2026年春季学期

数据库系统概论

An Introduction to Database Systems

第三章 关系数据库标准语言SQL

中国科学技术大学
人工智能与数据科学学院

黄振亚，huangzhy@ustc.edu.cn

第三章 关系数据库标准语言SQL

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.5.1 插入数据

- 两种插入数据方式

  1. 插入元组

  2. 插入子查询结果（学习过select后再介绍）

     - 可以一次插入多个元组

一、插入元组

- 语句格式

  INSERT

  INTO <表名> [(<属性列1>[，<属性列2 >…)]

  VALUES (<常量1> [，<常量2>] …)

- 功能
  - 将新元组插入指定表中

插入元组（续）

- 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

Into子句没有出现的属性列，新元组在这些列上取Null;
在表定义时说明了Not Null的属性列不能取空，否则报错;

插入元组（续）

- **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（续）

［例3.71］ 将一个新学生元组（学号： 20180009；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 ('20180009', '陈冬', '男', 'IS', 18);

```
INSERT  INTO  Student (Sno, Sname, Ssex, Sdept, Sage)
VALUES ('20180009', '陈冬', '男', 'IS', 18);
```

查看：

```
select * from Student where Sname='陈冬'
```

Result Grid Filter Rows: Search

Sno	Sname	Ssex	Sage	Sdept
20180009	陈冬	男	18	IS

插入元组（续）

［例3.72］ 将学生张成民的信息插入到Student表中。

INSERT INTO Student

VALUES (‘20180008’, ‘张成民’, ‘男’, 18, ‘CS’);

```
INSERT  INTO  Student
VALUES ('20180008', '张成民', '男', 18, 'CS');
```

查看：

```
select * from Student where Sname='张成民'
```

Result Grid | Filter Rows: Search

Sno	Sname	Ssex	Sage	Sdept
20180008	张成民	男	18	CS

插入元组（续）

［例3.72］ 将学生张成民的信息插入到Student表中。

```
INSERT INTO Student
VALUES ('20180008', '张成民', '男', 18, 'CS');
```

```
INSERT INTO Student (Sno, Sname, Ssex, Sdept, Sage)
VALUES ('20180008', '张成民', '男', 'CS', 18);
```

选择哪种？

插入元组（续）

［例3.73］ 插入一条选课记录(‘20180005’, ‘4’, ‘20202’)。

```
INSERT
INTO SC(Sno, Cno, Semester)
VALUES (‘20180005’, ‘4’, ‘20202’);
```

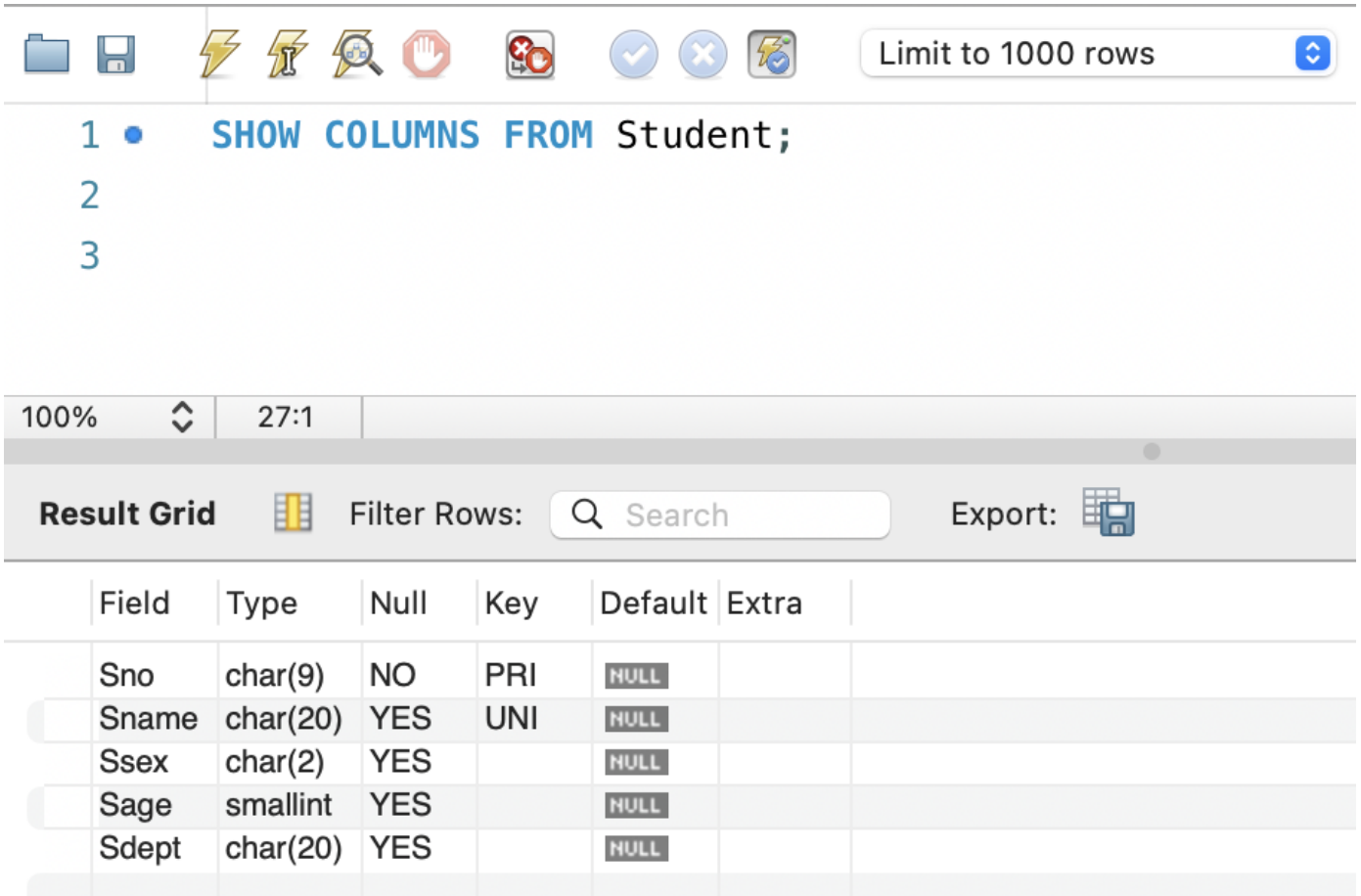
- RDBMS将在新插入记录的Grade列上自动地赋空值。

或者：

```
INSERT
INTO SC
VALUES (‘20180005’, ‘4’, NULL, ‘20202’);
```

查询表中有哪些属性列

- SHOW COLUMNS FROM 表名;



二、插入子查询结果（select学过以后再介绍）

- 语句格式

  INSERT

  INTO <表名> [(<属性列1> [，<属性列2>…)]

  子查询；

- 功能：将子查询结果插入指定表中
- INTO子句(与插入元组类似)
- 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（续）

[例3.74] 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;
```

插入数据

- RDBMS在执行插入语句时会检查所插元组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

PS：RDBMS检查功能有限，应用程序中做更复杂的检查数据完整性非常重要

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

修改数据（续）

- 三种修改方式
  1. 修改某一个元组的值
  2. 修改多个元组的值
  3. 带子查询的修改语句

3.4.2 修改数据

- 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（续）

- **SET子句**
  - 指定修改方式
  - 要修改的列
  - 修改后取值
- **WHERE子句**
  - 指定要修改的元组
  - 缺省表示要修改表中的所有元组

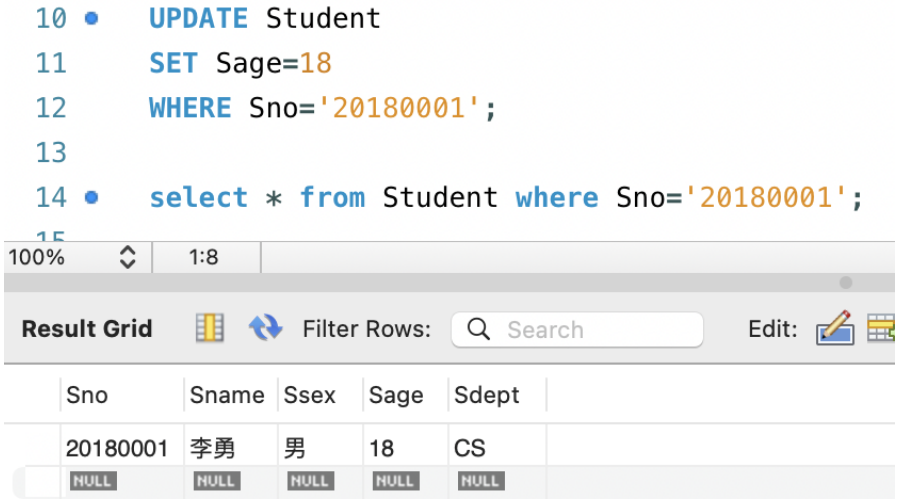
1. 修改某一个元组的值

[例3.75] 将学生20180001的年龄改为18岁

```
UPDATE  Student
SET Sage=18
WHERE  Sno='20180001';
```

修改前

Sno	Sname	Ssex	Sage	Sdept
20180001	李勇	男	20	CS



2. 修改多个元组的值

[例3.76] 将所有学生的年龄增加1岁

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

✕ Error Code: 1175. You are using safe update mode and you tried to...

执行后可能会报错！主要是因为Mysql运行在safe-updates模式下，该模式会导致非主键条件下无法执行update或者delete命令！

```
16 •  SET SQL_SAFE_UPDATES = 0;
17
18 •  UPDATE Student
19    SET Sage=Sage+1;
20
21 •  select * from Student where Sno='20180001' or Sno='20180008';
22
```

100% 25:14

Result Grid | Filter Rows: Search | Edit: | Export/Import:

Sno	Sname	Ssex	Sage	Sdept
20180001	李勇	男	19	CS
20180008	张成民	男	19	CS

3. 带子查询的修改语句

[例3.77] 将计算机科学系全体学生的成绩置0。

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

修改数据（续）

- RDBMS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
  - 主码不允许修改
  - 用户定义的完整性
    - NOT NULL约束
    - UNIQUE约束
    - 值域约束

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.5.3 删除数据

- 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（续）

- 三种删除方式

1. 删除某一个元组的值

2. 删除多个元组的值

3. 带子查询的删除语句

1. 删除某一个元组的值

[例3.78] 删除学号为20180008的学生记录。

```
DELETE
FROM Student
WHERE Sno= '20180008';
```

删除前

Sno	Sname	Ssex	Sage	Sdept
20180008	张成民	男	19	CS

```
24 •  DELETE
25    FROM Student
26    WHERE Sno='20180008';
27
28 •  select * from Student where Sno='20180008';
29
```

100% 1:22

Result Grid Filter Rows: Search Edit:

Sno	Sname	Ssex	Sage	Sdept
NULL	NULL	NULL	NULL	NULL

2. 删除多个元组的值

[例3.79] 删除所有的学生选课记录。

```
DELETE FROM SC;
```

Drop table SC ?

3. 带子查询的删除语句

[例3.80] 删除计算机科学系所有学生的选课记录。

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

MySQL会发生什么情况？

3. 带子查询的删除语句

[例3.80] 删除计算机科学系所有学生的选课记录(Mysql 例子)。

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

✕ Error Code: 1175. You are using safe update mode and you tried to...

执行后可能会报错！主要是因为Mysql运行在safe-updates模式下，该模式会导致非主键条件下无法执行update或者delete命令！

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(Mysql 例子)。

```
SET SQL_SAFE_UPDATES = 0;
```

解除安全模式后再运行上述删除指令，即可完成删除

删除前

Sno	Cno	Grade	Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201
NULL	NULL	NULL	NULL

删除后

Sno	Cno	Grade	Semester
NULL	NULL	NULL	NULL

第三章 关系数据库标准语言SQL

数据查询

- 语句格式

SELECT [ALL|DISTINCT] <目标列表达式>[,<目标列表达式>] ...

FROM <表名或视图名>[,<表名或视图名>]... | (SELECT 语句)

[AS]<别名>

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]];

数据查询

- SELECT子句：指定要显示的属性列
- FROM子句：指定查询对象（基本表或视图）
- WHERE子句：指定查询条件
- GROUP BY子句：对查询结果按指定列的值分组，该属性列值相等的元组为一个组。通常会在每组中作用聚集函数。
- HAVING短语：只有满足指定条件的组才予以输出
- ORDER BY子句：对查询结果表按指定列值的升序或降序排序

3.4 数据查询

- 3.4.1 单表查询
- 3.4.2 连接查询
- 3.4.3 嵌套查询
- 3.4.4 集合查询
- 3.4.5 基于派生表的查询
- 3.4.6 Select语句的一般形式

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 ORDER BY子句
  - 四、 聚集函数
  - 五、 GROUP BY子句

一、 选择表中的若干列

- 查询指定列

  [例3.16] 查询全体学生的学号与姓名。

```
SELECT Sno, Sname
FROM Student;
```

  [例3.17] 查询全体学生的姓名、学号、所在系。

```
SELECT Sname, Sno, Sdept
FROM Student;
```

  - <目标列表达式>属性顺序可以不一致
  - 查询结果是一个关系

一、选择表中的若干列


```
80 •    SELECT Sno, Sname
81      FROM Student;
82
```

100% 18:77

Result Grid Filter Rows: Searc

Sno	Sname
20180002	刘晨
20180004	张立
20180001	李勇
20180003	王敏
20180005	陈新奇
NULL	NULL

```
80 •    SELECT Sno, Sname, Sdept
81      FROM Student;
82
```

100% 14:76

Result Grid Filter Rows: Search

Sno	Sname	Sdept
20180001	李勇	CS
20180002	刘晨	CS
20180003	王敏	MA
20180004	张立	IS
20180005	陈新奇	DS
NULL	NULL	NULL

一、 选择表中的若干列

- 查询指定列

[例3.16] 查询全体学生的学号与姓名。

```
SELECT Sno, Sname
FROM Student;
```

[例3.17] 查询全体学生的姓名、学号、所在系。

```
SELECT Sname, Sno, Sdept
FROM Student;
```

- <目标列表达式>属性顺序可以不一致
- 查询结果是一个关系

类似于关系代数中的哪一个操作?

2. 查询全部列

- 选出所有属性列：
  - 在SELECT关键字后面列出所有列名
  - 将<目标列表达式>指定为 *

[例3.18] 查询全体学生的详细记录。

```
SELECT  Sno, Sname, Ssex, Sage, Sdept
FROM Student;
```

或

```
SELECT  *
FROM Student;
```

Result Grid Filter Rows: Search

Sno	Sname	Ssex	Sage	Sdept
20180001	李勇	男	20	CS
20180002	刘晨	女	19	CS
20180003	王敏	女	18	MA
20180004	张立	男	19	IS
20180005	陈新奇	男	19	DS
NULL	NULL	NULL	NULL	NULL

3. 查询经过计算的值

- SELECT子句的<目标列表达式>可以为：
  - 算术表达式
  - 字符串常量
  - 函数
  - 列别名

查询经过计算的值（续）

[例3.19] 查全体学生的姓名及其出生年份。

```
SELECT Sname, 2020-Sage
                                    /算术表达式，*假定当年的年份为2020年*/
FROM Student;
```

输出结果：

Sname	2020-Sage
李勇	2000
刘晨	2001
王敏	2002
张立	2001
陈新奇	2001

查询经过计算的值（续）

[例3.20] 查询全体学生的姓名、出生年份和所有系，要求用小写字母表示所有系名

```
SELECT  Sname, 'Year of Birth: ', 2020-Sage,
                LOWER(Sdept)
FROM Student;
```

输出结果：

Sname	Year of Birth:	2020-Sage	LOWER(Sdept)
李勇	Year of Birth:	2000	cs
刘晨	Year of Birth:	2001	cs
王敏	Year of Birth:	2002	ma
张立	Year of Birth:	2001	is
陈新奇	Year of Birth:	2001	ds

查询经过计算的值（续）

- 使用列别名改变查询结果的列标题:

```
SELECT  Sname NAME, 'Year of Birth: '  BIRTH,
        2020-Sage  BIRTHDAY, LOWER(Sdept)  DEPARTMENT
FROM Student;
```

输出结果(Mysql 例子) :

```
SELECT  Sname NAME, 'Year of Birth:'  BIRTH,
        2020-Sage  BIRTHDAY, LOWER(Sdept)  DEPARTMENT
FROM Student;
```

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	2000	cs
刘晨	Year of Birth:	2001	cs
王敏	Year of Birth:	2002	ma
张立	Year of Birth:	2001	is
陈新奇	Year of Birth:	2001	ds

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 ORDER BY子句
  - 四、 聚集函数
  - 五、 GROUP BY子句

二、选择表中的若干元组

- **1. 消除取值重复的行**

  如果没有指定DISTINCT关键词，则缺省为ALL

[例3.21] 查询选修了课程的学生学号。

SELECT Sno FROM SC;

等价于：

SELECT ALL Sno FROM SC;

执行上面的SELECT语句后，结果为：

Sno
20180001
20180001
20180001
20180002
20180002

与关系代数的区别?

消除取值重复的行（续）

- 指定DISTINCT关键词，去掉表中重复的行

```
SELECT DISTINCT Sno
FROM SC;
```

执行结果：

Sno
20180001
20180002

2. 查询满足条件的元组

表3.4 常用的查询条件

查询条件	谓　词
比　较	=，>，<，>=，<=，!=，<>，!>，!<；NOT+上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空　值	IS NULL，IS NOT NULL
多重条件（逻辑运算）	AND，OR，NOT

(1) 比较大小

[例3.22] 查询计算机科学系全体学生的名单。

```
SELECT Sname
FROM Student
WHERE Sdept='CS';
```

[例3.23] 查询所有年龄在20岁以下的学生姓名及其年龄。

```
SELECT Sname, Sage
FROM  Student
WHERE Sage < 20;
```

[例3.24] 查询考试成绩有不及格的学生的学号。

```
SELECT DISTINCT Sno
FROM  SC
WHERE Grade<60;
```

执行过程？

（2）确定范围

- 谓词：BETWEEN … AND …

  NOT BETWEEN … AND …

[例3.25] 查询年龄在20~23岁（包括20岁和23岁）之间的学生的姓名、系别和年龄

```
SELECT Sname, Sdept, Sage
FROM   Student
WHERE  Sage BETWEEN 20 AND 23;
```

[例3.26] 查询年龄不在20~23岁之间的学生姓名、系别和年龄

```
SELECT Sname, Sdept, Sage
FROM   Student
WHERE Sage NOT BETWEEN 20 AND 23;
```

(3) 确定集合

- 谓词：IN <值表>，NOT IN <值表>　　<值表>是一个集合

[例3.27]查询信息系（IS）和计算机科学系（CS）学生的姓名和性别

```
SELECT Sname, Ssex
FROM  Student
WHERE Sdept IN ( 'IS', 'CS' );
```

其他语句？逻辑表达式

[例3.28]查询既不是信息系也不是计算机科学系的学生的姓名和性别。

```
SELECT Sname, Ssex
FROM Student
WHERE Sdept NOT IN ( 'IS', 'CS' );
```

(4) 字符匹配

- 谓词： [NOT] LIKE ‘<匹配串>’ [ESCAPE ‘ <换码字符>’]
- <匹配串>可以是一个完整的字符串，也可以含有通配符%和 _
  - % （百分号） 代表任意长度（长度可以为0）的字符串
    - 例如，a%b表示以a开头，以b结尾的任意长度的字符串
  - _ （下横线） 代表任意单个字符。
    - 例如，a_b表示以a开头，以b结尾的长度为3的任意字符串

(4)字符匹配

1） 匹配串为固定字符串

[例3.29] 查询学号为20180003的学生的详细情况。

```
SELECT *
FROM  Student
WHERE  Sno LIKE '20180003';
```

等价于：

```
SELECT  *
FROM  Student
WHERE Sno = '20180003';
```

字符匹配（续）

2) 匹配串为含通配符的字符串

%：任意多个字符； _：单个字符（汉字为2个字符）

[例3.30] 查询所有姓刘学生的姓名、学号和性别。

```
SELECT Sname, Sno, Ssex
FROM Student
WHERE Sname LIKE '刘%';
```

```
101 •  SELECT Sname, Sno, Ssex FROM Student
102    WHERE  Sname  LIKE  '刘%';
103
```

100% 14:98

Result Grid Filter Rows: Search E

Sname	Sno	Ssex
刘晨	20180002	女
NULL	NULL	NULL

[例3.31] 查询姓"欧阳"且全名为三个汉字的学生的姓名。

```
SELECT Sname
FROM   Student
WHERE  Sname LIKE '欧阳_ _';（两个下划线）
```

字符匹配（续）

[例3.32] 查询名字中第2个字为"阳"字的学生的姓名和学号。

```
SELECT Sname, Sno
FROM Student
WHERE Sname LIKE '__阳%';
```

[例3.33] 查询所有不姓刘的学生姓名。

```
SELECT Sname, Sno, Ssex
FROM Student
WHERE Sname NOT LIKE '刘%';
```

字符匹配（续）

3) 使用换码字符ESCAPE '\'将通配符转义为普通字符

[例3.34] 查询DB_Design课程的课程号和学分。

```
SELECT Cno, Ccredit
FROM Course
WHERE Cname LIKE 'DB\_Design' ESCAPE '\';
```

[例3.35] 查询以"DB_"开头，且倒数第3个字符为 i的课程的详细情况。

```
SELECT *
FROM Course
WHERE Cname LIKE 'DB\_%i_ _' ESCAPE '\';
```

ESCAPE '\' 表示“ \ ” 为换码字符

实例-练习

Cno	Cname	Cpno	Ccredit
1	DB_Design	1	NULL
2	DBEDesign	2	NULL
NULL	NULL	NULL	NULL

SELECT Cno FROM Course WHERE Cname LIKE “DB_Design”;

SELECT Cno FROM Course WHERE Cname LIKE “DB_Design”;

SELECT Cno FROM Course WHERE Cname LIKE “DB/_Design” escape “/”;

结果分别是什么？

实例-练习

```
120 •   SELECT * FROM Course WHERE Cname LIKE "DB\_Design";
121
```

100%

```
121 •   SELECT * FROM Course WHERE Cname LIKE "DB_Design";
122
```

Result G

100%

```
122 •   SELECT * FROM Course WHERE Cname LIKE "DB/_Design" escape "/";
123
```

Cno

Result Gr

100% 53:121

1

NULL

Result Grid Filter Rows: Search Edit: Export/Import:

Cno

Cno	Cname	Cpno	Ccredit
1	DB_Design	1	NULL
NULL	NULL	NULL	NULL

1

2

NULL

#right 能找到DB_Design
#right,这里_表示通配符，它能找出来DB_Design 和 DBEDesign
#right,这里_不表示通配符，它能找出来DB_Design

(5) 涉及空值的查询

- 谓词： IS NULL 或 IS NOT NULL
  - “IS” 不能用 “=” 代替

[例3.36] 某些学生选修课程后没有参加考试，所以有选课记录，但没有考试成绩。查询缺少成绩的学生的学号和相应的课程号

```
SELECT Sno, Cno
FROM  SC
WHERE  Grade IS NULL;
```

[例3.37] 查所有有成绩的学生学号和课程号

```
SELECT Sno, Cno
FROM  SC
WHERE  Grade IS NOT NULL;
```

(6) 多重条件查询

- 逻辑运算符：AND和 OR来联结多个查询条件
  - AND的优先级高于OR
  - 可以用括号改变优先级
- 可用来实现多种其他谓词
  - [NOT] IN
  - [NOT] BETWEEN … AND …

多重条件查询（续）

[例3.38] 查询计算机系年龄在20岁以下的学生学号，姓名和性别。

```
SELECT Sno, Sname, Ssex
FROM  Student
WHERE Sdept= 'CS' AND Sage<20;
```

Sno	Sname	Ssex	Sage	Sdept
20180001	李勇	男	20	CS
20180002	刘晨	女	19	CS
20180003	王敏	女	18	MA
20180004	张立	男	19	IS
20180005	陈新奇	男	19	DS
NULL	NULL	NULL	NULL	NULL

Result Grid   Filter Rows:

Sno	Sname	Ssex
20180002	刘晨	女
NULL	NULL	NULL

多重条件查询（续）

- 改写[例3.27]

[例3.27] 查询信息系（IS）和计算机科学系（CS）学生的姓名和性别。

```
SELECT Sname, Ssex
FROM Student
WHERE Sdept IN ( 'IS', 'CS' )
```

可改写为：

```
SELECT Sname, Ssex
FROM   Student
WHERE  Sdept= 'IS' OR Sdept= 'CS';
```

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 ORDER BY子句
  - 四、 聚集函数
  - 五、 GROUP BY子句

三、ORDER BY子句

- **ORDER BY子句（对结果进行排序）**
  - 可以按一个或多个属性列排序
  - 升序：ASC；降序：DESC；缺省值为升序

ORDER BY子句 （续）

[例3.39] 查询选修了3号课程的学生的学号及其成绩，查询结果按分数降序排列。

```
SELECT Sno, Grade
FROM  SC
WHERE  Cno= '3'
ORDER BY Grade DESC;
```

[例3.40] 查询全体学生选修课程情况，查询结果先按照课程号升序排列，同一课程中按成绩降序排列。

```
SELECT  *
FROM  SC
ORDER BY Cno, Grade DESC;
```

ORDER BY子句 （续）

[例3.39] 查询选修了3号课程的学生的学号及其成绩，查询结果按分数降序排列(Mysql 例子)

```
SELECT
Sno,
Grade
FROM SC
WHERE Cno='3'
ORDER BY Grade DESC;
```

Result Grid | Filter F

Sno	Grade
20180001	88
20180002	80

[例3.40]查询全体学生选修课程情况，查询结果先按照课程号升序排列，同一课程中按成绩降序排列(Mysql 例子)

```
SELECT *
FROM SC
ORDER BY Cno,Grade DESC;
```

Result Grid | Filter Rows:

Sno	Cno	Grade	Semester
20180001	1	92	20192
20180002	2	90	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	3	80	20201

三、ORDER BY子句

- 当排序列含空值时，排序时显示的次序由具体系统实现决定
  - ASC：排序列为空值的元组最后显示
  - DESC：排序列为空值的元组最先显示
- MySQL的实现方式

```
SELECT *
FROM Course
ORDER BY Cpno ASC;
```

Cno	Cname	Cpno	Ccredit
2	MATH	NULL	2
6	Data Processing	NULL	2
3	Information System	1	4
1	DB_Design	5	4
4	Operating System	6	3
7	Python	6	4
8	Analysis and Practice of the Data	6	3
5	Data Structure	7	4

```
SELECT *
FROM Course
ORDER BY Cpno DESC;
```

Cno	Cname	Cpno	Ccredit
5	Data Structure	7	4
4	Operating System	6	3
7	Python	6	4
8	Analysis and Practice of the Data	6	3
1	DB_Design	5	4
3	Information System	1	4
2	MATH	NULL	2
6	Data Processing	NULL	2

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 ORDER BY子句
  - 四、 聚集函数
  - 五、 GROUP BY子句

四、聚集函数

❖ 聚集函数：

- 统计元组个数， COUNT(*)
- 统计一列中值的个数

  COUNT([DISTINCT|ALL] <列名>)
- 计算一列值的总和（此列必须为数值型）

  SUM([DISTINCT|ALL] <列名>)
- 计算一列值的平均值（此列必须为数值型）

  AVG([DISTINCT|ALL] <列名>)
- 求一列中的最大值和最小值

  MAX([DISTINCT|ALL] <列名>)

  MIN([DISTINCT|ALL] <列名>)

聚集函数（续）

[例3.41] 查询学生总人数。

```
SELECT COUNT(*)
FROM  Student;
```

[例3.42] 查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT Sno)
FROM SC;
```

[例3.43] 计算1号课程的学生平均成绩。

```
SELECT AVG(Grade)
FROM SC
WHERE Cno= ' 1 ';
```

聚集函数（续）

[例3.44] 查询选修1号课程的学生最高分数。

```
SELECT MAX(Grade)
FROM SC
WHER Cno=‘1’;
```

［例3.45］查询学生20180001选修课程的总学分数。

```
SELECT SUM(Ccredit)
FROM  SC，  Course （连接查询，后面介绍）
WHERE Sno=‘20180001' AND  SC.Cno=Course.Cno;
```

聚集函数（续）

[例3.44] 查询选修1号课程的学生最高分数(Mysql 例子)

```
SELECT MAX(Grade)
FROM SC
WHERE Cno='1';
```

MAX(Grade)
92

[例3.45] 查询学生20180001选修课程的总学分数(Mysql 例子)

```
SELECT SUM(Ccredit)
FROM SC, Course
WHERE Sno='20180001' AND SC.Cno=Course.Cno;
```

SUM(Ccredit)
10

聚集函数（续）

[例3.44] 查询选修1号课程的学生最高分数(Mysql 例子)

Cno	Cname	Cpno	Ccredit
1	DB_Design	5	4
2	MATH	NULL	2
3	Information System	1	4
4	Operating System	6	3
5	Data Structure	7	4
6	Data Processing	NULL	2
7	Python	6	4
8	Analysis and Practice of the Data	6	3

Sno	Cno	Grade	Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201

[例3.45] 查询学生20180001选修课程的总学分数(Mysql 例子)

```
SELECT SUM(Ccredit)
FROM SC, Course
WHERE Sno='20180001' AND SC.Cno=Course.Cno;
```

SUM(Ccredit)
10

聚集函数（续）

- 聚集函数遇到空值NULL？
  - Count（*）
  - Count (Sage)
  - MAX (grade)
  - SUM (Ccredit)

[例3.43] 计算1号课程的学生平均成绩。

```
SELECT AVG(Grade)
FROM SC
WHERE Cno= '1';
```

如果有学生没有成绩怎么办？

课后尝试

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 ORDER BY子句
  - 四、 聚集函数
  - 五、 GROUP BY子句

五、GROUP BY子句

- **GROUP BY子句分组：**

  细化聚集函数的作用对象

  - 未对查询结果分组，聚集函数将作用于整个查询结果
  - 对查询结果分组后，聚集函数将分别作用于每个组
  - 作用对象是查询的中间结果表
  - 按指定的一列或多列值分组，值相等的为一组

GROUP BY子句（续）

[例3.46] 求各个课程号及相应的选课人数。

```
SELECT Cno，COUNT(Sno)
FROM   SC
GROUP BY Cno;
```

查询结果：

Cno	COUNT(Sno)
1	22
2	34
3	44
4	33
5	48

GROUP BY子句（续）

[例3.46] 求各个课程号及相应的选课人数。

```
SELECT Cno，COUNT(Sno)
FROM   SC
GROUP BY Cno;
```

如果删除group by?

```
SELECT Cno, COUNT(Sno)
FROM SC
GROUP BY Cno;
```

```
61 •  SELECT Cno,COUNT(Sno)
62    FROM    SC;
```

	Cno	COUNT(Sno)
▶	1	1
	2	2
	3	2

⊗	24	19:25:53	SELECT Cno,COUNT(Sno) FROM SC LIMIT 0, 1000

Error Code: 1140. In aggregated query without GROUP BY, expressi...

GROUP BY子句（续）

[例3.47] 查询2019年第2学期选修课程数超过10门的学生学号。

```
SELECT Sno
 FROM  SC
WHERE Semester='20192'
 GROUP BY Sno
 HAVING  COUNT(*) > 10;
```

GROUP BY子句（续）

[例3.47]查询2019年第2学期选修课程数超过10门的学生学号。

```
SELECT Sno
FROM  SC
WHERE Semester='20192'
GROUP BY Sno
HAVING  COUNT(*) >10;
```

执行顺序

1. WHERE Semester='20192'
2. GROUP BY Sno
3. Count(*)
4. Having

	Sno	Cno	Grade	Semester
▸	20180001	1	92	20192
	20180001	2	85	20201
	20180001	3	88	20202
	20180002	2	90	20192
	20180002	3	80	20201

sno	Count(*)
20180001	1
20180002	1

sno	Count(*)

GROUP BY子句（续）

[例3.48]查询平均成绩大于等于90分的学生学号和平均成绩

下面的语句是不对的：

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

因为WHERE子句中不能用聚集函数作为条件表达式

正确的查询语句应该是：

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

GROUP BY子句（续）

[例3.48]查询平均成绩大于等于80分的学生学号和平均成绩(Mysql 例子)

```
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno
HAVING AVG(Grade)>=80;
```

Sno	Cno	Grade	Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201

Sno	AVG(Grade)
20180001	88.3333
20180002	85.0000

GROUP BY子句（续）

- **HAVING短语与WHERE子句的区别：**
  - 作用对象不同
  - **WHERE子句作用于基表或视图，从中选择满足条件的元组**
  - **HAVING短语作用于组，从中选择满足条件的组。**

注意：where子句不能用聚集函数作为条件表达式，聚集函数只用于select子句和group by子句中的having语句

3.4 数据查询

- 3.4.1 单表查询
- 3.4.2 连接查询
- 3.4.3 嵌套查询
- 3.4.4 集合查询
- 3.4.5 基于派生表的查询
- 3.4.6 Select语句的一般形式

回顾：关系代数中的连接

- 连接 join
  - 等值连接
  - 自然连接
- 外连接
  - 左外连接
  - 右外连接

3.4.2 连接查询

- 连接查询WHERE子句：同时涉及多个表的查询
- 连接条件或连接谓词：用来连接两个表的条件

  一般格式：

  - [<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>
  - [<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2> AND [<表名2>.]<列名3>
- 连接字段：连接谓词中的列名称
  - 连接条件中的各连接字段类型必须是可比的，但名字不必是相同的

连接查询（续）

一、等值与非等值连接查询

二、自身连接

三、外连接

四、多表连接

一、等值与非等值连接查询

- 等值连接：连接运算符为 =

[例3.51] 查询每个学生及其选修课程的情况

```
SELECT  Student.*, SC.*
FROM    Student,SC
WHERE Student.Sno=SC.Sno
```

SQL中没有Join操作，连接条件一定要写

等值与非等值连接查询（续）

查询结果：

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
20180001	李勇	男	20	CS	20180001	1	92
20180001	李勇	男	20	CS	20180001	2	85
20180001	李勇	男	20	CS	20180001	3	88
20180002	刘晨	女	19	CS	20180002	2	90
20180002	刘晨	女	19	CS	20180002	3	80

连接操作的执行过程

- 嵌套循环法(NESTED-LOOP)

Sno	Sname	Ssex	Sage	Sdept
20180001	LY	M	20	CS
20180002	LC	W	19	CS
20180003	WM	W	18	MA
20180004	ZL	M	19	IS
20180005	CXQ	M	19	DS

，然后从头开始扫描表2，找到后就将表1中的第形成结果表中一个元组。

第二个元组，然后再从头开始扫描表2，逐一找到后就将表1中的第二形成结果表中一个元组。

Sno	Cno	Grade	Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201

- 重复上述操作，直到表

连接操作的执行过程

2. 排序合并法（续）

常用于=连接

- 首先按连接属性对表1和表2排序
- 对表1的第一个元组，从头开始扫描表2，顺序查找满足连接条件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组。当遇到表2中第一条大于表1连接字段值的元组时，对表2的查询不再继续

连接操作的执行过程

2. 排序合并法（续）

Sno	Sname	Ssex	Sage	Sdept
20180001	LY	M	20	CS
20180002	LC	W	19	CS
20180003	WM	W	18	MA
20180004	ZL	M	19	IS
20180005	CXQ	M	19	DS

刚才的中断点处继续顺序扫

且，找到后就将表1中的第一

结果表中一个元组。直接遇

到表2中大于表1连接字段

续

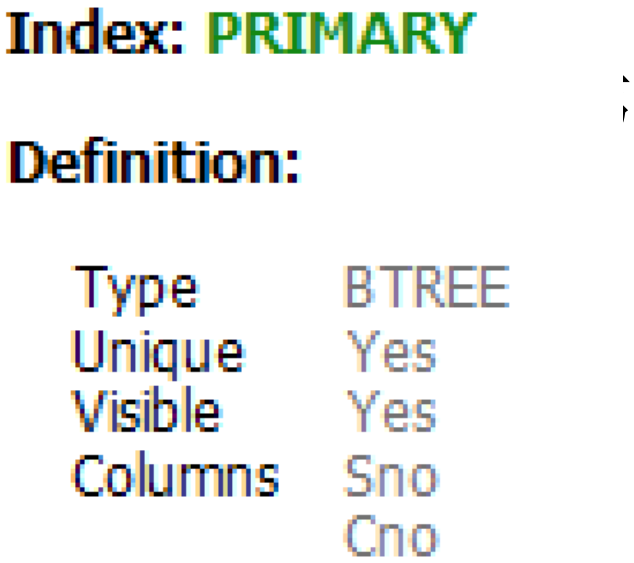
- 重复上述操作，直到表1或

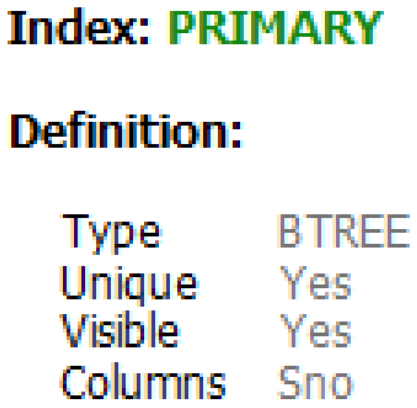
Sno	Cno	Grade	Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201

连接操作的执行过程

3. 索引连接(INDEX-JOIN)

- 对表2按连接字段建立索引
- 对表1中的每个元组，依次根据其连接字段值查询表2的索引，从中找到满足条件的元组，与之连接起来，形成一个





等值与非等值连接查询（续）

- 自然连接：将等值连接中重复的属性列去掉

[例3.52]　对[例3.51]用自然连接完成。

```
SELECT Student.Sno，Sname，Ssex，Sage，Sdept，Cno，Grade
FROM   Student，SC
WHERE  Student.Sno = SC.Sno;
```

Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
20180001	LY	M	20	CS	1	92
20180001	LY	M	20	CS	2	85
20180001	LY	M	20	CS	3	88
20180002	LC	W	19	CS	2	90
20180002	LC	W	19	CS	3	80

一、等值与非等值连接查询

- 等值连接：连接运算符为 =

[例3.49] 查询选修数据库系统概论课程且成绩排名在前10名的学生的学号。

```
SELECT  Sno
FROM    SC,Course
WHERE Course.Cname='数据库系统概论' AND
SC.Cno=Course.Cno
ORDER BY Grade DESC
LIMIT 10;          /*取前10行数据为查询结果*/
```

[例3.50] 查询平均成绩排名在第3~7名的学生的学号和平均成绩。

```
SELECT  Sno,AVG(Grade)
FROM SC
GROUP BY Sno
ORDER BY AVG(Grade) DESC
LIMIT 5 OFFSET 2;     /*取5行数据，忽略前2行，之后为查询结果数据*/
```

等值与非等值连接查询（续）

- 复合条件查询WHERE子句中含多个连接条件
- 一条SQL语句可以同时完成选择和连接查询，这时WHERE子句是由连接谓词和选择谓词组成的复合条件

[例 3.53]查询选修2号课程且成绩在90分以上的所有学生的学号和姓名

```
SELECT Student.Sno, Sname
FROM   Student, SC
WHERE  Student.Sno=SC.Sno  AND
       SC.Cno=‘ 2 ' AND SC.Grade>90;
```

- 执行过程:
  - 先从SC中挑选出Cno= ‘2'并且Grade>90的元组形成一个中间关系
  - 再和Student中满足连接条件的元组进行连接得到最终的结果关系

四、多表连接

[例3.53]查询选修2号课程且成绩在85分以上的所有学生(Mysql 例子)

```
SELECT
Student.Sno,
Sname
FROM Student, SC
WHERE Student.Sno=SC.Sno AND
SC.Cno='2' AND SC.Grade>85;
```

	Sno	Sname
▶	20180002	LC

连接查询（续）

一、等值与非等值连接查询

二、自身连接

三、外连接

四、多表连接

二、自身连接

- **自身连接：**一个表与其自己进行连接
- 需要给表起别名以示区别（空格）
- 由于所有属性名都是同名属性，因此必须使用别名前缀

[例3.54]查询每一门课的间接先修课（即先修课的先修课）

```
SELECT FIRST.Cno，SECOND.Cpno
FROM Course FIRST，Course SECOND
WHERE FIRST.Cpno = SECOND.Cno and
SECOND.Cpno IS NOT NULL;
```

自身连接（续）

FIRST表（Course表）

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	Python语言	6	4
8	数据分析及实践	6	3

SECOND表（Course表）

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	Python语言	6	4
8	数据分析及实践	6	3

自身连接（续）

[例3.54]查询每一门课的间接先修课（即先修课的先修课）

- 在同一个关系课程中进行连接，为加以区分引入别名。
- SELECT FIRST.Cno，SECOND.Cpno

  FROM Course FIRST， Course SECOND

  WHERE FIRST. Cpno = SECOND. Cno and SECOND.Cpno IS NOT NULL ；

Cno	Cpno
1	7
3	5
5	6

FIRST

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	Python语言	6	4
8	数据分析及实践	6	3

SECOND

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	Python语言	6	4
8	数据分析及实践	6	3

自身连接（续）

查询结果：

Cno	Cpno
1	7
3	5
5	6

连接查询（续）

一、等值与非等值连接查询

二、自身连接

三、外连接

四、多表连接

三、外连接

- 外连接与普通连接的区别
  - 普通连接操作只输出满足连接条件的元组
  - 外连接操作以指定表为连接主体，将主体表中不满足连接条件的元组一并输出

[例 3.55] 改写[例3.51]查询每个学生及其选修课程的情况

```
SELECT Student.Sno，Sname，Ssex，Sage，Sdept，Cno，Grade
FROM  Student  LEFT OUTER JOIN SC ON (Student.Sno=SC.Sno)；
```

外连接（续）

执行结果：

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
20180001	李勇	男	20	CS	1	92
20180001	李勇	男	20	CS	2	85
20180001	李勇	男	20	CS	3	88
20180002	刘晨	女	19	CS	2	90
20180002	刘晨	女	19	CS	3	80
20180003	王敏	女	18	MA	NULL	NULL
20180004	张立	男	19	IS	NULL	NULL

外连接（续）

- 左外连接
  - 列出左边关系（如本例Student）中所有的元组
- 右外连接
  - 列出右边关系中所有的元组
- 全外连接
  - 列出两边关系中所有的元组
- 不符合连接条件的元组的另一个关系的属性取空值

RIGHT OUTER JOIN
INNER JOIN ?

连接查询（续）

一、等值与非等值连接查询

二、自身连接

三、外连接

四、多表连接

四. 多表连接

- 多表连接：两个以上的表进行连接

[例3.56]查询每个学生的学号、姓名、选修的课程名及成绩

```
SELECT Student.Sno，Sname，Cname，Grade
FROM   Student，SC，Course   /*多表连接*/
WHERE Student.Sno = SC.Sno
      and SC.Cno = Course.Cno;
```