第三章　关系数据库标准语言SQL

第三章　关系数据库标准语言SQL

第三章 课程内容矩阵

知识单元	内容与要求		掌握程度
	一级知识点	二级知识点	
关系数据库 标准语言SQL	SQL 基本概念	视图、基本表、SQL标准、存储文件	3
	数据定义	模式定义、模式删除、基本表定义、表删除、表修改、数据类型、索引定义、索引删除、唯一性索引、聚簇索引	3
	数据查询	单表查询、多表查询、嵌套查询、聚集函数、集合查询	3
	数据更新	插入、删除、修改	3
	空值	空值的概念和处理	2
	视图	定义、删除、查询、更新、视图作用	3

SQL标准的进展过程

标准	大致页数	发布日期
SQL/86		1986.10
SQL/89(FIPS 127-1)	120页	1989年
SQL/92	622页	1992年
SQL99(SQL 3)	1700页	1999年
SQL2003	3600页	2003年
SQL2008	3777页	2006年
SQL2011		2010年
SQL2016		2016年

没有一个数据库系统能够支持SQL标准的所有概念和特性

3.1 SQL概述

- SQL的特点
  - 1. 综合统一
  - 2. 高度非过程化
  - 3. 面向集合的操作方式
  - 4. 以同一种语法结构提供两种使用方法
  - 5. 语言简洁，易学易用

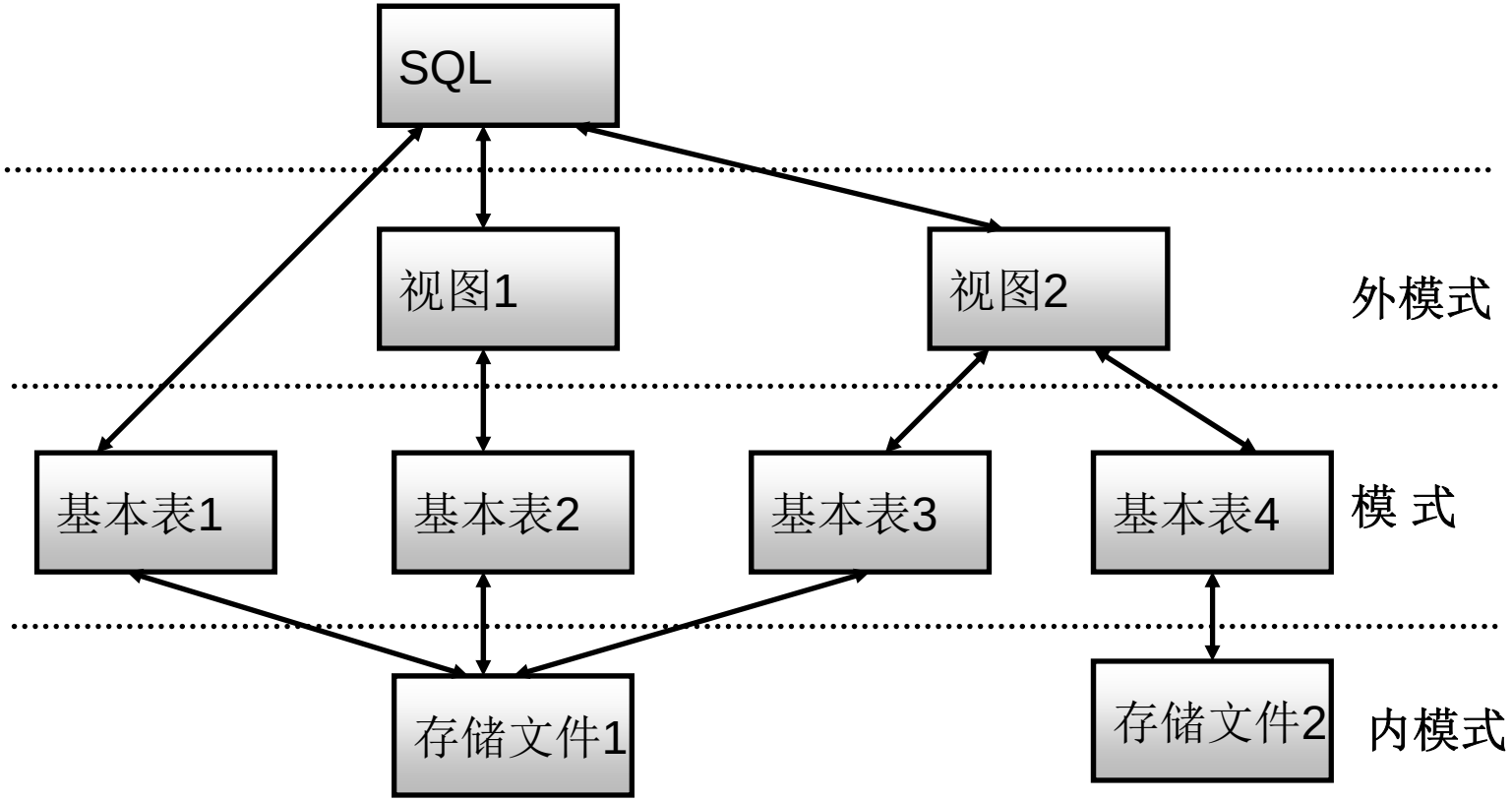
表 3.2　SQL 的动词

SQL 功　能	动词
数 据 查 询	SELECT
数 据 定 义	CREATE，DROP，ALTER
数 据 操 纵	INSERT, UPDATE, DELETE
数 据 控 制	GRANT，REVOKE

SQL功能极强，完成核心功能只用了9个动词

SQL的基本概念

SQL支持关系数据库三级模式结构



第三章　关系数据库标准语言SQL

3.1 SQL概述

3.2 数据定义

3.3 查询

3.4 数据更新

3.5 视图

3.6 数据控制

3.7 嵌入式SQL

3.8 小结

3.2 数 据 定 义

表 3.2 SQL 的数据定义语句

操作对象	操作方式		
	创 建	删 除	修 改
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视 图	CREATE VIEW	DROP VIEW	
索 引	CREATE INDEX	DROP INDEX	

3.2.1 定义基本表

CREATE TABLE <表名>

（<列名> <数据类型>[<列级完整性约束条件>]

[, <列名> <数据类型>[<列级完整性约束条件>]] …

[, <表级完整性约束条件>] ）；

- <表名>：所要定义的基本表的名字
- <列名>：组成该表的各个属性（列）
- <列级完整性约束条件>：涉及相应属性列的完整性约束条件
- <表级完整性约束条件>：涉及一个或多个属性列的完整性约束条件

例　题

[例1]　建立一个“学生”表Student，它由学号Sno、姓名Sname、性别Ssex、年龄Sage、所在系Sdept五个属性组成。其中学号不能为空，值是唯一的，并且姓名取值也唯一。

```
CREATE TABLE Student
        (  Sno       CHAR(5)   NOT NULL   UNIQUE,
           Sname     CHAR(20)   UNIQUE,
           Ssex      CHAR(1) ,
           Sage      INT,
           Sdept     CHAR(15)  );
```

例　　题　（续）

Sno	Sname	Ssex	Sage	Sdept

Sno	Sname	Ssex	Sage	Sdept
字符型 长度为5 不能为空值	字符型 长度为20	字符型 长度为1	整数	字符型 长度为15

定义基本表（续）

- 常用完整性约束
  - 主码约束： PRIMARY KEY
  - 唯一性约束：UNIQUE
  - 非空值约束：NOT NULL
  - 参照完整性约束

PRIMARY KEY与 UNIQUE的区别？

例　题　（续）

- [例2]　建立一个“学生选课”表SC，它由学号Sno、课程号Cno，修课成绩Grade组成，其中(Sno, Cno)为主码。

```
CREATE TABLE SC(
        Sno CHAR(5) ,
        Cno CHAR(3) ,
        Grade   int,
        Primary key (Sno, Cno) );
```

二、删除基本表

DROP TABLE <表名>;

基本表删除

数据、表上的索引、表上的视图都删除

删除基本表时，系统会从数据字典中删去有关该基本表及其索引的描述

例　　题

[例5]　删除Student表

```
DROP TABLE  Student ;
```

三、修改基本表

ALTER TABLE <表名>

[ADD <新列名><数据类型>[完整性约束]]

[DROP <完整性约束名>]

[AITER COLUMN <列名><数据类型>];

<表名>：要修改的基本表

ADD子句：增加新列和新的完整性约束条件

DROP子句：删除指定的完整性约束条件

例　　题

［例2］　向Student表增加“入学时间”列，其数据类型为日期型。

```
ALTER TABLE Student ADD Scome DATE;
```

- 不论基本表中原来是否已有数据，新增加的列一律为空值。

[例3]　将年龄的数据类型改为整数。

```
ALTER TABLE Student ALTER COLUMN Sage INT;
```

- 注：修改原有的列定义有可能会破坏已有数据

例　　题

例[4]　删除学生姓名必须取唯一值的约束。

```
ALTER TABLE Student DROP UNIQUE(Sname);
```

语句格式（续）

- 修改属性列列名　间接
- alter table 表名 add 修改后的属性列名 数据类型;
- update 表名 set 修改后的属性列名=原表需要修改的属性列名;
- alter table 表名 drop column原表需要修改的属性列名;
- 例:
- alter table student add name varchar(10) null
  update student set name=Sname
  alter table student drop column sname

- 删除属性列
- alter table 表名 drop column 属性列名
- 例：
- alter table SC drop column Grade ;

3.2.2 建立与删除索引

- 建立索引是加快查询速度的有效手段
- 建立索引
  - DBA或表的属主（即建立表的人）根据需要建立
  - 有些DBMS自动建立以下列上的索引
    - PRIMARY KEY
    - UNIQUE
- 维护索引
  - DBMS自动完成
- 使用索引
  - DBMS自动选择是否使用索引以及使用哪些索引

一、建立索引

- 语句格式

  CREATE [UNIQUE] [CLUSTER] INDEX <索引名> ON <表名>(<列名>[<次序>][,<列名>[<次序>]]…);

  - 用<表名>指定要建索引的基本表名字
  - 索引可以建立在该表的一列或多列上，各列名之间用逗号分隔
  - 用<次序>指定索引值的排列次序，升序：ASC，降序：DESC。缺省值：ASC
  - UNIQUE表明此索引的每一个索引值只对应唯一的数据记录
  - CLUSTER表示要建立的索引是聚簇索引

例　　题

[例6]　为学生-课程数据库中的Student，Course，SC三个表建立索引。其中Student表按学号升序建唯一索引，Course表按课程号升序建唯一索引，SC表按学号升序和课程号降序建唯一索引。

```
CREATE UNIQUE INDEX  Stusno ON Student(Sno);

CREATE UNIQUE INDEX Coucno ON Course(Cno);

CREATE UNIQUE INDEX SCno ON SC(Sno ASC, Cno DESC);
```

建立索引 （续）

- 唯一值索引
  - 对于已含重复值的属性列不能建UNIQUE索引
  - 对某个列建立UNIQUE索引后，插入新记录时DBMS会自动检查新记录在该列上是否取了重复值。这相当于增加了一个UNIQUE约束

建立索引 （续）

聚簇索引

- 建立聚簇索引后，基表中数据也需要按指定的聚簇属性值的升序或降序存放。也即聚簇索引的索引项顺序与表中记录的物理顺序一致

例：

```
CREATE CLUSTER INDEX Stusname ON Student(Sname);
```

在Student表的Sname（姓名）列上建立一个聚簇索引，而且Student表中的记录将按照Sname值的升序存放

建立索引 （续）

- 在一个基本表上最多只能建立一个聚簇索引
- 聚簇索引的用途：对于某些类型的查询，可以提高查询效率
- 聚簇索引的适用范围
  - 很少对基表进行增删操作
  - 很少对其中的变长列进行修改操作

二、删除索引

DROP INDEX <索引名>;

- 删除索引时，系统会从数据字典中删去有关该索引的描述。

[例7] 删除Student表的Stusname索引。

```
DROP INDEX Stusname;
```

3.3 查　　询

3.3.1 概述

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> ] …
[ WHERE <条件表达式> ]
[ GROUP BY <列名1> [ HAVING <条件表达式> ] ]
[ ORDER BY <列名2> [ ASC|DESC ] ];
```

语句格式

- 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集函数。
- HAVING短语：筛选出只有满足指定条件的组
- ORDER BY子句：对查询结果表按指定列值的升序或降序排序

3.3 查　　询

3.3.1 概述

3.3.2 单表查询

3.3.3 连接查询

3.3.4 嵌套查询

3.3.5 集合查询

3.3.6 小结

3.3.2 单表查询

查询仅涉及一个表，是一种最简单的查询操作

一、选择表中的若干列

二、选择表中的若干元组

三、对查询结果排序

四、使用集函数

五、对查询结果分组

查询指定列

[例1] 查询全体学生的学号与姓名。

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

[例2] 查询全体学生的姓名、学号、所在系。

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

查询全部列

[例3] 查询全体学生的详细记录。

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

或

```
SELECT  *
FROM Student;
```

3. 查询经过计算的值

SELECT子句的<目标列表达式>为表达式

- 算术表达式
- 字符串常量
- 函数
- 列别名
- 等

3. 查询经过计算的值

[例4] 查全体学生的姓名及其出生年份。

```
SELECT Sname, 2019-Sage
FROM Student;
```

输出结果：

Sname	2019-Sage
李勇	1976
刘晨	1977
王名	1978
张立	1978

3. 查询经过计算的值

[例5] 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: ', 2019-Sage,
                    LOWER(Sdept)
FROM Student;
```

例　题（续）

输出结果：

Sname	'Year of Birth:'	2019-Sage	LOWER(Sdept)
李勇	Year of Birth:	1976	cs
刘晨	Year of Birth:	1977	is
王名	Year of Birth:	1978	ma
张立	Year of Birth:	1977	is

[例5.1] 使用列别名改变查询结果的列标题

```
SELECT Sname NAME, 'Year of Birth:'  BIRTH,
            2019-Sage BIRTHDAY, LOWER(Sdept)  DEPARTMENT
FROM Student;
```

输出结果：

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	1976	cs
刘晨	Year of Birth:	1977	is
王名	Year of Birth:	1978	ma
张立	Year of Birth:	1977	is

二、选择表中的若干元组

- 消除取值重复的行

- 查询满足条件的元组

1. 消除取值重复的行

- 在SELECT子句中使用DISTINCT短语

假设SC表中有下列数据

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	80

ALL 与 DISTINCT

[例6] 查询选修了课程的学生学号。

(1) SELECT Sno
FROM SC;

或(默认 ALL)

SELECT ALL Sno
FROM SC;

结果：

Sno
95001
95001
95001
95002
95002

例　题（续）

(2)　SELECT DISTINCT Sno

FROM SC;

结果：

Sno
95001
95002

例　题（续）

- 注意 DISTINCT短语的作用范围是所有目标列

  例：查询选修课程的各种成绩

  错误的写法

```
SELECT DISTINCT Cno, DISTINCT Grade
FROM SC;
```

  正确的写法

```
SELECT DISTINCT Cno, Grade
FROM SC;
```

2. 查询满足条件的元组

WHERE子句常用的查询条件

表 3.3　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	AND，OR

(1) 比较大小

在WHERE子句的<比较条件>中使用比较运算符

- =, >, <, >=, <=, != 或 <>, !>, !<,
- 逻辑运算符NOT + 比较运算符

[例8] 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;
```

或

```
SELECT Sname, Sage
FROM   Student
WHERE NOT Sage >= 20;
```

(2) 确定范围

- 使用谓词　BETWEEN ... AND ...

  NOT BETWEEN ... AND …

[例10]　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　题（续）

[例11]　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

使用谓词　　IN <值表>,　NOT IN <值表>

<值表>：用逗号分隔的一组取值

[例12]查询信息系（IS）、数学系（MA）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', 'MA', 'CS' );
```

(3) 确定集合

[例13]查询既不是信息系、数学系，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', 'MA', 'CS' );
```

(4) 字符串匹配

- [NOT] LIKE ‘<匹配串>’ [ESCAPE ‘ <换码字符>’]

  <匹配串>：指定匹配模板

  匹配模板：固定字符串或含通配符的字符串

  当匹配模板为固定字符串时，

  可以用 = 运算符取代 LIKE 谓词

  用 != 或 < >运算符取代 NOT LIKE 谓词

通配符

- %（百分号） 代表任意长度（长度可以为0）的字符串
  - 例：a%b表示以a开头，以b结尾的任意长度的字符串。如acb，addgb，ab 等都满足该匹配串
- _（下横线） 代表任意单个字符
  - 例：a_b表示以a开头，以b结尾的长度为3的任意字符串。如acb，afb等都满足该匹配串

ESCAPE 短语：

- 当用户要查询的字符串本身就含有 % 或 _ 时，要使用ESCAPE '<换码字符>' 短语对通配符进行转义。

例　　题

1）匹配模板为固定字符串

[例14]　查询学号为95001的学生的详细情况。

```
SELECT *
FROM  Student
WHERE  Sno LIKE '95001';
```

等价于：

```
SELECT  *
 FROM  Student
 WHERE Sno = '95001';
```

例　题（续）

2）匹配模板为含通配符的字符串

[例15]　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%';
```

例　题（续）

匹配模板为含通配符的字符串（续）

[例16]　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_ _';
```

例　题（续）

匹配模板为含通配符的字符串（续）

[例17] 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　题（续）

匹配模板为含通配符的字符串（续）

[例18]　查询所有不姓刘的学生姓名。

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

例　题（续）

3）使用换码字符将通配符转义为普通字符

[例19]　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　题（续）

使用换码字符将通配符转义为普通字符(续)

[例20]　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 ' \ ';
```

(5) 涉及空值的查询

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

[例21] 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　题（续）

[例22]　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

用逻辑运算符AND和 OR来连接多个查询条件

- 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 ...

例　题

[例23]　查询计算机系年龄在20岁以下的学生姓名。

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

改　写　[例12]

[例12]　查询信息系（IS）、数学系（MA）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', 'MA', 'CS' );
```

可改写为：

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

改　写　[例10]

[例10]　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;
```

可改写为：

```
SELECT Sname, Sdept, Sage
FROM   Student
WHERE  Sage>=20 AND Sage<=23;
```

三、对查询结果排序

使用ORDER BY子句

- 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

当排序列含空值时

- 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

对查询结果排序（续）

[例24]　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;
```

查询结果

Sno	Grade
95010	
95024	
95007	92
95003	82
95010	82
95009	75
95014	61
95002	55

对查询结果排序（续）

[例25]　查询全体学生情况，查询结果按所在系的系号升序排列，同一系中的学生按年龄降序排列。

```
SELECT  *
FROM  Student
ORDER BY Sdept, Sage DESC;
```

四、使用集函数

5类主要集函数

- 计数

  COUNT（[DISTINCT|ALL] *）

  COUNT（[DISTINCT|ALL] <列名>）

- 计算总和

  SUM（[DISTINCT|ALL] <列名>）

- 计算平均值

  AVG（[DISTINCT|ALL] <列名>）

使用集函数（续）

求最大值

MAX（[DISTINCT|ALL] <列名>）

求最小值

MIN（[DISTINCT|ALL] <列名>）

- DISTINCT短语：在计算时要取消指定列中的重复值
- ALL短语：不取消重复值
- ALL为缺省值

使用集函数 （续）

[例26] 查询学生总人数。

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

[例27] 查询选修了课程的学生人数。

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

注：用DISTINCT以避免重复计算学生人数

使用集函数 （续）

[例28] 计算1号课程的学生平均成绩。

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

[例29] 查询选修1号课程的学生最高分数。

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

五、对查询结果分组

使用GROUP BY子句分组

细化集函数的作用对象

- 未对查询结果分组，集函数将作用于整个查询结果
- 对查询结果分组后，集函数将分别作用于每个组

对查询结果分组 （续）

- GROUP BY子句的作用对象是查询的中间结果表
- 分组方法：按指定的一列或多列值分组，值相等的为一组
- 使用GROUP BY子句后，SELECT子句的列名列表中只能出现分组属性和集函数

使用GROUP BY子句分组

[例30] 求各个课程号及相应的选课人数。

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

结果

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

使用HAVING短语筛选最终输出结果

- 只有满足HAVING短语指定条件的组才输出
- 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。

使用HAVING短语筛选最终输出结果

［例31］ 查询选修了3门以上课程的学生学号。

```
SELECT Sno
FROM   SC
GROUP BY Sno
HAVING   COUNT(*) >3;
```

例　　题

[例32]　查询有3门以上课程是90分以上的学生的学号及（90分以上的）课程数

```
SELECT   Sno,   COUNT(*)
 FROM    SC
 WHERE Grade>=90
 GROUP BY Sno
 HAVING COUNT(*)>=3;
```