可移植性较差。

6.2 自定义函数

function，可以使用自定义函数来扩展数据库的功能。

```
#创建函数
create function 函数名([参数1 数据类型 [, 参数2 数据类型, ...]])
returns 返回值类型
begin
    函数逻辑代码
end;
#调用函数
select 函数名([参数1, 参数2...]);
#删除函数
drop function [if exists] 函数名;
```

练习

创建函数:

实现检验登录的账号密码是否正常

1.创建账号信息表:(id,name,password)并存入数据

2.创建一个函数:传入两个参数

3.函数返回结果:(1)用户不存在(2)密码错误(3)登录成功

```
# 创建账号信息表
create table if not exists userdata(
  id int primary key auto_increment,
  username varchar(20) not null,
  pwd varchar(20) not null
)

# 插入数据
insert into userdata(username,pwd)
values
("weihong1","weihong1223"),
("weihong2","weihong1224"),
("weihong3","weihong1225");


# 函数
create procedure mypro2(in a varchar(20), in b varchar(20))
begin
  declare uname varchar(20);
  declare upwd varchar(20);
  select username into uname from userdata where username = a;
  select uname;
  if uname = null then
    select "用户不存在";
  else
    select pwd into upwd from userdata where pwd = b && username = a;
    if upwd = null then
      select "密码错误！";
    elseif upwd = b then
      select "登陆成功！";
    end if;
  end if;
end;



drop procedure if exists mypro2;

# 设置变量
set @uname = "weihong4";
```

```
set @pwd = "weihong1224";
select @uname;
select @pwd;

# 调用函数
call mypro2(@uname,@pwd);
```