数据库

DB（DataBase）数据库

DBA（DataBase Admin）数据库管理员

DBMS（DataBase Management System）数据库管理系统

SQL（Structure Query Language）结构化查询语言

ISO 国际化标准组织

DDL（Data Definition Language）数据定义语言 用来定义数据库对象

DML（Data Manipulation Language）数据操作语言 用来定义数据库记录

DCL（Data Control Language）数据控制语言 用来定义访问权限和安全级别

DQL（Data Query Language）数据查询语言 用来查询记录

关键字：select

select 列名 from 表名 [where🡪group by🡪having🡪order by🡪limit]

group by grouping\_columms 对结果分组

having condition 分组后的行条件

order by sorting\_columms 对结果分组

limit offiset\_start,row\_count 结果限定

SELECT \* FROM stu;

-- 条件查询

-- 1 性别为female并且年龄小于50

SELECT \* FROM stu WHERE gender='female' AND age<50;

-- 2 查询学号为1或者姓名为ss

SELECT \* FROM stu WHERE sid='1' OR sname='ss'

-- 3 查询学号为1 2 3的学生

SELECT \* FROM stu WHERE sid IN('1','3');

SELECT \* FROM stu WHERE sid NOT IN('1');

-- 5 年龄在20-40

SELECT \* FROM stu WHERE age>=20 AND age<=40;

SELECT \* FROM stu WHERE age BETWEEN 20 AND 40;

-- 6 查询性别非male的学生

SELECT \* FROM stu WHERE gender!='male';

SELECT \* FROM stu WHERE gender<>'male';

SELECT \* FROM stu WHERE NOT gender='male';

-- 7 查询姓名不为ss的学生

SELECT \* FROM stu WHERE sname IS NOT NULL;

SELECT \* FROM stu WHERE NOT sname='ss';

-- 模糊查询

-- 1 查询姓名为4个字母组成的记录（4\_）

SELECT \* FROM emp WHERE ename LIKE '\_\_\_\_';

-- 2 查询4个字母组成 最后一个是D

SELECT \* FROM emp WHERE ename LIKE '\_\_\_D';

-- 3 查询名字中以F开头的名字

SELECT \* FROM emp WHERE ename LIKE 'F%';

-- 4 查询姓名中包含N的名字

SELECT \* FROM emp WHERE ename LIKE '%N%';

-- 5 查询姓名中包含N但是是店员

SELECT \* FROM emp WHERE ename LIKE '%N%' AND job='销售';

-- 字段控制查询（去重）

SELECT DISTINCT sal FROM emp;

-- 1 查询表中 月薪和奖金之和(数据库中与null相加 结果还是null)

SELECT \*,sal+IFNULL(comm,0) FROM emp;

-- 如果列值为null替换成0

-- 给列添加别名 as(可省略)

SELECT \*,sal+IFNULL(comm,0) AS total FROM emp;

-- order by 排序（默认升序 ASC升序 DESC降序）

SELECT \* FROM emp ORDER BY mgr;

SELECT \* FROM emp ORDER BY ename DESC;

-- 查询所有员工信息 按照comm排序 如果comm相同按照empno查询

SELECT \* FROM emp ORDER BY comm DESC,empno ASC;

-- 查询二月份入职的人

SELECT \* FROM emp WHERE hiredate LIKE '%-02-%';

-- 聚合函数（sum,avg,max,min,count）

-- count（）统计制定列不为null的记录行数

-- max()计算指定列的最大值

-- min()计算指定列的最小值

-- sum()计算指定列的数值和

-- avg()计算指定列的平均值

-- 1 count查询表中记录数

SELECT COUNT(\*) AS ss FROM emp;

-- 2 查询有奖金人数

SELECT COUNT(comm) FROM emp;

-- 3 查询月薪大于2500的人数

SELECT COUNT(\*) FROM emp WHERE sal>2500;

-- 4 查询月薪和奖金只喝大于2500的人数

SELECT COUNT(\*) FROM emp WHERE sal+IFNULL(comm,0)>2500;

-- 5 查询有奖金是领导的人数

SELECT COUNT(comm),COUNT(mgr) FROM emp;

-- 查询所有员工工资之和，最高工资，最低工资,平均工资

SELECT SUM(sal),MAX(sal),MIN(sal),AVG(sal) FROM emp;

-- 分组 group by

-- 1 查询每个部门的部门编号 和 每个部门的工资和

SELECT deptno ,SUM(sal) FROM emp GROUP BY deptno;

-- 2 查询部门编号和部门人数

SELECT deptno ,COUNT(deptno) FROM emp GROUP BY deptno;

-- 3 查询每个部门中工资大于1500的人数

SELECT deptno ,COUNT(deptno) FROM emp WHERE sal>1500 GROUP BY deptno ORDER BY COUNT(deptno);

-- 4 查询工资总和大于9000的部门编号以及工资和

SELECT deptno,SUM(sal) FROM emp GROUP BY deptno HAVING SUM(sal)>9000;

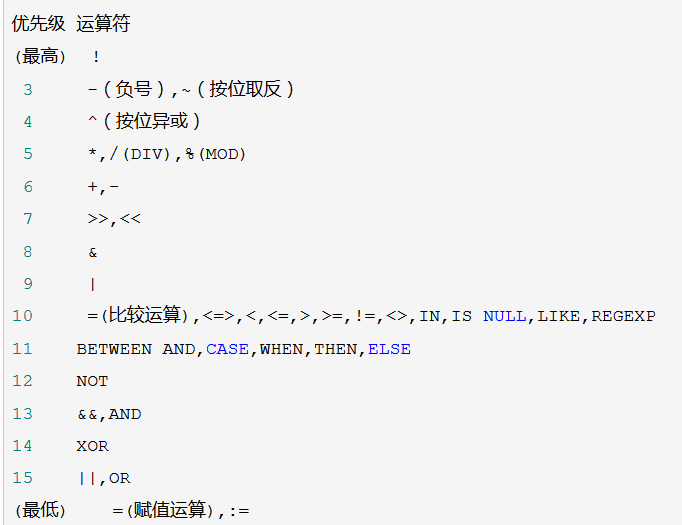
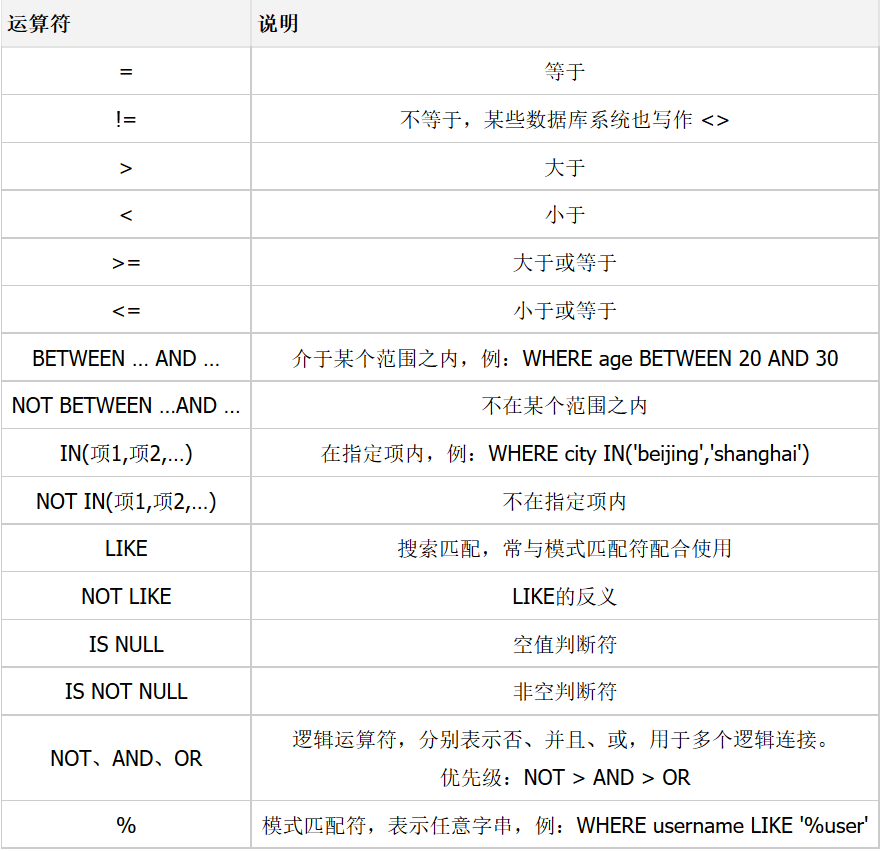
-- limit start(0开始),number

-- 1 查询五行记录

SELECT \* FROM emp LIMIT 3,10;

查询语句书写顺序

select from where group by having order by limit



约束类型

主键约束（primary key）

唯一约束（unique）数据不能重复，可以为空

自动增长列（auto\_increment）只能是整数类型

域完整性：限制此单元格的数据正确

外键（foreign key）

创建外键语法：

[constraint 外键关系名]foreign key (从表字段) references 主表（主表字段）

修改外键

alter table emp add foreign key（deptno）references dept(deptno)

删除外键

alter table 从表名 drop foreign key 外键关系名

多表联合查询

1. 合并结果集（union all（不去重）、union（去重））

被合并的两个表必须列和数据类型都一样

1. 连接查询（笛卡尔积）

join（多表连接）

内连接[INNER] join…on

查询结果必须满足条件

外连接[OUTER] join…on

外链接不但返回符合连接和查询条件的数据行，还返回不符合条件的一些行

左外连接left [OUTER] join

返回左表全部信息加上右表符合的信息

右外链接right[OUTER]join

返回右表全部信息加上左表符合的信息

全外链接full join（mysql不支持）

自然连接

natural join

natural left join

子查询（嵌套查询）

子查询出现位置

where后：作为查询条件的一部分

from后：作为表

all 、 any 、in、exists

数学函数

floor（float） 向下取整

rand（） 0-1随机数

round（float）四舍五入

round（float，保留小数位数）

日期和时间函数

curdate()当前日期

curtime()当前时间

now()当前时间和日期

year/month/day（date）获取指定月份

monthname/dayname（date）英文星期

dayofweek(date)星期日开始

week/hour/minute/second()

addtime(date1,date2) date1+date2

datediff(date1,date2) 相差天数

字符串函数

lower/upper/length/concat(连接)/

replace(s,search\_s,replace\_s)

substring(str,start,[length])

instr(str,substr) 返回substr出现的位置，没有返回0

一.数据库概述

什么是数据库:

DataBase DB 是指长期保存在计算机的存储设备上的,按照一定规则存储的数据集合

这个数据集合 可以让用户 或者 应用共享其中的数据

DBA 数据库管理员 BataBase Admin

数据库管理系统(DataBase Management System,DBMS)是一种操作和管理数据库的大型软件,

用于建立/使用和维护数据库 对数据库进行统一的管理和控制.以保证数据库的安全性和完整性,

用户通过数据库管理系统访问数据库中的数据,对其进行相对安全的操作.

二.常用的数据库

Access office里的

oracle 收钱 很大

db2 ibm的

sql server 微软的

mysql

LAMP(Linux Apache Mysql Php)

三.SQL概述

SQL 结构化查询语言(Structure Query Language)

SQL 被国际化标准组织(ISO)采纳为关系型数据库语言的国际标准

各个数据库厂商都支持SQL标准 -->普通话

各个数据库厂商在标准的基础上做了自己的扩展-->方言

四.SQL的分类

DDL\*\*(Data Definition Language)数据定义语言 用来定义数据库对象:库,表,列等

DML\*\*(Data Manipulation Language)数据操作语言,用来定义数据库记录(数据):增删改

DCL(Data Control Language)数据控制语言,用来定义访问权限和安全级别

DQL\*\*\*\*\*(Data Query Language)数据查询语言,用来查询记录(数据):查询

注意:

1.sql语句以; 英文分号作为结尾 所有的符号必须是英文的

2.sql大小写问题(表名 列名)

windows 不区分大小写

unix/linux下区分大小写

4.1 DDL 操作数据库/表/列等

数据定义语言 定义数据库/表/索引等对象 关键字 create / alter / drop

database 用来存放表的容器

对数据库的操作:

4.1.1 创建

cmd 进入数据库 mysql -u root -p

create database 数据库名;

create database 数据库名 character set utf8;//设置数据库的字符集

4.1.2 查询

查看当前数据库服务器中所有的数据库

show databases;

查看前面创建的数据库的定义信息

show create database 数据库名;

4.1.3 删除

drop database 数据库名;

4.1.4 修改

查询服务器中的数据库 并把编码集改成gbk

alter database dbcq character set gbk;

4.1.5 其他

查看当前正在使用的数据库

select database();

切换数据库

use 数据库名;

对表的操作:

4.1.6 创建 create

create table 表名(

字段1 字段类型 约束条件,

字段2 字段类型 约束条件,

....

字段n 字段类型 约束条件

);

约束条件 可有可无

mysql中常用的数据类型

int 整数

double 浮点 例如 double(5,2) 表示最多5位 其中必须有2位是小数 最大值 999.99

decimal 从mysql 5.1之后引入 浮点

decimal(M,D) decimal(5,2) 最大值是9999.99

float占4个字节 double占8个字节 decimal(M,D)占M+2个字节

和金融类有关的 钱..用这个

char 固定长度的字符串类型 char(10) 'aaa ' 占10位

varchar 可变长度的字符串类型 varchar(10) 'aaa'占3位

text 字符串类型 超长

blob 字节类型

date 日期类型 格式为 yyyy-MM-dd

time 时间类型 格式为 hh:mm:ss

timestamp 时间戳类型 yyyy-MM-dd hh:mm:ss 会自动赋值

datetime:日期时间类型 yyyy-MM-dd hh:mm:ss

create table stu(sid char(6),sname varchar(50),age int,gender varchar(50));

查看数据库中的表

show tables;

查看表结构

desc 表名;

查看创建表的语句

show create table 表名;

4.1.7 修改 alter

修改表的字符集为gbk

alter table stu character set gbk;

change 修改列名 用来给字段重命名 不能修改字段类型和约束条件

alter table stu change 旧字段名 新字段名 新的数据类型;

modify 修改列属性 不能用来给字段重命名 只能修改字段类型和约束条件

修改表中字段为stuname的列 让它的长度变为60

alter table 表名 modify 字段名 字段类型;

alter table stu modify stuname varchar(60);

新增字段

add

alter table 表名 add 新增的字段名 字段类型;

alter table stu add cname varchar(10);

删除字段 一次只能删除一列

alter table 表名 drop 字段名

修改表名

alter table 原表名 rename 新表名

rename table 原表名 to 新表名

4.1.8 删除表 drop

drop table 表名;

4.2 DML\*\*(Data Manipulation Language)数据操作语言,用来定义数据库记录(数据):增删改

不要与DDL混淆了 增删改是对表中的数据进行的操作

insert(增加) update(修改) delete(删除)

注意:

在mysql中 字符串类型和日期类型 都要用单引号 或者 双引号引起来

空值 null

4.2.1 insert 增加操作(插入操作)

语法:

insert into 表名[(列名1,列名2....)] values(列值1,列值2....);

注意: 列名 和 列值的类型/个数/顺序要一一对应

[]列名可不写方 但是存储的数据要和表结构完全一致

可以将列名理解为java中的形参 把列值理解为实参

如果插入空值 请使用null

日期类型 可以使用 - , .分割 或者不分割都可以存入

同时插入多条记录 (值1,值2...),(值1,值2...),(值1,值2...) 用逗号分隔

4.2.2 update 修改操作

语法:

update 表名 set 列名1=列值1,列名2=列值2... [where 列名=列值]

[where子句] 修改哪个符合条件的

可以不写 如果不写 就是修改当前表下所有记录

练习:

将所有员工薪水修改为5000元

将学生信息表中 性别为男的学生 成绩 在原有基础上+5分

update stu set score=score+5 where gender='male';

4.2.3 delete 删除操作

语法:

delete from 表名 [where 列名=值]

删除表中的所有记录

delete from 表名

删除表中的数据 表结构还在 删除后 数据可找回

truncate table 表名

删除是直接把表drop掉 然后创建一个同样的新表

删除的数据不能找回 执行速度比delete快

删除名字为zhangsan的学生信息

delete from stu where sname='zhangsan';

删除年龄小于20的学生

delete from stu where age<20;

4.3 DQL 数据查询语言(重要)

数据库执行DQL语句 不会对数据进行改变 而是让数据库发送结果集给客户端

查询返回的结果是一张虚拟表

关键字:select

语法 select 列名 from 表名 [where--> group by --> having --> order by --> limit]

\* 是查询所有列 实际开发中尽量少使用\* 查询所有列 只要获取想要的列就可以了

select 列名 /\*要查询的列名称\*/

from table\_list /\*要查询的表名称\*/

where condition /\*行条件\*/

group by grouping\_columms /\*对结果分组\*/

having condition /\*分组后的行条件\*/

order by sorting\_columms /\*对结果分组\*/

limit offiset\_start,row\_count /\*结果限定\*/

基础查询

查询所有列 select \* from 表名

查询指定列 select sid,sname from 表名

条件查询

条件查询 是在查询时 给出where子句 在where子句中可以一些关键字来用作条件匹配

符合条件的查询出来

符号:

> < != <> = >= <=

between...and... 包含边界 等同于 >= <=

in 在xxx范围中

is null 是空的

and 和

or 或者

not 不

/\*条件查询\*/

/\*1 性别为 female 并且 年龄小于 50 岁\*/

SELECT \* FROM stu WHERE gender='female' AND age<50;

/\*2 查询学号为 s\_1001 或者 姓名为 chenEr\*/

SELECT \* FROM stu WHERE sid='s\_1001' OR sname='chenEr';

/\*3 查询学号为 s\_1002,s\_1005,s\_1004的学生\*/

SELECT \* FROM stu WHERE sid IN ('s\_1002','s\_1004','s\_1005');

SELECT \* FROM stu WHERE sid NOT IN ('s\_1002','s\_1004','s\_1005');

/\*4.性别为null的记录\*/

SELECT \* FROM stu WHERE gender IS NULL;

/\*5.年龄在 20 到 40 之间的所有学生\*/

SELECT \* FROM stu WHERE age>=20 AND age<=40;

SELECT \* FROM stu WHERE age BETWEEN 20 AND 40;

/\*注意 实际mysql between and 大于等于 小于等于的意思\*/

/\*6.查询性别非male的学生\*/

SELECT \* FROM stu WHERE gender!='male';

SELECT \* FROM stu WHERE gender<>'male';

SELECT \* FROM stu WHERE NOT gender='male';

/\*7.查询姓名不为null的学生记录\*/

SELECT \* FROM stu WHERE sid IS NOT NULL;

SELECT \* FROM stu WHERE NOT sid IS NULL;

/\*模糊查询\*/

like

通配符:

\_ 任意一个字母

% 任意0-n个字母

/\*1.查询姓名由三个字组成的老师记录\*/

SELECT \* FROM teacher WHERE NAME LIKE '\_\_\_';

/\*2.查询 由四个字母组成 最后一个是i\*/

SELECT \* FROM stu WHERE sname LIKE '\_\_\_i';

/\*3.查询 姓名中以李开头的老师\*/

SELECT \* FROM teacher WHERE NAME LIKE '李%';

/\*4.查询 姓名包含五的老师\*/

SELECT \* FROM teacher WHERE NAME LIKE '%五%';

/\*字段控制查询\*/

1.去除重复记录

distinct 去除两行以上记录中数据相同的部分

SELECT DISTINCT sal FROM emp;

/\*1.查看emp表 月薪和奖金之和\*/

SELECT \*,sal+comm FROM emp;

/\*在数据库里 任何东西与null相加 结果还是null 所以结果可能会出现null的情况

将null转换成0 IFNULL(列名,值) 如果列的值是null 就替换为0

\*/

SELECT \*,sal+IFNULL(comm,0) FROM emp;

/\*给列添加别名 AS 给列起别名时 as 可省略 \*/

SELECT \*,sal+IFNULL(comm,0) AS total FROM emp;

/\*排序 order by... 根据xx来排序 默认排序是升序的\*/

SELECT \* FROM stu ORDER BY age DESC;

/\*ASC 升序 DESC 降序\*/

/\*ASC 升序 DESC 降序\*/

/\*查询所有员工信息 按照月薪降序排序 如果月薪相同 按照编号升序排序\*/

SELECT \* FROM emp ORDER BY sal DESC,empno ASC;

/\*聚合函数\*/

sum avg max min count

用来做纵向运算的函数

count() 统计制定列不为null的记录行数

max() 计算指定列的最大值 如果指定列是字符串 那么使用字符串排序运算

min() 计算指定列的最小值

sum() 计算指定列的数值和

avg() 计算指定列的平均值

聚合函数不能放在where子句之后

聚合函数 是针对结果集进行的 where条件并不是在查询出结果之后运行的

所以放在where语句中 会出错

sql语句的执行过程

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

select \* from stu where....

/\*count 查询表中记录数\*/

SELECT COUNT(\*) FROM emp;

/\*1.查询emp 表中有奖金的人数 非null 的行数\*/

SELECT COUNT(comm) FROM emp;

/\*2.查询表中月薪大于2500的人数\*/

SELECT COUNT(\*) FROM emp WHERE sal>2500;

/\*3.统计月薪与奖金之和大于2500的人数\*/

SELECT COUNT(\*) FROM emp WHERE sal+IFNULL(comm,0)>2500;

/\*4.查询有奖金的人数 以及有领导的人数\*/

SELECT COUNT(comm),COUNT(mgr) FROM emp;

/\*sum 和 avg\*/

/\*1.查询所有员工的薪资和\*/

SELECT SUM(sal) FROM emp;

/\*max 和 min\*/

/\*1.查询最高工资 和 最低工资\*/

SELECT MAX(sal),MIN(sal) FROM emp;

/\*分组查询\*/

group by 当需要分组查询的时候 使用group by 子句

/\*分组 group by\*/

/\*1.查询每个部门的部门编号 和 每个部门的工资和\*/

SELECT deptno,SUM(sal) FROM emp GROUP BY deptno;

SELECT \* FROM stu GROUP BY gender;

/\*2.查询每个部门的部门编号 以及每个部门的人数\*/

SELECT deptno,COUNT(\*) FROM emp GROUP BY deptno;

/\*3.查询每个部门的部门编号 以及 每个部门工资大于1500的人数\*/

SELECT deptno,COUNT(\*) FROM emp WHERE sal>1500 GROUP BY deptno;

where 先过滤 后分组

having 先分组 后过滤

/\*having\*/

/\*4.查询工资总和大于9000的部门编号以及工资和\*/

SELECT deptno,SUM(sal) FROM emp GROUP BY deptno HAVING SUM(sal)>9000;;

having与where的区别

1.having是在分组后对数据进行过滤

where是在分组前对数据进行过滤

2.having后面可以使用聚合函数(分组函数 统计函数)

where后面不可以

where 是对分组前记录的条件,如果某行记录没有满足where子句的条件,那么这行记录就不会参加分组

having 是对分组后数据的约束筛选

/\*limit\*/ 用来限定查询结果的起始行,以及总行数

/\*1查询5行记录 [offset] 从第几行开始 起始行是0 0代表第一行 rows 取几行\*/

SELECT \* FROM emp LIMIT 0,5;

select \* from emp 5;

/\*2查询10行记录 其实行从第四行开始 \*/

SELECT \* FROM emp LIMIT 3,10;

分页查询

如果一页记录为10条 希望查看第3页的记录应该怎么查呢?

先规定 每页显示条数 pageSize = 10;

所有的记录 count

共分几页?

page = count%pageSize==0 ? count/pageSize : count/PageSize+1;

第一页的记录 起始行为 0 一共查询10行 0-9

第二页 10-19

第三页 20-29

每页的起始行 (页码-1)\*pageSize

查询语句书写顺序:

select -- from -- where -- group by -- having -- order by -- limit

查询语句执行顺序:

from -- where -- group by -- having -- select -- order by – limit

一 约束条件

为了保证数据的完整性 可以给列加上约束条件

确保数据的完整性 = 在创建表的时候给表中添加约束

完整性的分类:

实体完整性

域完整性

引用完整性

1.实体完整性

实体:即表中的一行(一条记录)代表一个实体(entity)

实体完整性的作用:标识每一行数据不重复

约束类型:

主键约束(primary key)

唯一约束(unique)

自动增长列(auto\_increment)

1.1主键约束

每一个表中要有一个主键(可以是一列 也可以是多列联合组成)

特点:数据唯一 且不能为null

/\*主键约束

第一种添加方式 在创建表的时候 直接添加主键约束\*/

CREATE TABLE a(

id INT PRIMARY KEY,

NAME VARCHAR(10)

);

INSERT INTO a VALUES(1,'abc');

/\*主键约束

第二种添加方式 后补主键约束\*/

CREATE TABLE b(

id INT,

NAME VARCHAR(10)

);

ALTER TABLE b ADD PRIMARY KEY(id); -->括号里放的是要设置为主键的列名

/\*联合主键\*/

CREATE TABLE c(

id INT,

NAME VARCHAR(10),

PRIMARY KEY(id,NAME);-->括号里放的是要设置为主键的列名

);

/\*删除主键约束?\*/

ALTER TABLE 表名 DROP PRIMARY KEY;

1.2 唯一约束(unique)

特点:数据不能重复 值可以为null

/\*unique唯一约束\*/

CREATE TABLE c(

id INT PRIMARY KEY,

NAME VARCHAR(100) UNIQUE

);

ALTER TABLE 表名 ADD UNIQUE(列名);

1.3 自动增长列(auto\_increment)

mysql -->auto\_increment

sqlserver --> idenitiy

oracle --> sequence

注意:给主键设置自动增长的时候 列只能是整数类型

CREATE TABLE d(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(10)

);

可以设置自增起始值

2.域完整性

域完整性的作用:限制此单元格的数据正确,不对照此列的其他单元格

域代表当前单元格

域完整性约束:

数据类型 非空约束(not null) 默认值约束(default)

check约束 (mysql不支持) check(sex="男" or sex="女");

2.1数据类型(数值类型,日期类型,字符串类型)

2.2非空约束 not null

ALTER TABLE 表名 MODIFY 列名 VARCHAR(20) NOT NULL;

2.3默认值约束(default)

/\*默认约束 第一种添加方式\*/

CREATE TABLE e(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(10),

age INT,

sex VARCHAR(10) DEFAULT '男'

);

/\*默认约束 第二种添加方式\*/

ALTER TABLE 表名 ALTER 列名 SET DEFAULT 默认值;

3.引用完整性

外键约束: foreign key

用来约束表与表之间的关系

建立外键关联 首先要确定 主 从关系

外键一定是在从表中创建 从而找到与主表之间的联系

例子: 国家表 --> 人民表 国家表国家名字-->人民表的国家名字

部门表 --> 员工表

学生表 --> 分数表

外键必须满足的条件:

1.至少涉及到两张表

2.建立外键关系 对应列 必须具有相似的数据类型

3.必须建立索引 pk fk

注意: 外键可以有重复的 可以是空值

创建外键的两种方式

方式一.在创建表的时候 添加外键

方式二.表已经创建好了 后续修改表的结构来添加外键

方式一

语法 [constraint 外键关系名] foreign key (从表的字段) references 主表名(主表的字段)

[constraint 外键关系名] 可以不写 就是一个外键约束的名字 如果不写 系统会自己给

foreign key 将从表中的字段作为外键字段

references 映射到主表的字段

CREATE TABLE stu2(

sid INT PRIMARY KEY,

sname VARCHAR(10) NOT NULL,

sex VARCHAR(10) DEFAULT '男'

);

CREATE TABLE score(

id INT PRIMARY KEY,

score INT,

ssid INT,

CONSTRAINT fk\_stu2\_score FOREIGN KEY (ssid) REFERENCES stu2(sid)

);

方式二 表已经创建好了

alter table dept add primary key (deptno);

alter table emp add foreign key (deptno) references dept(deptno);

DELETE FROM dept WHERE deptno = 20;

注意:

如果两张表有了外键关联 并且从表中正在使用主表数据

那么就不能随便删除主表中的数据

必须先删除从表中的数据 再删除主表的数据

SHOW CREATE TABLE 从表名;//查询建表语句

删除外键

ALTER TABLE 从表名 DROP FOREIGN KEY 外键关系名;

二 表与表之间的关系

三种关系

1.一对一 1vs1

2.一对多(多对一) 1 vs N

3.多对多 N vs N

三 多表联合查询\*\*\*\*

多表连接简介

在关系型数据库中 一个查询往往会涉及到多表,多张表如何进行联合查询 就成为了学习sql 查询语言的关键

如果单纯的只是将数据存放在一张表中 那么这个表存在大量冗余数据 这种数据库设计是不合理的

1.合并结果集

union

union all

作用:合并结果集 把两个select语句的查询结果合并到一起

CREATE TABLE a(

aname VARCHAR(10),

aage INT

);

CREATE TABLE b(

bNAME VARCHAR(10),

bage INT

);

INSERT INTO a VALUES('a',1),('b',2),('c',3);

INSERT INTO b VALUES('d',1),('e',2),('c',3);

/\*union\*/

SELECT \* FROM a UNION SELECT \* FROM b; // 去除重复数据

SELECT \* FROM a UNION ALL SELECT \* FROM b; //不去除重复数据

要求:被合并的两个结果 :列数/列类型 必须相同

2.连接查询(重要)

连接查询 就是求多个表的乘积 例如 a表连接b表 那么查询出的结果就是 a表\*b表

连接查询 -->笛卡尔积(在sql中的实现方式交叉连接 cross join)

所有的连接方式 都会先生成临时笛卡尔积表 然后加上限制条件 才会有实际查询意义

join 就是把多个表连成一个表的手段

多表查询 分为 内/外连接

内连接 [INNER] join...on

外连接 [OUTER] join...on

左外连接 Left[OUTER] join

右外连接 Right[OUTER] join

全外连接 Full join (Mysql 不支持)

自然连接 Natural join

内连接

SELECT \* FROM 表1,表2 WHERE 表1.相同列 = 表2.相同列

可以给表设置别名 然后在引用列的时候 使用别名调用即可

SELECT e.ename,e.job,d.dname,d.loc FROM emp e,dept d WHERE e.deptno = d.deptno

以上连接语句就是内连接,但是不是sql标准中查询方法,可以理解为方言

有两种 隐式(没有 inner join 上面那种)和显式(有 inner join)

隐式:

SELECT e.ename,e.job,d.dname,d.loc FROM emp e,dept d WHERE e.deptno = d.deptno

显式: [inner]可省略 不使用 where 使用 on

SELECT e.ename,e.job,d.dname,d.loc FROM emp e [INNER] JOIN dept d ON e.deptno = d.deptno

特点: 查询结果必须满足条件

外连接

外连接不但返回符合连接和查询条件的数据行,还返回不符合条件的一些行

外连接分为三种

左外连接 Left[OUTER] join

右外连接 Right[OUTER] join

全外连接 Full join (Mysql 不支持)

左外连接 Left[OUTER] join

返回左表中不符合连接条件

右外连接 Right[OUTER] join

返回右表中不符合连接条件

SELECT \* FROM emp e LEFT [OUTER] JOIN dept d ON e.deptno = d.deptno;

左连接是先查询出左表(即以左表为主) 然后查询右表,右表中满足条件的显示 不满足条件的显示为null

SELECT \* FROM emp e RIGHT [OUTER] JOIN dept d ON e.deptno = d.deptno;

右连接是先查询出右表中所有记录,然后查询左表,左表中满足条件显示 不满足条件的显示为null

/\*练习 查询出 员工姓名 工作 部门 部门所在地 当部门为采购部的不显示 当名字为smith的不显示\*/

SELECT \* FROM emp LEFT JOIN dept ON emp.deptno = dept.deptno AND dept.dname !='采购部'

WHERE emp.ename !='smith'

/\*

1.笛卡尔积 select \* from emp cross join dept

2.应用 on 筛选器 emp.deptno = dept.deptno AND dept.dname !='采购部'

3.添加外部行 outer join起作用 如果是left join会以左表为保留表

如果是 right join 看右侧表

所谓的外部行 就是保留表中的行 即第二步on过滤掉了一些 在这里会添加上

4.where过滤器 过滤的是left join 左表部分的数据

\*/

总结:

1.如果 sql 用的是left join,on后面的条件 对left表没有作用,只对right表起过滤作用

where 语句可以对left的表有过滤作用

2.如果 sql 用的是right join,on后面的条件对right表没有作用,只对left表起过滤作用

where 语句可以对right的表右过滤作用

数据库在通过连接两张或多张表返回数据的时,都会生成一张中间的临时表,然后再将这张临时表的数据返回给用户

on条件与where条件的区别

在使用left join时,on和where区别如下

1.on条件是在 生成临时表 时 使用的条件 它不管on 的条件是否为真 都会返回左边表的记录

2.where条件是在 生成临时表 后 ,再对临时表进行过滤 如果条件不为真 就全部过滤

自然连接 Natural join

自然连接无需你去给出主外键等式 它会自动找到这一等式

两张连接的表中 名称 和 类型 完全一致的列 作为条件

SELECT \* FROM emp NATURAL JOIN dept;

SELECT \* FROM emp NATURAL LEFT JOIN dept;

SELECT \* FROM emp NATURAL RIGHT JOIN dept;

总结:

连接查询时sql查询的核心 按照实际需求来选择对应的连接类型,如果选择不当

非但不能提高效率 反而会带来一些逻辑错误或者性能下降

选择依据:

1. 查两表关联列相等的数据用 内连接

2. left表是right表的子集 用 右外连接

3. right表是left表的子集 用 左外连接

4. right和left彼此之间有交集 但是互不为子集 使用 用全外链接

子查询(非常重要)

一个select语句中 包含另一个 完整的 select语句

子查询 就是 嵌套查询

子查询出现的位置:

where 后 :作为查询条件的一部分

from 后 : 作为表

当子查询出现在where后 作为条件时,还可以使用如下关键字(很少用)

any任意一个 --->min

all全部 --->max

子查询结果集的形式

> 单行单列 (用于条件)

> 单行多列 (用于条件)

> 多行单列 (用于条件)

> 多行多列 (表)

1.子查询形式为 单行单列

2.子查询形式为 单行多列

3.子查询形式为 多行单列

多行子查询:是指返回多行数据 在where使用多行子查询时,要用 in/all/any来连接

/\*练习 工资高于 WARD 的员工\*/

SELECT sal FROM emp WHERE ename='ward';

SELECT ename,sal FROM emp WHERE sal > (SELECT sal FROM emp WHERE ename='ward');

/\*练习 查询与 SCOTT 同一部门的员工\*/

SELECT ename FROM emp WHERE deptno = (SELECT deptno FROM emp WHERE ename ='scott' );

/\*练习 查询工资比 7654高 与 7788从事相同工作的全部员工信息\*/

SELECT \* FROM emp WHERE sal>(SELECT sal FROM emp WHERE empno=7654) AND

job=(SELECT job FROM emp WHERE empno = 7788)

/\*练习 查询工作和工资与MARTIN 完全相同的员工信息\*/

SELECT job,sal FROM emp WHERE ename = 'MARTIN';

SELECT \* FROM emp WHERE (job,sal) IN (SELECT job,sal FROM emp WHERE ename = 'MARTIN');

/\*练习 查询有2个以上直接下属的员工信息\*/

SELECT mgr FROM emp GROUP BY mgr HAVING COUNT(mgr)>=2;

SELECT \* FROM emp WHERE empno IN (SELECT mgr FROM emp GROUP BY mgr HAVING COUNT(mgr)>=2);

/\*练习 工资高于 30 号部门所有人的员工信息\*/

SELECT MAX(sal) FROM emp WHERE deptno = 30;

SELECT \* FROM emp WHERE sal>(SELECT MAX(sal) FROM emp WHERE deptno = 30);

SELECT sal FROM emp WHERE deptno = 30;

SELECT \* FROM emp WHERE sal> ALL(SELECT sal FROM emp WHERE deptno = 30);

in 和 exists 区别

/\*会员表 订单表 1 vs N\*/

/\*in\*/

SELECT user\_id FROM ordertable

SELECT \* FROM usertable WHERE id IN(SELECT user\_id FROM ordertable);

/\*

执行流程

1.SELECT user\_id FROM ordertable

2.查询到的结果和原有usertable 表 做一个 笛卡尔积

3.where id in 条件 将结果进行筛选

usertable.id 和 ordettable.user\_id 是否相等 相等保留 不想等删除

\*/

/\*exists\*/

SELECT \* FROM usertable WHERE EXISTS

(SELECT user\_id FROM ordertable WHERE usertable.id = ordertable.user\_id );

/\*

执行流程

1.SELECT \* FROM usertable

2.根据表的每一条记录 执行下面的语句 依次去判断where后面的条件

EXISTS

(SELECT user\_id FROM ordertable WHERE usertable.id = ordertable.user\_id );

如果匹配 返回true 否则返回false 当匹配结束 只保留返回true的数据

\*/

select \* from 表1 where id in (select id from 表2);

查询过程 相当于一个双重for循环

for(int i=0 ;i<表1.length;i++){

for(int j=0 ;j<表2.length;j++){

if(表1[i].id = 表2[j].id){

判断 数据是否存在

break;

}

}

}

如果 表1有1000条记录 表2有10000条记录 最多遍历 1000\*10000

如果 表1有10000条记录 表2有100条记录 最多遍历 10000\*100

select \* from 表1 where exists(select id from 表2 where 表1.id = 表2.id);

exists

for(int i=0;i<表1.length;i++){

if(exists(表1[i].id)){

//select id from 表2 where 表1.id = 表2.id

//判断是否有记录返回

}

}

如果 表1有1000条记录 表2有10000条记录 最多跑了1000次去判断

如果 表1有1000条记录 表2有10000000条记录 exists 1000次

exists 执行了 表1.length次

结论:

in() 适合 表2比表1数据小的情况下

exists() 适合 表2比表1数据大的情况下

当 表1和表2的数据一样大时, in 与 exists效果差不多 可以任意选择

4 多行多列 (表)

form后面

/\*查询员工编号为 7788 的员工姓名 部门 部门地址\*/

SELECT e.ename,d.dname,d.loc

FROM emp e,dept d

WHERE e.deptno = d.deptno AND empno = 7788;

SELECT e.ename,d.dname,d.loc

FROM emp e,(SELECT deptno,dname,loc FROM dept) d

WHERE e.deptno = d.deptno AND empno = 7788;

/\*求7369员工 姓名 经理姓名 经理编号\*/

SELECT e1.empno,e1.ename,e2.ename,e2.empno

FROM emp e1,(SELECT empno,ename FROM emp) e2

WHERE e1.mgr = e2.empno AND e1.empno = 7369;

一函数

1.聚合函数

sum()

avg()

count()

min()

max()

2.数学函数

主要用于处理数字,包括整型,浮点型等

/\*数学函数\*/

SELECT FLOOR(1.5) /\*返回小于或等于参数的最大整数\*/

SELECT RAND()/\*返回0>1的随机数\*/

SELECT ROUND(RAND()\*10)/\*round(x) 返回离x最近的整数 小数点后四舍五入\*/

SELECT ROUND(1.2345,3);/\*保留x小数点后y位 但截取时要进行四舍五入\*/

3.字符串函数

最常用的一类函数 主要用于处理表中的字符串

SELECT LOWER(ename) FROM emp;/\*将字符串转换为小写\*/

SELECT UPPER(ename) FROM emp;/\*将字符串转换为大写\*/

/\*length(x) 返回字符串长度\*/

/\*获取表中 名字长度为5的所有员工\*/

SELECT ename FROM emp WHERE LENGTH(ename)=5

/\*将多个字符串 合并为一个字符串\*/

SELECT CONCAT('ab','23');

/\*替换 replace(str,search\_str,replace\_str)

在str中用 replace\_str 替换 search\_str

\*/

/\*

截取 substring(str,position[,length])

从str 的 position位置开始 取length个字符

\*/

/\*

出现的位置 instr(string,substring)

返回substring在string中出现的位置 没有返回0 默认索引从1开始

\*/

/\*

ltrim(string)

rtrim(string) 去除前端空格或者去除后端空格

\*/

4.日期和时间函数

主要用于处理日期和时间的

/\*日期和时间函数\*/

SELECT CURDATE()/\*返回当前日期\*/

SELECT CURTIME()/\*返回当前时间\*/

SELECT NOW()/\*返回当前日期和时间\*/

SELECT MONTH("2011-05-02")/\*获取指定时间中的 月的部分\*/

SELECT YEAR(NOW())/\*获取指定时间中 年的部分\*/

/\*返回日期当中的月份英文名\*/

SELECT MONTHNAME(NOW())

/\*返回日期是星期几 英文\*/

SELECT DAYNAME(NOW())

/\*返回一周的第几天 从星期日开始\*/

SELECT DAYOFWEEK("2017-8-21")

SELECT WEEK(NOW())/\*计算日期是本年的第几个星期 范围0-53\*/

SELECT HOUR(NOW())/\*时\*/

SELECT MINUTE(NOW())/\*分\*/

SELECT SECOND(NOW())/\*秒\*/

SELECT ADDTIME('1:1:1','2:2:2')/\*将时间相加\*/

SELECT DATEDIFF('1981-1-17','1980-12-17');

/\*获取今年2月份入职的员工列表\*/

SELECT \* FROM emp WHERE MONTH(biredate)=2 AND YEAR(NOW())=YEAR(biredate);

/\*获取入职时间在30年以上的员工信息\*/

SELECT \* FROM emp WHERE (YEAR(NOW())-YEAR(biredate))>30

/\*获取 7369 的年龄 \*/

5.条件判断函数

ifnull(x,y)

如果x的值不为null 则返回x 否则返回y

select ifnull(null,'hello word') -->hello word

if(expr,x,y)

如果表达式expr成立,返回结果x,否则,返回结果y

select if(1>0,'正确','错误') --> 正确

6.系统信息函数

ip地址互相转换

inet\_aton(ip)将ip地址转换为数字表示 ip需要加引号

inet\_ntoa(n)函数可以将数字n转换为ip地址

7.加密函数

mysql对数据进行加密的函数

1.password(str)

可以对字符串str进行加密,一般,用于给用户的密码加密的

2.md5(str)

md5(str)函数可以对字符串进行散列,可以用于一些普通的不需要解密的数据

SELECT PASSWORD('123')

SELECT MD5('123')

SELECT INET\_ATON('127.0.0.1')

SELECT INET\_NTOA(2130706433);

8.格式化函数

二mysql数据库的备份与恢复

1.生成sql脚本 导出数据

cmd控制台下的命令

mysqldump -u 用户名 -p 数据库名 > 盘符:\脚本文件名.sql

mysqldump命令可以用来生成指定数据库的脚本文件

但要注意,脚本文件中只包含数据库的内容,而不包含创建数据库语句

所以在恢复数据时,需要自己手动创建数据库后才可以恢复

注意:mysqldump命令是直接在windows控制台上执行的 无需登录mysql!!!

演示命令:

mysqldump -u root -p dbcq > d:\mydbcq.sql

2.执行sql脚本 导入数据

1. mysql -u 用户名 -p 数据库名 < 盘符:\脚本文件名.sql

无需登录mysql

2.source 盘符:\脚本文件名.sql

需要登录mysql 必须先创建库 进入库 才可以执行sql脚本

dbc

一.简介

JDBC (java database connectivity) SUN公司推出的一套操作数据库的标准规范

JDBC与数据库驱动的关系: 接口与实现的关系

二.开发步骤

0.搭建开发环境

把数据库驱动jar包加入到项目中

java工程

先在工程下创建lib文件件 将jar包拷入

buildpath ...add jars

web工程

直接复制jar包到webRoot/WEB-INF/lib目录下

1.注册驱动

2.获取与数据库的连接

3.创建代表sql语句的对象

4.执行sql语句

5.如果是查询语句:返回结果集

6.释放资源

对数据库的基本操作

CRUD : Create Retrive Update Delete

三.JDBC规范中常用的接口和类(四个核心对象)

DriverManager 类

管理一组 JDBC 驱动程序的基本服务。

static Connection getConnection(String url) 试图建立到给定数据库 URL 的连接。

Connection 接口

java.sql 接口 Connection

与特定数据库的连接（会话）。在连接上下文中执行 SQL 语句并返回结果。

Statement createStatement() 创建一个 Statement 对象来将 SQL 语句发送到数据库。

Statement 接口

PreparedStatement

用于执行静态 SQL 语句并返回它所生成结果的对象。

ResultSet executeQuery(String sql) 执行给定的 SQL 语句，该语句返回单个 ResultSet 对象。

int executeUpdate(String sql) 执行给定 SQL 语句，该语句可能为 INSERT、UPDATE 或 DELETE 语句，或者不返回任何内容的 SQL 语句（如 SQL DDL 语句）。

boolean execute(String sql) 执行给定的 SQL 语句，该语句可能返回多个结果。

//任何sql语句都能执行

方法 executeQuery

用于产生单个结果集的语句 返回 ResultSet

方法 executeUpdate

执行 insert update delete DDL 返回 int 影响了几行

方法 execute

执行 查询操作 更新操作 DDL语句

返回值类型是boolean

返回值为true时 表示执行的是查询语句 getResultSet()

返回值为false时 表示执行的是更新语句或DDL语句 getUpdateCount()

ResultSet 接口

表示数据库结果集的数据表，通常通过执行查询数据库的语句生成。

boolean next() 将光标从当前位置向前移一行。

getXX()

java类型 数据库类型

int getInt() int

String getString() varchar char text

java.sql.Date getDate() Date

double getDouble() double

float getFloat() float

short getShort() smallint

long getLong() bigint

int getInt(int columnIndex) /\*序号\*/

int getInt(String columnLabel) /\*数据库表中的列名\*/

以 Java 编程语言中 int 的形式获取此 ResultSet 对象的当前行中指定列的值。

/\*在获取数据库数据时 可以通过 列名调用 或者通过列的编号来调用

\* getString() 可以获取任意数据类型 但是不推荐

\* 数据库列是什么类型 就用getxxx()方法来调用

\* 在使用的时候 通过列名调用

\* \*/

JUnit 单元测试 以方法为单元 只要测试方法里面的内容是否通过就行了

语法:在声明方法的上一行 加入 @Test

注意:

把一个日期值写入数据库中

方法1

//将 util Date 转成 sql Date

java.util.Date utilDate = new java.util.Date();

java.sql.Date sqlDate = new java.sql.Date(utilDate.getTime());//getTime()获取毫秒 long型

System.out.println(utilDate);

//方法2

long lo = Calendar.getInstance().getTime().getTime();

java.sql.Timestamp ts = new java.sql.Timestamp(lo);

//将 字符串 转换为 sql Date

// String s = "2017-08-22 09:8:10";

// SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

// java.util.Date d1 = sdf.parse(s);

// java.sql.Date d2 = new java.sql.Date(d1.getTime());

获取数据库中时间的方式

1.getString()

尽量不要使用getString()来获取日期/时间类型的数据

jdbc在内部将日期/时间转换为String

但是这个字符串的格式 取决于 数据库

Oracle mysql ..时间格式可能不一样 所以不推荐使用getString()

2.获取到Date 然后用SimpleDateFormat来格式化

java.sql.Timestamp ts = //通过数据库访问获取到的数据

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String str = sdf.format(ts);

如果是 sql.Date / sql.Time 同上