

# 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

---

# 第三章 关系数据库标准语言SQL

## 本章主要内容

- ➡ SQL概述 ←
- ➡ 学生-课程数据库
- ➡ 数据定义
- ➡ 数据查询
- ➡ 数据更新
- ➡ 空值的处理
- ➡ 视图
- ➡ 小结



3.1.1

*SQL* 的产生与发展

3.1.2

SQL的特点

3.1.3

SQL的基本概念

### 3.1.1 SQL 的产生与发展

#### ➔ SQL (Structured Query Language)

结构化查询语言，是关系数据库的标准语言

#### ➔ SQL是一个通用的、功能极强的关系数据库语言

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	1700页	1999年
■ SQL2003	标准越来越细致	2003年
■ SQL:2008		2008年
■ SQL:2011		2011年

3.1.1

SQL 的产生与发展

3.1.2

*SQL*的特点

3.1.3

SQL的基本概念

### ➡ 1.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：
  - 定义关系模式，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、完整性控制等
- ▣ 用户数据库投入运行后，可根据需要随时逐步修改模式，不影响数据的运行。
- ▣ 数据操作符统一

#### ➡ 2.高度非过程化

- ❏ 非关系数据模型的数据操纵语言“**面向过程**”，必须制定存取路径
- ❏ **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。

### ➡ 3.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



#### ➡ 4.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

### 3.1.2 SQL的特点

#### ➡ 5.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。

表 3.1 SQL 语言的动词

SQL 功 能	动 词
数 据 查 询	<b>SELECT</b>
数 据 定 义	<b>CREATE, DROP, ALTER</b>
数 据 操 纵	<b>INSERT, UPDATE DELETE</b>
数 据 控 制	<b>GRANT, REVOKE</b>

3.1.1

SQL 的产生与发展

3.1.2

SQL的特点

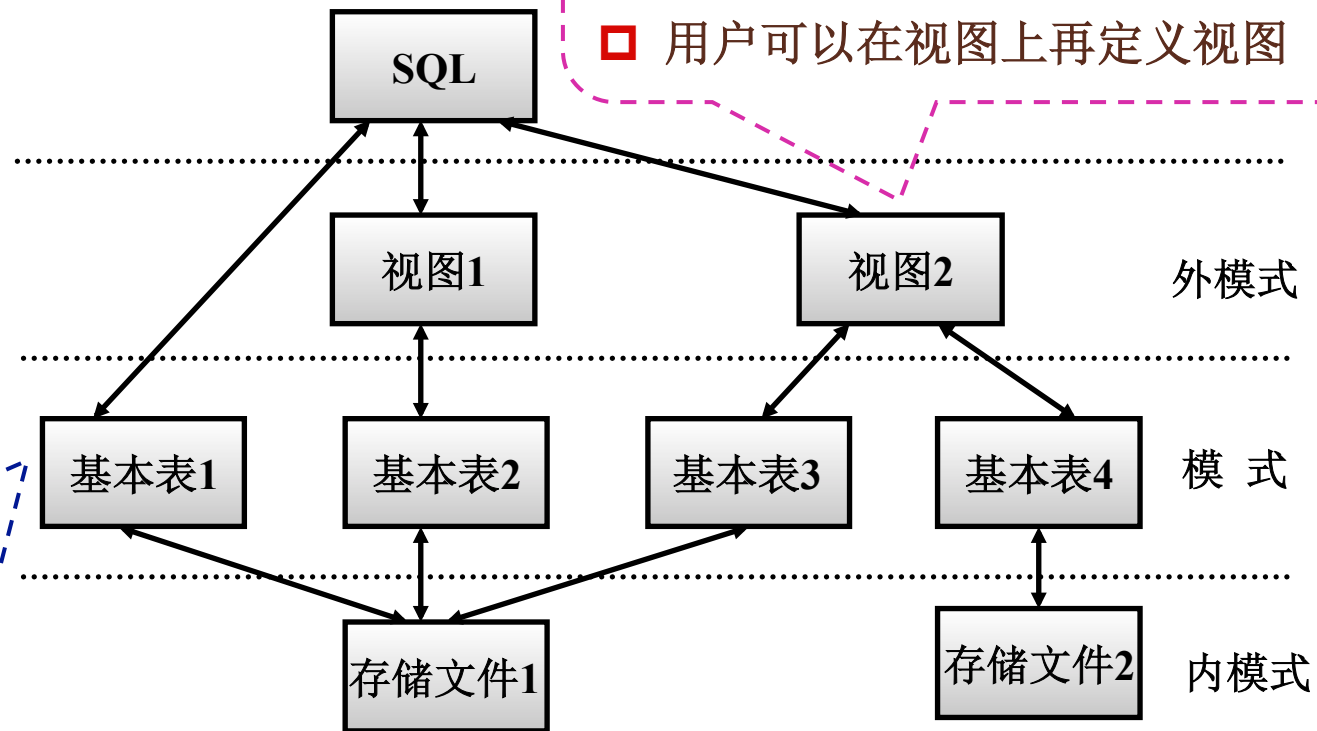
3.1.3

*SQL*的基本概念

### 3.1.3 SQL的基本概念

#### SQL支持关系数据库 三级模式结构

- 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
- 一个表可以有若干索引



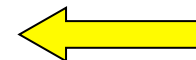
- 从一个或几个基本表导出的表
- 数据库中只存放视图的定义而不存放视图对应的数据视图是一个**虚表**
- 用户可以在视图上再定义视图

- 物理结构组成了关系数据库的内模式
- 物理结构是任意的，对用户透明 **由系统来完成，用户不用管**

# 第三章 关系数据库标准语言SQL

## 本章主要内容

- ➡ SQL概述
- ➡ 学生-课程数据库
- ➡ 数据定义
- ➡ 数据查询
- ➡ 数据更新
- ➡ 空值的处理
- ➡ 视图
- ➡ 小结



## 3.2 学生-课程 数据库

---

➡ 学生-课程模式 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)**

## 3.2 学生-课程 数据库

### ➔ Student表

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所 在 系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200515125	张立	男	19	IS

## 3.2 学生-课程 数据库

➔ **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





## 3.2 学生-课程 数据库

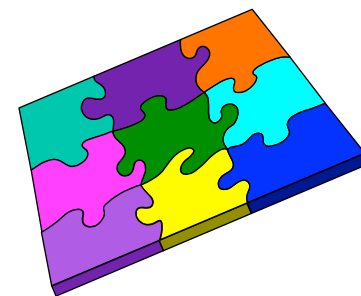
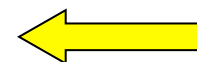
### ➡ SC表

学 号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

# 第三章 关系数据库标准语言SQL

## 本章主要内容

- ➡ SQL概述
- ➡ 学生-课程数据库
- ➡ 数据定义
- ➡ 数据查询
- ➡ 数据更新
- ➡ 空值的处理
- ➡ 视图
- ➡ 小结



### 3.3 数据定义

SQL的数据定义功能: 模式定义、表定义、视图和索引的定义

表 3.2 SQL 的数据定义语句 构成DDL

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	

索引可以视为内模式的一个定义



## 3.3 数据定义

---

### 3.3.1

模式的定义与删除

### 3.3.2

基本表的定义、删除与修改

### 3.3.3

索引的建立与删除

### 3.3.1 模式的定义与删除

#### ➡ 定义模式

**[例1]** 定义一个学生-课程模式**S-T**

**CREATE SCHEMA “S-T” AUTHORIZATION WANG;**

为用户**WANG**定义了一个模式**S-T**

**[例2] CREATE SCHEMA AUTHORIZATION WANG;**

<模式名>隐含为用户名**WANG**

■ 如果没有指定<模式名>，那么<模式名>隐含为<用户名>

### 3.3.1 模式的定义与删除

#### ➡ 定义模式

- 定义模式实际上定义了一个命名空间
- 在这个空间中可以定义该模式包含的数据库对象，例如基本表、视图、索引等。
- 在CREATE SCHEMA中可以接受CREATE TABLE，CREATE VIEW和GRANT子句。

授权

**CREATE SCHEMA** <模式名> **AUTHORIZATION** <用户名> [<表定义子句> | <视图定义子句> | <授权定义子句>]

### 3.3.1 模式的定义与删除

#### ➡ 定义模式

##### [例3]

```
CREATE SCHEMA TEST AUTHORIZATION ZHANG 主语句
CREATE TABLE TAB1( COL1 SMALLINT,
                    COL2 INT,
                    COL3 CHAR(20),
                    COL4 NUMERIC(10, 3), 对应属性
                    COL5 DECIMAL(5, 2)
                );
```

为用户**ZHANG**创建了一个模式**TEST**，并在其中定义了一个表**TAB1**。



## -- Syntax for SQL Server and Azure SQL Database


create schema的语法，可以在帮助文档输入关键字打开

```
CREATE SCHEMA schema_name_clause [ <schema_element> [ ...n ] ]

<schema_name_clause> ::=
{
    schema_name
  | AUTHORIZATION owner_name
  | schema_name AUTHORIZATION owner_name
}

<schema_element> ::=
{
    table_definition | view_definition | grant_statement |
    revoke_statement | deny_statement
}
禁止
```





---

```
CREATE SCHEMA Sprockets AUTHORIZATION Annik
CREATE TABLE NineProngs (source int, cost int, partnumber int)
GRANT SELECT ON SCHEMA::Sprockets TO Mandar
DENY SELECT ON SCHEMA::Sprockets TO Prasanna;
GO
```

### 3.3.1 模式的定义与删除

#### ➡ 删除模式

**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。
- 当该模式中没有任何下属的对象时 才能执行。

### 3.3.1 模式的定义与删除

---

#### ➡ 删除模式

**[例4] DROP SCHEMA ZHANG CASCADE;**

删除模式**ZHANG**

同时该模式中定义的表**TAB1**也被删除



# 使用T-SQL语言创建数据库

Transact-SQL 语言的语法约定及用于说明	
约定	用于
大写	<b>Transact-SQL</b> 关键字。
斜体	用户提供的 <b>Transact-SQL</b> 语法的参数。
粗体	数据库名、表名、列名、索引名、存储过程、实用工具、数据类型名以及必须按所显示的原样键入的文本。
下划线	指示当语句中省略了包含带下划线的值的子句时应用的默认值。
（竖线）	分隔括号或大括号中的语法项。只能使用其中一项。
[ ]（方括号）	可选语法项。不要键入方括号。
{ }（大括号）	必选语法项。不要键入大括号。
[,... <i>n</i> ]	指示前面的项可以重复 <i>n</i> 次。各项之间以逗号分隔。
[... <i>n</i> ]	指示前面的项可以重复 <i>n</i> 次。每一项由空格分隔。
[:]	可选的 <b>Transact-SQL</b> 语句终止符。不要键入方括号。
<label> ::=	语法块的名称。此约定用于对可在语句中的多个位置使用的过长语法段或语法单元进行分组和标记。可使用的语法块的每个位置由括在尖括号内的标签指示：<label>。



# 使用T-SQL语言创建数据库

---

## ➡ 创建数据库的命令

```
CREATE DATABASE database_name  
[ ON  
    [ < filespec > [ ,...n ] ]  
    [ , < filegroup > [ ,...n ] ]  
]  
[ LOG ON { < filespec > [ ,...n ] } ]  
[ COLLATE collation_name ]  
[ FOR LOAD | FOR ATTACH ]
```



# 使用T-SQL语言创建数据库

---

## ➡ 说明

☞ **< filespec > ::=**

**[ PRIMARY ]**

**( [ NAME = logical\_file\_name , ]**

**FILENAME = 'os\_file\_name'**

**[ , SIZE = size ]**

**[ , MAXSIZE = { max\_size | UNLIMITED } ]**

**[ , FILEGROWTH = growth\_increment ] ) [ ,...n ]**



## 使用T-SQL语言创建数据库

【例】创建一个指定主数据文件和事务日志文件的简单数据库，数据库名称为**Exercise\_db2**。

**CREATE DATABASE**

**Exercise\_db2**

**ON**

**PRIMARY**

( NAME=**Exercise\_Data**,  
FILENAME= 'F  
:\mydb\Exercise.MDF',  
SIZE=1,  
MAXSIZE=Unlimited,  
FILEGROWTH=10% )

**LOG ON**

( NAME=**Exercise\_LOG**,  
FILENAME= 'F  
:\mydb\Exercise.LDF',  
SIZE=1,  
MAXSIZE=10,  
FILEGROWTH=2 )



# 使用T-SQL语言删除数据库

---

## ➡ 语句格式

❏ **DROP DATABASE** database\_name [ ,...n ]

❏ 例：删除Test\_db1

**DROP DATABASE Test\_db1**

❏ 例：删除Test\_db2和Test\_db3.

**DROP DATABASE Test\_db2,Test\_db3**





### 3.3.1

### 模式的定义与删除

### 3.3.2

### 基本表的定义、删除与修改

### 3.3.3

### 索引的建立与删除

### 3.3.2 基本表的定义、删除与修改

#### 一、定义基本表

#### ➡ 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> ] ) ;

★ 数据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，否则既可以定义在列级也可以定义在表级。

### 3.3.2 基本表的定义、删除与修改

#### 一、定义基本表

**[例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,** /\* 列级完整性约束条件\*/  
注意逗号

**Sname CHAR(20) UNIQUE,** /\* Sname取唯一值\*/  
约束，说明sname必须取值唯一；两个红字都是列级  
约束，只约束本列

**Ssex CHAR(2),**

**Sage SMALLINT,**

**Sdept CHAR(20)**

**);**

主码

### 3.3.2 基本表的定义、删除与修改

#### 一、定义基本表

[例6] 建立一个“课程”表Course

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

先修课

表级约束的定义

自参照

Cpno是外码  
被参照表是Course  
被参照列是Cno

### 3.3.2 基本表的定义、删除与修改

#### 一、定义基本表

[例7] 建立一个“学生选课”表SC

**CREATE TABLE SC**

**(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\*/**  
**);**

- ➡ SQL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

数据类型	含义
CHAR(n)	长度为n的定长字符串
VARCHAR(n)	最大长度为n的变长字符串
INT	长整数（也可以写作INTEGER）
SMALLINT	短整数
NUMERIC(p, d)	定点数，由p位数字（不包括符号、小数点）组成，小数后面有d位数字
REAL	取决于机器精度的浮点数
Double Precision	取决于机器精度的双精度浮点数
FLOAT(n)	浮点数，精度至少为n位数字
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





# *SQL Data Types*

# SQL Data Types

ANSI SQL	MS Access	SQL Server 2000	IBM DB2	MySQL	Oracle 10
Character	char	char	char	char	char
Character varying	varchar	varchar	varchar	varchar	varchar
National character	char	nchar	graphic	char	nchar
National character varying	varchar	nvarchar	vargraphic	varchar	nvarchar
Integer	number (long integer)	int	int	int	int
Smallint	number (integer)	smallint	smallint	smallint	smallint
Real	number (double)	real	real	real	real
Decimal	number (decimal)	decimal	decimal	decimal	decimal
Date	date	datetime	date	date	date
Time	time	datetime	time	time	date



## SQL Data Types (Descriptions)

Data Type	Description	Storage Used	Example
character	Stores text data. A character can be any letter, number, or punctuation. You must specify how many characters you want to store in advance. If you actually store fewer than you allow for, the RDBMS adds spaces to the end to pad it out.	One byte per character allocated.	<code>char (8)</code> allocates space for eight characters and takes up approximately 8 bytes of space.
character varying	Similar to character except the length of the text is variable. Only uses up memory for the actual number of characters stored.	One byte per character stored.	<code>nchar (8)</code> allocates space for up to eight characters. However, storing only one character consumes only 1 byte of memory, storing two characters consumes 2 bytes of memory, and so on; up to 8 bytes allocated.
national character	Similar to character, except it uses two bytes for each character stored. This allows for a wider range of characters and is especially useful for storing foreign characters.	Two bytes per character allocated.	<code>nchar (8)</code> allocates space for eight characters and consumes 16 bytes of memory regardless of the number of characters actually stored.
national character varying	Similar to character varying, except it uses 2 bytes to store each character, allowing for a wider range of characters. Especially useful for storing foreign characters.	Two bytes per character stored.	<code>nvarchar (8)</code> allocates space for eight characters. How much storage is used depends on how many characters are actually stored.



## SQL Data Types (Descriptions)

Data Type	Description	Storage Used	Example
integer	A whole number between -2,147,483,648 and 2,147,483,647.	Four bytes.	int consumes 4 bytes regardless of the number stored.
smallint	A whole number between -32,768 and 32,767.	Two bytes.	smallint consumes 2 bytes of memory regardless of the number stored.
real	A floating-point number; range is from -3.40E+38 through 3.40E+38. It has up to eight digits after its decimal point, for example, 87.12342136.	Four bytes.	real consumes 4 bytes of memory regardless of the number stored.
decimal	A floating-point number. Allows you to specify the maximum number and how many digits after the decimal place. Range is from $-10^{38} + 1$ through $10^{38} - 1$ .	5–17 bytes.	decimal(38,12) sets a number that is up to 38 digits long with 12 digits coming after the decimal point.
date	Stores the date.	Four bytes.	date, for example, 1 Dec 2006 or 12/01/2006. Be aware of differences in date formats. In the U.K., for example, "12/01/2006" is actually January 12, 2006, whereas in the U.S. it's December 1, 2006.
time	Stores the time.	Three bytes.	time, for example, 17:54:45.

### 3.3.2 基本表的定义、删除与修改

#### 三、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 <完整性约束名> ]**

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

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

**[ALTER COLUMN<列名><数据类型>];**

**[例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;**

■ 不论基本表中原来是否已有数据，新增加的列一律为空值。

**[例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。

- 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

#### [例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



[例12] 若表上建有视图，选择**RESTRICT**时表不能删除

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

```
DROP TABLE Student RESTRICT;
```

**--ERROR: cannot drop table Student because other  
objects depend on it**

**[例12]**如果选择**CASCADE**时可以删除表，视图也自动被删除

```
DROP TABLE Student CASCADE;
```

```
--NOTICE: drop cascades to view IS_Student
```

```
SELECT * FROM IS_Student;
```

```
--ERROR: relation " IS_Student " does not exist
```

## 3.3.2 基本表的定义、删除与修改

### 四、删除基本表

#### DROP TABLE时，SQL99 与 3个RDBMS的处理策略比较

序号	标准及主流 数据库 的处理方式 依赖基本表的对象	SQL99		Kingbase ES		ORACLE 9i		MS SQL SERVER 2000
		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.	Foreign Key	×	√	×	√	×	√	×
5.	TRIGGER 触发器, 进行完整性控制	×	√	×	√	√	√	√
6.	函数或存储过程	×	√	√ 保留	√ 保留	√ 保留	√ 保留	√ 保留

视图还在, 但是  
select 视图没有值  
返回了

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

### 3.3.1

### 模式的定义与删除

### 3.3.2

### 基本表的定义、删除与修改

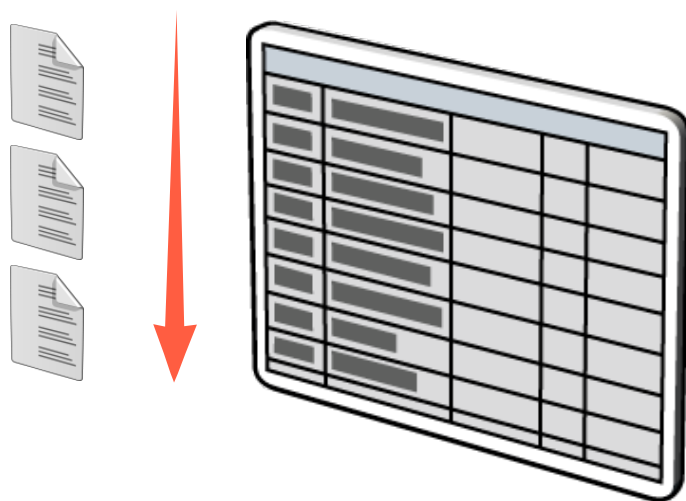
### 3.3.3

### 索引的建立与删除

### 3.3.3 索引的建立与删除

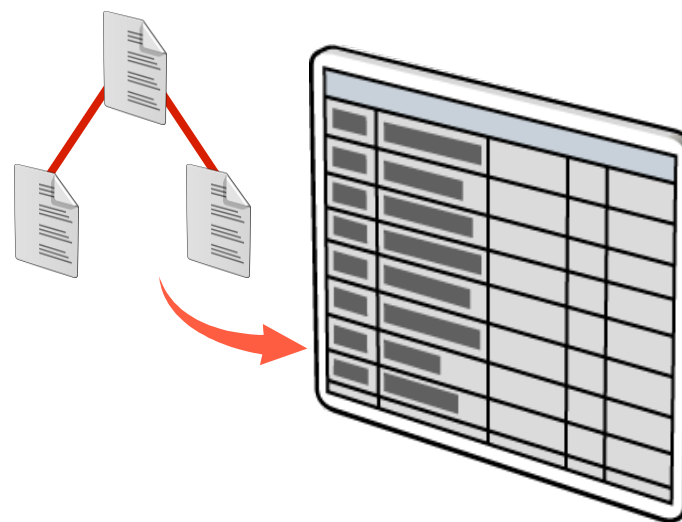
#### ➡ 表扫描

☞ **SQL Server** 扫描  
表的所有页



#### ➡ 索引

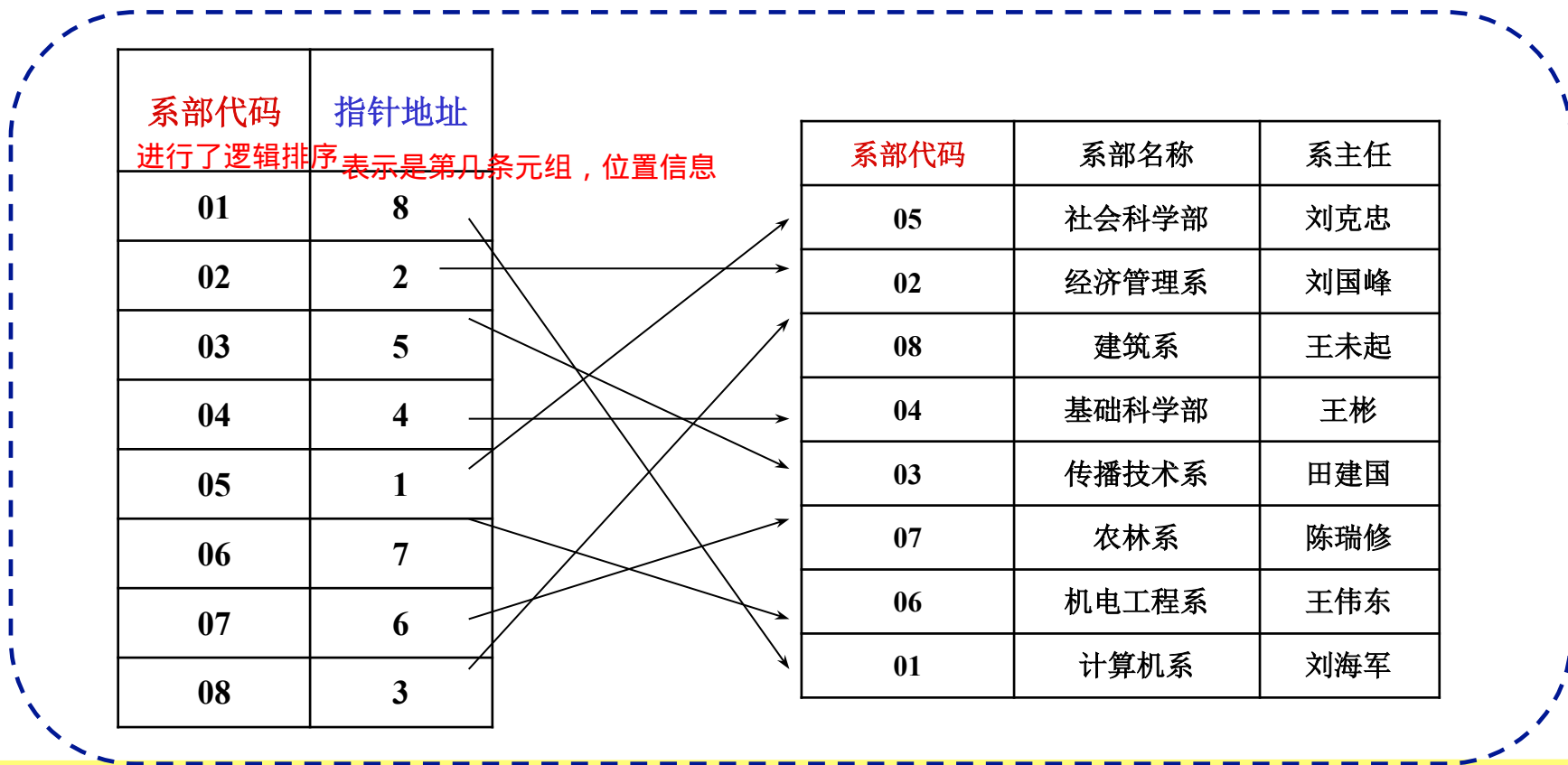
☞ **SQL Server** 使用  
索引页找到行 有一个专门的索引文件



### 3.3.3 索引的建立与删除

#### 索引的概念

索引是针对一个表，以表列为基础建立的数据库对象，它保存着表中排序的索引列，并且记录了索引列在数据表中的物理存储位置，实现了表中数据的逻辑排序。



### 3.3.3 索引的建立与删除

#### ➔ 定义索引

创建值唯一  
**CREATE** [**UNIQUE**] [**CLUSTER**] **INDEX** <索引名>  
**ON** <表名>(<列名>[<次序>][,<列名>[<次序>]]...);

##### ☰ 非聚簇索引

##### (Non-Clustered Index)

- 创建一个指定表的逻辑排序的对象。
- 对于非聚集索引，行的物理排序独立于索引排序。
- 非聚集索引的叶级包含索引行
- 每个索引行均包含非聚集键值和一个或多个行定位器（指向包含该值的行）

##### ☰ 聚簇索引

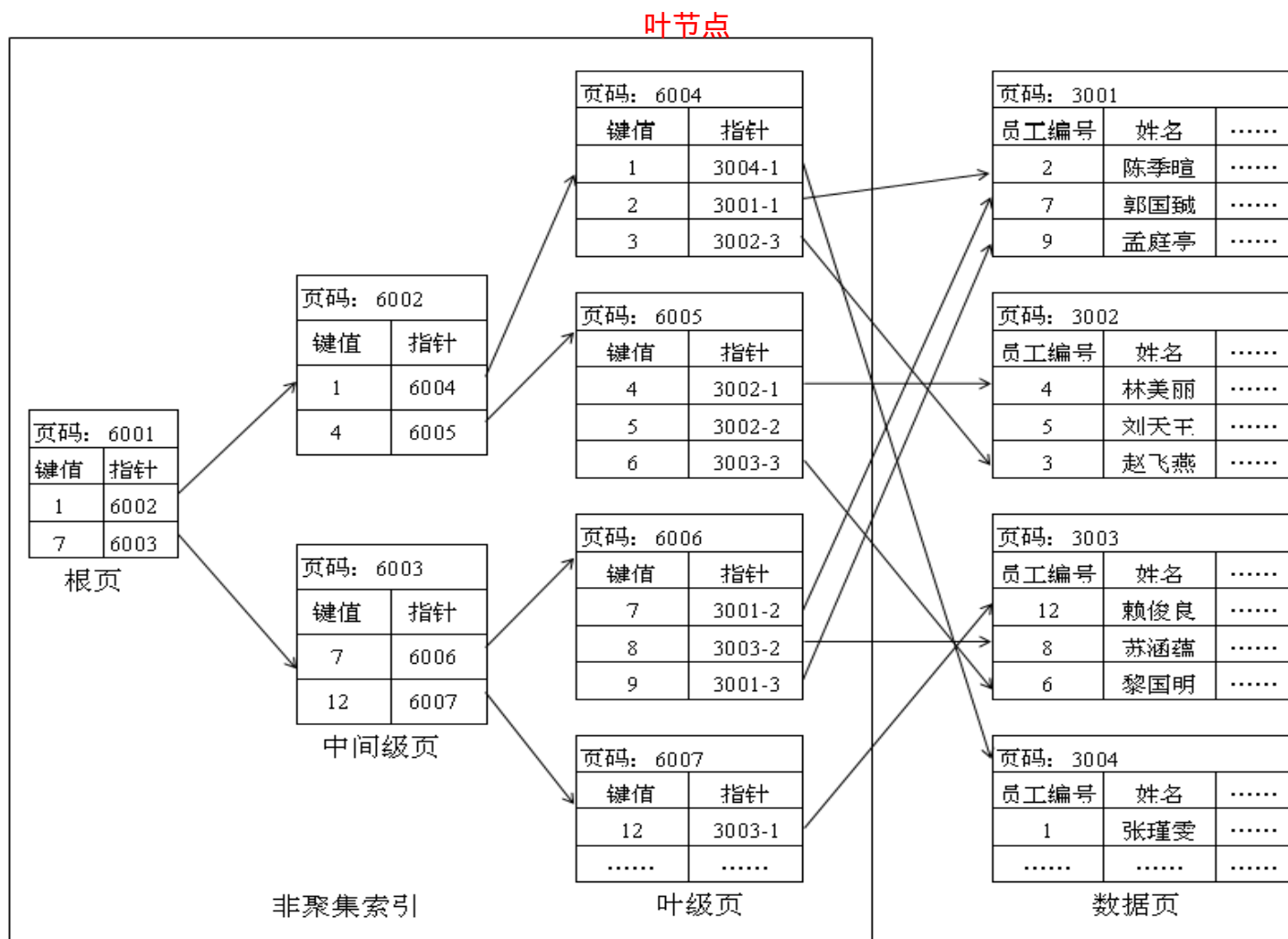
##### (Clustered Index)

注意是clustered不是cluster

- 表中的元组按聚簇索引的顺序物理地存放
- 根级页面---中间层页面---叶级页面（数据页面）
- 一个表中只能有一个聚簇索引
- 更新的复杂性，需要大量的临时空间

### 3.3.3 索引的建立与删除

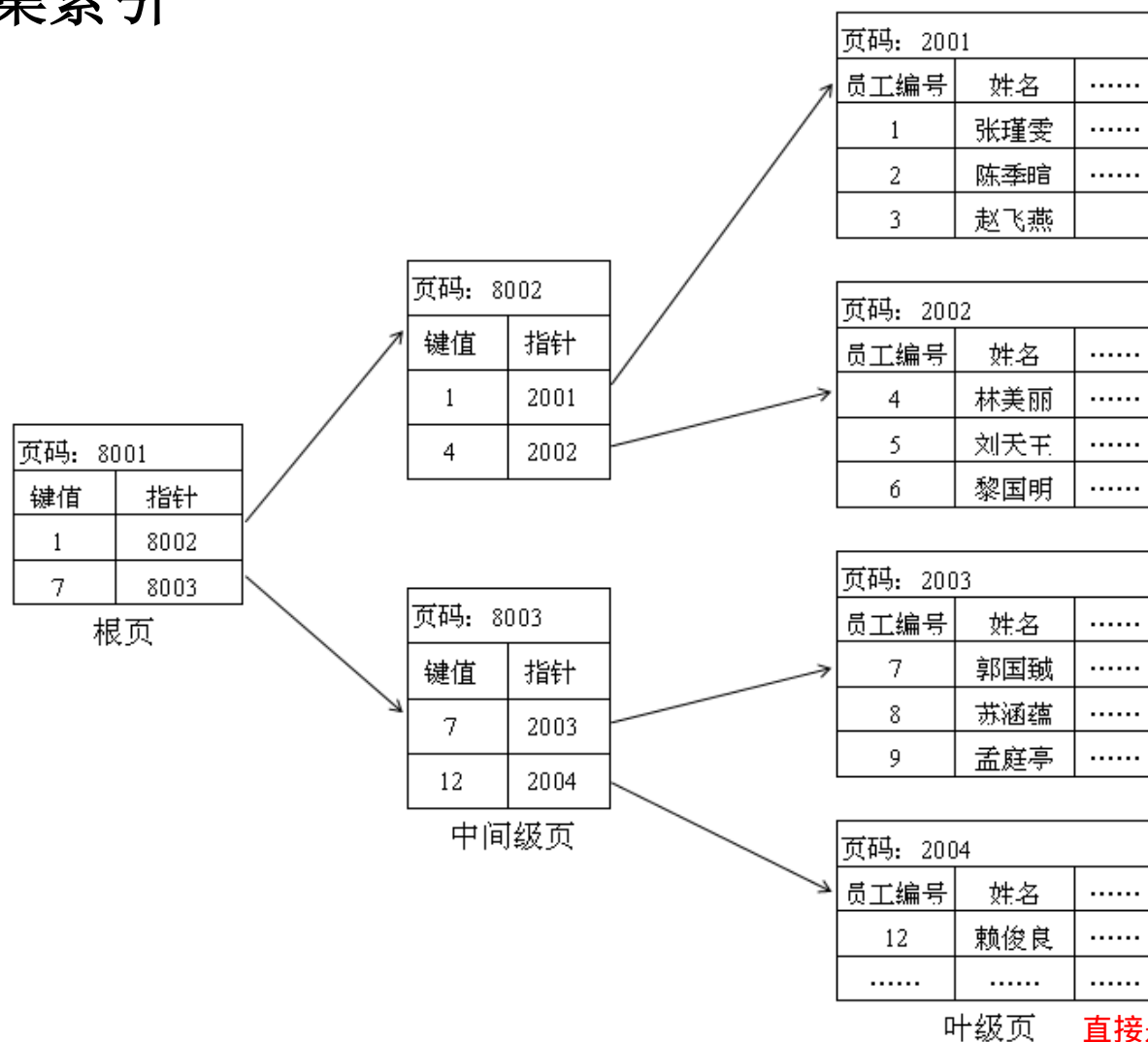
#### ➡ 非聚集索引





### 3.3.3 索引的建立与删除

#### ➡ 聚集索引



➡ 索引是关系数据库的内部实现技术，属于内模式的范畴

➡ 谁可以建立索引？

▣ **DBA** 或 表的属主（即建立表的人）

▣ **DBMS**一般会建立以下列上的索引

这是一些问题出现的原因

**PRIMARY KEY**

**UNIQUE**

➡ 谁维护索引？

**DBMS**自动完成

➡ 使用索引

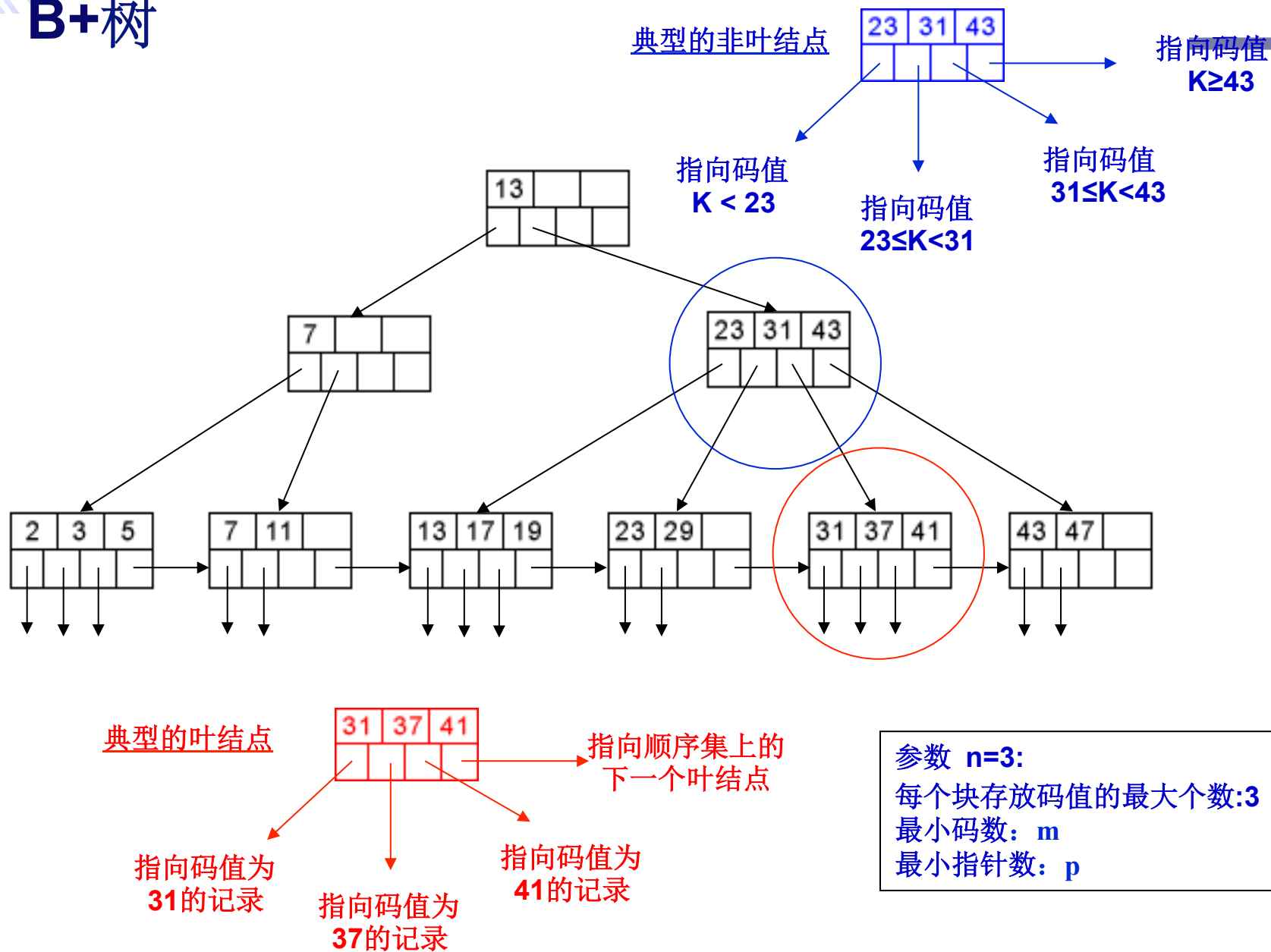
**DBMS**自动选择是否使用索引以及使用哪些索引

➡ **RDBMS**中索引一般采用**B+树**、**HASH**索引来实现

▣ **B+树**索引具有动态平衡的优点

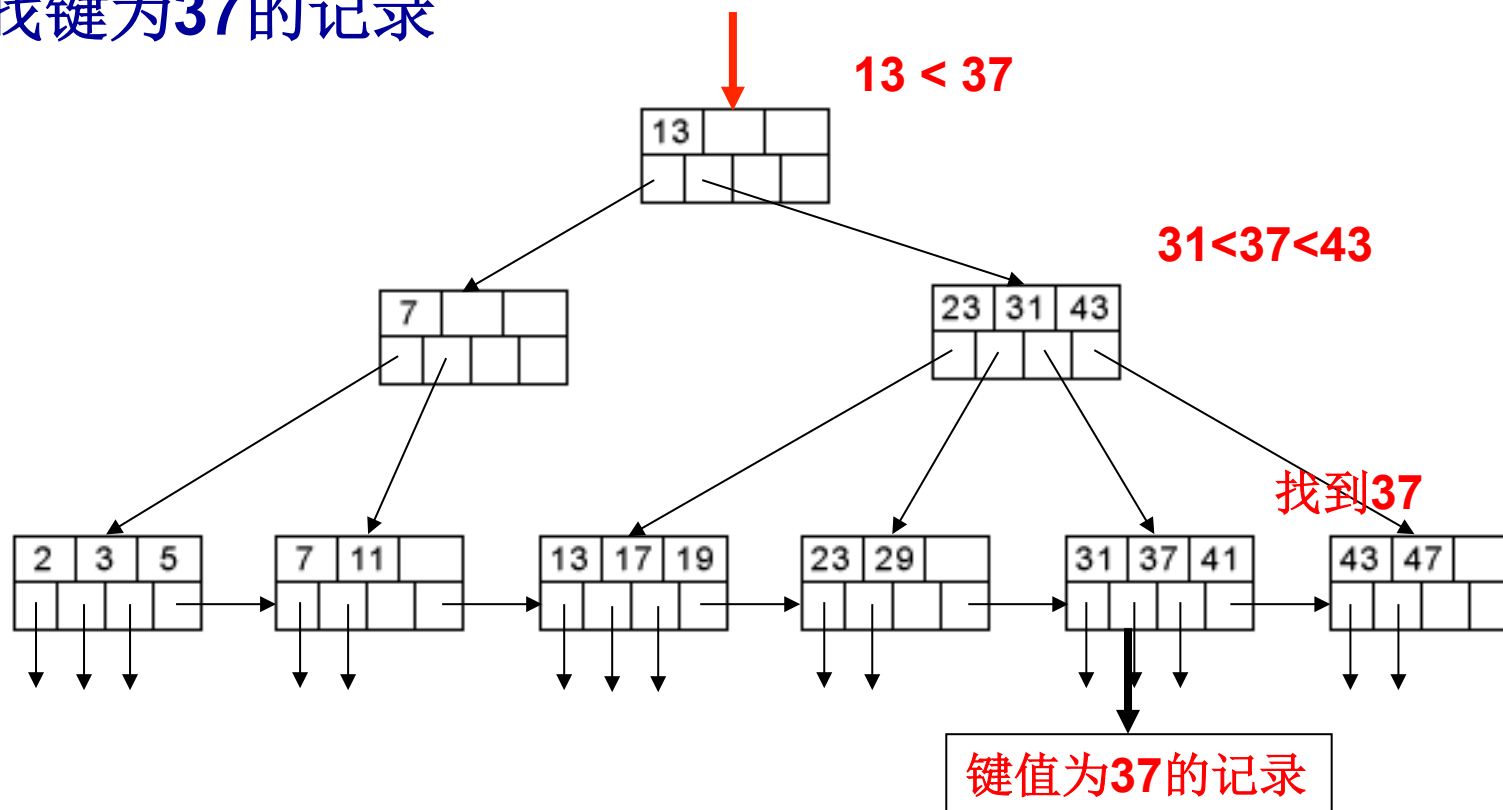
▣ **HASH**索引具有查找速度快的特点

# B+树



## B+树中的查找

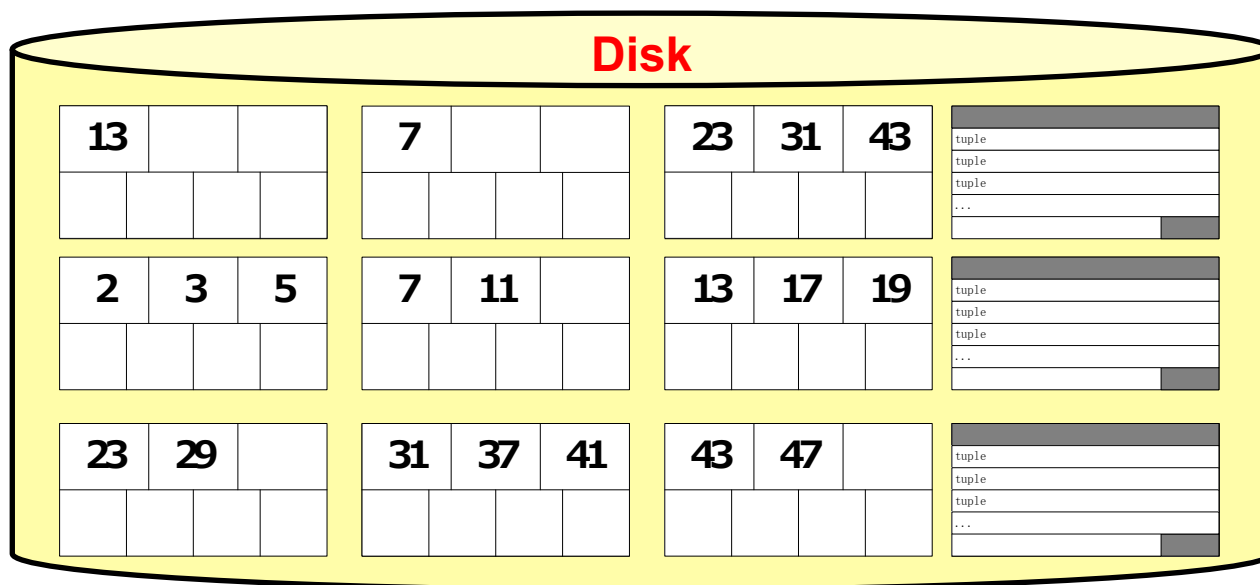
查找键为37的记录



IO数: 4 input或者output  
若把第一、第二层结点保存在缓冲区  
IO数: 2

## B+树中的查找(若索引块全在磁盘)

查找键为37的记录



读入root节点,IO=1  
根据条件 $13 < 37$ ,读取下一节点块

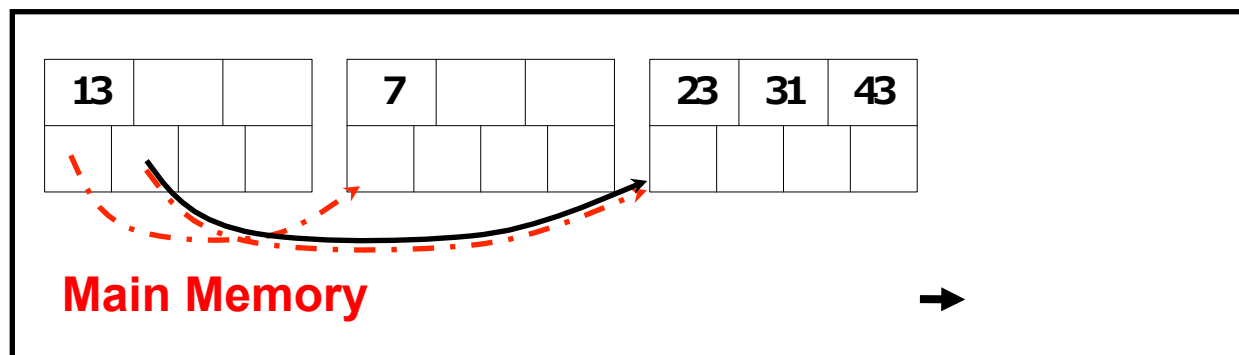
读入下一节点,IO=2  
根据条件 $31 < 37 < 43$ ,  
读取下一节点块

读入下一节点,IO=3,找到  
键值37,读取记录指针,  
读取记录所在的块

读入数据块,IO=4,根据记  
录指针找到键值为37的记录

## B+树中的查找(若第一、二层索引块在内存)

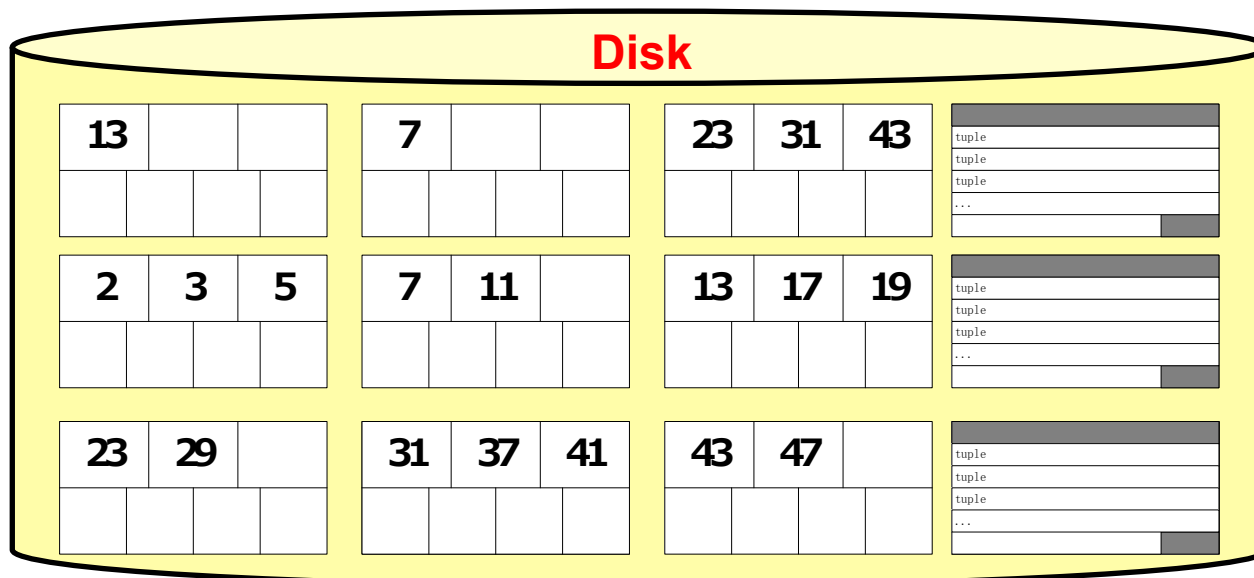
### 查找键为37的记录



读取root节点, 根据条件  $13 < 37$ , 读取下一节点块

读取下一节点, 根据条件  $31 < 37 < 43$ , 读取下一节点块

读入下一节点, **IO=1**, 找到键值37, 读取记录指针, 读取记录所在的块



读入数据块, **IO=2**, 根据记录指针找到键值为37的记录

**[例13] CREATE CLUSTER INDEX Stusname  
ON Student(Sname);**

在**Student**表的**Sname**（姓名）列上建立一个聚簇索引

- ➡ 业务规则、数据特征和数据的使用决定了创建索引的列
  - ☐ 在最经常查询的列上建立聚簇索引以提高查询效率
  - ☐ 一个基本表上最多只能建立一个聚簇索引
  - ☐ 经常更新的列不宜建立聚簇索引

### 3.3.3 索引的建立与删除

#### 一、建立索引

**[例14]**为学生-课程数据库中的**Student**，**Course**，**SC**三个表建立索引。

```
CREATE UNIQUE INDEX Stusno ON Student(Sno);  
CREATE UNIQUE INDEX Coucno ON Course(Cno);  
CREATE UNIQUE INDEX SCno ON SC(Sno ASC  
    , Cno DESC);
```

升序

降序

- 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



➡ 业务规则、数据特征和数据的使用决定了创建索引的列

➡ 建立索引的规则

☞ 主键

- 通常，检索、存取表是通过主键来进行的

☞ 连接中频繁使用的列

☞ 在某一范围内频繁搜索的列和按排序顺序频繁检索的列

#### ➡ 不考虑建立索引的列

- ☐ 很少或从来不在查询中引用的列，因为系统很少或从来 not 根据这个列的值去查找数据行。
- ☐ 只有两个或很少几个值的列（如性别，只有两个值“男”或“女”），以这样的列创建索引并不能得到建立索引的好处。  
这样可能没啥意义了
- ☐ 以 **bit**、**text**、**image** 数据类型定义的列。
- ☐ 数据行数很少的小表一般也没有必要创建索引。

➡ 修改索引  
语句格式

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

**[例14]** 将SC表的SCno索引名改为SCSno

**ALTER INDEX SCno RENAME TO SCSno**

➡ 删除索引  
语句格式

**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15]** 删除Student表的Stusname索引

**DROP INDEX Stusname;**

### 3.3.3 索引的建立与删除

#### ➡ 使用索引的意义

- ☐ 可以大大加快数据查询速度。
- ☐ 通过创建唯一索引，可以保证数据记录的唯一性。
- ☐ 在使用**ORDER BY**和**GROUP BY**子句进行检索数据时，可以显著减少查询中分组和排序的时间。
- ☐ 使用索引可以在检索数据的过程中使用优化隐藏器，提高系统性能。
- ☐ 可以加速表与表之间的连接，这一点在实现数据的参照完整性方面有特别的意义。

### 3.3.3 索引的建立与删除

#### ➡ 使用索引的代价

- ▮ 创建索引要花费时间和占用存储空间
- ▮ 建立索引加快了数据检索速度，却减慢了数据修改速度

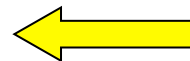
#### ➡ 在SQL Server 的数据库对象中，除了表之外，索引、视图、存储过程及触发器是数据库中最重要4种对象，能否恰当地使用这些对象，对于充分发挥SQL Server 的性能至关重要。

- ▮ 索引:提高查询速度
- ▮ 视图:定制数据
- ▮ 存储过程:定制功能
- ▮ 触发器:自动处理数据

# 第三章 关系数据库标准语言SQL

## 本章主要内容

- ➡ SQL概述
- ➡ 学生-课程数据库
- ➡ 数据定义
- ➡ 数据查询
- ➡ 数据更新
- ➡ 空值的处理
- ➡ 视图
- ➡ 小结



## 3.4 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> ] ...  
[ **WHERE** <条件表达式> ]  
[ **GROUP BY** <列名1> [ **HAVING** <条件表达式> ] ]  
[ **ORDER BY** <列名2> [ ASC|DESC ] ];



## 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

单表查询:

查询仅涉及一个表:

- 选择表中的若干列
- 选择表中的若干元组
- **ORDER BY**子句
- 聚集函数
- **GROUP BY**子句

### 3.4.1 单表查询

#### 一、 选择表中的若干列

#### ➡ 1.查询指定列 考试的时候会有select语句与关系代数之间的转换的题

**[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;
```

### 3.4.1 单表查询

#### 一、 选择表中的若干列

#### ➡ 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]** 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>可以为:

- 算术表达式
- 字符串常量
- 函数
- 列别名

## ➡ 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, 2004-Sage /*假定当年的年份为2004年*/  
FROM Student;
```

输出结果:

Sname	2004-Sage
李勇	1984
刘晨	1985
王敏	1986
张立	1985

### 3.4.1 单表查询

#### 一、选择表中的若干列

#### ➔ 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: ', 2004-Sage,  
        ISLOWER(Sdept)  
FROM Student;
```

大小写转换

输出结果:

Sname	'Year of Birth:'	2004-Sage	ISLOWER(Sdept)
-------	------------------	-----------	----------------

李勇	Year of Birth:	1984	cs
刘晨	Year of Birth:	1985	is
王敏	Year of Birth:	1986	ma
张立	Year of Birth:	1985	is

大写全部转化为了小写

### 3.4.1 单表查询

#### 一、 选择表中的若干列

#### ➡ 3. 查询经过计算的值

☞ 使用列**别名**改变查询结果的列标题:

```
SELECT Sname NAME, 'Year of Birth: ' BIRTH,  
        2000-Sage BIRTHDAY, LOWER(Sdept) DEPARTMENT  
FROM Student;
```

输出结果:

<b>NAME</b>	<b>BIRTH</b>	<b>BIRTHDAY</b>	<b>DEPARTMENT</b>
-----	-----	-----	-----
李勇	Year of Birth:	1984	cs
刘晨	Year of Birth:	1985	is
王敏	Year of Birth:	1986	ma
张立	Year of Birth:	1985	is

### 3.4.1 单表查询

## 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6]** 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**

---

200215121  
200215121  
200215121  
200215122  
200215122

📖 指定**DISTINCT**关键词，  
去掉表中重复的行

**SELECT DISTINCT Sno  
FROM SC;** 不加distinct会直接输出

执行结果：

---

200215121  
200215122



### ➡ 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

### 3.4.1 单表查询

#### 二、选择表中的若干元组

#### ➡ 2. 查询满足条件的元组

##### ☞ (1) 比较大小

[例7] 查询计算机科学系全体学生的名单。

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

[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;
```

[例9] 查询考试成绩有不及格的学生的学号。

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

### 3.4.1 单表查询

#### 二、选择表中的若干元组

#### ➔ 2. 查询满足条件的元组

##### ☞ (2) 确定范围

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;
```

### ➡ 2. 查询满足条件的元组

#### 📖 (3) 确定集合

谓词: **IN <值表>**, **NOT IN <值表>**

[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' );
```

[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姓名和性别。

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

## ➡ 2. 查询满足条件的元组

### 📖 (4) 字符匹配

谓词: **[NOT] LIKE ‘<匹配串>’ [ESCAPE ‘<换码字符>’]**

- ① 匹配串为固定字符串

**[例14]** 查询学号为**200215121**的学生的详细情况。

```
SELECT *  
FROM Student  
WHERE Sno LIKE ‘200215121’;
```

等价于:

```
SELECT *  
FROM Student  
WHERE Sno = ‘200215121’;
```

### 3.4.1 单表查询

#### 二、选择表中的若干元组

#### ➔ 2. 查询满足条件的元组

##### 📖 (4) 字符匹配

- ② 匹配串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%';
```

% (百分号) 代表  
任意长度 (长度可  
以为0) 的字符串

**[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__';
```

\_ (下划线) 代表  
任意单个字符

#### ➡ 2. 查询满足条件的元组

##### ☞ (4) 字符匹配

- ② 匹配串为含通配符的字符串

**[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18]** 查询所有不姓刘的学生姓名。

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

### 3.4.1 单表查询

#### 二、选择表中的若干元组

#### ➔ 2. 查询满足条件的元组

##### ☐ (4) 字符匹配

- 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 '\';
```

ESCAPE '\ ' 表示 “ \ ” 为换码字符

[例20] 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



#### ➡ 2. 查询满足条件的元组

##### ☞ (5) 涉及空值的查询

谓词: **IS NULL** 或 **IS NOT NULL**

[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;
```

[注]: “IS” 不能用 “=” 代替

#### ➡ 2. 查询满足条件的元组

##### 📖 (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 ...**

**[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;
```

#### ➡ 2. 查询满足条件的元组

##### 📋 (6) 多重条件查询

※※ 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 ';
```

#### ➡ 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

**[例24]** 查询选修了3号课程的学生学号及其成绩，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;
```

➡ 聚集函数:

☞ 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>)**

☞ 最大最小值

**MAX ([DISTINCT|ALL] <列名>)**

**MIN ([DISTINCT|ALL] <列名>)**

**[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;
```

**[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
WHERE Cno= '1' ;
```

**[例30]** 查询学生200215012选修课程的总学分数。

```
SELECT SUM(Ccredit)
FROM SC, Course
WHERE Sno='200215012' AND
SC.Cno=Course.Cno;
```

from不是单表查询，而是=有两个



**[例31]** 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

**[例32]** 查询选修了3门以上课程的学生学号。

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

➡ **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。

