数据库(1)(2)

数据库的基本常识：

1、分类：

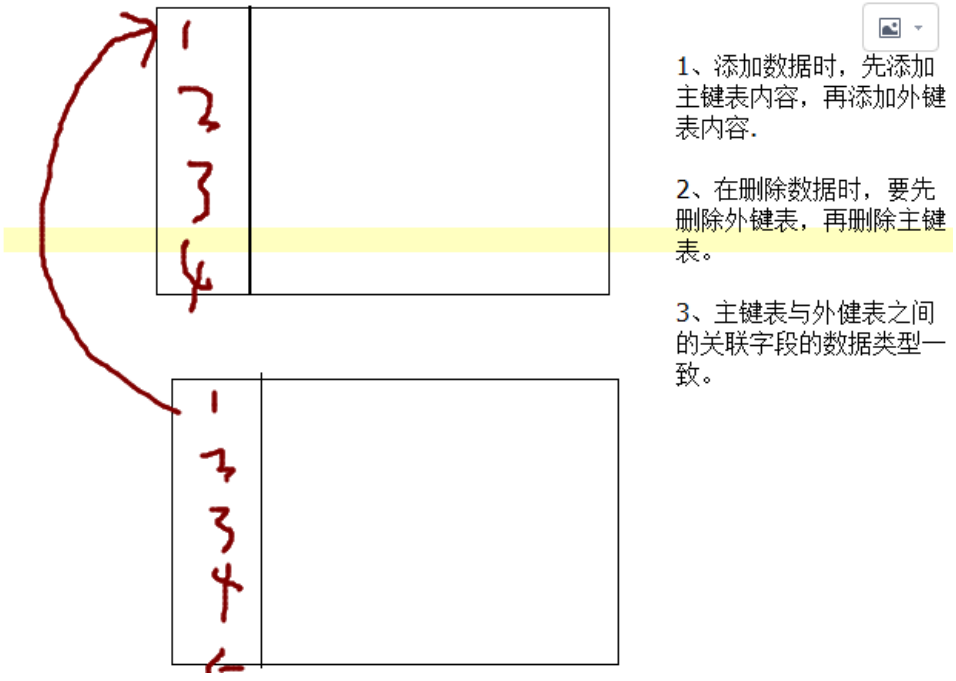
关系型数据库(oracle， SQLServer, IBM（DB2），MySql)， 嵌入式数据库(Sqlite)，用于微型移动设备

面向对象数据

2、数据库工作的分类:

DBA(DataBase Administrator)

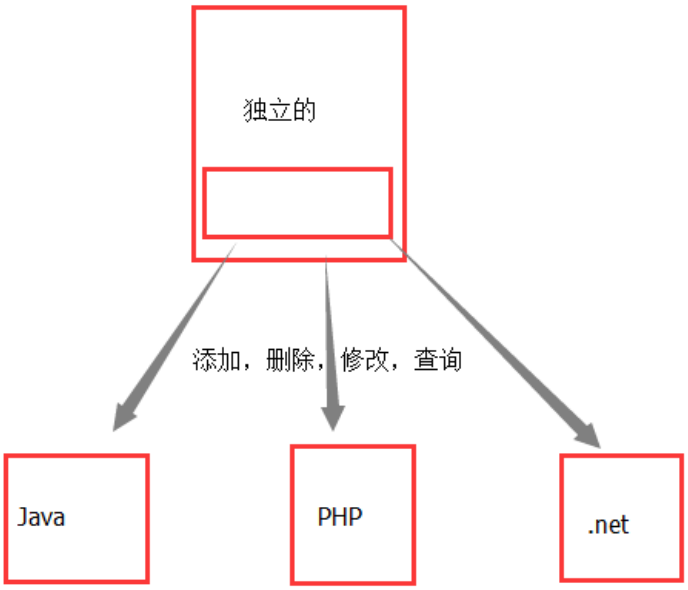
大数据处理

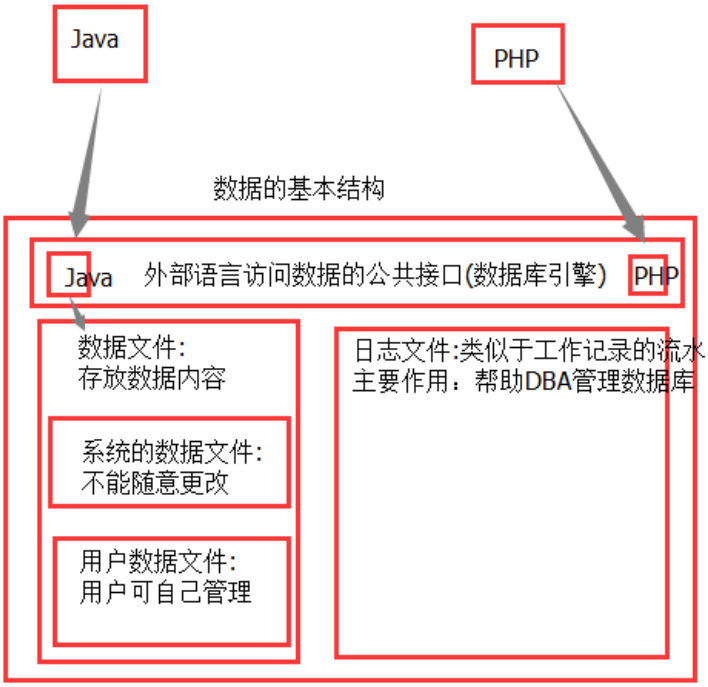


1、添加数据时，先添加主键表内容，再添加外键表内容.

2、在删除数据时，要先删除外键表，再删除主键表。

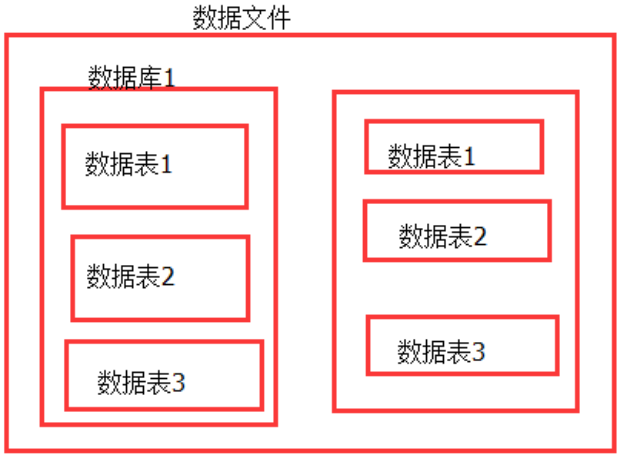
3、主键表与外健表之间的关联字段的数据类型一致。


独立的
添加，删除，修改，查询
Java
PHP
.net


Java
PHP
数据的基本结构
Java
外部语言访问数据的公共接口(数据库引擎)
PHP
数据文件:
存放数据内容
系统的数据文件:
不能随意更改
用户数据文件:
用户可自己管理
日志文件:类似于工作记录的流水
主要作用：帮助DBA管理数据库

本周学习的主要内容:

1、创建数据库
2、修改数据库
3、创建数据表
4、修改数据表的结构
5、对数据表的（增，删，改，查）
6、数据表的设计
7、通过JAVA连接并操作数据库(JDBC, C3P0)
重点，难点：数据表的查询及数据表的设计

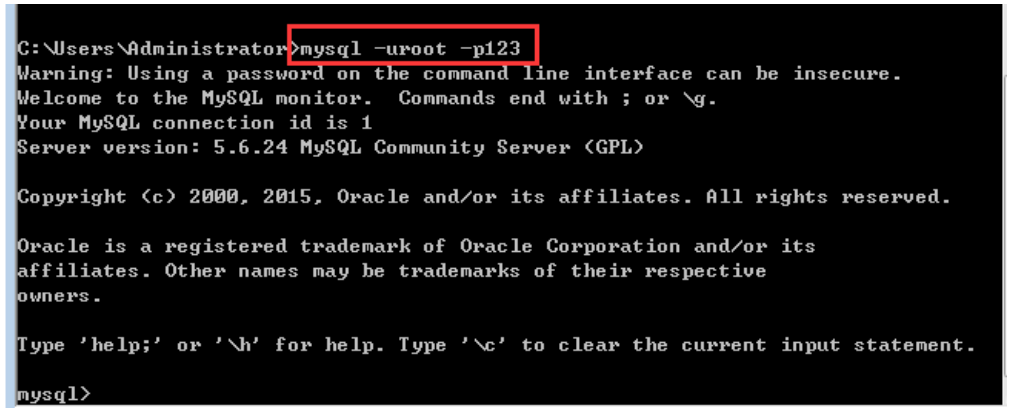


在DOC环境下，使用MySQL。

1、启动数据库服务：net start mysql，启动成功后，有正确提示

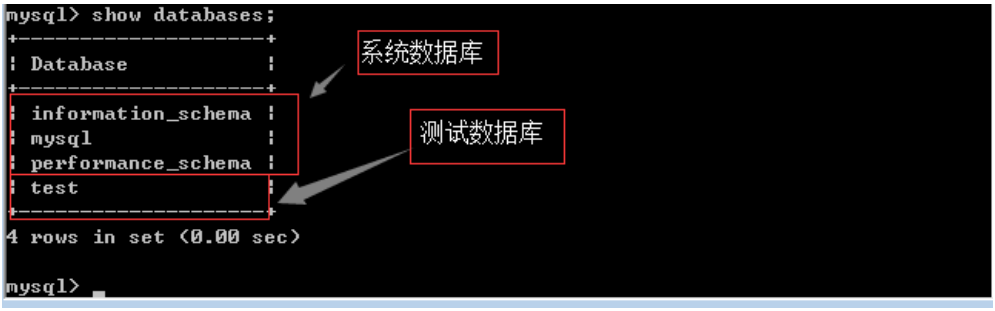
2、关闭数据库服务: net stop mysql, 关闭数据服务，正确操作后，有提示信息。

3、数据库的登录: mysql -u用户名 -p用户名对应的密码。登录成功提示如下图。

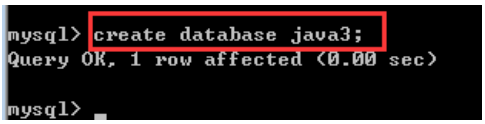


4、查看当前用户下，可以操作的数据库: show databases;

注：其中系统自带的数据库，不要随意修改.



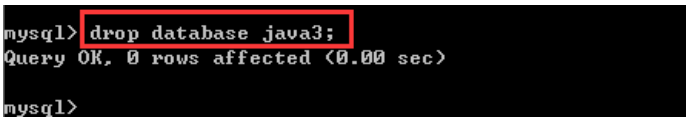
5、创建数据库的最简命令：create database 数据库名; 创建成功后，有相应提示信息



创建数据库并指定字符集:

指定数据库的字符集是utf8格式：CREATE DATABASE java4 DEFAULT CHARSET utf8 COLLATE utf8_general_ci;

6、删除数据库: drop database 数据库名;



7、mysql的数据类型：第一大类：数值类型. 第二大类:日期，日期、时间类型. 第三大类：字符串类型

数字类型：

tinyint	smallint	int	bigint	float	double	decimal
小整数	大整数	大整数	极大整数	浮点数	双精度	小数字
年龄等字段						

日期，时间类型：

data	time	year	datatime	timestamp
YYYY-MM-DD	HH:MM:SS	YYYY	YYYY-MM-DD HH:DD:SS	YYYYMMDD HHMMSS

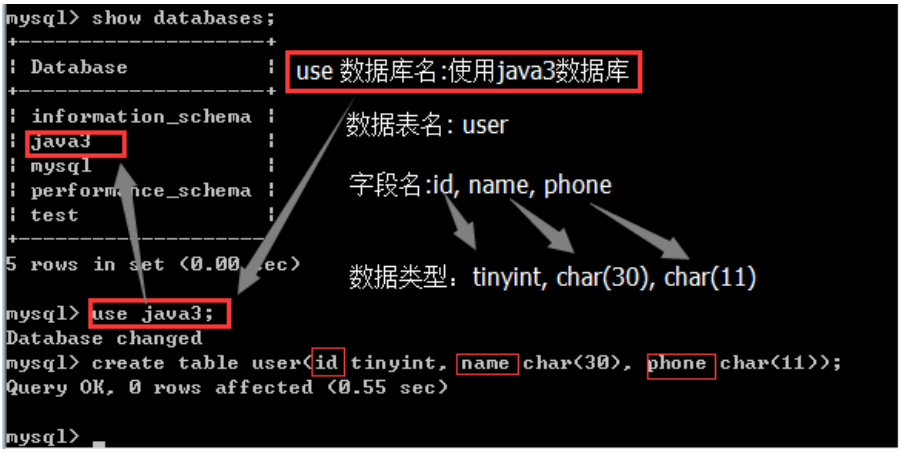
字符串类型：

char	varchar	text	longtext	blob	longblob		
定长字符串	变长字符串	长文本数据	极大文本数据	二进制数据	极大二进制数据		

注：如果字段类型是二进制数据，则该字段可以存放流媒体文件（图片，音频，视频），但一般不要这样存放数据。

8、创建数据表及删除数据表

create table 表名(字段名1 字段类型, 字段名2 字段类型, ... 字段n 字段类型);



删除数据表:

drop table 表名;

```
mysql> drop table stu;
Query OK, 0 rows affected (0.40 sec)
```

9、向数据表中，插入数据。

insert into 表名(字段一, 字段二, 字段三,.....字段n) values (value1, value2, value3..... valueN);

```
mysql> insert into user(id, name, phone) values(1, 'huyang', '13999999999');
Query OK, 1 row affected (0.08 sec)
```

注：a、在向字段中添加数据时，要注意对应字段的数据类型.

b、要注意value中的字段书写顺序与表名后面的字段顺序。

10、（1）最简查询数据表: select * from 表名;

```
mysql> select * from user;
+------+--------+-------------+
| id   | name   | phone       |
+------+--------+-------------+
|    1 | huyang | 13999999999 |
|    1 | huyang | 13999999999 |
|    0 | 1      | 13999999999 |
+------+--------+-------------+
3 rows in set (0.00 sec)
```

（2）、查询数据表中的指定字段. select 字段1, 字段2.... from 表名;

```
mysql> select id, name from user;
+------+--------+
| id   | name   |
+------+--------+
|    1 | huyang |
|    1 | huyang |
|    0 | 1      |
+------+--------+
3 rows in set (0.00 sec)
```

11、修改数据字段名：

修改字段名: id --> user_id;

alter table user change id user_id tinyint;

12、修改字段属性：

格式： alter table 表名 字段名 字段名 新的数据类型。 与修改字段名类似。

修改字段属性： tinyint ---> int

alter table user change user_id user_id int ;

13、修改数据表的表名：

格式：alter table 表名 rename 新的表名;

修改数据表的表名称:

alter table user rename student;

14、删除表中的列:

格式：alert tabel 表名 drop column 需要删除的列;

实例：alter table student drop column user_pwd;

15、增加新的字段:

格式：alter table 表名 add column 新的字段名 字段类型;

alter table student add column user_pwd varchar(10);

alert table 表名 具体的修改方式(如，修改属性，添加字段，删除字段)

注：凡是涉及修改数据库，数据表，数据字段，字段类型时，都是由DBA或项目设计者完成。

1、添加数据

2、删除数据

3、删除表格内容

4、修改数据

16、表的主键: 关键字: primary key（主键也称为主关键字, 也称为主索引）

主键的作用： a、能够唯一标识一条记录，防止出现重复记录。 b、提高查询效率。

注：一个表中，只能不一个主键.(即一个主关键字)

给原有的数据表添加主健。	**在创建数据表时，同时给定表的主键**
alter table student add primary key (user_id);	create table user(id tinyint primary key, user_name varchar(30), us

17、删除主键。 注：该操作一般不要去进行。

格式：alter table 表名 drop primary key;

实例，删除student表的主键：alter table student drop PRIMARY KEY;

18、设置字段自增长： auto_increment关键字

```
create table user(
    id tinyint primary key auto_increment,   //设置id字段自增长
    user_name varchar(30) unique,
    user_pwd varchar(10)
);
```

19、索引：

索引即目录。索引主要是提高查询效率。

索引的分类：

1）、主索引：主键

2）、唯一索引: 也能唯一标识一条记录

3）、普通索引

4）、复合索引(多列索引)

5）、全文索引

注：a、一张表中，主索引只能有一个，但其他索引的个数不限。

b、一个字段设置了索引之后，可再次为该字段添加其他索引，但在主索引字段上，不能添加全文索引

c、下面实例中，方括号中的内容是可选的。即可以没有.

sql语句也称为t-sql

添加唯一索引: unique

格式：alter table 表名 add unique [索引名] (字段1, 字段2....，字段n);

alter table student add unique name_pwd (user_name, user_pwd); //为多个字段同时设置唯一索引

alter table student add unique (user_pwd);

添加普通索引

格式：alter table 表名 add index 索引名(字段名)

实例， 为student表的user_pwd添加普通索引: alter table student add index index_pwd(user_pwd);

添加复合索引: 几个字段合起来作为索引。

格式：alter table 表名 add index 复合索引名(字段1, 字段2....字段n)

alter table student add index index_name_pwd (user_name, user_pwd, user_id);

添加全文索引

alter table student add fulltext (user_id);

20、外键（外部关键字）

前提：1）、设置一个表的外部关联时，一定相关表的主键。

2）、两个对应的字段的数据类型必须一致。

3）、先有主健表，再有外键表.

注：a、两个表的对应字段的名称尽量一致。 b，外健名称尽量以fk_开头。

(1)、在创建表时，设置外键（外部关健字）

在创建表代码的最后一行，添加外键的代码：

格式：constraint 外键名称 foreign key (外键字段) references 主键表名(字段)

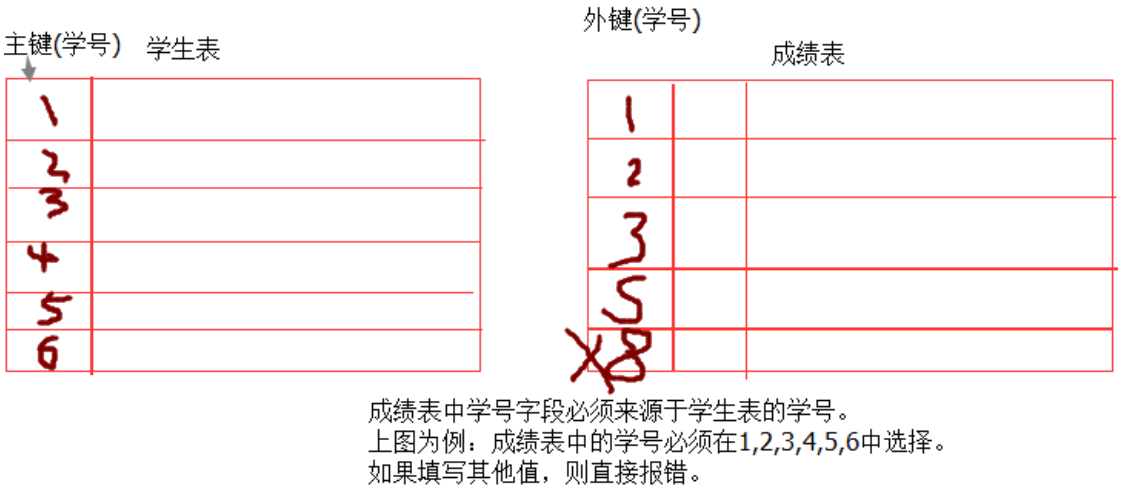
```
create table score(
    user_id int,
    javaSE float default 0.0,
    javascript float,
    mysql float,
    servlet float,
    constraint fk_user_id foreign key(user_id) references student(user_id)
);
```

(2)、通过修改表的形式，为表添加外键关联：

格式：alter table 表名 add foreign key 外键名 (外键字段) references 主键表(主健字段)

alter table score add foreign key fk_user_id (user_id) references student(user_id);

主键与外键之间 的关系如下图：



21、enum:关键字：限定字段只能在指定范围内取值. 适用于取值范围较少的场景。比如性别字段等。

decimal: 小数据类型。限制字段最大取值的数字。

//通过enum关键字设置字段的取值范围

//enum：枚举类型, enum不是所有数据都支持的语法。比如SQL Server不支持enum类型。在SQL Server中，使用check语句完成enum类似的功能。

//1、在mysql, oracle，sqlite中，都支持enum类型完成字段内容的限定，但微软公司的数据，则使用check语句完成任务类似的功能。

//2、mysql中，check语句是合法的，但不起作用.

比如，性别只能在"男","女", "妖"三者之间选择

decimal:关键字. 1、可以用于限制数字的位数，及小数位数. 2、第一个参数要大于第二个参数

如decimal(4, 2):最大取值是99.99.

```
create table aa(
    sex enum ('男', '女', '妖'),
    age decimal(4, 2)
)
```

22、向表中添加数据

```
insert into score(user_id, javaSE, javascript, mysql, servlet) values(1, 90, 80, 95, 96);
```

向表中所有字段都添加新的内容

```
insert into score values(2, 90, 80, 95, 96);
```

23、删除数据

删除指定的记录: where条件表达式，如果没有指定的记录，不执行删除操作，也不会报错。

格式：delete from 表名 where 字段 = "字段内容".如下删除user_id = 2的记录.

```
delete from score where user_id = 2;
```

删除表的全部内容，该操作请谨慎使用。

```
delete from score;
```

注：在删除数据记录时，要注意以下内容：

1）如果要删除的记录，与其他表有关联关系时，要仔细一点。

2）在删除数据时，如果有主外键关联关系，要先删除外键表中的记录，再删除主键表中的记录。

24、修改数据表：

格式：update 表名 set 字段名=value, 字段 名=value where 条件

注：如果没有where条件，则把字段中所有的记录都修改为一样的值

实例：把yg表中的y_name等于'张三'修改为'李四'

```
update yg set y_name = '李四' where y_name = '张三'
```

24、补充：Java的枚举类型

```java
public class EnumObj {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //System.out.println(Color.a);
        enumFun(Color.a);
    }

    public static void enumFun(Color color){
        switch(color){
        case BLACK:
            System.out.println("这是黑色");
            break;
        case RED:
            System.out.println("这是红色");
            break;
        case GREEN:
            System.out.println("这是绿色");
            break;
        case YELLOW:
            System.out.println("这是黄色");
            break;
        default:
            System.out.println("其他");
        }
    }

}
//enum:枚举类型关键字
//格式: enum 枚举名称{枚举的值}
//注：枚举类型的每个值一般使用全大写字母.
enum Color {
    RED, GREEN, BLACK, YELLOW, a
}
```

注：''(字串)与NULL不是同一个值.

25、t-sql查询：

1)、*：查询表中所有数据， 一般用于测试

select * from student;

2)、查询指定字段:

格式：select 字段名, 字段2， 字段3.... from 表名.

select user_name, user_pwd from student;

3)、where条件度语句: 从结果集中，筛选出适合条件的记录

格式：where 条件表达式

例：从student中，查询出"胡洋"的信息: select user_name, user_pwd from student where user_name = '胡洋'

4)条件表达式中的运算

=， >, < , !=, <>,

5）使用计算列

如总成绩: 字段相加即可.

格式：select 字段+字段+字段 from 表名

select javaSE + javascript + mysql + servlet from score;

6)、增加说明了列：as关键字

作用：要以隐藏数据表的真实结构，增加数据库的安全性，也可防止数据库的注入攻击。

as：为表，列添加说明

为查询出来的字段，添加新的别名

select javaSE + javascript + mysql + servlet as 总成绩 from score;

给字段及表添加别名

select user_name as 姓名, user_pwd as 密码 from student as 学生表;

7）、模糊查询: like, not like

模糊查询的语法：_:任意一个字符, %:任意多个字符

select * from student where user_name like '杨_'

select * from student where user_name like '杨%'

8）、范围之内查询:

between A and B: 在a到b的范围之内。类似于 and运算符

select * from student where user_age >= 24 and user_age <= 26;

select * from student where user_age between 24 and 26;

9）、in:在指定范围之内

格式：in(值1, 值2, 值班3..... 值n). 适用于小范围查询。类似于 or 运算符.

not in（值1, 值2, 值3.....值n）:不在指定范围之内

select * from student where user_age = 24 or user_age = 26;

select * from student where user_age in(24, 26);

select * from student where user_age not in(24, 26);

10）、is null:为空的判断.

is not null:不为空的判断

实例1，查询出student表中，user_age为空的记录：select * from student where user_age is null;

实例2，查询出student表中，user_age不为空的记录：select * from student where user_age is not null;

11)、去掉重复的查询记录: distinct

格式：select distinct 字段 from 表名

查询成绩小于9分的学号，去掉重复记录：select distinct sno from score where grade < 9;

12)、对查询结果排序：order by关键字

a、order by单字段排序:

查询排序： order by: asc/desc

asc:顺序排序。

desc:倒序排序

格式：order by 字段名 asc/desc

对student表按学号倒序排序：select * from student order by sno desc;

b、order by复合排序:

格式：order by 字段1， 字段子： 按字段1排序，如果字段一相同，则按字段2排序.

实例：在student表中，按所在系(sdept)字段排序，如果是同一个系的，则按年龄(sage)排序

select * from student order by sdept, sage;

13）、查询前面n条记录：limit关键字

查询记录数: limit

limit n:查询前面n条记录，如果记录数不够，也能正常运行。

select * from student limit 3;

limit n, m: n-->开始记录位置,记录从0开始计算， m --->一次性查询的记录数

select * from student limit 1, 3;

14）、分组查询：group by

group by : 分组查询, 先分组，再查询

按user_sex进行分组查询

select user_sex from student group by user_sex;

按 sno进行分组查询

select * from score group by sno;

按cno进行分组查询

select * from score group by cno;

实例：查询出男性，女性中年龄最大的姓名及年龄.

基本思路：1、把查询年龄及姓名 2、再按分组查询，得到每个分组中，年龄最大的数值. 3、合并两个查询语句

第一步：select user_name, user_age from student

第二步：取得每个组中的最大值。把下面的查询语句看是函数，结果则是函数的返回值。

select max(user_age) from student group by user_sex;

第三步：把前面两段sql合并，完成需求。

select user_name, user_age from student where user_age in(select max(user_age) from student group by user_sex);

group by的使用方式 : 一般与聚合函数(统计函数)一起使用。

练习：得到每个学号的总成绩, 成绩表的结构如下：

Sno	Cno	Grade
S_001	C_001	10.00
S_001	C_002	9.50
S_001	C_003	9.00
S_001	C_004	4.00
S_002	C_001	9.00
S_002	C_004	10.00
S_002	C_005	7.00
S_002	C_006	10.00
S_002	C_007	10.00
S_003	C_001	10.00
S_003	C_005	7.00
S_003	C_006	10.00
S_003	C_007	10.00
S_004	C_002	10.00
S_004	C_003	9.00
S_005	C_001	10.00
S_005	C_002	9.40
S_005	C_003	8.60
S_005	C_004	3.00

结果：select sno, sum(grade) from score group by sno;

15）、having子句的用法。及having与where的区别:

where条件执行的时间及限制:

1、执行时间：在分组之前执行

2、执行限制：where条件中，不能有聚合函数（统计函数）

3、where条件不能对group by进行分组

having：对group by之后的记录进行筛选。

1、having子句执行在group by之后

2、having过滤的字段，一定要在select的字段之中，否则会出现语法错误。

3、havig过滤的字段，不是聚合函数(统计函数)之中的字段

group by 与 having大多数是结合使用。使用having对group by 进行筛选。

实例：得到总学分大于40的学号及总成绩：

select sno as 学号, sum(grade) as 总成绩 from score group by sno having 总成绩 > 40

所有学号的总成绩：

学号	总成绩
S_001	32.50
S_002	46.00
S_003	37.00
S_004	19.00
S_005	31.00

通过having过滤后的结果：

学号	总成绩
S_002	46.00

16）、多表查询

多个表之间的联系查询，如果表之间有相同的字段，可根据字段相同，排除重复的记录.

格式：select 表1.字段, 表1.字段, 表2.字段, 表2.字段, 表2.字段 from 表1, 表2 where 表1.字段 = 表2.字段

实例：从学生表(student), 课程表(course), 成绩表(score)中，查询出每个学生的姓名及所选科目的成绩。

思路：从学生表中，得到姓名, 从课程表中得到课程名，从成绩表得到成绩

查询姓名，课程名, 成绩

select student.sname, course.cname, score.grade from student, course, score where student.sno = score.sno and course.cno = score.cno;

17）、左连接(left join)

格式：select 字段, 字段 from 表1 left join 表1 on 表1.字段 = 表2.字段

//左连接后面应加上连接时的条件: on条件语句

//左表与右表进行连接查询，可以减少重复的记录数，提高查询效率。

select student.sname, student.sno, score.grade from student left join score on score.sno = student.sno where grade is not null;

数据表的查询

a、指定列。

b、选择所有列

c、使用计算列

d、增加说明列

e、改变列标题

f、单表查询(简单查询)

(1)、使用比较运算符（=， <, >, <>, !=, !>, !<, >=, <=）

(2)、使用逻辑运算符（and, or, not）

(3)、使用字符串模糊匹配（%：任意长度的字符串， _：任意一个字符， []：范围内的任意一个字符， [^]:不在指定范围内的任意字符）

(4)、使用查询范围 between and.

(5)、使用查询列表(in, not in)

(6)、空值判断(is null, is not null)

(7)、排序（order by asc/desc）

(8)、使用limit,

26、常见的统计函数

嵌套查询时，可以把子查询看当是函数的返回值。

1)、max()：最大值, min()：最小值, avg()：平均值

max:取得字段的最大值。

格式: max(字段)

//取得年龄的最大值.

select max(sage) from student;

取得最大sage记录的详细信息

select * from student where sage in (select max(sage) from student);

2）、min:取得字段的最小值

格式：min(字段)

取得sage最小的值

select min(sage) from student;

取得最小sage记录的详细信息

select * from student where sage in (select min(sage) from student);

3）、avg:取得字段的平均值

格式：avg(字段)

取得sage的平均数

select avg(sage) from student;

取得比sage平均数小记录的详细信息

select * from student where sage < (select avg(sage) from student);

4）、统计记录个数： count

格式：count(字段)

根据sname取得记录个数

select count(sname) from student;

select count(*) from student;

5）、求合统计: sum, 取得字段相加过后的值。

格式：sum(字段)

select sum(sage) from student;

27、视图：虚拟表

1）、视图的主要功能：

a、提高查询效率, b、数据安全。 c、用法灵活

2）、视图的用途：

查询功能，不要在视图上进行数据的修改。如果在视图上对数据做了修改，实体表也会相应修改

视图主要有以下作用：

a、安全。一些数据表有着重要的信息。有些字段是保密的，不能让用户直接看到。这时就可以创建一个视图，在这张视图中只保留一部分字段。这样，用户就可以查询自己需要的字段，不能查看保密的字段。

b、性能。关系数据库的数据常常会分表存储，使用外键建立这些表的之间关系。这时，数据库查询通常会用到连接（JOIN）。这样做不但麻烦，效率相对也比较低。如果建立一个视图，将相关的表和字段组合在一起，就可以避免使用JOIN查询数据。

c、灵活。如果系统中有一张旧的表，这张表由于设计的问题，即将被废弃。然而，很多应用都是基于这张表，不易修改。这时就可以建立一张视图，视图中的数据直接映射到新建的表。这样，就可以少做很多改动，也达到了升级数据表的目的。

格式： create view 视图名as (查询语句)

实例：在student实体表上，建立视图，视图包括了student表中的sno及sname字段.

create view stu_view as

select sno as 学号, sname as 姓名 from student;

28、存储过程：procedure

概念：存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的SQL 语句集

作用：1、提高查询效率：存储在数据库中，经过第一次编译后再次调用不需要再次编译，

2、提高SQL语句的灵活性：用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它， 把存储过程当成是函数进行调用。

语法：创建存储过程: create procedure 存储过程名字([in, inout, out] 变量名 参数类型)

```
BEGIN
    SQL语句集....
END;
```

存储过程的调用：call 存储过程名(参数)

存储过程的不足：在不同数据库系统中，定义在存储过程的语法格式是不同的，调用存储过程的语法也是不一样的。所以存储过程不具有跨平台性。

实例：

存储过程参数类型说明：

in:输入参数。在存储过程中，重新给参数赋值后，也不会影响到实际的变量值。

out:输出参数。在存储过程中，重新给参数赋值后，要改变变量的初始值。

inout:输入输出参数。在存储过程中，重新给参数赋值后，要改变变量的初始值。

```
CREATE PROCEDURE sp_demo_in_parameter(IN p_in INT)
BEGIN
 SET p_in=2;
 select p_in as fun;  //输出结果：2
END;

set @p_in=1;
call sp_demo_in_parameter(@p_in);
select @p_in as outfun;  //输出结果：1


CREATE PROCEDURE sp_demo_out_parameter(out p_out INT)
BEGIN
 SET p_out=2;
 select p_out as fun;  //输出结果：2
END;

set @p_out=1;
call sp_demo_out_parameter(@p_out);
select @p_out;  //输出结果：2


CREATE PROCEDURE sp_demo_inout_parameter(inout p_inout INT)
BEGIN
 SET p_inout=2;
 select p_inout as fun;  //输出结果：2
END;
```

```
set @p_inout=1;
call sp_demo_inout_parameter(@p_inout);
select @p_inout;  //输出结果：2
```

实例：分页显示的存储过程

创建存储过程：

```
create procedure student_page(in sta int, in size int)
begin
  select * from student limit sta, size;
end;
```

调用存储过程：

```
set @sta = 0;  #起始位置，（起始位置 + 每页记录数量 = 下一页的数据记录的开始位置）
set @size = 3;  #每页显示的数据条数
call student_page(@sta, @size);
```

25、数据库的概念总结：

数据库的元素：

1、数据表
2、视图
3、存储过程
4、触发器(用于数据完整性及安全性， 缺点：执行效率较低)

数据表的元素：

1、字段
2、记录
3、索引(主索引，普通索引，复合索引， 全文索引)
4、主键
5、外键
6、字段的数据类型

保证数据安全及完整性的方式：

1、取别名: as关键字
2、视图
3、主键(数据不重复)
4、外键(保证数据的录入是来源于主键表)
5、事物(在向数据库中添加数据时，所有操作都是正确的。如果在添加数据过程中，出现了任何异常，都不进行数据库操作)
6、触发器

注：数据库中需要自学的内容: 1、数据库的事物处理. 2、触发器（可选）

MySQL的补充知识点：

mysql的语法补充：

1、在查询时，添加新的一列：

```
#在查询表时，添加新的一列: 在查询student表时，添加age列,且该列的默认值是'age'
select sno, sname, sex, '年龄' as age from student;
```

2、union:连接多个子查询，但这多个字查询的字段个数及字段类型必须一致.

3、#case 条件 when 判断条件 then 条件成立时执行 else 条件不成立时执行 end：类似于switch... catch的作用

4、横纵表之间的转换：

1）、横表转纵表：union关键字，连接各个子查询

2）、纵表转横表：case ... when进行条件判断。

一、横表转纵表实例：

id	姓名	数学	语文	英语
1	胡洋	98.0	88.0	89.0
2	杨诚	79.0	99.9	99.9

需要得到的结果如下图：

name	subject	成绩
胡洋	语文	88.0
杨诚	语文	99.9
胡洋	数学	98.0
杨诚	数学	79.0
胡洋	英语	89.0
杨诚	英语	99.9

横转纵的t-sql语句：

```
select 姓名, '语文' as 科目, 语文 as 成绩 from 成绩表
    union
select 姓名, '数学' as 科目, 数学 as 成绩 from 成绩表
```

二、纵转横向的实例：

name	subject	成绩
胡洋	语文	88.0
杨诚	语文	99.9
胡洋	数学	98.0
杨诚	数学	79.0
胡洋	英语	89.0
杨诚	英语	99.9

需要得到的结果如下图：、

姓名	语文	数学	英语
杨诚	99.9	79.0	99.9
胡洋	88.0	98.0	89.0

纵转横的t-sql语句：

```
select
    name as '姓名',
    max(case subject when '语文' then 成绩 else 0 end) 语文,
    max(case subject when '数学' then 成绩 else 0 end) 数学,
    max(case subject when '英语' then 成绩 else 0 end) 英语
from 成绩表_view group by name
```

26、关系型数据库的常用三大范式：

1）、字段不能再分(不能像excel那样，进行表格的合并及拆分操作。)。比如：学号字段，不能再分成其他字段。（关系数据库中的数据表必须满足的条件）

专业解释：确保每列的原子性.

2）、每一张表中，都有唯一标识记录的字段(每个表要有主键)

专业解释：确保表中的每列都和主键相关

3）、表中字面要有外键关联。

专业解释：目标是确保每列都和主键列直接相关,而不是间接相关

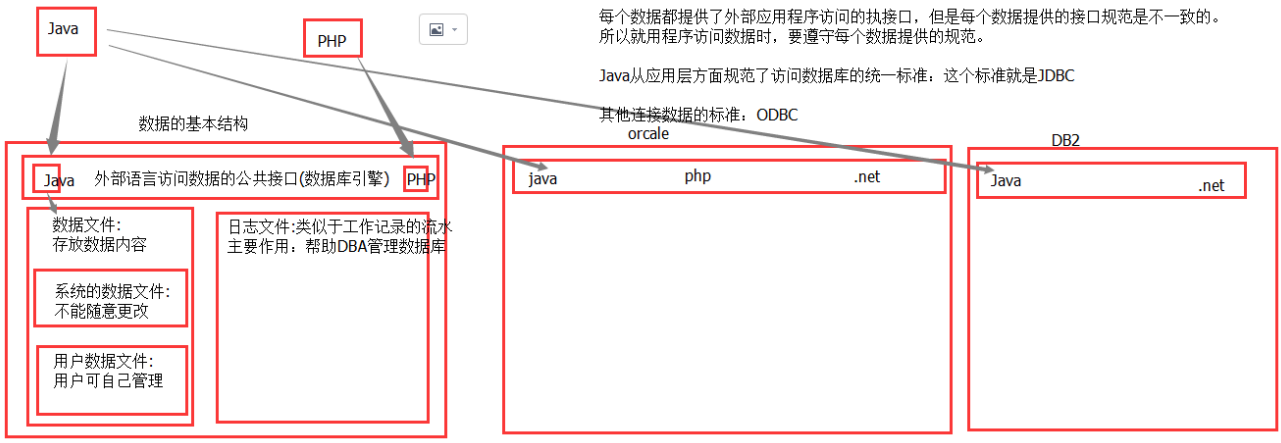
27、表与表之间的关联关系

1）、一对一关系：比如、公民与身份证的关系。员工表与工资表之间的关系，学生与学号

2）、一对多关系：比如、一个用户可以设置多个头像， 一个身份证号码可以有多个银行卡号， 一个国家对应一个国家元首

3）、多对多关系：学生与选课(一个学生可以选择多门课程，同一门课程也可多名学生选择。)

28、JDBC连接mysql数据库



Java连接数据库的步骤：

1、下载对应数据库的连接jar包.

2、集成jar

3、使用jar包

JDBC连接数据实例代码：

MySqlFactory.java
2016/12/23 14:30, 6.32KB

数据库的JDBC连接

获取连接：

```
private static Connection getConn() {          String driver = "com.mysql.jdbc.Driver";  //连接mysql数据库驱动的名称
    //jdbc:jdbc连接数据库; mysql:mysql数据库. 172.18.2.86:数据库所在的IP地址。 obj:数据库名.
    //useUnicode=true&characterEncoding=UTF-8: 连接数据时，指定连接时的编码集。
    String url = "jdbc:mysql://172.18.2.86/obj?useUnicode=true&characterEncoding=UTF-8"; //连接数据库的格式
    String username = "root"; //数据库的登录名
    String password = "123";  //数据库的登录密码
    Connection conn = null;   //Connection: 数据库的连接对象
    try {
        Class.forName(driver);        //第一步: classLoader,加载对应驱动
        conn = (Connection) DriverManager.getConnection(url, username, password); //第二步: 打开数据库连接(数据库地址及名称,
用户名, 密码)
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return conn;}
```

insert：添加记录

```
private static int insert(Student student) {    Connection conn = getConn();    int i = 0;    String sql = "insert into
students (Name,Sex,Age) values(?,?,?)";    PreparedStatement pstmt;    try {        pstmt = (PreparedStatement)
conn.prepareStatement(sql);        pstmt.setString(1, student.getName());        pstmt.setString(2, student.getSex());
pstmt.setString(3, student.getAge());        i = pstmt.executeUpdate();        pstmt.close();        conn.close();    }
catch (SQLException e) {        e.printStackTrace();    }    return i;}
```

update：修改记录

```
private static int update(Student student) {    Connection conn = getConn();    int i = 0;    String sql = "update
students set Age='" + student.getAge() + "' where Name='" + student.getName() + "'";    PreparedStatement pstmt;    try
```

```
{        pstmt = (PreparedStatement) conn.prepareStatement(sql);        i = pstmt.executeUpdate();
System.out.println("resutl: " + i);        pstmt.close();        conn.close();    } catch (SQLException e) {
e.printStackTrace();    }    return i;}
```

select:查询记录

```
private static Integer getAll() {    Connection conn = getConn();    String sql = "select * from students";
PreparedStatement pstmt;    try {        pstmt = (PreparedStatement)conn.prepareStatement(sql);        ResultSet rs =
pstmt.executeQuery();        int col = rs.getMetaData().getColumnCount();
        System.out.println("==========================");        while (rs.next()) {            for (int i = 1; i <=
col; i++) {                System.out.print(rs.getString(i) + "\t");                if ((i == 2) &&
(rs.getString(i).length() < 8)) {                    System.out.print("\t");                }            }
System.out.println("");        }            System.out.println("==========================");    } catch (SQLException
e) {        e.printStackTrace();    }    return null;}
```

delete:删除记录

```
private static int delete(String name) {    Connection conn = getConn();    int i = 0;    String sql = "delete from
students where Name='" + name + "'";    PreparedStatement pstmt;    try {        pstmt = (PreparedStatement)
conn.prepareStatement(sql);        i = pstmt.executeUpdate();        System.out.println("resutl: " + i);
pstmt.close();        conn.close();    } catch (SQLException e) {        e.printStackTrace();    }    return i;}
```

一、存储过程, 分页储存过程

二、数据库的设计：表与表之间的关联关系，数据库的前三个范式

三、JDBC, C3p0.

- where 子句的作用是在对查询结果进行分组前，将不符合where条件的行去掉，即在分组之前过滤数据，where条件中不能包含聚组函数，使用where条件过滤出特定的行。
- having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚组函数，使用having 条件过滤出特定的组，也可以使用多个分组标准进行分组。

数据安全约束的方式：

1、通过主外键的关联关系。

2、enum枚举进行限制

3、数据类型进行限制。如：char(30)：字段长度为30，超出范围无效. varchar(30):最大长度为30. not null：该字段不能为空.

```
create view my_view as
select * from ghs;

select * from my_view;

create or replace view syntax as
select * from ghs
select * from syntax;
```

1、 视图能简化用户操作

视图机制使用户可以将注意力集中在所关心地数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使数据库看起来结构简单、清晰，并且可以简化用户的的数据查询操作。

2、 视图使用户能以多种角度看待同一数据

视图机制能使不同的用户以不同的方式看待同一数据，当许多不同种类的用户共享同一个数据库时，这种灵活性是非常必要的。

3、 视图对重构数据库提供了一定程度的逻辑独立性

輸入参数：

```
CREATE PROCEDURE sp_demo_in_parameter(IN p_in INT)
BEGIN
SELECT p_in;
SET p_in=2;
select p_in;
END;

set @p_in=1;
call sp_demo_in_parameter(@p_in)

select @p_in;

CREATE PROCEDURE sp_demo_out_parameter(OUT p_out INT)
BEGIN
SELECT p_out;
SET p_out=2;
SELECT p_out;
END;


SET @p_out=1;
CALL sp_demo_out_parameter(@p_out)

select @p_out;



CREATE PROCEDURE sp_demo_inout_parameter(INOUT p_inout INT)
BEGIN
SELECT p_inout;
SET p_inout=2;
SELECT p_inout;
END;

set @p_inout=1;
call sp_demo_inout_parameter(@p_inout);
```