知识点44： 数据查询语句

数据查询

- 数据查询是数据库的核心操作，SQL使用SELECT语句进行数据库的查询，其一般格式为：

  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]];

说明

- **[ ]：表示[ ]中的内容是可选的，比如[WHERE ＜条件表达式>]，表示可以使用WHERE也可以不使用。**
- **<>：表示<>中的内容必须出现。比如< 表名或视图名 >，这个部分表示从哪个地方获取数据，是不可或缺的。**
- **｜：表示选择其一，例如 < 列名2> [ASC ｜DESC]]，表示列名2后只能用ASC或者DESC其中之一来进行结果的排序，前者代表升序，后者代表降序。**
- **[，…]：表示括号中的内容可以重复出现1至多次。**

说明（续）

- SELECT子句用于指定输出的目标列；
- FROM子句用于指定数据所依赖的表或者视图；
- WHERE子句用于指定数据的选择条件；
- GROUP BY子句用于对检索到的记录进行分组；
- HAVING子句用于指定组的选择条件；
- ORDER BY子句用于对查询的结果进行排序。
- 在这些句子中，SELECT子句和FROM子句是必须出现的，其它子句都是可选的。

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

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
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

知识点45：　选择表中的若干列

选择表中的若干列

- 查询指定列

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

[例1] 查询全体学生的学号与姓名。

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

Sno	Sname
200215121	李勇
200215122	刘晨
200215123	王敏
200515125	张立

[例2] 查询全体学生的姓名、学号、所在系。

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

Sname	Sno	Sdept
李勇	200215121	CS
刘晨	200215122	CS
王敏	200215123	MA
张立	200515125	IS

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;

Sno	Sname	Ssex	Sage	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

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4] 查全体学生的姓名及其出生年份。

```
SELECT Sname，2016-Sage
FROM Student;
```

输出结果：

Sname	2016-Sage
李勇	1996
刘晨	1997
王敏	1998
张立	1997

查询经过计算的值（续）

[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: '，2016-Sage，LOWER(Sdept)
FROM Student；
```

输出结果：

Sname	'Year of Birth:'	2004-Sage	ISLOWER(Sdept)
李勇	Year of Birth:	1996	cs
刘晨	Year of Birth:	1997	is
王敏	Year of Birth:	1998	ma
张立	Year of Birth:	1997	is

查询经过计算的值（续）

- 使用列别名改变查询结果的列标题
  - as+别名，as可省略

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

输出结果：

NAME	BIRTH	BIRTHDAY	DEPARTMENT
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

选择表中的若干元组

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

- 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

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
200215121
200215122

知识点45：　选择表中的若干元祖

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>]

查询满足条件的元组

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

比较大小

Student(Sno, Sname, Ssex, Sage, Sdept)
Course(Cno,Cname,Cpno,Ccredit)
SC(Sno,Cno,grade)

［例7］ 查询计算机科学系(CS)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;
```

确定范围

Student(Sno, Sname, Ssex, Sage, Sdept)
Course(Cno,Cname,Cpno,Ccredit)
SC(Sno,Cno,grade)

- 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;
```

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的姓名和性别。

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

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 ';
```

字符匹配（续）

- <匹配串>中可以使用通配符“%”（百分号）和“_”（下横线）。
  - %：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。字符串ab、axb、agxb都满足该匹配串。
  - _：代表任意单个字符。
    - 例如，a_b表示以a开头，以b结尾，中间夹一个字符的任意字符串。如axb、a!b等都满足要求。

字符匹配（续）

Student(Sno, Sname, Ssex, Sage, Sdept)
Course(Cno,Cname,Cpno,Ccredit)
SC(Sno,Cno,grade)

- 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%’；
```

[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__';
```

字符匹配（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18] 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）

- 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  ‘ \  ‘;
```

- **ESCAPE '\' 表示“ \ ” 为换码字符**

涉及空值的查询

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

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;
```

多重条件查询（续）

- 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 ';
```

知识点46： ORDER BY子句

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
- 当排序列含空值时
  - **DESC**：排序列为空值的元组最后显示
  - **ASC**：排序列为空值的元组最先显示

ORDER BY子句（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;

或：

SELECT Sno，Grade FROM SC WHERE Cno='3' ORDER BY 2 DESC;

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

SELECT * FROM Student ORDER BY Sdept，Sage DESC;

Sno	Sname	Ssex	Sage	Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200515125	张立	男	19	IS
200215123	王敏	女	18	MA

抽取来自顶部的记录

- 抽取来自顶部的记录
  - **TOP *n* [PERCENT]**
  - ***n*** **指定返回的行数。如果未指定 PERCENT，*n* 就是返回的行数。如果指定了 PERCENT，*n* 就是返回的结果集行的百分比**
  - **如果一个 SELECT 语句既包含 TOP 又包含 ORDER BY 子句，那么返回的行将会从排序后的结果集中选择。整个结果集按照指定的顺序建立并且返回排好序的结果集的前 *n* 行。**

练习

- 为了从没有e-mail id 的职工表中显示所有的职工材料，可用以下查询中哪一个？

A. **Select * from employee where email = NULL**

B. **Select * from employee where email is NULL**

C. **Select * from employee where email = 0**

D. **Select * from employee where email is not NULL**

练习

ABC公司举行一个聚会，庆祝它成立一周年。聚会由30岁以下的所有职工组织。从HR 部保存的EmployeePersonal表中检索30 以下的人员的列表。

为产生满足上述要求的列表，你应使用以下查询中哪一个？

1）SELECT * from EmployeePersonal where iAge like 30

2）SELECT * from EmployeePersonal where iAge=30

3）SELECT * from EmployePersonal where iAge IN (30)

4）SELECT * from EmployeePersonal where iAge <30

练习

Student_Marks 表包含学生花名册号、名字、及获得的总分。编写一个查询，它将显示前5名成绩的列表。

A. Select TOP 5 * from Student_Marks
B. Select TOP 5 * from Student_Marks order by marks asc
C. Select TOP 5 * from Student_Marks order by marks desc
D. Select TOP 5 PERCENT * from Student_Marks order by marks desc

练习

某个电视商人用TVType 表来跟踪他仓库中的TV。为显示3种最昂贵的TV的描述，你将使用以下查询中哪一个？

A. SELECT TOP 3 cDescription FROM TVType ORDER BY iPrice asc
B. SELECT TOP 3 cDescription from TVType ORDER BY iPrice desc
C. SELECT cDescription from TVType where max(iPrice) > 3
D. SELECT cDescription, max(iPrice) from TVType ORDER BY iPrice.

知识点47： 字符串函数

函数

- **Select function_name(parameters)**

字符串函数

- **ASCII**
  - 返回字符表达式最左端字符的 ASCII 代码值。
  - 语法:ASCII (*character_expression*)
  - 参数:*character_expression*是类型为 char 或 varchar的表达式
  - 返回类型:int
  - 例子:

    Select ASCII（’abcd’）　　--结果为字符a的ASCII码97。

ASCII码

- 为保证人类和设备，设备和计算机之间能进行正确的信息交换，人们编制的统一的信息交换代码，这就是ASCII码表，它的全称是“美国信息交换标准代码”。

值	符号	值	符号	值	符号
0	空字符	44	,	91	[
32	空格	45	-	92	\
33	!	46	.	93	]
34	"	47	/	94	^
35	#	48 ~ 57	0 ~ 9	95	-
36	$	58	:	96	`
37	%	59	;	97 ~ 122	a ~ z
38	&	60	<	123	{
39	'	61	=	124	|
40	(	62	>	125	}
41	)	63	?	126	~
42	*	64	@	127	DEL (Delete 键)
43	+	65 ─ 90	A ~ Z		

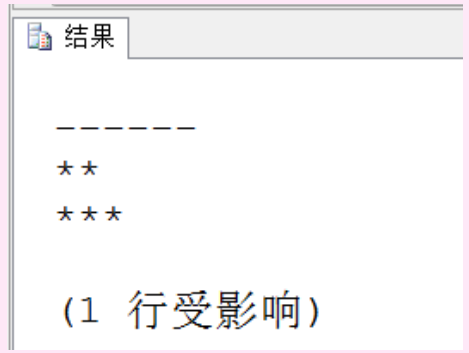
Char函数

- **Char**
  - 将 int ASCII 代码转换为字符的字符串函数。
  - 语法：**CHAR (*integer_expression*)**
  - 参数：***integer_expression***介于 **0** 和 **255** 之间的整数。如果整数表达式不在此范围内，将返回 **NULL** 值。
  - 返回类型：**char(1)**
  - 例：**Select CHAR（97）　　--结果为'a'**

CHAR 可用于将控制字符插入字符串中。

控制字符	值
制表符	CHAR(9)
换行符	CHAR(10)
回车	CHAR(13)

- 例：select '**'+char(13)+'***'



函数

- **NCHAR**
  - 根据 Unicode 标准所进行的定义，用给定整数代码返回 Unicode 字符。
  - 语法:NCHAR (*integer_expression*)
  - 参数:*integer_expression*介于 0 与 65535 之间的所有正整数。如果指定了超出此范围的值，将返回 NULL。
  - 返回类型:nchar(1)
- **UNICODE**
  - 按照 Unicode 标准的定义，返回输入表达式的第一个字符的整数值。
  - 语法:UNICODE ('*ncharacter_expression*')
  - 参数:'*ncharacter_expression*'是 nchar 或 nvarchar 表达式。
  - 返回类型:int
  - Select unicode(N’kerge’)

Charindex函数

- **Charindex**
  - 返回字符串中指定表达式的起始位置。
  - 语法：**CHARINDEX (*expression1* , *expression2* [, *start_location*])**
  - 参数：
    - ***expression1***一个表达式，其中包含要寻找的字符的次序。***expression1*** 是一个短字符数据类型分类的表达式。
    - ***expression2***一个表达式，通常是一个用于搜索指定序列的列。***expression2*** 属于字符串数据类型分类。
    - ***start_location***在 ***expression2*** 中搜索 ***expression1*** 时的起始字符位置。如果没有给定 ***start_location***，而是一个负数或零，则将从 ***expression2*** 的起始位置开始搜索。
  - 返回类型：**int**
  - 如果在 ***expression2*** 内没有找到 ***expression1***，则 **CHARINDEX** 返回 **0**。
  - 例子：**Select CHARINDEX('ab', '123abc123abc')　　--结果为4。**

LEFT函数

- **LEFT**
  - 返回从字符串左边开始指定个数的字符。
  - 语法：**LEFT (*character_expression* , *integer_expression*)**
  - 参数：
  - ***character_expression***字符或二进制数据表达式。***character_expression*** 可以是常量、变量或列。
  - ***integer_expression***是正整数。如果 ***integer_expression*** 为负，则返回空字符串。
  - 返回类型：**varchar**
  - 例：**LEFT(‘abcde’, 3)　　--结果为‘abc’。**

RIGHT函数

- **RIGHT**
  - 返回字符串中从右边开始指定个数的 ***integer_expression*** 字符。
  - 语法：**RIGHT (*character_expression* , *integer_expression*)**
  - 例：**Select RIGHT('abcde', 3)　　--结果为'cde'**

LEN函数

- LEN
  - 返回给定字符串表达式的字符（而不是字节）个数，其中不包含尾随空格。
  - 语法：LEN (*string_expression*)
  - 参数：*string_expression*要计算的字符串表达式。
  - 返回类型：int
  - 例：Select LEN('abcde　') --结果为5。

    select len('刘') --结果为1。

DATALENGTH函数

- **DATALENGTH (*expression*)**
  - 返回用于表示任何表达式的字节数。
  - 例：**Select DATALENGTH(‘abcde　’)　　--结果为8（含3个空格）**

    select DATALENGTH('刘')　　--结果为2。

- **LOWER**
  - 将大写字符数据转换为小写字符数据后返回字符表达式。
  - 语法：**LOWER (*character_expression*)**
  - 参数：***character_expression***是字符或二进制数据表达式。***character_expression*** 可以是常量、变量或列。
  - 返回类型：**varchar**
  - 例：**Select LOWER('12ABC45*%^def')　　--结果为'12abc45*%^def'。**

- **UPPER**
  - 返回将小写字符数据转换为大写的字符表达式。
  - 语法：**UPPER (*character_expression*)**
  - 例：**Select UPPER(‘12ABC45*%^def’)　--结果为'12ABC45 *%^DEF’**

- **LTRIM**
  - 删除起始空格后返回字符表达式。
  - 语法：**LTRIM (*character_expression*)**
  - 参数：***character_expression***是字符或二进制数据表达式。***character_expression*** 可以是常量、变量或列
  - 返回类型：**varchar**
  - 例：**Select LTRIM(' 12AB ') --结果为'12AB '**
- **RTRIM**
  - 截断所有尾随空格后返回一个字符串。
  - 语法：**RTRIM (*character_expression*)**
-

REPLACE

- **REPLACE**
  - 用第三个表达式替换第一个字符串表达式中出现的所有第二个给定字符串表达式。
  - 语法:REPLACE ('*string_expression1*' , '*string_expression2*' , '*string_expression3*')
  - 参数:
    - '*string_expression1*'待搜索的字符串表达式。*string_expression1* 可以是字符数据或二进制数据。
    - '*string_expression2*'待查找的字符串表达式。*string_expression2* 可以是字符数据或二进制数据。
    - '*string_expression3*'替换用的字符串表达式。*string_expression3* 可以是字符数据或二进制数据。
  - 返回类型:如果 *string_expression*（1、2 或 3）是支持的字符数据类型之一，则返回字符数据。
  - 例子:

```
Select REPLACE('abcde','de','12')     --结果为'abc12'
```

STUFF

- **STUFF**
  - 删除指定长度的字符并在指定的起始点插入另一组字符。
  - 语法:STUFF (*character_expression* , *start* , *length* , *character_expression*)
  - 参数
    - *character_expression*由字符数据组成的表达式。*character_expression* 可以是常量、变量，也可以是字符或二进制数据的列。
    - *start*是一个整形值，指定删除和插入的开始位置。如果 *start* 或 *length* 是负数，则返回空字符串。如果 *start* 比第一个 *character_expression* 长，则返回空字符串。
    - *length*是一个整数，指定要删除的字符数。如果 *length* 比第一个 *character_expression* 长，则最多删除到最后一个 *character_expression* 中的最后一个字符。
  - 返回类型:如果 *character_expression* 是一个支持的字符数据类型，则返回字符数据。
  - 例：Select STUFF('abcde',4,2,'12')　　--结果为'abc12'

REPLICATE

- **REPLICATE**
  - 以指定的次数重复字符表达式。
  - 语法:REPLICATE (*character_expression* , *integer_expression*)
  - 参数:
    - *character_expression*由字符数据组成的字母数字表达式。*character_expression* 可以是常量或变量，也可以是字符列或二进制数据列。
    - *integer_expression*是正整数。如果 *integer_expression* 为负，则返回空字符串。
  - 返回类型:varchar
  - 例子：replicate(' ',3)

SPACE

- **SPACE**
  - 返回由重复的空格组成的字符串。
  - 语法:**SPACE (*integer_expression*)**
  - 参数:***integer_expression***是表示空格个数的正整数。如果 ***integer_expression*** 为负，则返回空字符串。
  - 返回类型:**char**

STR

- **STR**
  - 由数字数据转换来的字符数据。
  - 语法:**STR (*float_expression* [, *length* [, *decimal*]])**
  - 参数:
    - ***float_expression*是带小数点的近似数字 (float) 数据类型的表达式。**
    - ***length*是总长度，包括小数点、符号、数字或空格。默认值为 10。**
    - ***decimal*是小数点右边的位数。**
  - 返回类型:**char**
  - 如果为 STR 提供 *length* 和 *decimal* 参数值，则这些值应该是正数。在默认情况下或者小数参数为 0 时，数字四舍五入为整数。指定长度应该大于或等于小数点前面的数字加上数字符号（若有）的长度。短的 *float_expression* 在指定长度内右对齐

STR（续）

- select str(123,6) --结果为‘ 123’
- select str(123.456,5) --结果为‘ 123’
- select str(123.456,5,2) --结果为‘123.5’
- select str(123.456,8,2) --结果为' 123.46

SUBSTRING

- **SUBSTRING**
  - 返回字符、binary、text 或 image 表达式的一部分。
  - 语法:SUBSTRING (*expression* , *start* , *length*)
  - 参数:
    - *expression*是字符串、二进制字符串、text、image、列或包含列的表达式。不要使用包含聚合函数的表达式。
    - *start*是一个整数，指定子串的开始位置。
    - *length*是一个整数，指定子串的长度（要返回的字符数或字节数）。
  - 例子：返回product表中name中第四个字符为c的产品id，name

Select id,name from product where substring(name,4,1)='c'

知识点48：　日期函数

日期函数

- **GETDATE**
  - 返回当前系统日期和时间。
  - 语法：**GETDATE ( )**
  - 返回类型：**datetime**
  - **Select getdate()　　--结果为2016-04-13 21:51:32.390**

YEAR函数

- **YEAR**
  - 返回表示指定日期中的年份的整数。
  - 语法：**YEAR (*date*)**
  - **Select year('2004-3-5')　　--结果为2004**
  - **select sname as学号, BirthDate as 出生日期, YEAR(getdate())-YEAR(birthdate) as 年龄 from dbo.student**

日期函数

- **MONTH**
  - 返回代表指定日期月份的整数。
  - 语法：**MONTH (*date*)**
- **DAY**
  - 返回代表指定日期的天的日期部分的整数。
  - 语法：**DAY (*date*)**

DATENAME

- **DATENAME**
  - 返回代表指定日期的指定日期部分的字符串。
  - 语法：**DATENAME (*datepart* , *date*)**
  - 参数：
    - ***datepart*是指定应返回的日期部分的参数**
  - **select datename(weekday,getdate())**
  - **select datename(yy, getdate())　　--返回当前年份**
  - **select datename (m, getdate())　　--返回当前月份**
  - **select datename (WEEKDAY, getdate()) --结果为当前日期是星期几**

日期部分	缩写
年份	**yy、yyyy**
季度	**qq、q**
月份	**mm、m**
每年的某一日	**dy、y**
日期	**dd、d**
星期	**wk、ww**
工作日	**dw**
小时	**hh**
分钟	**mi、n**
秒	**ss、s**
毫秒	**ms**

DATEPART

- **DATEPART**
  - 返回代表指定日期的指定日期部分的整数。
  - 语法：**DATEPART (*datepart* , *date*)**
  - **select datepart(weekday,getdate())**

DATEADD

- **DATEADD**
  - 在向指定日期加上一段时间的基础上，返回新的 datetime 值。
  - 语法：DATEADD (*datepart* , *number*, *date*)
  - 参数
    - *datepart*是规定应向日期的哪一部分返回新值的参数。
    - *number*是用来增加 *datepart* 的值。如果指定一个不是整数的值，则将废弃此值的小数部分。例如，如果为 *datepart* 指定 day，为 *number* 指定 1.75，则 *date* 将增加 1。
    - *date*是返回 datetime 或 smalldatetime 值或日期格式字符串的表达式
- 例子：
  - **Select dateadd(yy,2,'2012-3-4') --结果为’2014-03-04 00:00:00.000'**
  - **Select dateadd(m,2,'2012-3-4') --结果为'2012-05-04 00:00:00.000’**
  - **Select dateadd(d,2,'2012-3-4') --结果为'2012-03-06 00:00:00.000'**

DATEDIFF

- **DATEDIFF**
  - 返回跨两个指定日期的日期和时间边界数。
  - 语法：**DATEDIFF (*datepart* , *startdate* , *enddate*)**
  - 参数
    - ***datepart*是规定了应在日期的哪一部分计算差额的参数。**
    - ***startdate*是计算的开始日期。**
    - ***enddate*是计算的终止日期。*enddate* 是返回 datetime 或 smalldatetime 值或日期格式字符串的表达式。**
    - 返回类型：**integer**
  - 例子： **select sname as学号, BirthDate as 出生日期, DATEDIFF(yy, birthdate,getdate()) as 年龄 from dbo.student**

知识点49：　数学函数

数学函数

- **ABS (*numeric_expression*)**
  - 返回给定数字表达式的绝对值。
- **COS (*float_expression*)**
  - 在给定表达式中以弧度形式返回给定角度的三角余弦的数学函数。
- **SIN (*float_expression*)**
  - 以弧度形式返回近似数字 (float) 表达式中给定角度的三角正弦。
- **COT (*float_expression*)**
  - 在给定 float 表达式中以弧度形式返回给定角度的三角余切的数学函数。
- **TAN (*float_expression*)**
  - 返回输入表达式的正切。

数学函数

- **ACOS (*float_expression*)**
  - 返回以弧度表示的角度，其余弦为给定的浮点表达式。也称为反余弦。
- **ASIN (*float_expression*)**
  - 返回以弧度表示的角度，其正弦为给定的浮点表达式。也称为反正弦。
- **ATAN (float_expression)**
  - 返回以弧度表示的角，其正切为指定的 **float** 表达式。它也称为反正切函数。
- **ATN2 (*float_expression, float_expression*)**
  - 返回以弧度表示的角度，其正切为两个给定 **float** 表达式的商。也称为反正切函数。

数学函数

- **CEILING (*numeric_expression*)**
  - 返回大于或等于给定数字表达式的最小整数。
- **FLOOR (*numeric_expression*)**
  - 返回小于或等于给定数字表达式的最大整数。
- **DEGREES (*numeric_expression*)**
  - 如果给出以弧度为单位的角度，则返回相应的以度数为单位的角度。
- **RADIANS (*numeric_expression*)**
  - 当输入以度数表达的数字表达式时，返回弧度。

数学函数

- **EXP (*float_expression*)**
  - 返回给定 float 表达式的指数值。
- **LOG (*float_expression*)**
  - 返回给定 float 表达式的自然对数。
- **LOG10 (*float_expression*)**
  - 返回给定 float 表达式的以 10 为底的对数。
- **PI ()**
  - 返回数学常量 pi 的常量值。
- **RAND ([*seed*])**
  - 返回介于 0 和 1 之间的随机 float 值。
- **SIGN (*numeric_expression*)**
  - 返回给定表达式的正号 (+1)、零 (0) 或负号 (-1)。

数学函数

- **POWER (*numeric_expression, y*)**
  - 返回给定表达式的指定幂的值。
- **SQUARE(numeric_expression)**
  - 返回数值表达式的平方。
- **SQRT (*float_expression*)**
  - 返回给定表达式的平方根。

数学函数

- **ROUND**
  - 返回数字表达式并四舍五入为指定的长度或精度。
- 语法：**ROUND (*numeric_expression* , *length* [, *function*])**
- 参数
  - ***numeric_expression*:**精确数字或近似数字数据类型类别的表达式（**bit** 数据类型除外）。
  - ***length*是 *numeric_expression*** 将要四舍五入的精度。当 ***length*** 为正数时，***numeric_expression*** 四舍五入为 ***length*** 所指定的小数位数。当 ***length*** 为负数时，***numeric_expression*** 则按 ***length*** 所指定的在小数点的左边四舍五入。
  - ***function*是要执行的操作类型。*function*** 必须是 **tinyint**、**smallint** 或 **int**。如果省略 ***function*** 或 ***function*** 的值为 **0**（默认），***numeric_expression*** 将四舍五入。当指定 **0** 以外的值时，将截断 ***numeric_expression***。
- 返回类型:返回与 ***numeric_expression*** 相同的类型。

数学函数

- 注释
  - **ROUND 始终返回一个值。如果 *length* 是负数且大于小数点前的数字个数，ROUND 将返回 0。**
  - **ROUND(748.58, -4)**
  - **当 *length* 是负数时，无论什么数据类型，ROUND 都将返回一个四舍五入的 *numeric_expression*。**
  - **select ROUND(534.56, 1)　　--结果为534.60**
  - **select ROUND(534.56, 0)　　--结果为535.00**
  - **select ROUND(534.56, -1)　　--结果为530.00**
  - **select ROUND(534.56, -2)　　--结果为500.00**
  - **select ROUND(534.56, -3)　　--结果为1000.00**
  - **select ROUND(534.56, -4)　　--结果为0.00**

知识点50：　系统函数

系统函数

- **HOST_ID**
  - 返回工作站标识号。
  - 语法:HOST_ID ( )
- **HOST_NAME**
  - 返回工作站名称。
  - 语法:HOST_NAME ( )

系统函数

- **SUSER_SID**
  - 返回用户登录名的安全标识号 (SID)。
  - 语法:SUSER_SID (['*login*'])
  - 参数*login*是用户的登录名。
- **SUSER_ID**
  - 返回用户的登录标识号。
  - 语法:SUSER_ID (['*login*'])
  - 参数*login*是用户的登录标识名。

系统函数

- **SUSER_SNAME**
  - 从用户的安全标识号 (SID) 返回登录标识名。
  - 语法:SUSER_SNAME ([*server_user_sid*])
  - 参数*server_user_sid*是用户的安全标识号。
- **SUSER_NAME**
  - 返回用户的登录标识名。
  - 语法：SUSER_NAME ([*server_user_id*])
  - 参数*server_user_id*是用户的登录标识号。

系统函数

- **USER_ID**
  - 返回用户的数据库标识号。
  - 语法：**USER_ID (['*user*'])**
  - 参数：‘*user*’要使用的用户名
- **USER_NAME**
  - 返回给定标识号的用户数据库用户名。
  - 语法**USER_NAME ([*id*])**
  - 参数*id*用来返回用户名的标识号。

系统函数

- **DB_ID**
  - 返回数据库标识 (ID) 号。
  - 语法DB_ID (['*database_name*'])
  - 参数‘*database_name*’是用来返回相应数据库 ID 的数据库名。
- **DB_NAME**
  - 返回数据库名。
  - 语法DB_NAME (*database_id*)
  - 参数*database_id*是应返回数据库的标识号 (ID)。

系统函数

- **OBJECT_ID**
  - 返回数据库对象标识号。
  - 语法**OBJECT_ID ('*object*')**
  - 参数‘*object*’要使用的对象。
- **OBJECT_NAME**
  - 返回数据库对象名。
  - 语法**OBJECT_NAME (*object_id*)**
  - 参数*object_id*要使用的对象的 ID。

函数练习

- 预测以下陈述的输出：**Select Round(1234.567,1)**

A. **1234.5**
B. **1234.6**
C. **1234**
D. **1234.56**

函数练习

预测以下SQL 语句的输出。

Select floor(1234.567)

1）1234.56

2）1234

3）1235

4）12345.67

函数练习

一名学生执行以下SELECT语句：

SELECT substring('Microsoft SQL Sever is a great product',2,4).

此substring 函数将返回以下串中哪一个？

A. icro

B. icr

C. ir

D. ro

函数练习

预测以下SQL 语句的输出：

```
Select * from sales  where tran_date >=dateadd(dd, -3, getdate())
```

a. 显示销售日期在当前系统日期之后3天的所有行。
b. 显示销售日期在当前系统日期之前3天的所有行。
c. 显示销售日期是当前系统日期的所有行。
d. 显示销售日期在当前系统日期之后3周的所有行。

函数练习

预测以下SQL 语句的输出，如果给定产品的销售日期是July 13, 2000，定单日期是July 1, 2000：

```
Select datediff(yy, sale_dt, order_dt)  from transaction where
prod_id = '10202'
```

A. 1

B. -1

C. 0

D. 13

函数练习

预测以下SQL语句的输出，如果某产品销售的日期是July 23, 2001 （2001年7月23日），
定单日期是 July 1, 2001（2001年7月1日）。

```
Select datediff(dd, sale_dt, order_dt)
from transaction where prod_id = ‘10202’
```

A. 22
B. -22
C. 18
D. 21

知识点51：　聚集函数

聚集函数

函数	描述
COUNT（[DISTINCT|ALL]*）	计算元组的个数
COUNT（[DISTINCT|ALL]<列名>）	对一列中的值计算个数
SUM（[DISTINCT|ALL]<列名>）	求某一列值的总和（此列的值必须是数值）
AVG（[DISTINCT|ALL] <列名>）	求某一列值的平均值（此列的值必须是数值）
MAX（[DISTINCT|ALL] <列名>）	求某一列值中的最大值
MIN（[DISTINCT|ALL] <列名>）	求某一列值中的最小值

聚集函数（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;
```

[例28] 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29] 查询选修1号课程的学生最高分数。

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

［例30］查询学生200215012选修课程的总学分数。

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

知识点52： GROUP BY子句

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

说明

- 在用分组语句时，SELECT后跟的列只能是聚集函数或者是出现在GROUP BY之后的分组列。

```
SELECT  Lang 语种, COUNT(*) as 数量
FROM Songs
GROUP BY Lang
```

语种	数量
英文	3
中文	2

错误的写法：

```
SELECT  SongID, COUNT(*) as数量
FROM Songs
GROUP BY Lang
```

GROUP BY子句（续）

- 对分组进行选择操作由HAVING子句完成
- HAVING <条件表达式>
  - 条件表达式是由分组列、聚集函数和常数构成的有意义的式子。

[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
HAVING  COUNT(*) >3;
```

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。
  - Where 后不可以直接接聚合函数

练习

- 预测以下语句的输出：

select stor_id, count(stor_id) from sales group by stor_id having avg(qty)>=20 and count（stor_id)>3

a. 查询将显示那些商店的材料，它们的订单在3次以上，订单的平均数量大于等于20。

b. 查询将显示那些商店的材料，它们订单的平均数量大于等于20。

c. 查询将显示那些商店的材料，它们的订单在3次以上。

d. 查询将显示那些商店的材料，它们的订单在3次以上，订单的数量大于等于20。

练习

- 设有关系模式：
- SB（SN，SNAME，CITY），其中，S表示供应商，SN为供应商代号，SNAME为供应商名字，CITY为供应商所在城市，主关键字为SN。
- PB（PN，PNAME，COLOR，WEIGHT），其中P表示零件，PN为零件代号，PNAME为零件名字，COLOR为零件颜色，WEIGHT为零件重量，主关键字为PN。
- JB（JN，JNAME，CITY），其中，J表示工程，JN为工程编号，JNAME为工程名字，CITY为工程所在城市，主关键字为JN。
- SPJB（SN，PN，JN，QTY），其中，SPJ表示供应关系，SN是为指定工程提供零件的供应商代号，PN为所提供的备件代号，JN为工程编号，QTY表示提供的零件数量，主关键字为SN，PN，JN，外关键字为SN，PN，JN。

SB

SN	SNAME	CITY
S1	N1	上海
S2	N2	北京
S3	N3	北京
S4	N4	上海
S5	N5	南京

PB

PN	PNAME	COLOR	WEIGHT
P1	PN1	红	12
P2	PN2	绿	18
P3	PN3	蓝	20
P4	PN4	红	13
P5	PN5	蓝	11
P6	PN6	绿	15

JB

JN	JNAME	CITY
J1	JN1	上海
J2	JN2	广州
J3	JN3	南京
J4	JN4	南京
J5	JN5	上海
J6	JN6	武汉
J7	JN7	上海

SN	PN	JN	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P3	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	1000
S5	P3	J4	1200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

1. 查询所有工程的全部细节；
2. 查询所在城市为上海的所有工程的全部细节；
3. 查询提供零件数量大于300的供应商的编号；
4. 查询为工程J1提供零件的供应商代号：
5. 查询为工程J1提供零件P1的供应商代号；
6.取出为工程J1或J2提供零件的供应商代号；
7.取出由供应商S1提供零件的工程的代号；
8.取出零件重量在10-20之间的零件信息
9.取出城市以“北”开头的供应商的编号、名称、城市
10.按供应商编号升序、数量降序输出SPJ信息
11.取出s1供应商供应的最大数量、最小数量、及其两者之差，平均数量
12.求各供应商供应零件的平均数量，并按供应商号降序排序
13.查询供应零件总数量在800以上的工程编号和其供应的总数量

知识点53： 连接查询

连接查询

- 连接查询：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

等值与非等值连接查询

- 等值连接：连接运算符为=

[例33] 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；
```

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
200215121	李勇	男	20	CS	200215121	1	92
200215121	李勇	男	20	CS	200215121	2	85
200215121	李勇	男	20	CS	200215121	3	88
200215122	刘晨	女	19	CS	200215122	2	90
200215122	刘晨	女	19	CS	200215122	3	80

等值与非等值连接查询（续）

- 自然连接：

[例34]　对[例33]用自然连接完成。

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

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
- 由于所有属性名都是同名属性，因此必须使用别名前缀

[例35]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;
```

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	PASCAL语言	6	4

自身连接（续）

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	PASCAL语言	6	4

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	PASCAL语言	6	4

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	PASCAL语言	6	4

自身连接（续）

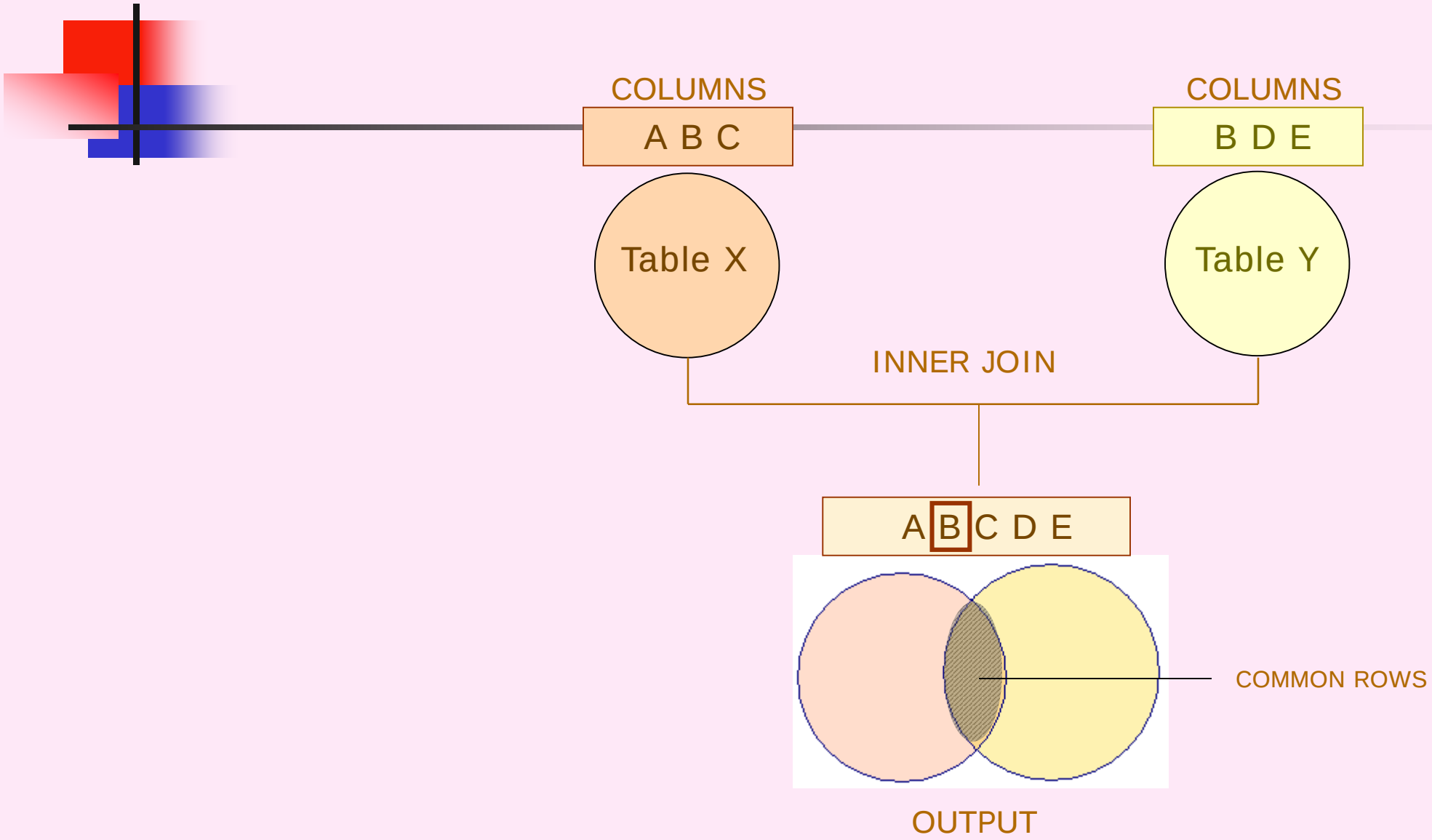
查询结果：

Cno	Pcno
1	7
3	5
5	6

知识点54： 外连接

表的连接方法

- 表的连接方法有两种：
  - 方法1：表之间满足一定的条件的行进行连接，此时FROM子句中指明进行连接的表名，WHERE子句指明连接的列名及其连接条件。
  - 方法2：利用关键字JOIN进行连接。


COLUMNS
A B C
COLUMNS
B D E
Table X
Table Y
INNER JOIN
A B C D E
COMMON ROWS
OUTPUT

内连接

- 内连接：
  - 内连接在公共的列上使用比较操作符从多表中抽取数据。
  - 仅抽取满足连接条件的行。
  - 语法：

```
SELECT column_name, column_name [,column_name]
FROM table1_name JOIN table2_name
ON table1_name.ref_column_name join_operator
table2_name.ref_column_name
```

- 内连接是默认连接

内连接

- 连接查询的运算过程：在表1中找到第一条记录，逐行扫描表2中的所有记录，若有满足连接条件的就组合表1和表2的字段为一个新的记录。依此类推，在表1中扫描完所有的记录后就组合成了连接查询的结果。
- 连接运算中的列名如果不唯一，必须指明表名或表别名。

几种连接

具体分为以下几种：

- **INNER JOIN**：显示符合条件的记录，此为默认值；
- **LEFT （OUTER） JOIN**：显示符合条件的数据行以及左边表中不符合条件的数据行，此时右边数据行会以NULL来显示，此称为左连接；
- **RIGHT （OUTER） JOIN**：显示符合条件的数据行以及右边表中不符合条件的数据行，此时左边数据行会以NULL来显示，此称为右连接；
- **FULL （OUTER） JOIN**：显示符合条件的数据行以及左边表和右边表中不符合条件的数据行，此时缺乏数据的数据行会以NULL来显示；
- **CROSS JOIN**：会将一个表的每一笔数据和另一表的每笔数据匹配成新的数据行。

- 当将JOIN 关键词放于FROM子句中时，应有关键词ON与之相对应，以表明连接的条件。

外连接

- 外连接：
  - 外连接显示包含来自一个表中所有行和来自另一个表中匹配行的结果集。
  - 外连接显示没有找到匹配记录的相关表的列为NULL。
  - 有三种类型：
    - 左外连接
    - 右外连接
    - 完全外连接

外连接

- 语法：

```
SELECT column_name, column_name [,column_name]
FROM table1_name [LEFT | RIGHT | FULL] OUTER JOIN
table2_name
ON table1_name.ref_column_name join_operator
table2_name.ref_column_name
```

COLUMNS

A B C

COLUMNS

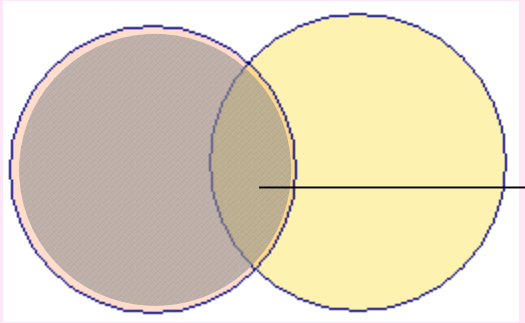
B D E

Table X

Table Y

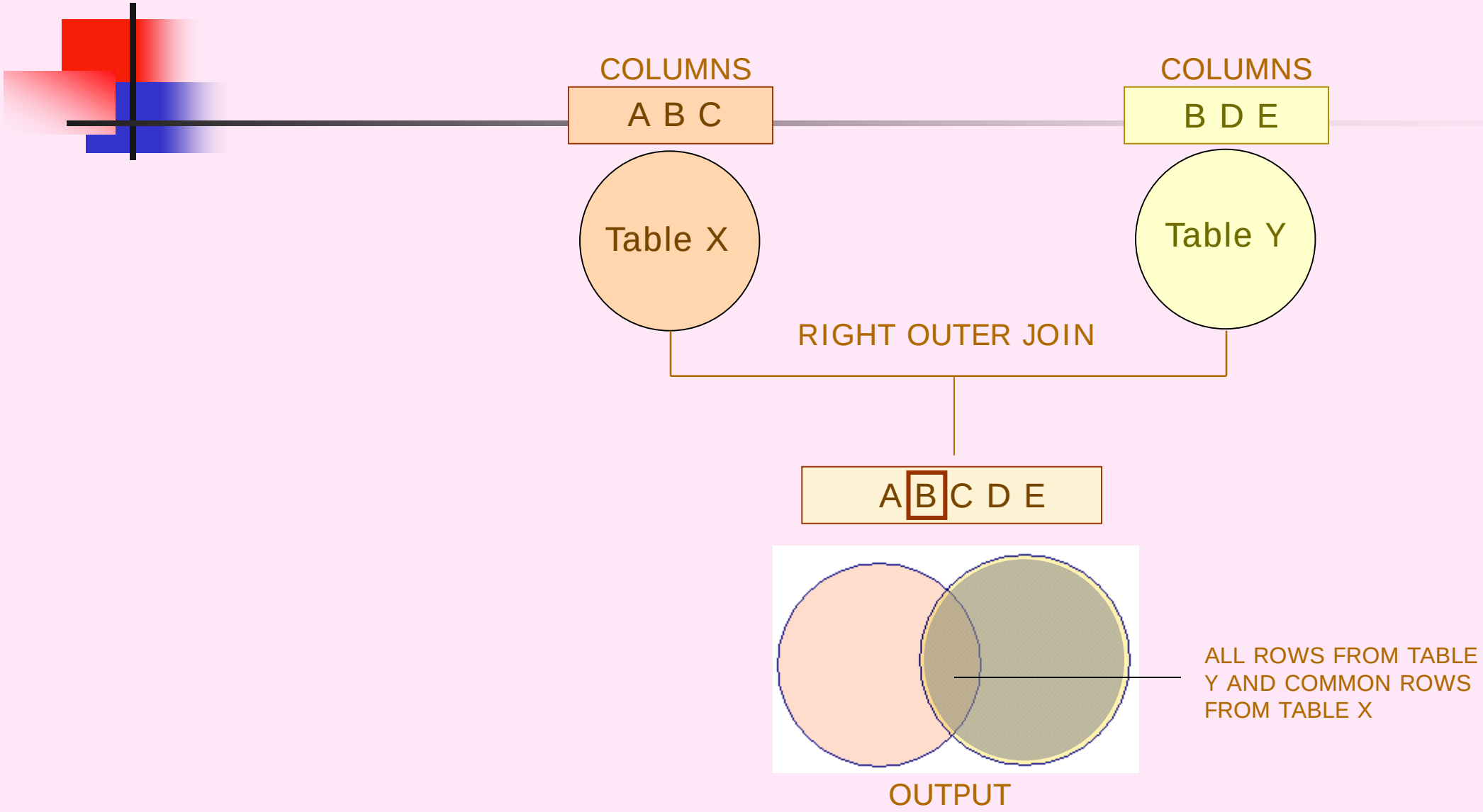
LEFT OUTER JOIN

A B C D E



ALL ROWS FROM TABLE X AND COMMON ROWS FROM TABLE Y

OUTPUT


COLUMNS
A B C
COLUMNS
B D E
Table X
Table Y
RIGHT OUTER JOIN
A B C D E
ALL ROWS FROM TABLE Y AND COMMON ROWS FROM TABLE X
OUTPUT

COLUMNS

A B C

COLUMNS

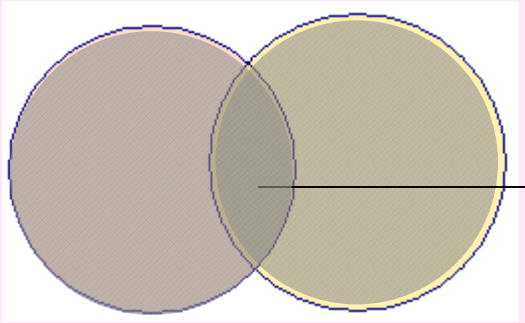
B D E

Table X

Table Y

FULL OUTER JOIN

A B C D E



ALL ROWS FROM TABLE Y AND TABLE Y AND COMMON ROWS ONLY ONCE

OUTPUT

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 36] 改写[例33]

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

外连接（续）

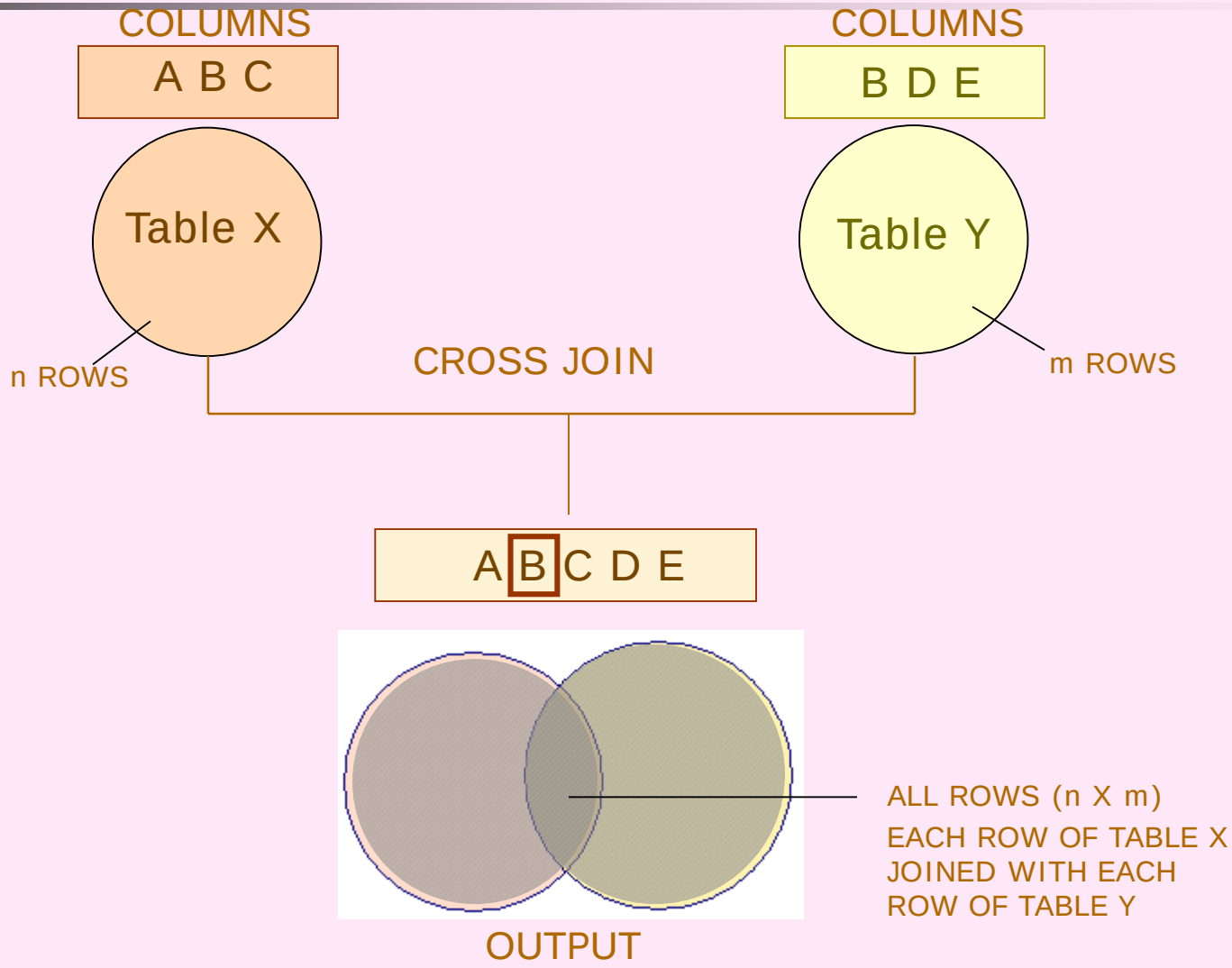
执行结果：

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
200215121	李勇	男	20	CS	1	92
200215121	李勇	男	20	CS	2	85
200215121	李勇	男	20	CS	3	88
200215122	刘晨	女	19	CS	2	90
200215122	刘晨	女	19	CS	3	80
200215123	王敏	女	18	MA	NULL	NULL
200215125	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

交叉连接


COLUMNS
A B C
COLUMNS
B D E
Table X
Table Y
n ROWS
CROSS JOIN
m ROWS
A B C D E
ALL ROWS (n X m)
EACH ROW OF TABLE X
JOINED WITH EACH
ROW OF TABLE Y
OUTPUT

交叉连接

- 交叉连接：在两个表中将一个表中的每行与另一个表中的每行连接。
  - 结果集中行的数量是第一个表中行的数量与第二个表中行的数量的乘积。

复合条件连接

- 复合条件连接：WHERE子句中含多个连接条件

[例37]查询选修2号课程且成绩在90分以上的所有学生

```
SELECT Student.Sno, Sname
FROM    Student, SC
WHERE Student.Sno = SC.Sno AND
                        /* 连接谓词*/
      SC.Cno=‘2’ AND SC.Grade > 90;
                        /* 其他限定条件 */
```

复合条件连接（续）

[例38]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；
```

知识点55：　嵌套查询

嵌套查询

- 嵌套查询概述
  - 一个SELECT-FROM-WHERE语句称为一个查询块
  - 将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中的查询称为嵌套查询

嵌套查询(续)

```
SELECT Sname                          /*外层查询/父查询*/
FROM Student
WHERE Sno IN
        （SELECT Sno          /*内层查询/子查询*/
          FROM SC
          WHERE Cno= ' 2 '）；
```

嵌套查询(续)

- 子查询的限制
  - 不能使用**ORDER BY**子句
- 层层嵌套方式反映了 **SQL**语言的结构化
- 有些嵌套查询可以用连接运算替代

嵌套查询求解方法

- 不相关子查询：

  子查询的查询条件不依赖于父查询

  - 由里向外 逐层处理。即每个子查询在上一级查询处理之前求解，子查询的结果用于建立其父查询的查找条件。

嵌套查询求解方法（续）

- 相关子查询：子查询的查询条件依赖于父查询
  - 首先取外层查询中表的第一个元组，根据它与内层查询相关的属性值处理内层查询，若**WHERE**子句返回值为真，则取此元组放入结果表
  - 然后再取外层表的下一个元组
  - 重复这一过程，直至外层表全部检查完为止

知识点56：带有比较运算符的子查询

带有比较运算符的子查询

- 当能确切知道内层查询返回单值时，可用比较运算符（>，<，=，>=，<=，!=或< >）。
- 与ANY或ALL谓词配合使用

带有比较运算符的子查询（续）

例：假设一个学生只可能在一个系学习，并且必须属于一个系：

```
SELECT Sno，Sname，Sdept
FROM    Student
WHERE Sdept   =
          (SELECT Sdept
           FROM    Student
           WHERE Sname= ‘刘晨’);
```

带有比较运算符的子查询（续）

子查询一定要跟在比较符之后。

错误的例子：

```
SELECT  Sno，Sname，Sdept
FROM    Student
WHERE ( SELECT Sdept
        FROM Student
        WHERE Sname=‘刘晨’ )
        = Sdept;
```

知识点57：带有IN谓词的子查询

带有IN谓词的子查询

[例39] 查询与“刘晨”在同一个系学习的学生。

此查询要求可以分步来完成

① 确定“刘晨”所在系名

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

结果为： CS

带有IN谓词的子查询（续）

② 查找所有在IS系学习的学生。

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

结果为：

Sno	Sname	Sdept
200215121	李勇	CS
200215122	刘晨	CS

带有IN谓词的子查询（续）

将第一步查询嵌入到第二步查询的条件中

```
SELECT Sno，Sname，Sdept
FROM Student
WHERE Sdept  IN
      (SELECT Sdept
       FROM Student
       WHERE Sname=‘刘晨’);
```

此查询为不相关子查询。

带有IN谓词的子查询（续）

用自身连接完成[例39]查询要求

```
SELECT  S1.Sno，S1.Sname，S1.Sdept

FROM    Student S1，Student S2

WHERE  S1.Sdept = S2.Sdept  AND

          S2.Sname = '刘晨';
```

带有IN谓词的子查询（续）

[例40]查询选修了课程名为“信息系统”的学生学号和姓名

```
SELECT Sno，Sname
FROM   Student
WHERE Sno  IN
      (SELECT Sno
       FROM   SC
       WHERE  Cno IN
            (SELECT Cno
             FROM Course
             WHERE Cname= ‘信息系统’
            )
       );
```

③ 最后在Student关系中取出Sno和Sname

② 然后在SC关系中找出选修了3号课程的学生学号

① 首先在Course关系中找出“信息系统”的课程号，为3号

带有IN谓词的子查询（续）

用连接查询实现[例40]

```
SELECT Sno，Sname
FROM   Student，SC，Course
WHERE Student.Sno = SC.Sno  AND
          SC.Cno = Course.Cno AND
          Course.Cname=‘信息系统’；
```

知识点57：带有ANY（SOME）或ALL谓词的子查询

三、带有ANY（SOME）或ALL谓词的子查询

谓词语义

- **ANY**：任意一个值
- **ALL**：所有值

带有ANY（SOME）或ALL谓词的子查询 （续）

需要配合使用比较运算符

> ANY	大于子查询结果中的某个值
> ALL	大于子查询结果中的所有值
< ANY	小于子查询结果中的某个值
< ALL	小于子查询结果中的所有值
>= ANY	大于等于子查询结果中的某个值
>= ALL	大于等于子查询结果中的所有值
<= ANY	小于等于子查询结果中的某个值
<= ALL	小于等于子查询结果中的所有值
= ANY	等于子查询结果中的某个值
=ALL	等于子查询结果中的所有值（通常没有实际意义）
!=（或<>）ANY	不等于子查询结果中的某个值
!=（或<>）ALL	不等于子查询结果中的任何一个值

带有ANY（SOME）或ALL谓词的子查询 （续）

[例42] 查询其他系中比计算机科学某一学生年龄小的学生姓名和年龄

```
SELECT Sname，Sage
FROM    Student
WHERE Sage < ANY (SELECT  Sage
                  FROM    Student
                  WHERE Sdept= ' CS ')
    AND Sdept <> ‘CS ' ;        /*父查询块中的条件 */
```

带有ANY（SOME）或ALL谓词的子查询 （续）

结果：

Sname	Sage
王敏	18
张立	19

执行过程：

1.RDBMS执行此查询时，首先处理子查询，找出CS系中所有学生的年龄，构成一个集合(20，19)

2. 处理父查询，找所有不是CS系且年龄小于20 或 19的学生

带有ANY（SOME）或ALL谓词的子查询 （续）

用聚集函数实现[例42]

```
SELECT Sname，Sage
 FROM   Student
 WHERE Sage <
                (SELECT MAX(Sage)
                  FROM Student
                  WHERE Sdept= ‘CS ')
     AND Sdept <> ' CS ’;
```

带有ANY（SOME）或ALL谓词的子查询 （续）

[例43] 查询其他系中比计算机科学系所有学生年龄都小的学生姓名及年龄。

方法一：用ALL谓词

```
SELECT Sname，Sage
FROM Student
WHERE Sage < ALL
            (SELECT Sage
             FROM Student
             WHERE Sdept= ' CS ')
    AND Sdept <> ' CS ’;
```

带有ANY（SOME）或ALL谓词的子查询 （续）

方法二：用聚集函数

```
SELECT Sname，Sage
FROM Student
WHERE Sage <
                (SELECT MIN(Sage)
                 FROM Student
                 WHERE Sdept= ' CS ')
    AND Sdept <>' CS ’;
```

带有ANY（SOME）或ALL谓词的子查询 （续）

表3.5 ANY（或SOME），ALL谓词与聚集函数、IN谓词的等价转换关系

	=	<>或!=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>= MIN
ALL	--	NOT IN	<MIN	<= MIN	>MAX	>= MAX

知识点58：集合查询

集合查询

- 集合操作的种类
  - 并操作UNION
  - 交操作INTERSECT
  - 差操作EXCEPT
- 参加集合操作的各查询结果的列数必须相同；对应项的数据类型也必须相同

集合查询（续）

[例48] 查询计算机科学系的学生及年龄不大于19岁的学生。

方法一：

```
SELECT *
FROM Student
WHERE Sdept= 'CS'
UNION
SELECT *
FROM Student
WHERE Sage<=19;
```

- UNION：将多个查询结果合并起来时，系统自动去掉重复元组。
- UNION ALL：将多个查询结果合并起来时，保留重复元组

集合查询（续）

方法二：

```
SELECT  DISTINCT  *
FROM Student
WHERE Sdept= 'CS'  OR  Sage<=19;
```

集合查询（续）

[例49] 查询选修了课程1或者选修了课程2的学生。

```
SELECT Sno
FROM SC
WHERE Cno=' 1 '
UNION
SELECT Sno
FROM SC
WHERE Cno= ' 2 ';
```

集合查询（续）

[例50] 查询计算机科学系的学生与年龄不大于19岁的学生的交集

```
SELECT *
FROM Student
WHERE Sdept='CS'
INTERSECT
SELECT *
FROM Student
WHERE Sage<=19
```

集合查询（续）

- **[例50] 实际上就是查询计算机科学系中年龄不大于19岁的学生**

```
SELECT *
FROM Student
WHERE Sdept= 'CS' AND  Sage<=19;
```

集合查询（续）

[例51] 查询选修课程1的学生集合与选修课程2的学生集合的交集

```
SELECT Sno
FROM SC
WHERE Cno=' 1 '
INTERSECT
SELECT Sno
FROM SC
WHERE Cno='2 ';
```

集合查询（续）

[例51]实际上是查询既选修了课程1又选修了课程2的学生

```
SELECT Sno
FROM SC
WHERE Cno=' 1 ' AND Sno IN
                    (SELECT Sno
                     FROM SC
                     WHERE Cno=' 2 ');
```

集合查询（续）

[例52] 查询计算机科学系的学生与年龄不大于19岁的学生的差集。

```
SELECT *
FROM Student
WHERE Sdept='CS'
EXCEPT
SELECT  *
FROM Student
WHERE Sage <=19;
```

集合查询（续）

[例52]实际上是查询计算机科学系中年龄大于19岁的学生

```
SELECT *
FROM Student
WHERE Sdept= 'CS' AND  Sage>19;
```

知识点59：插入元组

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

     - 可以一次插入多个元组

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
  - 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
  - 指定部分属性列
- 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1］ 将一个新学生元组（学号：200215128；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 ('200215128'，'陈冬'，'男'，'IS'，18);
```

插入元组（续）

［例2］ 将学生张成民的信息插入到Student表中。

```
INSERT
INTO  Student
VALUES (‘200215126’，  ‘张成民’，  ‘男’，18，'CS');
```

插入元组（续）

［例3］ 插入一条选课记录('200215128'，'1 ')。

```
INSERT
INTO SC(Sno，Cno)
VALUES (‘ 200215128 ’，  ‘ 1 ’);
```

RDBMS将在新插入记录的Grade列上自动地赋空值。

或者：

```
INSERT
INTO SC
VALUES (' 200215128 '，' 1 '，NULL);
```

知识点60：插入子查询

插入子查询结果

- 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

插入子查询结果（续）

- **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4] 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*/
```

插入子查询结果（续）

第二步：插入数据

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

插入子查询结果（续）

- 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

说明

- 使用Column_list说明：
  - 必须用括号将 column_list 括起来，并且用逗号进行分隔
  - 以下类型可不包含在column_list 中，数据库引擎自动为其提供值：
    - 具有 IDENTITY 属性，使用下一个增量标识值。
      - 当向标识列中插入显式值时，必须使用 column_list 和 VALUES 列表，并且表的 SET IDENTITY_INSERT 选项必须为 ON。
    - 有默认值，使用列的默认值。
    - 可为空值，使用空值。
    - 是计算列，使用计算值。

知识点61：数据更新

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

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

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

修改某一个元组的值

[例5] 将学生200215121的年龄改为22岁

```
UPDATE  Student
SET Sage=22
WHERE  Sno=' 200215121 ';
```

修改多个元组的值

[例6] 将所有学生的年龄增加1岁

```
UPDATE Student

SET Sage= Sage+1;
```

带子查询的修改语句

[例7] 将计算机科学系全体学生的成绩置零。

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

知识点62：删除数据

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

删除某一个元组的值

[例8] 删除学号为200215128的学生记录。

```
DELETE
FROM Student
WHERE Sno= 200215128 ';
```

删除多个元组的值

[例9] 删除所有的学生选课记录。

```
DELETE
FROM SC;
```

带子查询的删除语句

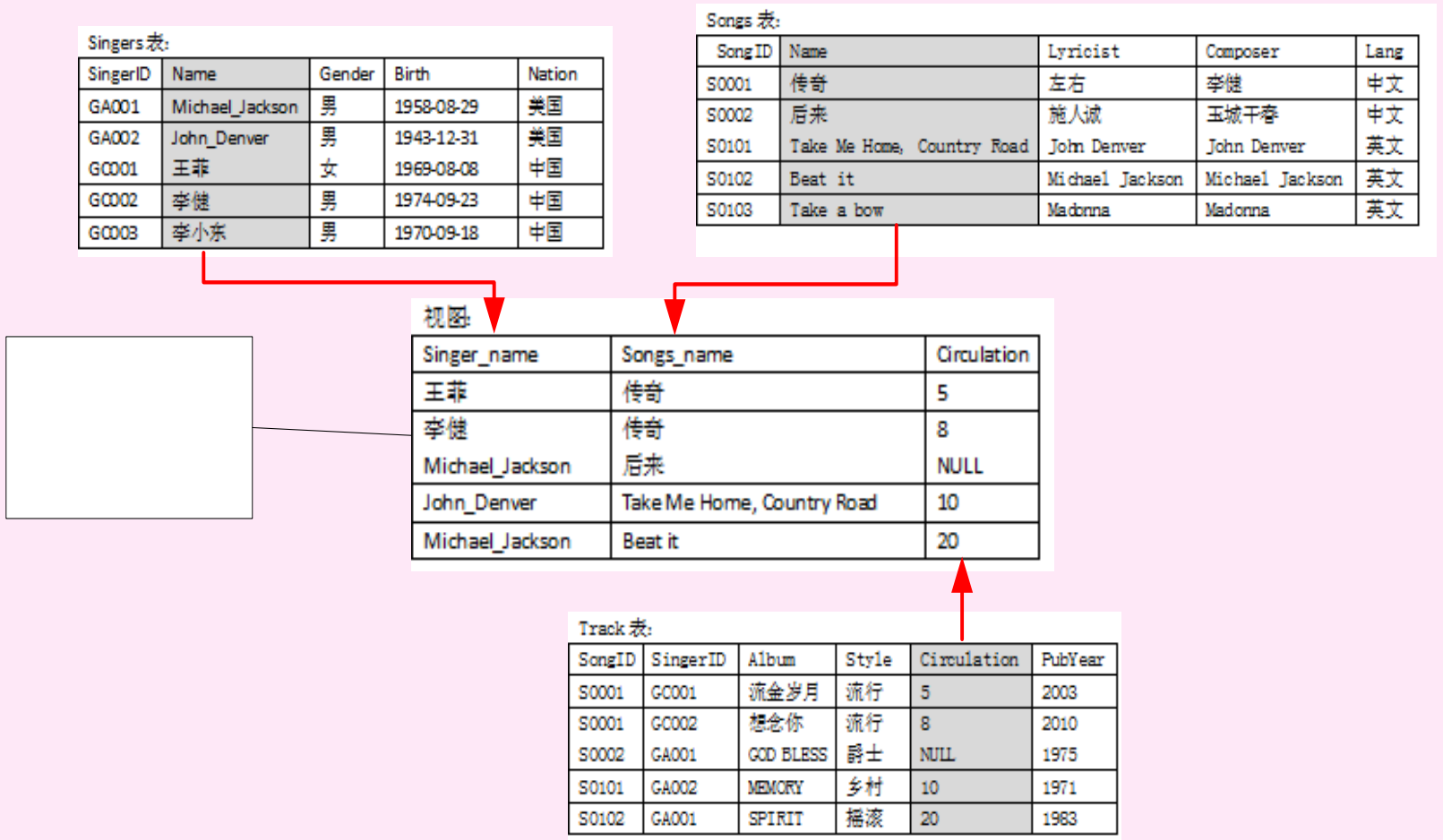
[例10] 删除计算机科学系所有学生的选课记录。

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

知识点63：视图的定义

引入视图的原因

- 从业务数据角度来看
  - 由于数据库设计时考虑到数据异常等问题，同一种业务数据有可能被分散在不同的表中
  - 对这种业务数据的使用经常是同时使用的，要实现这样的查询，需要通过连接查询或嵌套查询来实现。



Singers表:

SingerID	Name	Gender	Birth	Nation
GA001	Michael_Jackson	男	1958-08-29	美国
GA002	John_Denver	男	1943-12-31	美国
GC001	王菲	女	1969-08-08	中国
GC002	李健	男	1974-09-23	中国
GC003	李小东	男	1970-09-18	中国

Songs 表:

SongID	Name	Lyricist	Composer	Lang
S0001	传奇	左右	李健	中文
S0002	后来	施人诚	玉城千春	中文
S0101	Take Me Home, Country Road	John Denver	John Denver	英文
S0102	Beat it	Michael Jackson	Michael Jackson	英文
S0103	Take a bow	Madonna	Madonna	英文

视图:

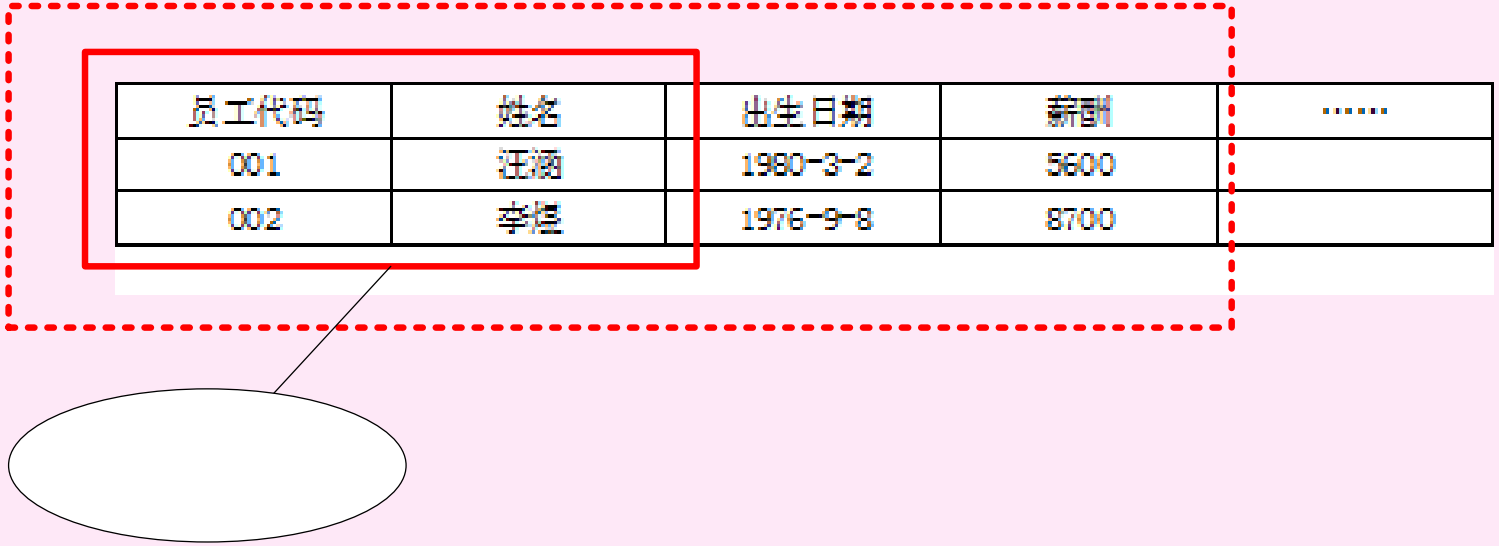
Singer_name	Songs_name	Circulation
王菲	传奇	5
李健	传奇	8
Michael_Jackson	后来	NULL
John_Denver	Take Me Home, Country Road	10
Michael_Jackson	Beat it	20

Track 表:

SongID	SingerID	Album	Style	Circulation	PubYear
S0001	GC001	流金岁月	流行	5	2003
S0001	GC002	想念你	流行	8	2010
S0002	GA001	GOD BLESS	爵士	NULL	1975
S0101	GA002	MEMORY	乡村	10	1971
S0102	GA001	SPIRIT	摇滚	20	1983

引入视图的原因（续）

- 从数据安全角度来看
  - 由于工作性质和需求不同，不同的操作人员只能查看表中的部分数据，不能查看表中的所有数据。



员工代码	姓名	出生日期	薪酬	……
001	汪洋	1980-3-2	5600	
002	李怪	1976-9-8	8700	

引入视图的原因（续）

- 从数据的应用角度来看
  - 一个报表中的数据往往来自于多个不同的表中。在设计报表时，需要明确地指定数据的来源途径和方式。通过视图机制，可以提高报表的设计效率。
- 视图是RDBMS提供给用户以多种角度来观察数据库中数据的重要机制。

视图是什么

- 视图是查看表中数据的一种方式
- 当一些用户需要经常访问和查询数据表中的某些字段构成的数据，但DBA从安全角度考虑不希望用户直接接触数据表时，可以利用视图
- 视图犹如数据表的窗户，管理员定义这些“窗户”的位置时，用户就只能查看他可以看到的数据

视图是什么

- 视图不是数据表，仅仅是一些SQL查询语句的集合，作用是按照不同的要求从数据表中提取的数据
- 对普通的用户而言，视图如图一张真实的表

视图和表的区别

- 与真实的表不同，视图中没有存储任何数据——虚表
- 在视图中被查询的表被称为基表
- 仅仅是一种较简单的访问数据库里其他表中的数据的方式
- 数据仍然存储在表中

视图的优点

- 可以使视图集中数据、简化和定制不同用户对数据库的不同数据要求
- 使用视图可以屏蔽数据的复杂性，用户不必了解数据库的结构，可以方便的使用和管理数据，简化数据权限管理

视图的优点

- 视图可以使用用户只关心他感兴趣的某些特定数据和他们所负责的特定任务，而那些不需要的或者无用的数据则不在视图中显示
- 在某些情况下，由于表中数据量太大，因此在表的设计时常将表进行水平或者垂直分割，但表的结构的变化会对应用程序产生不良的影响。视图提供了简单有效的安全机制。

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

知识点64：定义视图

基于视图的操作

基于视图的操作

- 查询
- 删除
- 受限更新
- 定义基于该视图的新视图

定义视图

- 建立视图
- 删除视图

建立视图

- 语句格式

  CREATE VIEW

  <视图名> [(<列名> [，<列名>]…)]

  AS <子查询>

  [WITH CHECK OPTION];

- **子查询不允许含有ORDER BY子句和DISTINCT短语**
- **RDBMS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。**

说明

- 组成视图的属性列名：全部省略或全部指定
  - 如果省略了各个属性列名，则隐含该视图由子查询中select子句目标列中的各字段组成；
  - 在下列情况下必须明确指定组成视图的所有列名：
    - 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定义新的名字。

创建视图的规则

- 只能在当前数据库中创建视图
- 如果视图引用的基表或者视图被删除，则该视图不能再被使用，直到创建新的基表或者视图
- 如果视图某一列是函数、数学表达式、常量或者来自多个表的列名相同，则必须为列定义名称

创建视图的规则

- 在通过视图查询数据时，SQL SERVER 要检查以确保语句数据的数据库对象存在，而且数据修改语句不能违法数据完整性规则
- 视图的名称是唯一的，且不能和表名相同
- 视图不能从临时表中产生数据

建立视图（续）

[例1] 建立信息系学生的视图。

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

建立视图（续）

[例2]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;
```

行列子集视图

- 行列子集视图
  - 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或某些列，但保留了主键，称为行列子集视图

建立视图（续）

- 基于多个基表的视图

[例3] 建立信息系统选修了1号课程的学生视图。

```
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'；
```

建立视图（续）

- 基于视图的视图

[例4] 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；
```

建立视图（续）

- 带表达式的视图

[例5] 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，2016-Sage
FROM  Student;
```

带表达式的视图必须明确组成视图的各个属性列名。

建立视图（续）

- 分组视图

[例6] 将学生的学号及他的平均成绩定义为一个视图

假设SC表中“成绩”列Grade为数字型

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

知识点65：删除视图

删除视图

- 语句的格式：

  DROP VIEW <视图名>；

  - 该语句从数据字典中删除指定的视图定义
  - 由该视图导出的其他视图定义仍在数据字典中，但已不能使用，必须使用**drop view**语句显式删除
  - 删除基表时，由该基表导出的所有视图定义都必须显式地使用**DROP VIEW**语句删除

删除视图(续）

［例8］ 删除视图BT_S： DROP VIEW BT_S；

删除视图IS_S1：DROP VIEW IS_S1；

知识点66：查询视图

查询视图（续）

[例9] 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;
```

IS_Student视图的定义：

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

查询视图（续）

[例11]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;
```

知识点67：更新视图

视图更新

- 更新视图包括insert、delete、update操作
- 由于视图是“虚表”，所以对视图的更新最终转化为对基表的更新
- 一般的，行列子集视图都是可以更新的
- 对视图进行更新操作时，还要注意基本表对数据的各种约束和规则要求。

更新视图（续）

[例12] 将信息系学生视图IS_Student中学号200215122的学生姓名改为“刘辰”。

```
UPDATE  IS_Student
SET  Sname= '刘辰'
WHERE  Sno= ' 200215122 ';
```

转换后的语句：

```
UPDATE  Student
SET Sname= '刘辰'
WHERE Sno= ' 200215122 ' AND Sdept= 'IS';
```

IS_Student视图的定义：

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

更新视图（续）

[例13] 向信息系学生视图IS_S中插入一个新的学生记录：200215129，赵新，20岁

```
INSERT
INTO IS_Student
VALUES(‘95029’，‘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（续）

[例14]删除信息系学生视图IS_Student中学号为200215129的记录

```
DELETE
FROM IS_Student
WHERE Sno= ' 200215129 ';
```

转换为对基本表的更新：

```
DELETE
FROM Student
WHERE Sno= ' 200215129 ' AND Sdept= 'IS';
```

更新视图（续）

- 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

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;
```

例：视图S_G为不可更新视图。

```
UPDATE  S_G
SET     Gavg=90
WHERE  Sno= '200215121';
```

这个对视图的更新无法转换成对基本表SC的更新

可更新视图的条件

- 创建视图的select 语句中没有聚合函数，且没有top、group by、having及distinct 关键字；
- 创建视图的select 语句的各列必须来自于基表（视图）的列，不能是表达式；
- 视图定义必须是一个简单的SELECT语句，不能带连接、集合操作。即SELECT语句的FROM子句中不能出现多个表，也不能有JOIN、EXCEPT、UNION、INTERSECT ；

```
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';
```

例3:向视图is_s1中插入一个新的学生记录，学号为200515027，姓名为王唔，成绩为60；

Insert into is_s1

Values(‘200515027’,‘王唔',60)

系统将发出错误信息：“视图或函数 is_s1 不可更新，因为修改会影响多个基表”。在表sc中，只有成绩而主键课程号cno不确定，显然不能把数据插入sc表中。

通过视图插入、修改、删除数据时的限制

- 对视图的修改一次仅影响一张表
- 不能修改计算列

知识点68：索引概念

索引

- 实现数据快速查询的数据对象。
- 索引是与表或视图关联的磁盘上的结构，可以加快从表或视图中检索行的速度。
- 索引包含由表或视图中的一列或多列生成的键，以及映射到指定数据的存储位置的指针。
  - 这些键存储在一个结构（B树，即平衡树）中，使SQL Server可以快速有效地查找与键值有关的行。

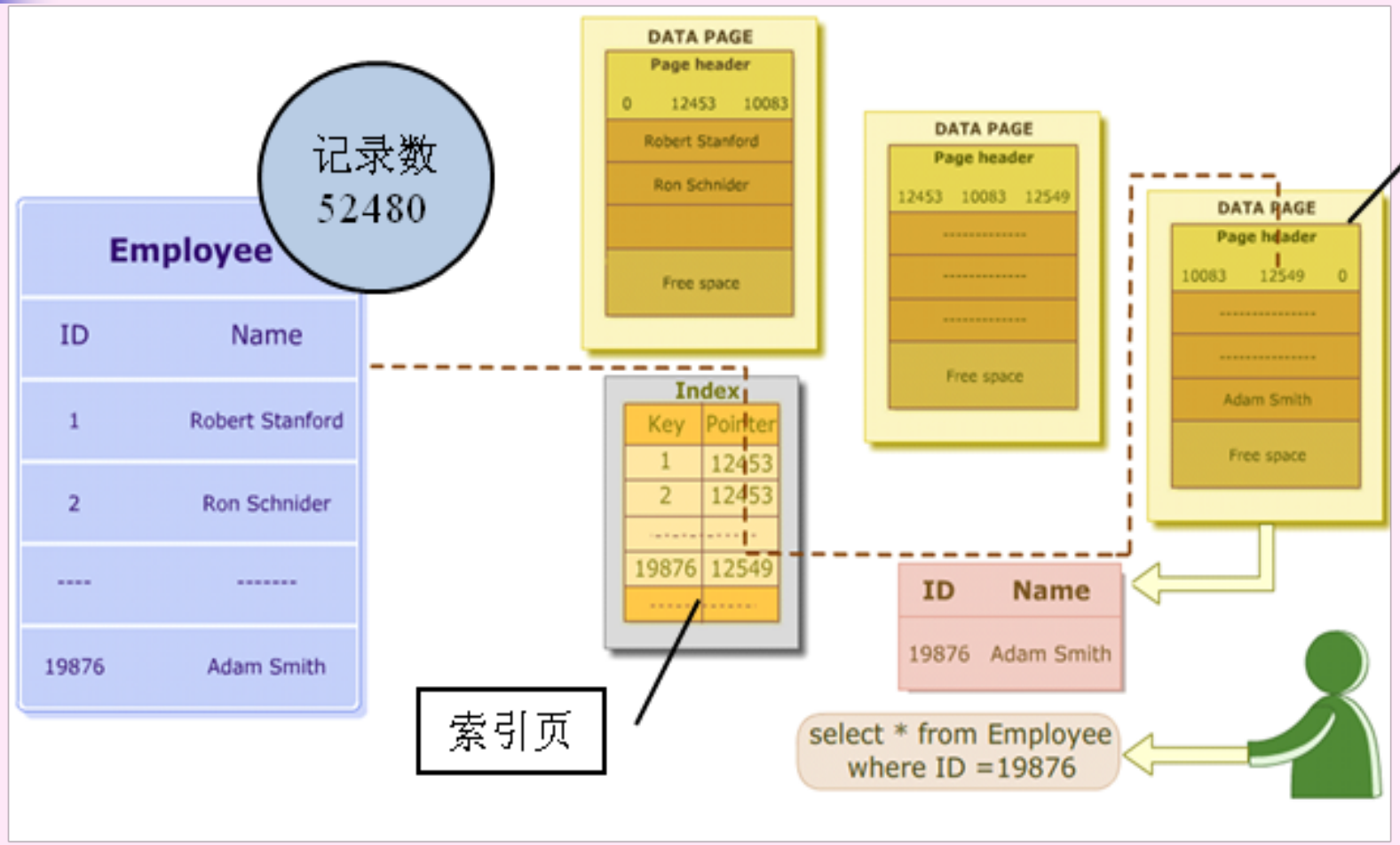
索引的作用

- 全表扫描：这是一种比较直接的处理。
  - 扫描表时，查询优化器读取表中的所有行，并提取满足查询条件的行。如果不使用索引，就需将数据文件分块，逐个读到内存中进行查找的比较操作
  - 因此扫描表会有许多磁盘 I/O 操作，并占用大量资源。
  - 但是，如果查询的结果集是占表中较高百分比的行（比如总归100条记录，需要返回80条记录），则扫描表会是最为有效的方法。

索引的作用（续）

- 索引扫描：
  - 查询优化器使用索引时，搜索索引键列，查找到查询所需行的存储位置，然后从该位置提取匹配行。
  - 通常，搜索索引比搜索表要快很多，因为索引与表不同，一般每行包含的列比全表要少得多，且行遵循排序顺序。
  - 因此，可以极大地提高查询的速度。

索引的作用（续）


记录数
52480
Employee
ID
Name
1
Robert Stanford
2
Ron Schnider

19876
Adam Smith
DATA PAGE
Page header
0 12453 10083
Robert Stanford
Ron Schnider
Free space
DATA PAGE
Page header
12453 10083 12549
Free space
DATA PAGE
Page header
10083 12549 0
Adam Smith
Free space
数据页
Index
Key
Pointer
1
12453
2
12453
19876
12549
索引页
ID
Name
19876
Adam Smith
select * from Employee
where ID =19876

- 好处：
  - 确保快速访问数据
  - 加快连接表的查询
  - 在使用**GROUP BY**和**ORDER BY**子句进行查询时，利用索引可以减少排序和分组的时间。
  - 增强行的唯一性
- 索引将占用用户数据库的空间

说明

- 说明：
  - 一般不用对小数据量的表创建索引，因为查询优化器在遍历索引时，花费的时间可能比执行简单的表扫描还长。
  - 查询优化器在执行查询时，通常会选择最有效的方法。如果没有索引，则查询优化器必须扫描表。用户的任务是设计并创建最适合自己环境的索引，以便查询优化器可以从多个有效的索引中选择。
  - 索引总是在最常查询的列上创建
  - **Sql server** 在表上创建**primary key**或**unique**约束时，索引以与约束同样的名称被自动创建

SQL SERVER 的索引类型

- 聚簇索引（Cluster Index）
- 非聚簇索引（Non-Cluster Index ）
- 唯一索引（Unique Index）
- 复合索引

知识点69：聚簇索引

聚簇索引的特点

- 聚簇索引是一种对磁盘上实际数据重新组织以按指定的一个或多个列的值排序。
- 创建聚簇索引时，需要对已有表数据重新进行排序（若表中已有数据），故聚簇索引建立完毕后，建立聚簇索引的列中的数据已经全部按序排列。
- 由于聚簇索引的索引页面指针指向数据页面，所以使用聚簇索引查找数据几乎总是比使用非聚簇索引快。
- 表的数据按照索引的数据顺序排列
  - 建立聚簇索引后，更新索引列数据，往往会导致表中物理记录的存储顺序的变化，维护的代价比较大
  - 对于需要经常更新的列，不宜建立聚簇索引,因为码值修改后，数据行必须移动到新的位置。

聚簇索引

- 聚簇索引确定表中数据的物理顺序。
- 每个表只能建立一个聚簇索引，但该索引可以包含多个列。
- 往往在主码所在的列或者最常查询的列上建立聚簇索引
- 每个表几乎都对列定义聚集索引来实现下列功能：
  - 可用于经常使用的查询。
  - 提供高度唯一性。
  - 适合范围查询：使用聚簇索引找到包含第一个值的行后，便可以确保包含后续索引值的行在物理相邻。
  - 在聚簇索引下，数据在物理上按顺序排在数据页上，重复值也排在一起，因而在那些包含范围检查(between、<、<=、>、>=)或使用group by或orderby的查询时，一旦找到具有范围中第一个键值的行，具有后续索引值的行保证物理上毗连在一起而不必进一步搜索，避免了大范围扫描，可以大大提高查询速度。

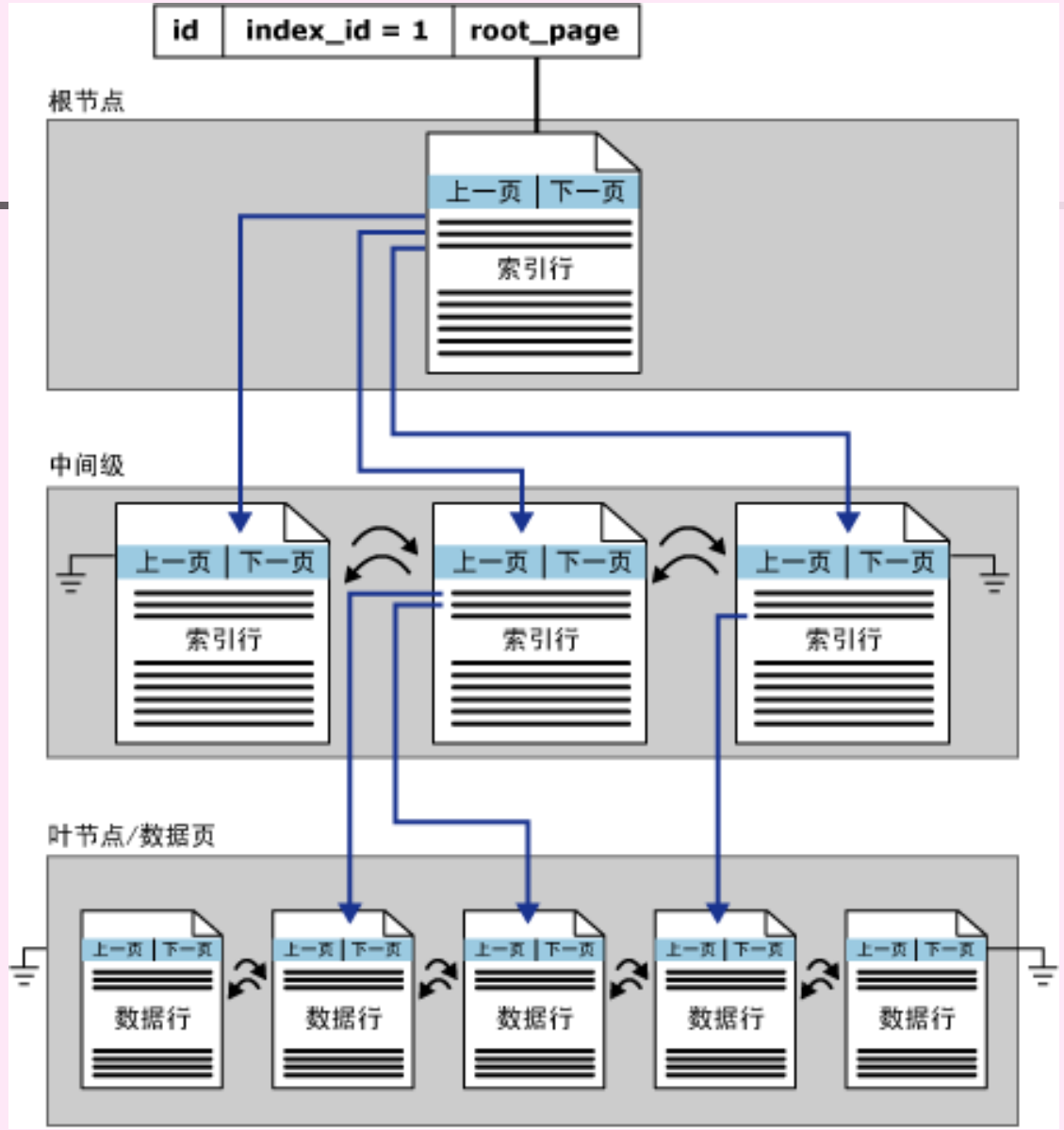
聚簇索引的侯选列

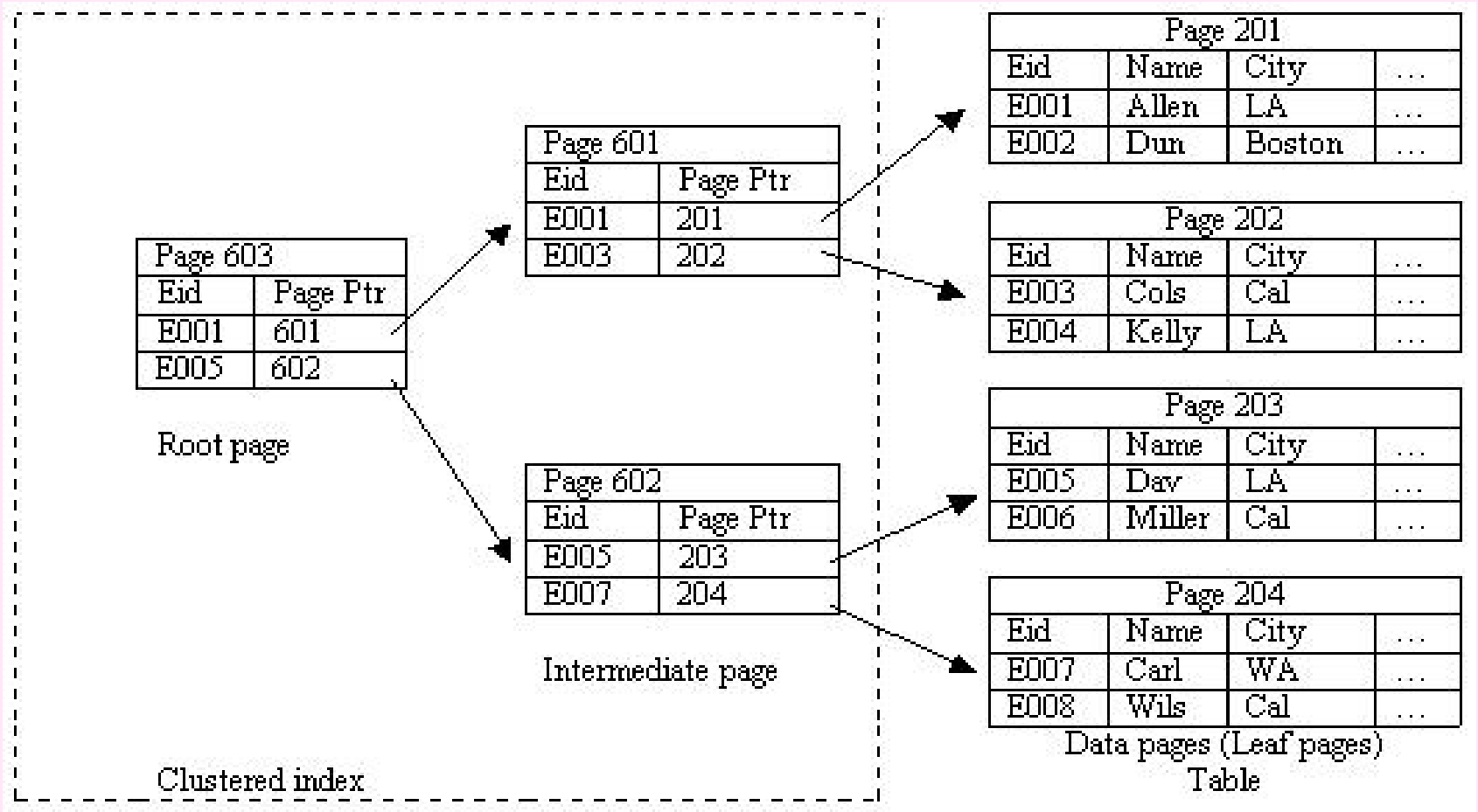
- 聚簇索引的侯选列
  - 主键列,该列在where子句中使用并且插入是随机的。
  - 按范围存取的列，

    如pri_order > 100 and pri_order < 200。
  - 在group by或order by中使用的列。
  - 不经常修改的列。
  - 在连接操作中使用的列。

说明

- 创建 **PRIMARY KEY** 约束时，将在列上自动创建唯一索引。默认情况下，此索引是聚集索引，但是在创建约束时，可以指定创建非聚集索引。
- 对于经常更新的列，不宜建立聚集索引。
- 叶节点就是数据节点
- 只有当表包含聚集索引时，表中的数据行才按排序顺序存储；如果表没有聚集索引，则其数据行存储在一个成为堆（**Heap**）的无序结构中。

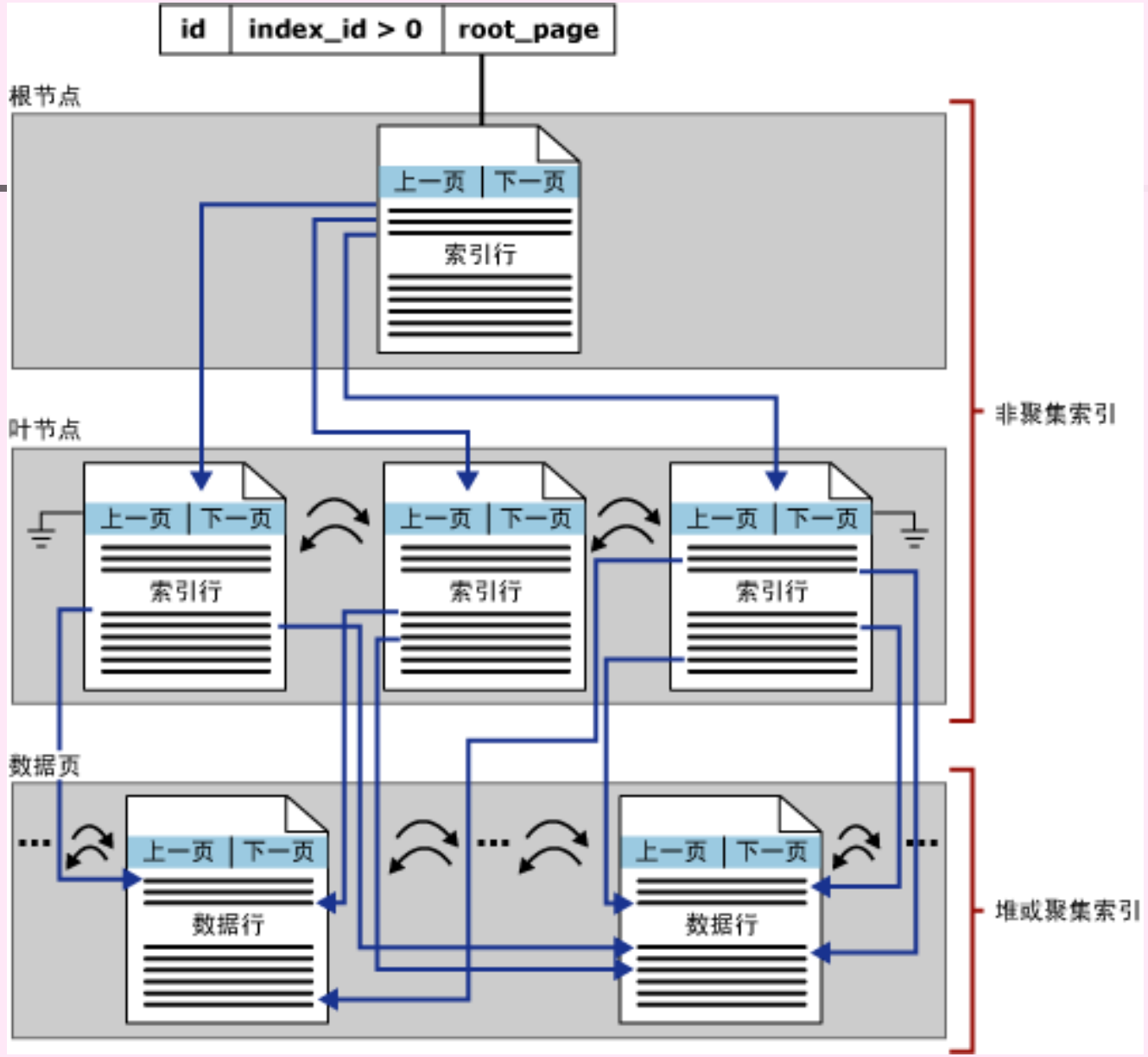

id
index_id = 1
root_page
根节点
上一页
下一页
索引行
中间级
上一页
下一页
索引行
上一页
下一页
索引行
上一页
下一页
索引行
叶节点/数据页
上一页
下一页
数据行
上一页
下一页
数据行
上一页
下一页
数据行
上一页
下一页
数据行
上一页
下一页
数据行

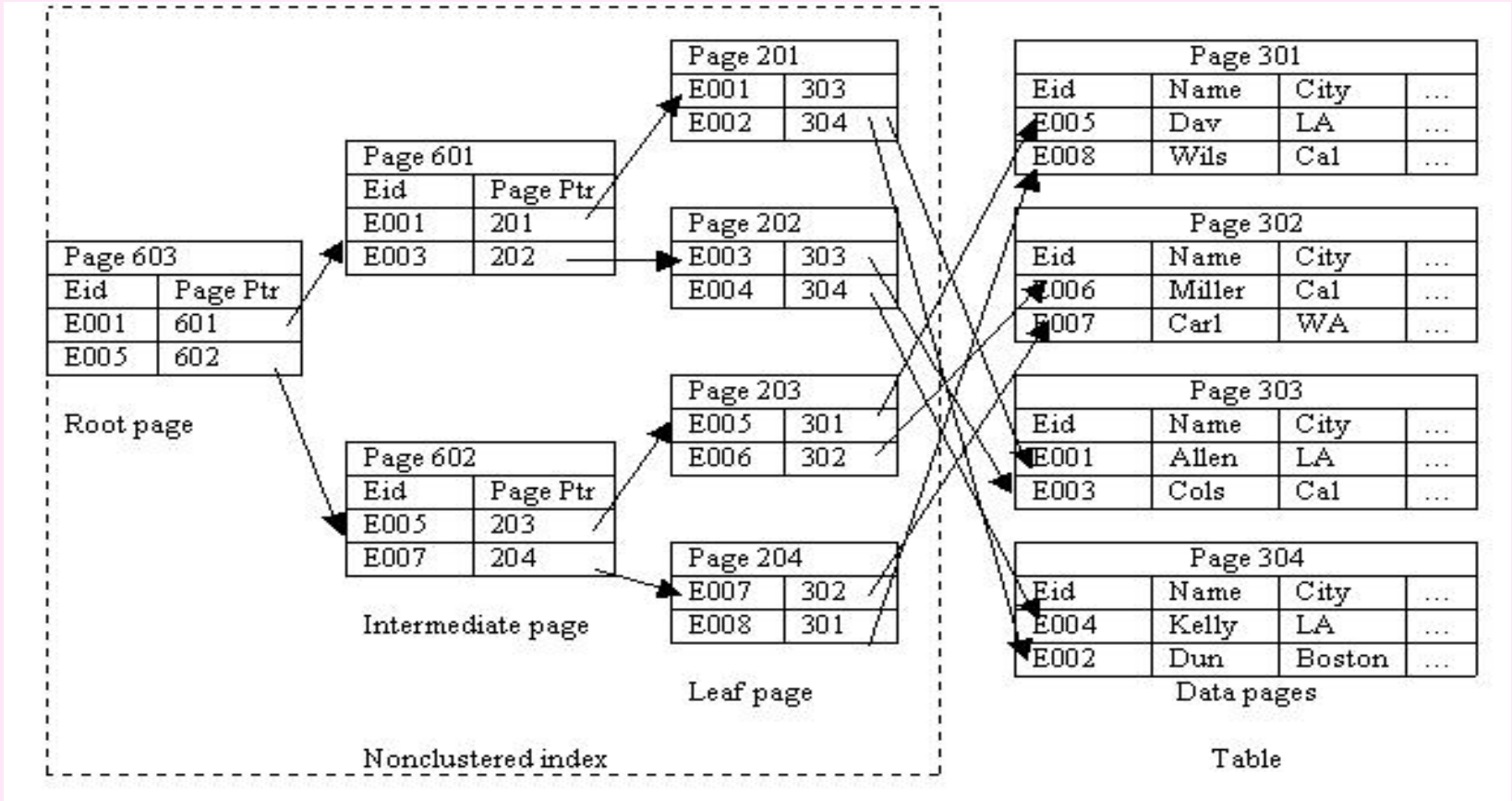

Page 603
Eid
Page Ptr
E001
601
E005
602
Root page
Page 601
Eid
Page Ptr
E001
201
E003
202
Page 602
Eid
Page Ptr
E005
203
E007
204
Intermediate page
Clustered index
Page 201
Eid
Name
City
...
E001
Allen
LA
...
E002
Dun
Boston
...
Page 202
Eid
Name
City
...
E003
Cols
Cal
...
E004
Kelly
LA
...
Page 203
Eid
Name
City
...
E005
Dav
LA
...
E006
Miller
Cal
...
Page 204
Eid
Name
City
...
E007
Carl
WA
...
E008
Wils
Cal
...
Data pages (Leaf pages)
Table

知识点70：非聚簇索引

非聚簇索引

- 非聚簇索引存储的数据一般和表的物理数据的存储不同
- 索引是有序的，而表中的数据是无序的。
- 尽管查询的速度慢一些，但是维护的代价小
- 表中最多可以建立249个非聚簇索引以满足多种查询的需要
- 叶节点仍然是索引节点，只不过有一个指针指向对应的数据块。


id
index_id > 0
root_page
根节点
上一页
下一页
索引行
叶节点
上一页
下一页
索引行
上一页
下一页
索引行
上一页
下一页
索引行
非聚集索引
数据页
上一页
下一页
数据行
上一页
下一页
数据行
堆或聚集索引


Page 603
Eid
Page Ptr
E001
601
E005
602
Root page
Page 601
Eid
Page Ptr
E001
201
E003
202
Page 602
Eid
Page Ptr
E005
203
E007
204
Intermediate page
Page 201
E001
303
E002
304
Page 202
E003
303
E004
304
Page 203
E005
301
E006
302
Page 204
E007
302
E008
301
Leaf page
Nonclustered index
Page 301
Eid
Name
City
...
E005
Dav
LA
...
E008
Wils
Cal
...
Page 302
Eid
Name
City
...
E006
Miller
Cal
...
E007
Carl
WA
...
Page 303
Eid
Name
City
...
E001
Allen
LA
...
E003
Cols
Cal
...
Page 304
Eid
Name
City
...
E004
Kelly
LA
...
E002
Dun
Boston
...
Data pages
Table

唯一索引

- 唯一索引是指索引存储的值必须是唯一的
- 不允许两行具有相同的索引值，包括NULL值
- 主码索引是唯一索引

知识点71：建立和删除索引

建立索引

■ 建立与删除索引

通常索引的建立和删除由DBA或DB　owner负责.

❖建立索引

- ■ 格式

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

举例

- **举例:**

例6:

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

例7:

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

例8:

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

删除索引

❖ 删除索引

1.格式

DROP　　INDEX　　<索引名>;

2.举例

例7:删除Student表的Stusname索引.

DROP　　INDEX　　Stusname　on student

或　DROP　　INDEX　　student.Stusname

主键或唯一约束创建的索引不能使用DROP INDEX语句删除。为了删除这些索引，需要删除约束