南京大学 NANJING UNIVERSITY

奋进行动 团结奋斗 争先进位 坐言起行 应势而动

数 据 管 理 基 础

第3章 关系数据库标准语言SQL

（数据操纵 & 视图）

智能软件与工程学院

数据管理基础

数据更新 -- 插入、修改、删除
空值的处理
视图

智能软件与工程学院

插入元组 1

- 将新元组插入指定表中，语句格式

```
INSERT
INTO  <表名> [(<属性列1>[,<属性列2 >…)]
VALUES  (<常量1> [,<常量2>]… );
```

- INTO子句
  - 指定要插入数据的表名及属性列
  - 属性列的顺序可与表定义中的顺序不一致
  - 没有指定属性列：表示要插入的是一条完整的元组，且属性列属性与表定义中的顺序一致
  - 指定部分属性列：插入的元组在其余属性列上取空值

- VALUES子句
  - 提供的值必须与INTO子句匹配：值的个数 & 值的类型

插入元组 2

属性名列表可以被省略。在此情况下，属性的排列顺序采用基表定义中的顺序

```
INSERT
INTO  tabname [ ( colname { , colname … } ) ]
VALUES  ( expr | NULL { , expr | NULL … } ) | subquery;
```

属性值，NULL表示空值

插入单个常量元组

被插入的常量元组值。其中属性值的数量及其排列顺序必须与INTO子句中的属性名列表一致

将子查询的查询结果插入到表tabname中。子查询结果属性的排列顺序必须与INTO子句中的顺序一致

插入元组 3

- [例3.69]将一个新学生元组（学号：201215128;姓名：陈冬;性别：男;所在系：IS;年龄：18岁）插入到Student表中。

```
INSERT
INTO  Student (Sno,Sname,Ssex,Sdept,Sage)
VALUES ('201215128','陈冬','男','IS',18);
```

- [例3.70]将学生张成民的信息插入到Student表中。

```
INSERT
INTO  Student
VALUES ('201215126','张成民','男',18,'CS');
```

插入元组 4

- [例3.71] 插入一条选课记录（ '200215128', '1' ）

```
INSERT
INTO SC(Sno,Cno)
VALUES ('201215128', '1');
```

  - 关系数据库管理系统将在新插入记录的Grade列上自动地赋空值。
  - 或者：

```
INSERT
INTO SC
VALUES ('201215128', '1', NULL);
```

插入子查询结果 1

- 语句格式

  INSERT
  INTO <表名> [(<属性列1> [,<属性列2>…)]
  子查询;

- INTO子句

- 子查询
  - SELECT子句目标列必须与INTO子句匹配
    - 值的个数 & 值的类型

插入子查询结果 2

[例3.72]对每一个系，求学生的平均年龄，并把结果存入数据库

- 第一步：建表

```
CREATE  TABLE  Dept_age (
    Sdept  CHAR(15)                  /* 系名 */
    Avg_age  SMALLINT );             /* 学生平均年龄 */
```

- 第二步：插入数据

```
INSERT
INTO  Dept_age(Sdept,  Avg_age)
      SELECT  Sdept,  AVG(Sage)
      FROM    Student
      GROUP BY  Sdept;
```

修改数据 1

❑语句格式

```
UPDATE  <表名>
SET  <列名> = <表达式>  [ , <列名> = <表达式> ] …
[WHERE  <条件>] ;
```

❑功能

- ➢修改指定表中满足WHERE子句条件的元组
- ➢SET子句给出<表达式>的值用于取代相应的属性列
- ➢如果省略WHERE子句，表示要修改表中的所有元组

修改数据 2

修改某一个元组的值

[例3.73]将学生201215121的年龄改为22岁

```
UPDATE  Student
SET  Sage=22
WHERE  Sno = '201215121';
```

修改多个元组的值

[例3.74]将所有学生的年龄增加1岁。

```
UPDATE Student
SET Sage = Sage + 1;
```

修改数据 3 -- 带子查询的修改语句

- [例3.75] 将计算机科学系全体学生的成绩置零。

```
UPDATE  SC
SET   Grade=0
WHERE  Sno  IN (
                    SELETE  Sno
                    FROM   Student
                    WHERE  Sdept = 'CS' );
```

修改数据 4 -- 修改数据与完整性

- 关系数据库管理系统在执行修改语句时会检查修改操作是否破坏表上已定义的完整性规则
  - 实体完整性
  - 参照完整性
  - 用户定义的完整性
    - NOT NULL约束
    - UNIQUE约束
    - 值域约束

删除数据 1

- 语句格式

```
DELETE
FROM    <表名>
[ WHERE <条件> ];
```

- 功能
  - 删除指定表中满足WHERE子句条件的元组

- WHERE子句
  - 指定要删除的元组
  - 缺省表示要删除表中的全部元组，表的定义仍在字典中

删除数据 2

删除某一个元组的值

[例3.76] 删除学号为201215128的学生记录。

```
DELETE
FROM Student
WHERE Sno = '201215128';
```

删除多个元组的值

[例3.77] 删除所有的学生选课记录。

```
DELETE
FROM SC;
```

删除数据 3 -- 带子查询的删除语句

[例3.78] 删除计算机科学系所有学生的选课记录。

```
DELETE
FROM   SC
WHERE  Sno   IN (
                  SELETE   Sno
                  FROM     Student
                  WHERE    Sdept= 'CS') ;
```

数据管理基础

3.6 SQL中的空值

智能软件与工程学院

空值的处理

- 空值就是“不知道”或“不存在”或“无意义”的值。

- 一般有以下几种情况：
  - 该属性应该有一个值，但目前不知道它的具体值
  - 该属性不应该有值
  - 由于某种原因不便于填写

- 空值是一个很特殊的值，含有不确定性。对关系运算带来特殊的问题，需要做特殊的处理。下面从空值的产生、判断、完整性约束、运算等方面来阐述空值处理规则。

空值的产生 1

- [例3.79]向SC表中插入一个元组，学生的学号是“201215126”，课程号是“1”，成绩为空。

```
INSERT  INTO SC(Sno, Cno, Grade)
VALUES('201215126', '1', NULL);
/* 该学生还没有考试成绩，故在成绩字段上写入的是空值NULL */
```

- 或

```
INSERT  INTO  SC(Sno, Cno)
VALUES('201215126', '1');
/* 在插入元组时没有赋值的属性，DBMS自动将该属性赋为空值 */
```

空值的产生 2

- [例3.80]将Student表中学生号为“201215200”的学生所属的系改为空值。

```
UPDATE Student
SET Sdept = NULL
WHERE Sno='201215200';
```

空值的判断

- 判断一个属性的值是否为空值，用IS NULL或IS NOT NULL来表示。

- [例3.81]从Student表中找出漏填了数据的学生信息

```
SELECT *
FROM   Student
WHERE  Sname  IS NULL  or
       Ssex  IS NULL  or
       Sage  IS NULL  or
       Sdept  IS NULL;
```

空值相关的完整性约束

- 属性定义（或者域定义）中
  - 有 NOT NULL 约束条件的不能取空值
  - 主码（primary key）属性不能取空值
  - UNIQUE唯一性约束：当属性值非空时，其值在表中具有唯一性

- 可以通过 UNIQUE + NOT NULL 约束来确保表中元组的唯一性

空值的算术运算、比较运算和逻辑运算 1

- 运算规则
  - 空值与另一个值（包括另一个空值）的算术运算的结果为空值
  - 空值与另一个值（包括另一个空值）的比较运算的结果为UNKNOWN
  - 有UNKNOWN后，传统二值（TRUE，FALSE）逻辑就扩展成了三值逻辑

- 在数据查询语句的WHERE子句和HAVING子句中，只有条件表达式的判断结果为TRUE时，当前元组（或组GROUP）才被作为结果输出。

- 因此，在关系数据库管理系统中，一般不支持三值逻辑，统一规定：空值参与比较运算的结果为逻辑假FALSE！

空值的算术运算、比较运算和逻辑运算 2

- [例3.82] 找出选修1号课程不及格的学生。

```
SELECT  Sno
FROM    SC
WHERE   Grade < 60  AND  Cno = '1' ;
```

- 查询结果不包括缺考的学生，因为他们的Grade值为NULL。

空值的算术运算、比较运算和逻辑运算 3

❑[例3.83] 选出选修1号课程的不及格的学生以及缺考的学生。

❑ (解1)

```
SELECT  Sno
FROM    SC
WHERE  Grade < 60  AND
           Cno = '1'

    UNION

SELECT  Sno
FROM    SC
WHERE  Grade IS NULL  AND
           Cno = '1';
```

❑ (解2) 或者

```
SELECT  Sno
FROM    SC
WHERE  Cno = '1'  AND
      ( Grade < 60  OR
            Grade IS NULL );
```

❑[思考]能否用下述语句来实现查询3.83?

```
SELECT  Sno
FROM    SC
WHERE  Cno = '1'  AND
      NOT ( Grade >= 60 );
```

数据管理基础

3.7 视图

智能软件与工程学院

视图

❑视图的特点

- ➢虚表，是从一个或几个基本表（或视图）导出的表
- ➢只存放视图的定义，不存放视图对应的数据
- ➢基表中的数据发生变化，从视图中查询出的数据也随之改变

定义视图-建立视图 1

- 语句格式

```
CREATE VIEW <视图名> [(<列名> [,<列名>]...)]
    AS <子查询>
    [WITH CHECK OPTION];
```

- WITH CHECK OPTION
  - 对视图进行UPDATE，INSERT和DELETE操作时要保证更新、插入或删除的行满足视图定义中的谓词条件（即子查询中的条件表达式）

- 子查询可以是任意的SELECT语句，是否可以含有ORDER BY子句和DISTINCT短语，取决于具体系统的实现。
  - 在标准SQL中，视图对应的子查询中不允许有ORDER BY子句，但可以使用DISTINCT短语

定义视图-建立视图 2

- 组成视图的属性列名：全部省略 或 全部指定
  - 全部省略
    - 由子查询中SELECT目标列中的诸字段的名字作为视图中对应列的列名
  - 明确指定视图的所有列名
    - 具体到每一列，可以继续沿用子查询中对应目标列的列名，也可以启用一个新的更合适的列名，但必须保证视图中列名的唯一性
    - 当出现下列情况时，必须明确指定视图的所有列名
      - 某个目标列是聚集函数或列表达式
      - 多表连接时选出了几个同名列作为视图的字段
      - 需要在视图中为某个列启用新的更合适的名字

定义视图-建立视图 3

- 关系数据库管理系统执行CREATE VIEW语句时只是把视图定义存入数据字典，并不执行其中的SELECT语句。
- 在对视图查询时，按视图的定义从基本表中将数据查出。

- [例3.84] 建立信息系学生的视图。

```
CREATE  VIEW  IS_Student
AS
        SELECT  Sno, Sname, Sage
        FROM    Student
        WHERE  Sdept = 'IS';
```

定义视图-建立视图 4

- [例3.85]建立信息系学生的视图，并要求进行修改和插入操作时仍需保证该视图只有信息系的学生。

```
CREATE  VIEW  IS_Student
AS
    SELECT Sno,Sname,Sage
    FROM  Student
    WHERE  Sdept = 'IS'
    WITH CHECK OPTION;
```

- 定义IS_Student视图时加上了WITH CHECK OPTION子句，其作用是：如果允许在该视图上执行更新操作，则其更新后的结果元组仍然必需满足视图的定义条件。即通过该视图插入或修改后的新元组能够通过该视图上的查询操作查出来。

- 视图定义中的WITH CHECK OPTION子句，不影响直接在基本表Student上的元组插入和修改操作的执行。

WITH CHECK OPTION 示例

❑例：有如下的信息系学生视图。

```
CREATE  VIEW  IS_Stud
AS
    SELECT Sno,Sname,Sdept
    FROM  Student
    WHERE  Sdept = 'IS'
    WITH CHECK OPTION;
```

❑例：下述元组插入和修改操作都不会得到执行：

```
INSERT  INTO  IS_Stud
VALUES(231215101,‘王海’,‘CS’);

UPDATE  IS_Stud
SET  Sdept = ‘CS’
WHERE  Sno = ‘201215121’;
```

❑ 但是，在基本表Student上可以执行

```
INSERT
INTO  Student(Sno,Sname,Sdept)
VALUES(231215101,‘王海’,‘CS’);

UPDATE  Student
SET  Sdept = ‘CS’
WHERE  Sno = ‘201215121’;
```

定义视图-行列子集视图

- [例3.85]信息系学生视图

```
CREATE  VIEW  IS_Student
AS
    SELECT Sno,Sname,Sage
    FROM  Student
    WHERE  Sdept = 'IS'
    WITH CHECK OPTION;
```

- 若一个视图是从单个基本表导出的，并且只是去掉了基本表的某些行和某些列，但保留了主码，我们称这类视图为行列子集视图。
  - IS_Student视图就是一个行列子集视图。

定义视图-建立视图 5

基于多个基表的视图

[例3.86]建立信息系选修了1号课程的学生的视图（包括学号、姓名、成绩）。

```
CREATE  VIEW  IS_S1(Sno, Sname, Grade)
AS
    SELECT  Student.Sno, Sname, Grade
    FROM  Student,  SC
    WHERE  Sdept = 'IS'  AND
                Student.Sno = SC.Sno  AND
                SC.Cno = '1';
```

基于视图的视图

[例3.87]建立信息系选修了1号课程且成绩在90分以上的学生的视图。

```
CREATE  VIEW  IS_S2
AS
    SELECT  Sno, Sname, Grade
    FROM   IS_S1
    WHERE  Grade >= 90;
```

定义视图-建立视图 6

带表达式的视图

[例3.88]定义一个反映学生出生年份的视图。

```
CREATE  VIEW BT_S(Sno, Sname, Sbirth)
AS
    SELECT  Sno,  Sname,  2014 – Sage
    FROM  Student;
```

分组视图

[例3.89]将学生的学号及平均成绩定义为一个视图。

```
CREAT  VIEW    S_G(Sno, Gavg)
AS
    SELECT  Sno, AVG(Grade)
    FROM    SC
    GROUP BY  Sno;
```

定义视图-建立视图 7

- [例3.90] 将Student表中所有女生记录定义为一个视图

```
CREATE VIEW F_Student(F_Sno,name,sex,age,dept)
AS
        SELECT *   /*不指定属性列*/
        FROM    Student
        WHERE  Ssex = '女';
```

- 缺点：
  - 修改基表Student的结构后，Student表与 F_Student视图 的映象关系被破坏，导致该视图不能正确工作。

定义视图-删除视图 1

- 语句的格式：

  DROP VIEW <视图名> [CASCADE];

  - 该语句从数据字典中删除指定的视图定义
  - 如果该视图上还导出了其他视图，使用CASCADE级联删除语句，把该视图和由它导出的所有视图一起删除。
    - 否则，系统将拒绝当前的视图删除操作。
  - 在删除基本表时如果使用了CASCADE选项，由该基本表导出的所有视图定义都会被连带一起删除；否则，必须先显式地使用DROP VIEW语句删除所有相关的视图，然后才能再删除基本表。

定义视图-删除视图 2

- [例3.91] 删除视图BT_S和IS_S1

```
DROP VIEW BT_S;        /*成功执行*/
DROP VIEW IS_S1;       /*拒绝执行*/
```

- 要删除IS_S1，需使用级联删除：

```
DROP VIEW IS_S1 CASCADE;
```

视图上的操作

- 对视图可以作查询操作
  - 视图上的查询操作将首先被改写为基表上的查询操作，然后才能得到执行
- 一般不允许执行视图上的更新操作，只有满足以下条件才可以：
  - ① 视图的每一行必须对应基表的惟一一行
  - ② 视图的每一列必须对应基表的惟一一列
- 同时满足上述两个条件的视图被称为‘可更新视图’(updateable view)。
- 视图上的数据更新操作需要被转化为基表上的数据更新操作，只有在可更新视图上才允许执行数据更新操作(insert, update, delete)。

查询视图 1

- 用户角度：查询视图与查询基本表相同

- 关系数据库管理系统实现视图查询的方法
  - 视图消解法（View Resolution）
    - 进行有效性检查
    - 转换成等价的对基本表的查询
    - 执行修正后的查询

查询视图 2

- [例3.92]在信息系学生的视图中找出年龄小于20岁的学生。

```
SELECT   Sno,Sage
FROM     IS_Student
WHERE    Sage<20;
```

- [例3.93]查询选修了1号课程的信息系学生

```
SELECT  IS_Student.Sno, Sname
FROM    IS_Student, SC
WHERE   IS_Student.Sno=SC.Sno
        AND SC.Cno= '1';
```

- 视图消解转换后的查询语句为：

```
SELECT  Sno,Sage
FROM  Student
WHERE  Sdept= 'IS'  AND
       Sage<20;
```

查询视图 3

- [例3.94]在S_G视图中查询平均成绩在90分以上的学生学号和平均成绩

```
SELECT  *
FROM    S_G
WHERE  Gavg>=90;
```

- S_G视图的子查询定义：

```
CREATE VIEW  S_G (Sno,Gavg)
AS
     SELECT  Sno, AVG(Grade)
     FROM   SC
     GROUP BY  Sno;
```

- 视图消解后得到的查询：

```
SELECT  Sno,AVG(Grade)
FROM  SC
GROUP BY Sno
HAVING AVG(Grade)>=90;
```

- 或者，直接在基本表上组织查询

```
SELECT *
FROM  (SELECT Sno,AVG(Grade)
          FROM  SC
          GROUP BY Sno)
          AS S_G(Sno,Gavg)
WHERE Gavg>=90;
```

更新视图 1

- [例3.95] 将信息系学生视图IS_Student中学号“201215122”的学生姓名改为“刘辰”。

```
UPDATE  IS_Student
SET  Sname = '刘辰'
WHERE  Sno = '201215122';
```

  - 转换后的语句：

```
UPDATE  Student
SET  Sname = '刘辰'
WHERE  Sno = '201215122'  AND  Sdept = 'IS';
```

更新视图 2

- [例3.96] 向信息系学生视图IS_S中插入一个新的学生记录，其中学号为“201215129”，姓名为“赵新”，年龄为20岁

```
INSERT
INTO  IS_Student
VALUES(‘201215129’, ’赵新’, 20);
```

- 转换为对基本表的更新：

```
INSERT
INTO  Student(Sno, Sname, Sage, Sdept)
VALUES(‘200215129 ', '赵新', 20, 'IS' );
```

更新视图 3

- [例3.97]删除信息系学生视图IS_Student中学号为“201215129”的记录

```
DELETE
FROM  IS_Student
WHERE  Sno = '201215129';
```

- 转换为对基本表的更新：

```
DELETE
FROM  Student
WHERE  Sno = '201215129'  AND  Sdept = 'IS';
```

更新视图 4

- 允许对行列子集视图进行更新
- 对其他类型视图的更新不同系统有不同限制
  - 更新视图的限制：一些视图是不可更新的，因为对这些视图的更新不能唯一地有意义地转换成对相应基本表的更新

- 例：例3.89定义的视图S_G为不可更新视图。

```
UPDATE  S_G
SET     Gavg = 90
WHERE  Sno = '201215121';
```

- 这个对视图的更新无法转换成对基本表SC的更新

更新视图 5

- 例：将SC中成绩在平均成绩之上的元组定义成一个视图

```
CREATE VIEW  GOOD_SC
AS
    SELECT  Sno,Cno,Grade
    FROM    SC
    WHERE  Grade > (
                    SELECT  AVG(Grade)
                    FROM    SC              );
```

- 在一个不允许更新的视图上定义的新视图也不允许更新。

视图的作用 - 视图能够简化用户的操作

- 当视图中数据不是直接来自基本表时，定义视图能够简化用户的操作
  - 基于多张表连接形成的视图
  - 基于复杂嵌套查询的视图
  - 含导出属性的视图

视图的作用 - 视图使用户能以多种角度看待同一数据

- 视图机制能使不同用户以不同方式看待同一数据，
- 适应数据库共享的需要

视图的作用 - 视图对重构数据库提供了一定程度的逻辑独立性

- 例：学生关系Student(Sno,Sname,Ssex,Sage,Sdept)
  - “垂直”地分成两个基本表：

    SX(Sno, Sname, Sage)　和　SY(Sno, Ssex, Sdept)

- 通过建立一个视图Student：

```
CREATE VIEW  Student(Sno,Sname,Ssex,Sage,Sdept)
AS
      SELECT  SX.Sno,SX.Sname,SY.Ssex,SX.Sage,SY.Sdept
      FROM  SX,SY
      WHERE  SX.Sno = SY.Sno;
```

- 使用户的外模式保持不变，用户应用程序通过视图仍然能够查找数据
- 由于对视图的更新是有条件的，因此应用程序中修改数据的语句可能仍会因基本表结构的改变而改变。

视图的作用 - 视图能够对机密数据提供安全保护

- 对不同用户定义不同视图，使每个用户只能看到他有权看到的数据

视图的作用 - 适当的利用视图可以更清晰的表达查询

- 经常需要执行这样的查询“对每个同学找出他获得最高成绩的课程号”。
- 可以先定义一个视图，求出每个同学获得的最高成绩

```
CREATE VIEW  VMGRADE
AS
    SELECT  Sno, MAX(Grade)  Mgrade
    FROM  SC
    GROUP BY  Sno;
```

- 然后用如下的查询语句完成查询：

```
SELECT  SC.Sno,Cno
FROM  SC, VMGRADE
WHERE  SC.Sno=VMGRADE.Sno  AND
          SC.Grade=VMGRADE .Mgrade;
```

复习思考题 4

8. 在使用字符匹配查询谓词like时，<匹配串>中可以使用哪些通配符？它们的含义分别是什么？

9. 关于聚集函数
   ① 在SQL语言中，引入了哪几种聚集函数？
   ② 哪一种聚集函数的统计结果是与被统计对象的元素类型无关？
   ③ 在被统计的对象集合中存在取值为空值的元素时，对聚集函数的计算将会产生什么样的影响？
   ④ count(*)、count(cid)、count(distinct cid)三者之间有什么区别？

10. 在使用到group by子句的SQL查询中，目标列可以由哪些部分组成？

11. 关于视图（view）
   ① SQL语言中的‘基本表’和关系模型中的‘关系’有什么区别？
   ② 在SQL语言中，‘视图’和‘基本表’有什么联系与区别？
   ③ 在视图创建命令中，WITH CHECK OPTION选项的作用是什么？
   ④ 在视图创建命令和基表创建命令中，定义内容有什么区别？
   ⑤ 在SQL中创建视图的作用是什么？

复习思考题 5

12. 设有一个教务管理数据库，其中的关系模式如下：

关系	属性	关系模式
学生	学号, 姓名, 就读院系, 入学年份	Student (sno, sname, dept, regyear)
教师	工号, 姓名, 工作院系	Teacher (tno, tname, dept)
课程	课程号, 课程名, 开课院系, 学分, 课程类型	Course (cno, cname, dept, credit, optional)
选课	学号, 课程号, 授课教师工号, 选修年份, 选修学期, 成绩	SC (sno, cno, tno, scyear, semester, grade)
授课	主讲教师工号, 课程号, 授课年份, 授课学期, 课时数	TC (tno, cno, tcyear, semester, hours)

其中：

- 学号、工号、课程号分别是学生、教师、课程表的主码；
- 同一门课同一个学生只能有一条选课记录，一门课一位老师在同一年的同一个学期内只能有一条授课记录；
- 成绩全部采用百分制，成绩大于或等于60分才能获得该门课程的学分；
- 单门课程的学分绩计算公式是：成绩$grade \times 0.05 \times$学分$credit$；
- 课程类型分‘核心’和‘选修’两种。

复习思考题 6

（第12题）请用SQL语言来表示下述数据访问请求。

1. 查询满足下述条件的课程的课程名及开课院系：课程名中包含有‘数据库’；
2. 查询满足下述条件的课程的课程号和课程名：‘开课院系’为空；
3. 查询满足下述条件的教师的工号和姓名：在2021至2023年间没有承担过授课任务；
4. 查询满足下述条件的教师的工号和姓名：只讲授过自己工作所在院系开设的课程；
5. 查询满足下述条件的学生的学号和姓名：修读并获得了‘计算机’系开设的所有‘核心’课的学分；
6. 查询满足下述条件的学生的学号、姓名和就读院系：2020年之前入学，但还没有修读完自己就读院系开设的所有‘核心’课；
7. 查询每一位老师第一次上课和最后一次上课的年份，结果返回教师工号和起、止授课年份；
8. 查询每一位教师的授课次数及累计课时数，结果只返回累计课时数满1000小时的教师的工号、授课次数、累计课时数，并按照累计课时数的降序输出结果；
9. 针对所有‘核心’课程都及格的同学，统计每一个人的平均学分绩（不含不及格课程），结果返回学号、就读院系、平均学分绩，并按照‘就读院系的升序’和‘平均学分绩的降序’输出结果；
10. 创建一个学生学习情况统计视图，视图中的属性包括：学号，姓名，就读院系，已获得的学分总数，‘核心’课程的平均学分绩。