

- ❖ 准备上课了，请投票签到
- ❖ 现在教师端的麦克风是静音状态，如果发现有声
音传出，请检查自己的麦克风；如果发现其他同
学的麦克风未关，请在群里相互提醒。
- ❖ 请做好上课准备了~

数据库系统概论

An Introduction to Database System

第三章 关系数据库标准语言SQL

SQL概述

❖ SQL (Structured Query Language)

结构化查询语言，是关系数据库的标准语言

SQL历史

- ❖ 1970年Codd提出了关系模型之后，由于关系代数太抽象了，难以被普通用户接受，于是1973年IBM开展了System R的研制工作
- ❖ System R 以关系模型为基础，但是摒弃了数学语言，以自然语言为方向，结果诞生了结构化英语查询语言（Structured English Query Language, SEQUEL），负责人为Don Chamberlin博士。
- ❖ 后来更名为SQL，发音不变。
- ❖ System R获得1988年度ACM “软件系统奖”

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

目前,没有一个数据库管理系统能够支持**SQL**标准的所有概念和特性

SQL的特点

❖ SQL示例

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

读取SC表中元组，从中选择选修了3号课程的选课元组，然后输出这些元组的学号Sno和成绩Grade属性列的值，按照 成绩Grade降序排列输出

SQL的特点

❖ 综合统一

- 集数据定义语言（**DDL**），数据操纵语言（**DML**），数据控制语言（**DCL**）功能于一体。
- 可以独立完成数据库生命周期中的全部活动：
 - 定义和修改、删除关系模式，定义和删除视图，插入数据，建立数据库；
 - 对数据库中的数据进行查询和更新；
 - 数据库重构和维护
 - 数据库安全性、完整性控制，以及事务控制
 - 嵌入式**SQL**和动态**SQL**定义

SQL的特点

❖ 高度非过程化

- 非关系数据模型的数据操纵语言“**面向过程**”，必须指定存取路径。
- **SQL**只要提出“做什么”，无须了解存取路径。
- 存取路径的选择以及**SQL**的操作过程由系统自动完成。

SQL的特点

❖ 面向集合的操作方式

- 非关系数据模型采用面向记录的操作方式，操作对象是一条记录
- **SQL**采用集合操作方式
- 操作对象、查找结果可以是元组的集合
- 一次插入、删除、更新操作的对象可以是元组的集合

SQL的特点

❖ 以同一种语法结构提供多种使用方式

■ SQL是独立的语言

能够独立地用于联机交互的使用方式

■ SQL又是嵌入式语言

SQL能够嵌入到高级语言（例如**C**，**C++**，**Java**）程序中，供程序员设计程序时使用

SQL的特点

❖ 语言简洁，易学易用

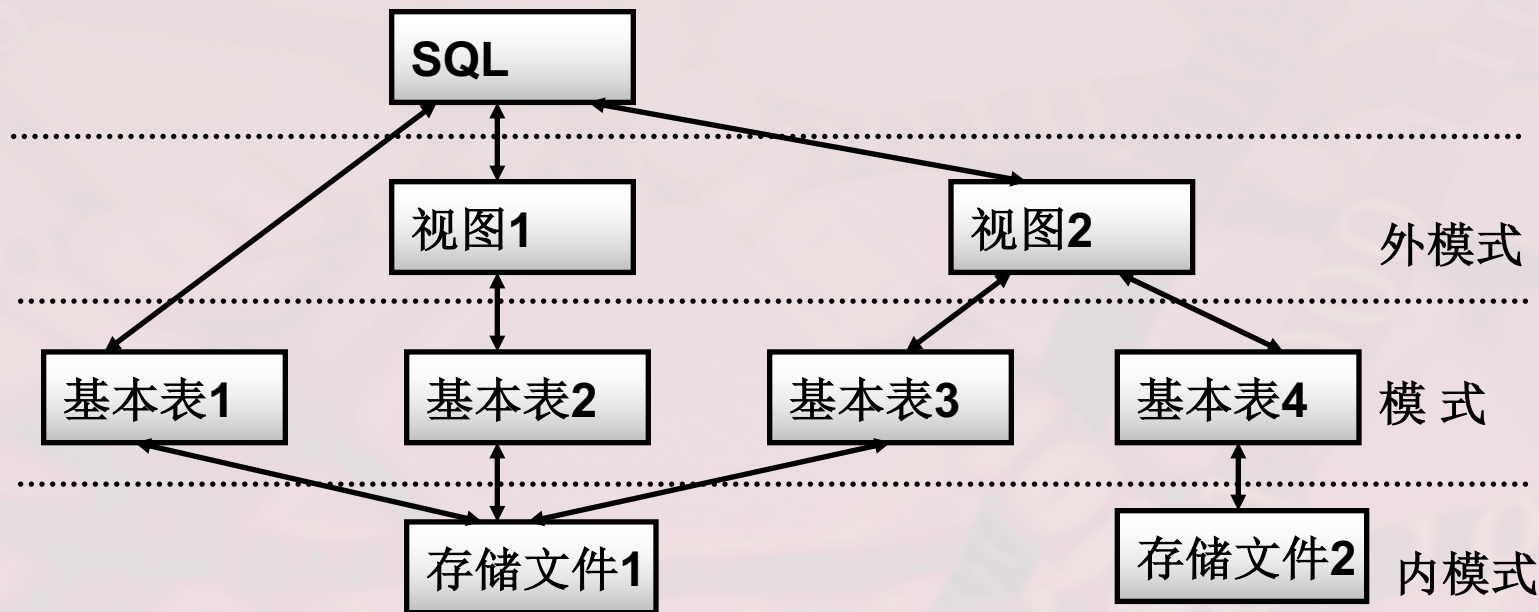
■ SQL功能极强，完成核心功能只用了**9**个动词。

□ 3.2 SQL □ □ □

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的基本概念（续）

SQL支持关系数据库三级模式结构



SQL的基本概念（续）

❖ 基本表

- 本身独立存在的表
- **SQL**中一个关系就对应一个基本表
- 一个（或多个）基本表对应一个存储文件
- 一个表可以带若干索引

SQL的基本概念（续）

❖ 存储文件

- 逻辑结构组成了关系数据库的内模式
- 物理结构对用户是隐蔽的

SQL的基本概念（续）

❖ 视图

- 从一个或几个基本表导出的表
- 数据库中只存放视图的定义而不存放视图对应的数据
- 视图是一个虚表
- 用户可以在视图上再定义视图

学生-课程 数据库

❖ 示例数据库

❖ 学生-课程数据库S-T :

学生表: **Student(Sno,Sname,Ssex,Sage,Sdept)**

课程表: **Course(Cno,Cname,Cpno,Ccredit)**

学生选课表: **SC(Sno,Cno,Grade)**

Student表

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

Course表

课程号 Cno	课程名 Cname	先行课 Cpno	学分 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	PASCAL语言	6	4

SC表

学号 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

数据定义

数据定义

❖ **SQL的数据定义功能:** 定义各种数据库的“对象”

- 模式定义
- 表定义
- 视图定义
- 索引定义

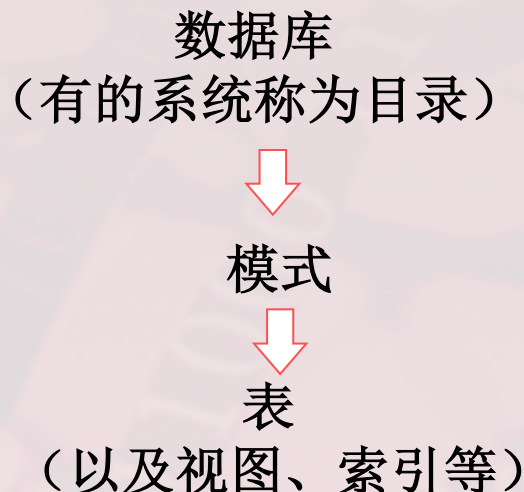
□ 3.3 SQL □ □ □ □ □ □ □

操 作 对 象	操 作 方 式		
	创 建	删 除	修 改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	ALTER INDEX

各种数据库“对象”

❖ 现代关系数据库管理系统提供了一个层次化的数据库对象命名机制

- 一个数据库中 can 建立多个模式
- 一个模式下通常包括多个表、视图和索引等数据库对象



数据定义

- 1 **SCHEMA**定义（为了和前面的模式概念区别）
- 2 表定义—重点
- 3 索引定义

1.SCHEMA定义

❖ SQL模式的创建和撤销

■ 注

- 模式（**Schema**）一词，来自于“**ISO SQL标准**”协议
- **Schema**在**SQL**数据库中相当于一个容器
 - 数据库所有的对象如表、视图、索引、用户、存储过程、触发器等都位于容器内
 - 创建**SQL**模式，就是定义一个存储空间
- 在商业**DBMS**中，大多都使用**Database**代替**Schema**

1.SCHEMA定义（续）

❖ SQL模式的创建和撤销

■ 创建

- **CREATE SCHEMA <模式名> AUTHORIZATION <用户名>**

■ 撤销

- **DROP SCHEMA <模式名> [CASCADE | RESTRICT]**
- 方式
 - **CASCADE**（级联式）
 - **RESTRICT**（约束式）

❖ 商业DBMS中

■ **CREATE Database**

■ **DROP Database**

1.SCHEMA定义（续）

❖ **DROP SCHEMA <模式名> <CASCADE|RESTRICT>**

■ **CASCADE**（级联）

- 删除模式的同时把该模式中所有的数据库对象全部删除

■ **RESTRICT**（限制）

- 如果该模式中定义了下属的数据库对象（如表、视图等），则拒绝该删除语句的执行。
- 仅当该模式中没有任何下属的对象时才能执行。

2 基本表的定义

❖ 定义基本表（关系模式）

```
CREATE TABLE <表名>                                /* 基本表的名称 */  
(<列名> <数据类型>[ <列级完整性约束条件> ] /*组成该表的列*/  
[,<列名> <数据类型>[ <列级完整性约束条件>]]  
...  
[,<表级完整性约束条件> ] );
```

- <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
- 如果完整性约束条件涉及到该表的多个属性列，则必须定义在表级上。

2 基本表的定义(续)

- 列级完整性约束条件
 - **NOT NULL / [NULL]** : 列值是否可以为空
 - **UNIQUE** : 列值唯一, 不得重复
 - **DEFAULT** : 列值空缺时, 由系统填写默认值
- 表级完整性约束条件
 - 主码 (**PRIMARY KEY**) 子句
 - » 格式: **PRIMARY KEY (<列名表>)**
 - » 作用: 提供实体完整性约束的说明
 - 说明
 - » 主码为单属性时, 可直接在属性后的列级完整性约束条件中, 使用**PRIMARY KEY**定义主码

2 基本表的定义(续)

- 表级完整性约束条件
 - 外码 (**FOREIGN KEY**) 子句
 - » 格式: **Foreign key** [外码名] (<列名表1>)
references <主表名> [(列名表2)]
 - 作用: 提供参照完整性约束的说明
 - 检查 (**CHECK**) 子句
 - » 格式: **CHECK** (约束表达式)
 - » 作用: 对某元组某属性取值的约束说明
- 说明
 - 表创建后是一个空表, 需要使用**DML** (**insert, update, delete**) 语句装入或维护数据行

学生表Student

[例3.5] 建立“学生”表Student。学号是主码，姓名取值唯一。

```
CREATE TABLE Student
```

```
(Sno CHAR(9) PRIMARY KEY,
```

主码

/* 列级完整性约束条件,Sno是主码*/

```
Sname CHAR(20) UNIQUE,
```

/* Sname取唯一值*/

```
Ssex CHAR(2),
```

```
Sage SMALLINT,
```

```
Sdept CHAR(20)
```

UNIQUE
约束

```
);
```

Student(Sno,Sname,Ssex,Sage,Sdept)

课程表Course

[例3.6] 建立一个“课程”表Course

CREATE TABLE Course

(Cno CHAR(4) PRIMARY KEY,

Cname CHAR(40),

先修课

Cpno CHAR(4),

Ccredit SMALLINT,

Cpno是外码
被参照表是Course
被参照列是Cno

FOREIGN KEY (Cpno) REFERENCES Course(Cno)
);

Course(Cno,Cname,Cpno,Ccredit)

学生选课表SC

[例3.7] 建立一个学生选课表SC

CREATE TABLE SC

SC(Sno,Cno,Grade)

(Sno CHAR(9),

Cno CHAR(4),

Grade SMALLINT,

PRIMARY KEY (Sno,Cno),

/ 主码由两个属性构成，必须作为表级完整性进行定义*/*

FOREIGN KEY (Sno) REFERENCES Student(Sno),

/ 表级完整性约束条件，Sno是外码，被参照表是Student */*

FOREIGN KEY (Cno)REFERENCES Course(Cno),

/ 表级完整性约束条件，Cno是外码，被参照表是Course*/*

Check (Grade Between 0 And 100)) ;

数据类型

- ❖ 关系模型中“域”的概念用数据类型来实现
- ❖ 定义表的属性时需要指明其数据类型及长度
- ❖ 选用哪种数据类型
 - 取值范围
 - 要做哪些运算
 - 不同的关系数据库管理系统支持的数据类型不完全相同

数据类型（续）

数据类型	含义
CHAR(<i>n</i>) , CHARACTER(<i>n</i>)	长度为 <i>n</i> 的定长字符串
VARCHAR(<i>n</i>) , CHARACTERVARYING(<i>n</i>)	最大长度为 <i>n</i> 的变长字符串
CLOB	字符串大对象
BLOB	二进制大对象
INT , INTEGER	长整数（4字节）
SMALLINT	短整数（2字节）
BIGINT	大整数（8字节）
NUMERIC(<i>p</i>, <i>d</i>)	定点数，由 <i>p</i> 位数字（不包括符号、小数点）组成，小数后面有 <i>d</i> 位数字
DECIMAL(<i>p</i>, <i>d</i>) , DEC(<i>p</i>, <i>d</i>)	同NUMERIC
REAL	取决于机器精度的单精度浮点数
DOUBLE PRECISION	取决于机器精度的双精度浮点数
FLOAT(<i>n</i>)	可选精度的浮点数，精度至少为 <i>n</i> 位数字
BOOLEAN	逻辑布尔量
DATE	日期，包含年、月、日，格式为YYYY-MM-DD
TIME	时间，包含一日的时、分、秒，格式为HH:MM:SS
TIMESTAMP	时间戳类型
INTERVAL	时间间隔类型

修改基本表

ALTER TABLE <表名>

[ADD[COLUMN] <新列名> <数据类型> [完整性约束]]

[ADD <表级完整性约束>]

[DROP [COLUMN] <列名> [CASCADE| RESTRICT]]

[DROP CONSTRAINT<完整性约束名>[RESTRICT | CASCADE]]

[ALTER COLUMN <列名><数据类型>] ;

修改基本表（续）

- <表名>是要修改的基本表
- **ADD**子句用于增加新列、新的列级完整性约束条件和新的表级完整性约束条件
- **DROP COLUMN**子句用于删除表中的列
 - 如果指定了**CASCADE**短语，则自动删除引用了该列的其他对象
 - 如果指定了**RESTRICT**短语，则如果该列被其他对象引用，关系数据库管理系统将拒绝删除该列
- **DROP CONSTRAINT**子句用于删除指定的完整性约束条件
- **ALTER COLUMN**子句用于修改原有的列定义，包括修改列名和数据类型

修改基本表（续）

[例3.8] 向**Student**表增加“入学时间”列，其数据类型为日期型

```
ALTER TABLE Student ADD S_entrance DATE;
```

不管基本表中原来是否已有数据，新增加的列一律为空值

修改基本表（续）

[例3.9] 将年龄的数据类型由字符型（假设原来的数据类型是字符型）改为整数。

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

[例3.10] 增加课程名称必须取唯一值的约束条件。

```
ALTER TABLE Course ADD UNIQUE(Cname);
```

删除基本表

DROP TABLE <表名> [RESTRICT| CASCADE] ;

❖ **RESTRICT:** 删除表是有限制的。

- 欲删除的基本表不能被其他表的约束所引用
- 如果存在依赖该表的对象，则此表不能被删除

❖ **CASCADE:** 删除该表没有限制。

- 在删除基本表的同时，相关的依赖对象一起删除

删除基本表（续）

[例3.11] 删除Student表

DROP TABLE Student CASCADE;

- 基本表定义被删除，数据被删除
- 表上建立的索引、视图、触发器等一般也将被删除

删除基本表（续）

DROP TABLE时，SQL2011 与 3个RDBMS的处理策略比较

序号	标准及主流数据库 依赖基本表 的处理方式 的对象	SQL2011		Kingbase ES		Oracle 12c		MS SQL Server 2012
		R	C	R	C		C	
1	索引	无规定		√	√	√	√	√
2	视图	×	√	×	√	√ 保留	√ 保留	√ 保留
3	DEFAULT, PRIMARY KEY, CHECK (只含该表的列) NOT NULL 等约束	√	√	√	√	√	√	√
4	外码FOREIGN KEY	×	√	×	√	×	√	×
5	触发器TRIGGER	×	√	×	√	√	√	√
6	函数或存储过程	×	√	√ 保留	√ 保留	√ 保留	√ 保留	√ 保留

R表示RESTRICT，C表示CASCADE

'×'表示不能删除基本表，'√'表示能删除基本表，‘保留’表示删除基本表后，还保留依赖对象

3 索引的定义

- ❖ 建立索引的目的：加快查询速度
- ❖ 关系数据库管理系统中常见索引：
 - 顺序文件上的索引、**B+**树索引、散列（**hash**）索引、位图索引
- ❖ 概念理解：
 - 属于内模式范畴
 - **RDBMS**通常在主码上自动建立索引
 - 查询、更新时自动起作用（适当建立索引会提高查询速度）
 - 索引属性值和相应的元组指针
 - 索引越多越好吗？

索引

❖ 谁可以建立索引

- 数据库管理员 或 表的属主（即建立表的人）

❖ 谁维护索引

- 关系数据库管理系统自动完成

❖ 使用索引

- 关系数据库管理系统自动选择合适的索引作为存取路径，用户不必也不能显式地选择索引

(1) . 建立索引

❖ 语句格式

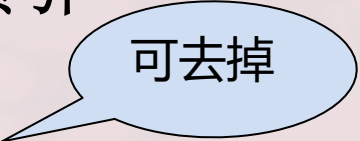
CREATE [**UNIQUE**] [**CLUSTER**] **INDEX** <索引名>

ON <表名>(<列名>[<次序>][,<列名>[<次序>]]...);

- **<表名>**: 要建索引的基本表的名字
- **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
- **<次序>**: 指定索引值的排列次序, 升序: **ASC**, 降序: **DESC**。缺省值: **ASC**
- **UNIQUE**: 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——会改变数据记录的物理顺序使之与索引项值的排列顺序相同, 一个表只能建立一个聚簇索引。

建立索引（续）

[例3.13] 为学生-课程数据库中的**Student**，**Course**，**SC**三个表建立索引。**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



可去掉

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

```
CREATE UNIQUE INDEX Coucno ON Course(Cno);
```

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

(2) 修改索引

❖ **ALTER INDEX** <旧索引名> **RENAME TO** <新索引名>

■ [例3.14] 将SC表的SCno索引名改为SCSno

ALTER INDEX SCno RENAME TO SCSno;

(3) 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3.15] 删除Student表的Stusname索引

```
DROP INDEX Stusname;
```

```
CREATE TABLE Course
(Cno    CHAR(4) PRIMARY KEY,
Cname  CHAR(40),
Cpno   CHAR(4),
Ccredit SMALLINT,
FOREIGN KEY (Cpno)
REFERENCES Course(Cno)
);
```

SQL语言
翻译与处理
程序

数据字典
(以表的方式存储)

数据字典

- ❖ 数据字典是关系数据库管理系统内部的一组系统表，它记录了数据库中所有定义信息：
 - 关系模式定义
 - 视图定义
 - 索引定义
 - 完整性约束定义
 - 各类用户对数据库的操作权限
 - 统计信息等
- ❖ 关系数据库管理系统在执行SQL的数据定义语句时，实际上就是在更新数据字典表中的相应信息。

数据查询

（单表查询）

数据查询

❖ 语句格式

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

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

[**AS**]<别名>

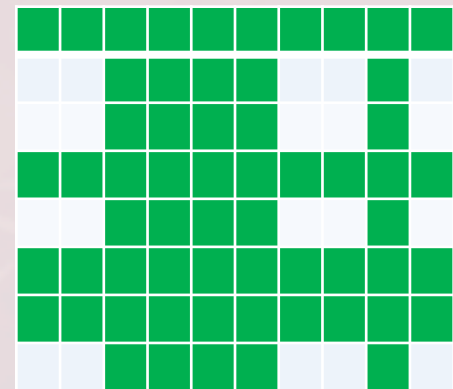
[**WHERE** <条件表达式>]

[**GROUP BY** <列名1> [**HAVING** <条件表达式>]]

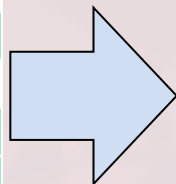
[**ORDER BY** <列名2> [**ASC|DESC**]];

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



SELECT-FROM-WHERE



201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

GROUP BY-HAVING
(常伴有聚集函数)



男	2
女	2



女	2
男	2



ORDER BY

1.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;
```

选择表中的若干列（续）

❖ 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;
```

查询经过计算的值（续）

❖ “虚”列

- **SELECT**子句的<目标列表达式>不仅可以为表中的属性列，也可以是表达式

[例3.19] 查全体学生的姓名及其出生年份。

```
SELECT Sname,2014-Sage  
FROM Student;
```

Sname	2014-Sage
李勇	1994
刘晨	1995
王敏	1996
张立	1995

查询经过计算的值（续）

[例3.20] 查询全体学生的姓名、出生年份和所在的院系，要求用小写字母表示系名。

```
SELECT Sname,'Year of Birth: ',2014-Sage,LOWER(Sdept)
FROM Student;
```

输出结果：

Sname	'Year of Birth: '	2014-Sage	LOWER(Sdept)
-------	-------------------	-----------	--------------

李勇	Year of Birth:	1994	cs
刘晨	Year of Birth:	1995	cs
王敏	Year of Birth:	1996	ma
张立	Year of Birth:	1995	is

查询经过计算的值（续）

❖ 使用别名改变查询结果的列标题:

```
SELECT Sname NAME,'Year of Birth:' BIRTH,  
       2014-Sage BIRTHDAY,LOWER(Sdept) DEPARTMENT  
FROM Student;
```

输出结果:

<u>NAME</u>	<u>BIRTH</u>	<u>BIRTHDAY</u>	<u>DEPARTMENT</u>
李勇	Year of Birth:	1994	cs
刘晨	Year of Birth:	1995	cs
王敏	Year of Birth:	1996	ma
张立	Year of Birth:	1995	is

2. 选择表中的若干元组

❖ 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;
```

结果为：

<u>Sno</u>
201215121
201215121
201215121
201215122
201215122

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：

<u>Sno</u>
201215121
201215122

(2) 查询满足条件的元组

表3.6 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;
```

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;
```


② 确定范围

❖ 谓词: **BETWEEN ... AND ...**
NOT BETWEEN ... AND ...

[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>

[例3.27]查询计算机科学系（**CS**）、数学系（**MA**）和信息系（**IS**）学生的姓名和性别。

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

③ 确定集合

❖ 谓词: **IN <值表>, NOT IN <值表>**

[例3.28]查询既不是计算机科学系、数学系，也不是信息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

字符匹配（续）

■ 匹配串为固定字符串

[例3.29] 查询学号为**201215121**的学生的详细情况。

```
SELECT *  
FROM Student  
WHERE Sno LIKE '201215121';
```

等价于：

```
SELECT *  
FROM Student  
WHERE Sno = '201215121';
```

字符匹配（续）

- 匹配串为含通配符的字符串

【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%';
```

字符匹配（续）

- 匹配串为含通配符的字符串

[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__';
```


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、学号和性别。

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

字符匹配（续）

- 使用换码字符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 '\' ;
```

字符匹配（续）

- 使用换码字符将通配符转义为普通字符

[例3.35] 查询以"DB_"开头，且倒数第3个字符为 i 的课程的具体情况。

```
SELECT *  
FROM Course  
WHERE Cname LIKE 'DB\__%i\__' ESCAPE '\';
```

ESCAPE '\ ' 表示 “ \ ” 为换码字符

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
```

⑤ 涉及空值的查询

❖ 谓词: **IS NULL** 或 **IS NOT NULL**

■ “IS” 不能用 “=” 代替

[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

[例3.38] 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;
```

多重条件查询（续）

❖ 改写[例3.27]

[例3.27] 查询计算机科学系（**CS**）、数学系（**MA**）和信息系（**IS**）学生的姓名和性别。

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

可改写为：

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


3.ORDER BY子句

❖ 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

❖ 对于空值, 排序时显示的次序由具体系统实现来决定

ORDER BY子句（续）

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

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

ORDER BY子句（续）

[例3.40]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;
```

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

4. 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
WHERE Cno='1';
```

[例3.45] 查询学生201215012选修课程的总学分数。

```
SELECT SUM(Ccredit)
FROM SC, Course
WHERE Sno='201215012' AND SC.Cno=Course.Cno;
```


5. GROUP BY子句

❖ GROUP BY子句分组:

细化聚集函数的作用对象

- 如果未对查询结果分组，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
- 按指定的一系列或多列值分组，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;
```

查询结果可能为：

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

GROUP BY子句（续）

[例3.47] 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;
```

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**子句中是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）

❖ **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。

应用示例

C (CNO, CNAME, Credit, CreditHours, CPNO, TNO)

S (SNO, SNAME, AGE, SEX, NativePlace)

T (TNO, TNAME, TITLE, SEX)

SC (SNO, CNO, Grade)

1.统计每门课程的学生选修人数，要求显示课程号、课程名和学生人数

– Select C.Cno, Cname, COUNT(Sno) as 学生人数

From C, SC

Where C.Cno = SC.Cno

Group By C.Cno, Cname ;

	Cno	Cname	学生人数
1	C1	Math	2
2	C2	English	3
3	C3	PM	2
4	C4	DB	4

CNO	CNAME	Credit	CreditHours	CPNO	TNO
C1	Math	3	48	NULL	T1
C2	English	4	64	NULL	T2
C3	PM	2	32	C2	T2
C4	DB	3.5	56	C1	T1

SNO	CNO	Grade
S1	C2	80
S1	C3	70
S1	C4	85
S2	C1	60
S2	C2	75
S2	C3	90
S2	C4	NULL
S3	C1	85
S3	C4	80
S4	C2	85
S4	C4	75

应用示例（续）

2.按教师号统计每位教师每门课程的学生选修人数，要求：

- 1) 仅显示选修人数在3人 (≥ 3) 以上的信息
- 2) 显示TNO、CNO和选修人数
- 3) 显示时，查询结果按选修人数降序排列，人数相同按TNO升序、CNO降序排列

应用示例（续）

C (CNO, CNAME, Credit, CreditHours, CPNO, TNO)

S (SNO, SNAME, AGE, SEX, NativePlace)

T (TNO, TNAME, TITLE, SEX)

SC (SNO, CNO, Grade)

- Select Tno, C.Cno, COUNT(Sno) as 选修人数
From C, SC Where C.Cno = SC.Cno
Group By Tno, C.Cno
Having COUNT(*)>=3
Order By 3 DESC, Tno, C.Cno DESC

SNO	CNO	Grade
S1	C2	80
S1	C3	70
S1	C4	85
S2	C1	60
S2	C2	75
S2	C3	90
S2	C4	NULL
S3	C1	85
S3	C4	80
S4	C2	85
S4	C4	75

	Tno	Cno	选修人数
1	T1	C4	4
2	T2	C2	3
3	T1	C1	2
4	T2	C3	2

	Tno	Cno	选修人数
1	T1	C4	4
2	T2	C2	3

CNO	CNAME	Credit	CreditHours	CPNO	TNO
C1	Math	3	48	NULL	T1
C2	English	4	64	NULL	T2
C3	PM	2	32	C2	T2
C4	DB	3.5	56	C1	T1

重点回顾

■ [GROUP BY <列名序列> [HAVING <组条件表达式>]]

• 作用

- 1) 数据按**GROUP BY**子句列名序列中的列值进行分组
- 2) 组内数据按**SELECT**子句中的聚集函数进行计算
- 3) 提取满足**HAVING**子句的条件表达式值的分组

• 注意

- **HAVING**子句支持聚集函数
- **WHERE**子句不支持聚集函数
- **SELECT**子句只能取聚集函数或**GROUP BY**子句指的列

{列名表 1} $\subset$ {列名表 2}

Select {列名表 1}, 聚集函数
... ..
Group By {列名表 2}
... .. ;

重点回顾（续）

❖ **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

重点回顾（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。

重点回顾（续）

■ **[ORDER BY <列名 | 列序号> [ASC | DESC]
[{, <列名 | 列序号> [ASC | DESC] }]**

- 对查询结果按子句中指定列的值排序，如果**ORDER BY**后有多多个列名
 - 先按第一列名值排序
 - 再对于第一列值相同的行，按第二列名值排序
 - 依次类推... ..
- 列序号是在**SELECT**子句中出现的序号（选的列是聚集函数或表达式时）
- **ASC**表示升序，**DESC**表示降序，缺省时表示升序