

3

单行函数

中国科学院西安网络中心 编译 2005

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

进度表:	时间	主题
	55 分钟	讲演
	30 分钟	练习
	85 分钟	总共

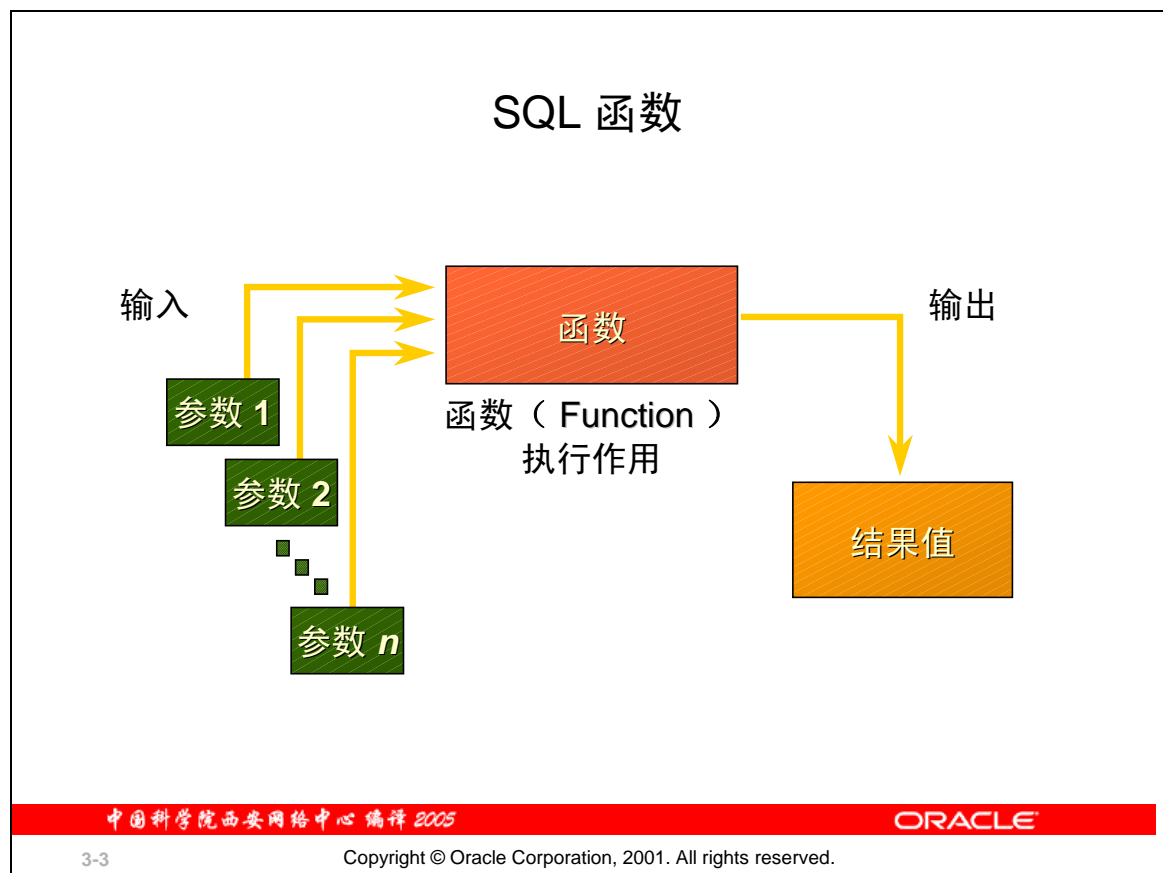
目标

完成本课后，您应当能够执行下列操作：

- 描述在 SQL 中可用的函数的变量类型
- 在 **SELECT** 语句中使用字符，数字和日期函数
- 描述转换函数的使用

课程目标

函数使得基本查询块更强大，函数用于操纵数据值。本课是探索函数的两节课程之一，内容集中在单行字符、数字和日期函数，以及转换数据类型的函数，例如，字符数据到数字数据的转换。



SQL 函数

函数是 SQL 的一个非常强有力的特性，函数能够用于下面的目的：

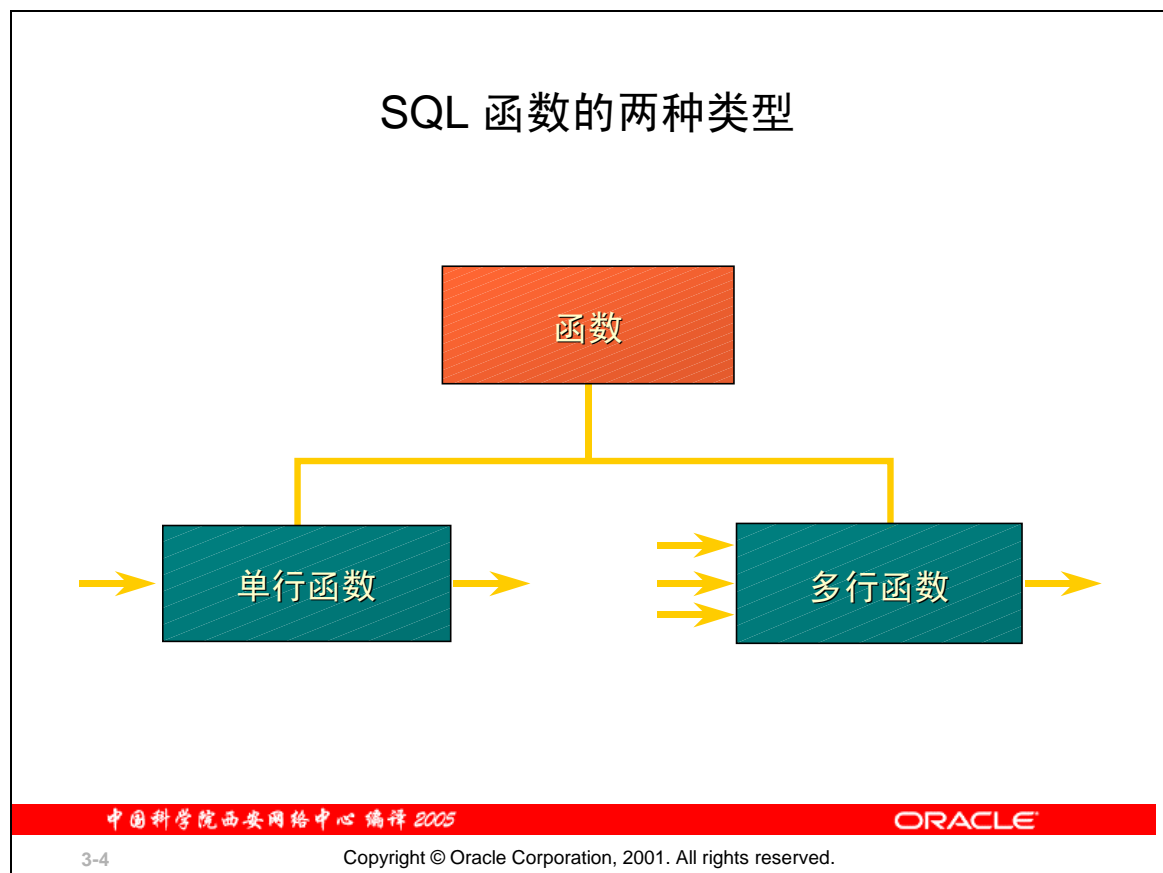
- 执行数据计算
- 修改单个数据项
- 操纵输出进行行分组
- 格式化显示的日期和数字
- 转换列数据类型

SQL 函数有输入参数，并且总有一个返回值。

注：在本课中讲述的大多数函数是针对 SQL 的 Oracle 版的。

教师注释

本课没有非常详细地讨论所有的函数，课程中用简要的说明介绍了最常用的函数。



SQL 函数 (续)

有两种截然不同的函数：

- 单行函数
- 多行函数

单行函数

这些函数仅对单个行进行运算，并且每行返回一个结果。有不同类型的单行函数，本课下面的函数类型：

- 字符
- 数字
- 日期
- 转换

多行函数

这些函数能够操纵成组的行，每个行组给出一个结果，这些函数也被称为组函数。多行函数在后面的课程中介绍。

为了得到更多的信息，请看 *Oracle9i SQL Reference* (参考)，以获得函数以及他们的语法的完整列表。

单行函数

单行函数：

- 操纵数据项
- 接受多个参数并返回一个值
- 作用于每一个返回行
- 每行返回一个结果
- 可以修改数据类型
- 可以嵌套
- 接受多个参数，参数可以是一个列或者一个表达式

```
function_name [(arg1, arg2,...)]
```

中国科学院西安网络中心 编译 2005

ORACLE

3-5

Copyright © Oracle Corporation, 2001. All rights reserved.

单行函数

单行函数用于操纵数据项，他们接受一个或多个参数，并且对查询的每个返回行返回一个值。一个参数可以是下列数据之一：

- 用户提供的常数
- 变量值
- 列名
- 表达式

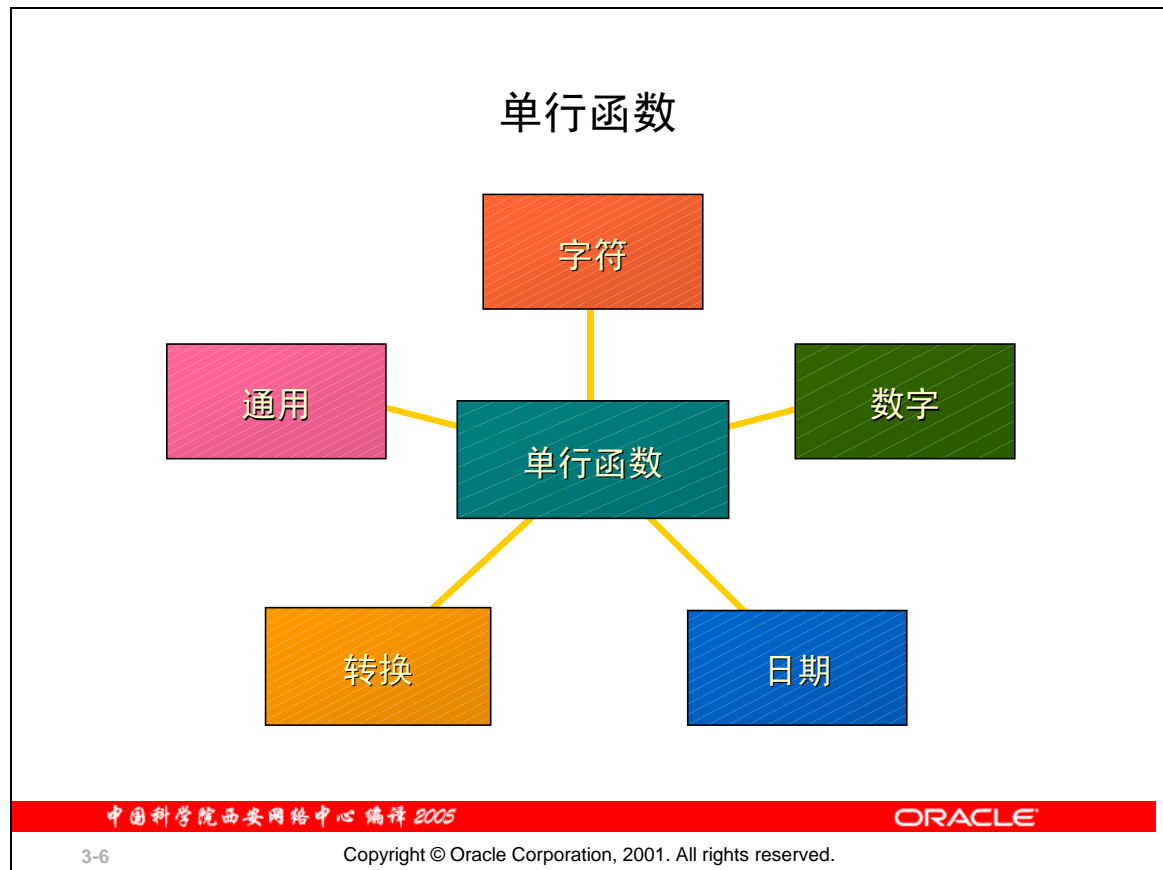
单行函数的特性包括：

- 作用于查询中返回的每一行
- 每行返回一个结果
- 可能返回一个与参数不同类型的数据值
- 可能需要一个或多个参数
- 能够用在 SELECT、WHERE 和 ORDER BY 子句中；可以嵌套

在语法中：

function_name 是函数的名字。

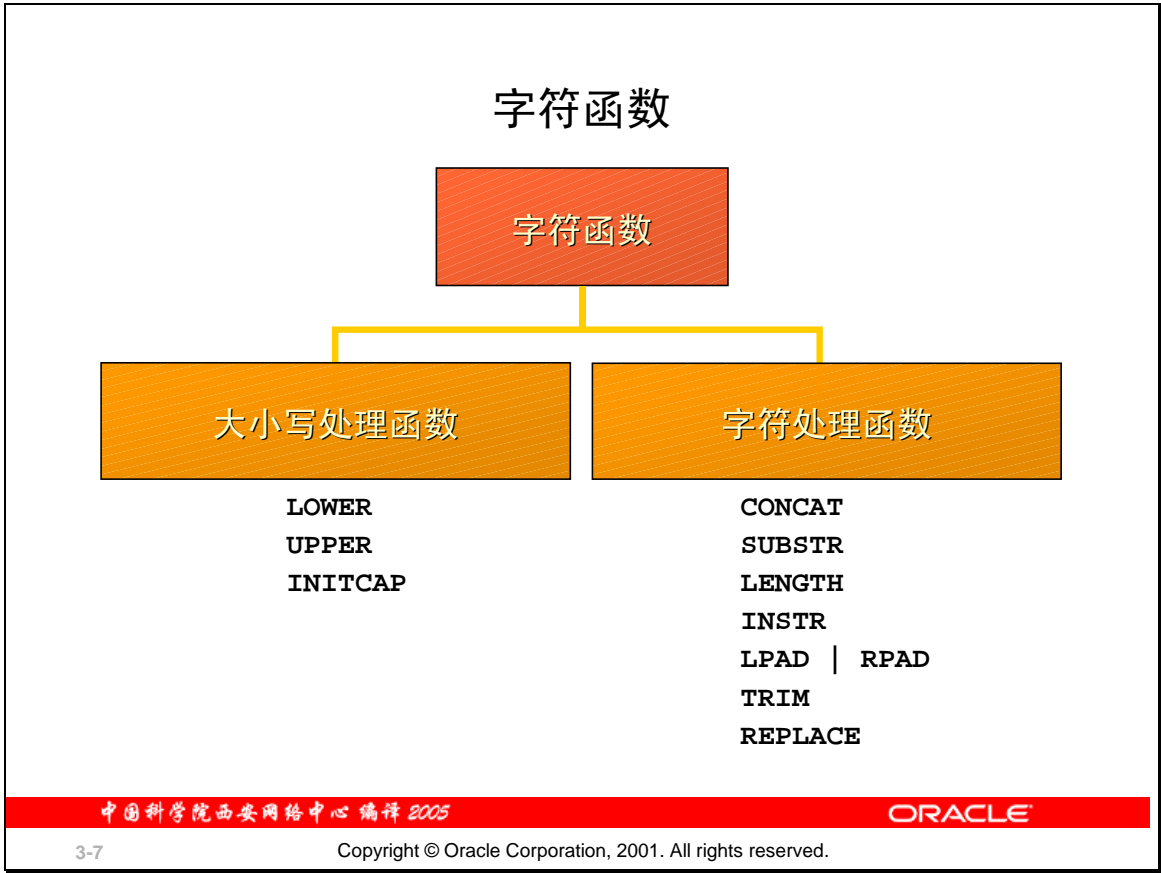
arg1, arg2 是由函数使用的任意参数，可以由一个列名或者一个表达式提供。



单行函数 (续)

本课包括下面的单行函数：

- 字符函数：接受字符输入，可以返回字符或者数字值
- 数字函数：接受数字输入，返回数字值
- 日期函数：对 DATE 数据类型的值进行运算（除了 MONTHS_BETWEEN 函数返回一个数字，所有日期函数都返回一个 DATE 数据类型的值。）
- 转换函数：从一个数据类型到另一个数据类型转换一个值
- 通用函数：
 - NVL
 - NVL2
 - NULLIF
 - COALSECE
 - CASE
 - DECODE



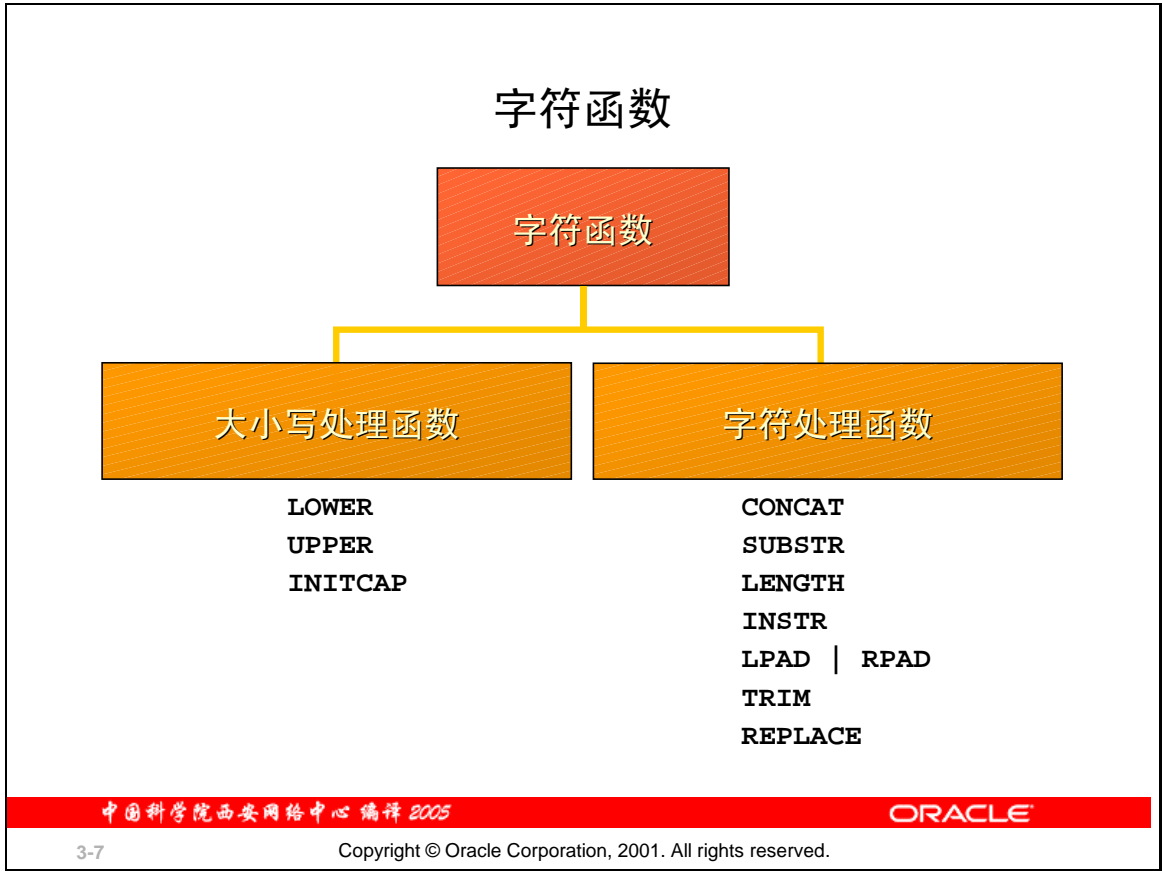
字符函数

单行字符函数接受字符数据作为输入，既可以返回字符值也可以返回数字值。字符函数可以被分为下面两种：

- 大小写处理函数
- 字符处理函数

LOWER(<i>column</i> / <i>expression</i>)	转换字符值为小写
UPPER(<i>column</i> / <i>expression</i>)	转换字符值为小写
INITCAP(<i>column</i> / <i>expression</i>)	转换每个单词的首字母值为大写，所有其它值为小写
CONCAT(<i>column1</i> / <i>expression1</i> , <i>column2</i> / <i>expression2</i>)	连接第一个字符值到第二个字符值；等价于连接运算符 ()
SUBSTR(<i>column</i> / <i>expression</i> , <i>m</i> [, <i>n</i>])	从字符返回值中回指定的字符，开始位置在 <i>m</i> , <i>n</i> 字符长度 (如果 <i>m</i> 是负数，计数从字符值末尾开始；如果 <i>n</i> 被忽略，返回到串结束的所有字符)。

注：在本课中只讨论这些函数中常用的几个。



字符函数 (续)

<code>LENGTH(column expression)</code>	返回表达式中的字符数
<code>INSTR(column expression, 'string', [,m], [n])</code>	返回一个命名串的数字位置。随意地，你可以提供一个位置 <i>m</i> 作为查找的开始，在字符串中第 <i>n</i> 次发现的位置。 <i>m</i> 和 <i>n</i> 的默认值是 1，意味着在起始开始查找，并且报告第一个发现的位置。
<code>LPAD(column expression, n, 'string')</code> <code>RPAD(column expression, n, 'string')</code>	填充字符值左、右调节到 <i>n</i> 字符位置的总宽度
<code>TRIM(leading/trailing/both, trim_character FROM trim_source)</code>	使你能够从一个字符串修整头或尾字符(或两者)。如果 <i>trim_character</i> 或 <i>trim_source</i> 是字符文字，你必须放在单引号中。
<code>REPLACE(text, search_string, replacement_string)</code>	从字符串查找一个文本表达式，如果找到，用指定的置换串代替它

大小写处理函数

这些函数转换字符串的大小写

函数	结果
<code>LOWER('SQL Course')</code>	<code>sql course</code>
<code>UPPER('SQL Course')</code>	<code>SQL COURSE</code>
<code>INITCAP('SQL Course')</code>	<code>Sql Course</code>

大小写处理函数

LOWER、UPPER 和 INITCAP 是三个大小写转换函数。

- LOWER: 转换大小写混合的字符串为小写字符串
- UPPER: 转换大小写混合的字符串为大写字符串
- INITCAP: 将每个单词的首字母转换为大写, 其他字母为小写

```
SELECT 'The job id for ' || UPPER(last_name) || ' is '
      || LOWER(job_id) AS "EMPLOYEE DETAILS"
FROM   employees;
```

EMPLOYEE DETAILS
The job id for KING is ad_pres
The job id for KOCHHAR is ad_vp
The job id for DE HAAN is ad_vp
The job id for HUNOLD is it_prog
The job id for ERNST is it_prog
The job id for HIGGINS is ac_mgr
The job id for GIETZ is ac_account

20 rows selected.

使用大小写处理函数

显示雇员 Higgins 的雇员号、姓名和部门号：

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

中国科学院西安网络中心 编译 2005

ORACLE

3-9

Copyright © Oracle Corporation, 2001. All rights reserved.

大小写处理函数 (续)

幻灯片中的例子显示雇员 Higgins 的雇员号、名字和部门号。

第一个 SQL 语句的 WHERE 子句指定雇员的名字是 higgins。因为在 EMPLOYEES 表中存储的所有数据是大小写混合的，名字 higgins 在表中找不到能够匹配的值，所以没有返回的行。

第二个 SQL 语句的 WHERE 子句指定在 EMPLOYEES 表中的雇员的名字与 higgins 相比较，为了比较，转换 LAST_NAME 列为小写。由于现在两个名字都是小写的，一个匹配值被发现，并且一行被返回。WHERE 子句可以写为下面的方式，而产生同样的结果：

```
...WHERE last_name = 'Higgins'
```

在输出中的名字看起来好象存储在数据库中。

为了以大写字母显示名字，在 SELECT 语句中使用 UPPER 函数。

```
SELECT employee_id, UPPER(last_name), department_id
FROM   employees
WHERE  INITCAP(last_name) = 'Higgins';
```

字符处理函数

字符串处理函数：

函数	结果
<code>CONCAT('Hello', 'World')</code>	<code>HelloWorld</code>
<code>SUBSTR('HelloWorld',1,5)</code>	<code>Hello</code>
<code>LENGTH('HelloWorld')</code>	<code>10</code>
<code>INSTR('HelloWorld', 'W')</code>	<code>6</code>
<code>LPAD(salary, 10, '*')</code>	<code>*****24000</code>
<code>RPAD(salary, 10, '*')</code>	<code>24000*****</code>
<code>TRIM('H' FROM 'HelloWorld')</code>	<code>elloWorld</code>

中国科学院西安网络中心 编译 2005

ORACLE

3-10

Copyright © Oracle Corporation, 2001. All rights reserved.

字符处理函数

CONCAT、SUBSTR、LENGTH、INSTR、LPAD、RPAD 和 TRIM 是本课中讲述的字符处理函数。

- **CONCAT**：连接值在一起 (CONCAT 函数有两个输入参数)
- **SUBSTR**：选取给定位置和长度的子字符串
- **LENGTH**：以数字值显示一个字符串的长度
- **INSTR**：找到一个给定字符的数字位置
- **LPAD**：用给定的字符左填充字符串到给定的长度
- **RPAD**：用给定的字符右填充字符串到给定的长度
- **TRIM**：从一个字符串中去除头或尾的字符 (或头和尾) (如果 *trim_character* 或 *trim_source* 是一个文字字符，必须放在单引号中。)

教师注释

明确地给学生指出 RPAD 函数，因为对该函数需要实际练习。TRIM 函数是 Oracle8i 以后新增加的函数，相当于 LTRIM 函数加上 RTRIM 函数的功能。

使用字符处理函数

1

2

3

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
```

EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
174	EllenAbel	SA_REP	4	0
176	JonathonTaylor	SA_REP	6	2
178	KimberelyGrant	SA_REP	5	3
202	PatFay	MK_REP	3	2

1

2

3

字符处理函数 (续)

幻灯片中的例子显示所有工作岗位名称从第 4 个字符位置开始，包含字符串 REP 的雇员的信息，将雇员的姓和名连接显示在一起，还显示雇员名的长度，以及名字中字母 a 的位置。

例子

修改幻灯片中的 SQL 语句，显示那些名字是以 n 结束的雇员的数据。

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
```

EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
102	LexDe Haan	7	5
200	JenniferWhalen	6	3
201	MichaelHartstein	9	2

数字函数

- **ROUND:** 四舍五入指定小数的值

ROUND(45.926, 2) → **45.93**

- **TRUNC:** 截断指定小数的值

TRUNC(45.926, 2) → **45.92**

- **MOD:** 返回除法的余数

MOD(1600, 300) → **100**

数字函数

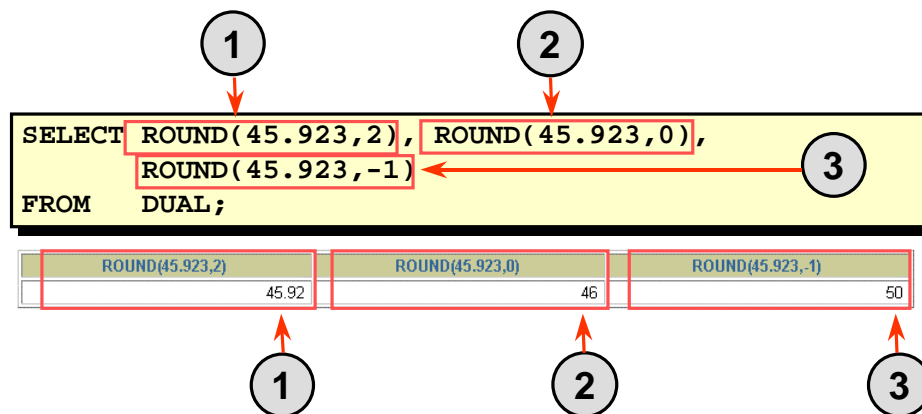
数字函数接受数字输入并且返回数字值，这一部分介绍一些数字函数。

函数	用途
ROUND(column expression, n)	四舍五入列、表达式或值为 <i>n</i> 位小数位，或者，如果 <i>n</i> 被忽略，无小数位。(如果 <i>n</i> 是负值，小数点左边左边的数被四舍五入)
TRUNC(column expression, n)	截断列、表达式或值到 <i>n</i> 位小数，或者，如果 <i>n</i> 被忽略，那么 <i>n</i> 默认为 0
MOD(m, n)	返回 <i>m</i> 除以 <i>n</i> 的余数

注：该列表仅包含常用的数字函数。

更多的信息，见 *Oracle9i SQL Reference*，“数字函数”

使用 ROUND 函数



DUAL 是一个虚拟表，你可以用它来查看函数和计算的结果

ROUND 函数

ROUND 函数四舍五入列、表达式或者 n 位小数的值。如果第二个参数是 0 或者缺少，值被四舍五入为整数。如果第二个参数是 2，值被四舍五入为两位小数。如果第二个参数是 -2，值被四舍五入到小数点左边两位。

ROUND 函数也能够被用于日期函数，在本课的后面会有例子。

DUAL 表

DUAL 表的所有者是用户 SYS，并且可以被所有的用户访问。它只包含一行，DUMMY，和带有值 X 的一行。当你想要返回一个值仅一次时，DUAL 表是有用的，例如，常数值、伪列或者不是来自用户数据表的表达式。DUAL 表通常用于 SELECT 子句语法的完整，因为不管是 SELECT 还是 FROM 子句都是强制的，并且一些计算不需要从实际的表中选择。

例：

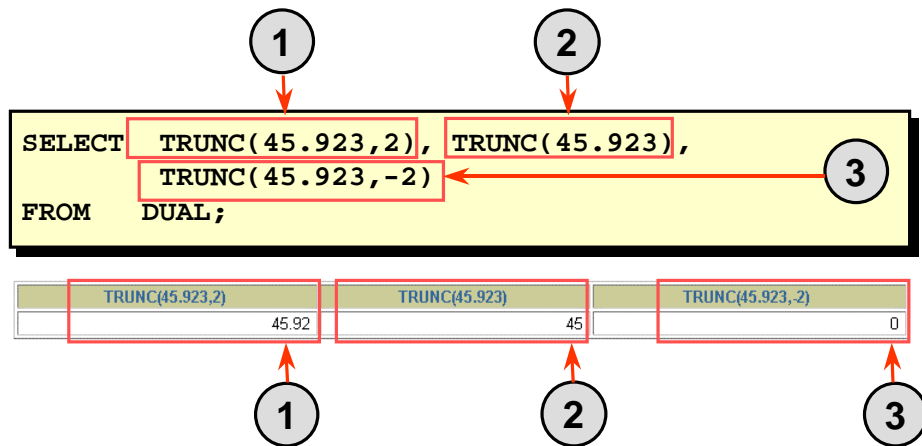
```
SELECT ROUND(45.923,2) FROM DUAL
```

SQL SERVER

在 SQL Server 中

```
SELECT ROUND(45.923,2)
```

使用 TRUNC 函数



TRUNC 函数

TRUNC 函数截断列、表达式或者 n 位小数值。

TRUNC 函数对参数起的作用类似于 ROUND 函数。如果第二个参数是 0 或者缺少，值被截断为整数。如果第二个参数是 2，值被截断为两位小数。如果第二个参数是 -2，值被截断到小数点左边两位。

象 ROUND 函数，TRUNC 函数也可以被用于日期函数。

使用 MOD 函数

计算所有是销售代表的雇员的工资被5000除后的余数

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

MOD 函数

MOD 函数找出值 1 除以值 2 的余数。幻灯片中的例子计算所有工作岗位是 SA_REP 的雇员的薪水除以 5,000 以后的余数。

注：MOD 函数经常用于确定一个值是奇数还是偶数。

教师注释 (3-17 页)

通过执行该命令，你能够改变一个用户会话的默认日期显示设置：

```
ALTER SESSION SET NLS_DATE_FORMAT = 'date format model';
```

DBA 能够为一个数据库设置不同于默认格式的日期格式。在任何一种情况下，改变这些设置通常不是开发人员的角色。

日期的使用

- Oracle 数据库用内部数字格式存储日期：世纪，年，月，日，小时，分钟和秒
- 默认日期显示格式是 DD-MON-RR.
 - 仅指定年的最后两位数字，允许你存储21世纪日期在20世纪中
 - 用同样的方式，允许你存储20世纪的日期在21世纪中

```
SELECT last_name, hire_date
FROM employees
WHERE last_name like 'G%';
```

LAST_NAME	HIRE_DATE
Gietz	07-JUN-94
Grant	24-MAY-99

中国科学院西安网络中心 编译 2005

ORACLE

3-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle 日期格式

Oracle 数据库以内部数字格式存储日期，表示世纪、年、月、日、小时、分和秒。对于任何日期默认显示和输入格式是 DD-MON-RR。有效的 Oracle 日期在公元前 4712 年 1 月 1 日和公元 9999 年 12 月 31 日之间。

在幻灯片的例子中，雇员 Gietz 的 HIRE_DATE 是以默认格式 DD-MON-RR 显示的，然而，存储在数据库的日期不是这种格式，所有的日期和时间的组成部分都会被存储。所以，尽管一个 HIRE_DATE，例如 07-JUN-94 被显示为天、月和年，也还有时间和世纪信息伴随它。完整的日期可能是 1994 年 7 月 7 日 5:10:43 p.m (June 7th, 1994 5:10:43 p.m)。

该日期在内部存储如下：

CENTURY	YEAR	MONTH	DAY	HOURL	MINUTE	SECOND
19	94	06	07	5	10	43

世纪和 2000 年

Oracle 服务器是 2000 年兼容的。当一个带有日期字段的记录被插入到表中时，世纪信息可从 SYSDATE 函数获得，可是，当日期字段显示到屏幕上时，默认情况下世纪部分不显示。

DATE 数据类型总是以 4 位内部数字存储年信息：两位数字代表世纪，两位数字代表年。例如，Oracle 数据库存储年为 1996 或 2001，而不是仅仅存 96 或 01。

日期的使用

SYSDATE 函数返回:

- Date
- Time

中国科学院西安网络中心 编译 2005

ORACLE

3-17

Copyright © Oracle Corporation, 2001. All rights reserved.

SYSDATE 函数

SYSDATE 是一个日期函数，它返回当前数据库服务器的日期和时间。可以象使用任何其它列名一样使用 **SYSDATE**，例如，你可以从一个表中选择 **SYSDATE** 来显示当前日期。习惯上我们是从一个被称为 **DUAL** 的虚拟表中选择 **SYSDATE**。

例子

用 **DUAL** 表显示当前日期。

```
SELECT SYSDATE
FROM DUAL;
```

SYSDATE
08-MAR-01

SQL SERVER

在 SQL Server 中

```
SELECT GETDATE()
```

用日期计算

- 从日期加或者减一个数，结果是一个日期值
- 两个日期相减，得到两个日期之间的天数
- 用小时数除以24，可以加小时到日期上

用日期计算

既然数据库以数字方式存储日期，你就可以用算术运算符进行计算，例如，加或减。你可以加或减数字常数以及日期。

你可以进行下面的运算：

运算	结果	说明
date + number	日期	加一个天数到一个日期上
date - number	日期	从一个日期上减一个天数
date - date	天数	用一个日期减另一个日期
date + number/24	日期	加一个小时数到一个日期上

用日期做算术运算

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS  
FROM employees  
WHERE department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

用日期计算 (续)

幻灯片中的例子显示所有在部门 90 中的雇员的名字和从业的周数。雇员的总工作时间以周计算，用当前日期 (SYSDATE) 减去雇员的受顾日期，再除以 7。

注：SYSDATE 是一个 SQL 函数，该函数返回当前日期和时间。你的结果可能与例子中的不同。

如果被减数大于当前日期，差是负数。

日期函数

函数	说明
MONTHS_BETWEEN	两个日期之间的月数
ADD_MONTHS	加日历月到日期
NEXT_DAY	下个星期几是几号
LAST_DAY	指定月的最后一天
ROUND	四舍五入日期
TRUNC	截断日期

中国科学院西安网络中心 编译 2005

ORACLE

3-20

Copyright © Oracle Corporation, 2001. All rights reserved.

日期函数

日期函数对 Oracle 日期进行操作，除了 MONTHS_BETWEEN 返回一个数字值，所有日期函数都返回一个 DATE 数据类型。

- MONTHS_BETWEEN(*date1*, *date2*): 计算 *date1* 和 *date2* 之间的月数，其结果可以是正的也可以是负的。如果 *date1* 大于 *date2*，结果是正的，反之，结果是负的。结果的小数部分表示月的一部分。
- ADD_MONTHS(*date*, *n*): 添加 *n* 个日历月到 *date*。*n* 的值必须是整数，但可以是负的。
- NEXT_DAY(*date*, '*char*'): 计算在 *date* 之后的下一个周('*char*')的指定天的日期。*char* 的值可能是一个表示一天的数或者是一个字符串。
- LAST_DAY(*date*): 计算包含 *date* 的月的最后一天的日期。
- ROUND(*date*[, '*fmt*']): 返回用格式化模式 *fmt* 四舍五入到指定单位的 *date*，如果格式模式 *fmt* 被忽略，*date* 被四舍五入到最近的天。
- TRUNC(*date*[, '*fmt*']): 返回用格式化模式 *fmt* 截断到指定单位的带天的时间部分的 *date*，如果格式模式 *fmt* 被忽略，*date* 被截断到最近的天。

该列表是一个可用日期函数的一个子集。格式化模式的内容含盖在本课后面。格式化模式的例子是月和天。

月的 3 字母缩写

JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC

使用日期函数

- **MONTHS_BETWEEN** ('01-SEP-95', '11-JAN-94')
→ 19.6774194
- **ADD_MONTHS** ('11-JAN-94', 6) → '11-JUL-94'
- **NEXT_DAY** ('01-SEP-95', 'FRIDAY') 下个星期五是几号
→ '08-SEP-95'
- **LAST_DAY**('01-FEB-95') → '28-FEB-95'

中国科学院西安网络中心 编译 2005

ORACLE

3-21

Copyright © Oracle Corporation, 2001. All rights reserved.

日期函数 (续)

例如, 显示所有受雇在 3 年 (36 个月) 以内的雇员的 employee_id, hire_date, 已被雇用的月, 6 个月的试用期, 受雇日期后的第一个星期五, 受雇月的最后一天。

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
FROM   employees
WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) < 36;
```

EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY(LAST_DAY(
107	07-FEB-99	25.0548529	07-AUG-99	12-FEB-99	28-FEB-99
124	16-NOV-99	15.7645303	16-MAY-00	19-NOV-99	30-NOV-99
143	15-MAR-98	35.7967884	15-SEP-98	20-MAR-98	31-MAR-98
144	09-JUL-98	31.9903368	09-JAN-99	10-JUL-98	31-JUL-98
149	29-JAN-00	13.3451755	29-JUL-00	04-FEB-00	31-JAN-00
176	24-MAR-98	35.5064658	24-SEP-98	27-MAR-98	31-MAR-98
178	24-MAY-99	21.5064658	24-NOV-99	28-MAY-99	31-MAY-99

7 rows selected.

周

SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY

使用日期函数

假定 `SYSDATE = '25-JUL-95':`

- `ROUND(SYSDATE, 'MONTH')` → 01-AUG-95
- `ROUND(SYSDATE, 'YEAR')` → 01-JAN-96
- `TRUNC(SYSDATE, 'MONTH')` → 01-JUL-95
- `TRUNC(SYSDATE, 'YEAR')` → 01-JAN-95
- `TRUNC(TO_DATE('25-JUL-95'), 'YEAR')` → 01-JAN-95

中国科学院西安网络中心 编译 2005

ORACLE

3-22

Copyright © Oracle Corporation, 2001. All rights reserved.

日期函数 (续)

`ROUND` 和 `TRUNC` 函数能够用于数字和日期值。在用于日期时，这些函数四舍五入或者以指定的格式化模板截断。因此，你可以四舍五入日期到最近的年或月。

例子

比较所有雇员的受雇日期，找出 1997 年开始工作的哪些人。用 `ROUND` 和 `TRUNC` 函数显示开始的月份。

```
SELECT employee_id, hire_date,
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
FROM   employees
WHERE  hire_date LIKE '%97';
```

教师注释

如果格式化模板是月，日期 1-15 结果在当前月的第一天，日期 16-31 结果在下月的第一天。如果格式化模板是年，1-6 月结果在当前年的 1 月 1 日，7-12 月结果在下年的 1 月 1 日。

EMPLOYEE_ID	HIRE_DATE	ROUND(HIR	TRUNC(HIR
142	29-JAN-97	01-FEB-97	01-JAN-97
202	17-AUG-97	01-SEP-97	01-AUG-97

练习 3, 第一部分: 概览

本部分练习包括下面的主题:

- 写一个查询, 显示当前日期
- 创建查询, 要求使用数字、字符和日期函数
- 执行对一个雇员已服务年和月的计算

练习 3, 第一部分: 概览

该练习为你设计了使用各种不同的字符、数字和日期数据类型函数的习题。
完成本课结尾的问题 1-5。

更多例子:

```
select TRUNC(TO_DATE('25-JUL-95'),'YEAR') from dual
```

第 1 行出现错误:

ORA-01843: 无效的月份

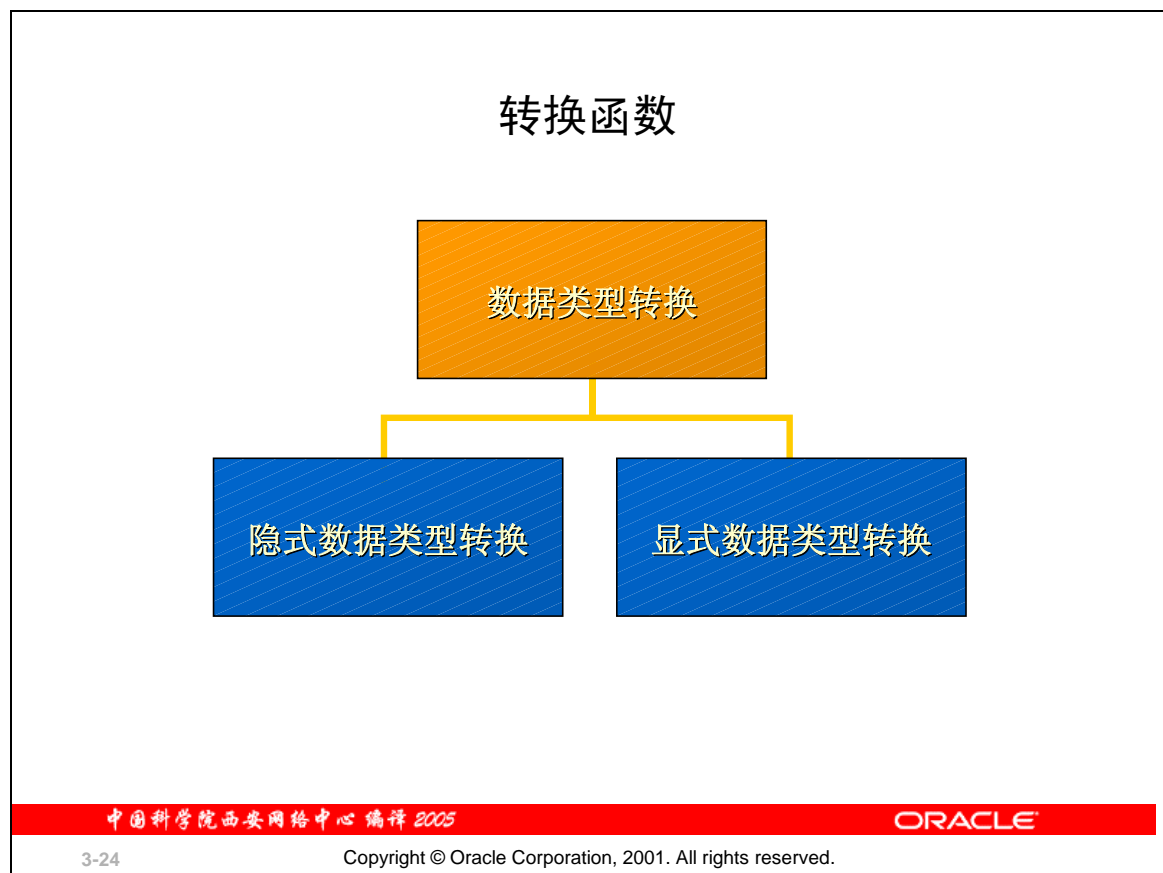
出现错误的原因是语言和区域设置 NLS_LANG 不匹配, 用下面的命令测试一下:

```
select to_char(SYSDATE) from dual
```

结果是: 18-4 月 -05

我们再试下面的语句:

```
select TRUNC(TO_DATE('25-7 月-95'),'YEAR') from dual
```

转换函数

除 Oracle 数据类型之外, Oracle9i/10g 数据库中表的列可以用 ANSI、DB2 和 SQL/DS 数据类型定义。不管用何种方法, Oracle 服务器内部将转换这些数据类型为 Oracle 数据类型。

在某些情况下, Oracle 服务器使用一种数据类型的数据, 而在另外一种情况下我们希望使用一种不同数据类型的数据, 如果这种情况发生, Oracle 服务器自动转换数据为期望的数据类型。这种数据类型的转换可以被 Oracle 服务器隐式进行, 或由用户显式进行。

隐式数据类型转换工作依照下面两张幻灯片解释的规则进行。
显式数据类型转换用转换函数进行。转换函数转换从一种数据类型转换值到另一种数据类型。通常, 函数名的构成遵循 *数据类型* 到 *数据类型* 的约定, 第一个数据类型是输入数据类型; 后一个数据类型是输出数据类型。

注: 尽管隐式数据类型转换是可用的, 但建议你做显式数据类型转换以确保 SQL 语句的可靠性。

隐式数据类型转换

对于直接赋值， Oracle 服务器能够自动地进行下面的转换：

从	到
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

中国科学院西安网络中心 编译 2005

ORACLE

3-25

Copyright © Oracle Corporation, 2001. All rights reserved.

隐式数据类型转换

如果 Oracle 服务器能够转换在任务中使用的值的数据类型到任务的目标，则任务成功。

教师注释

在 Oracle9i/10g 中有一些新的关于时间的数据类型可用，这包括：TIMESTAMP、TIMESTAMP WITH TIME ZONE、TIMESTAMP WITH LOCAL TIME ZONE、INTERVAL YEAR、INTERVAL DAY，这些函数在本课的后面讨论。

你也可以告诉学生参考 *Oracle9i SQL Reference*, “Basic Elements of Oracle SQL”。

隐式数据类型转换

对于表达式赋值，Oracle 服务器能自动地进行下面的转换：

从	到
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

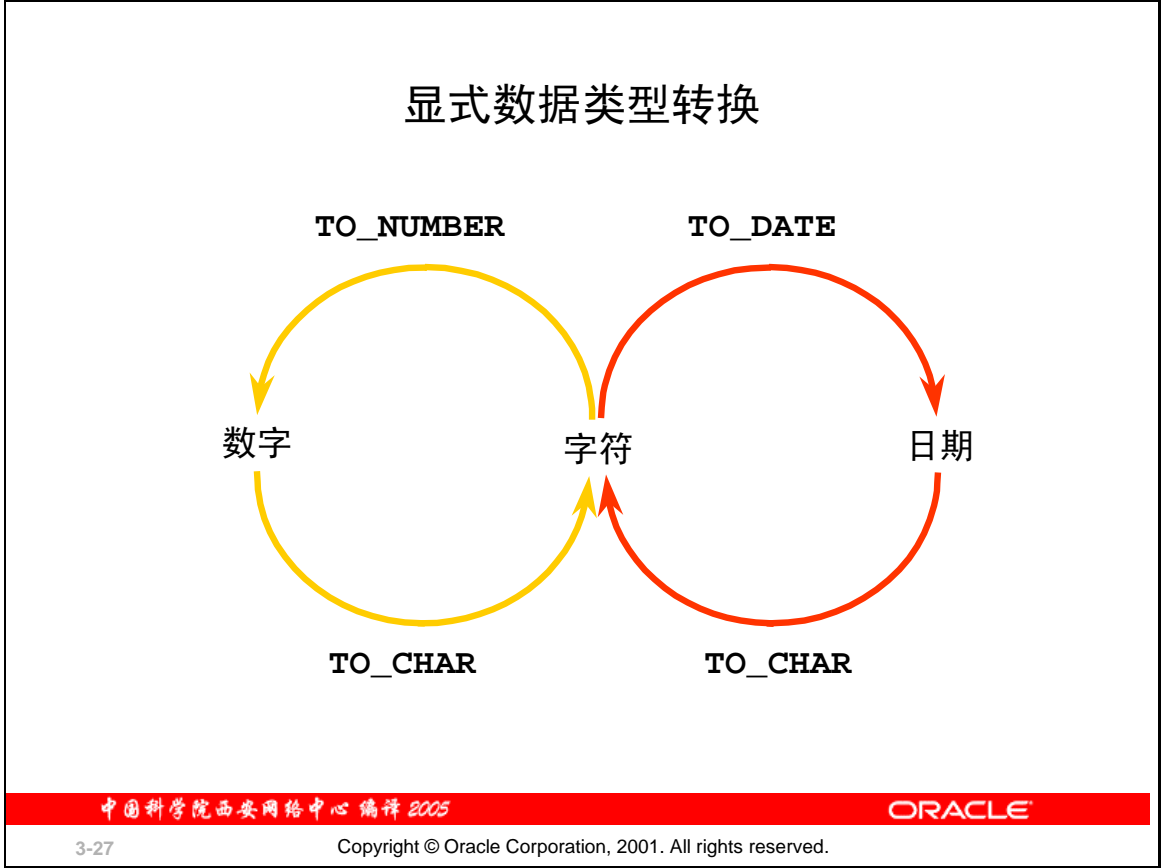
隐式数据类型转换 (续)

通常，当一种数据类型转换在某些地方被需要而又未不包括指定的转换规则时，Oracle 服务器用表达式规则。

注：只有当字符串表示一个有效的数时，CHAR 到 NUMBER 转换才能成功。

教师注释

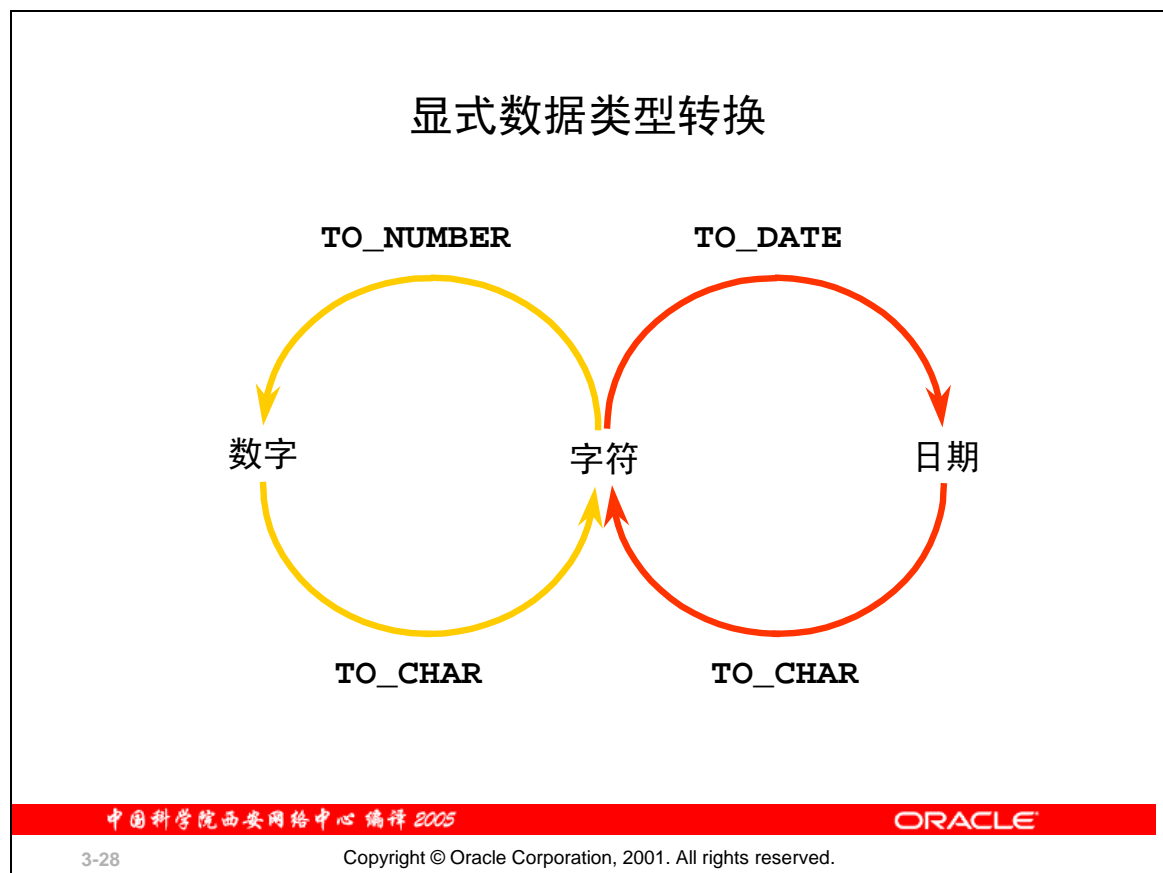
隐式数据转换不只是在前面提到的数据类型之间进行，还有其他一些隐式数据转换可以进行，例如，VARCHAR2 可以被隐式转换为 ROWID。



显式数据类型转换

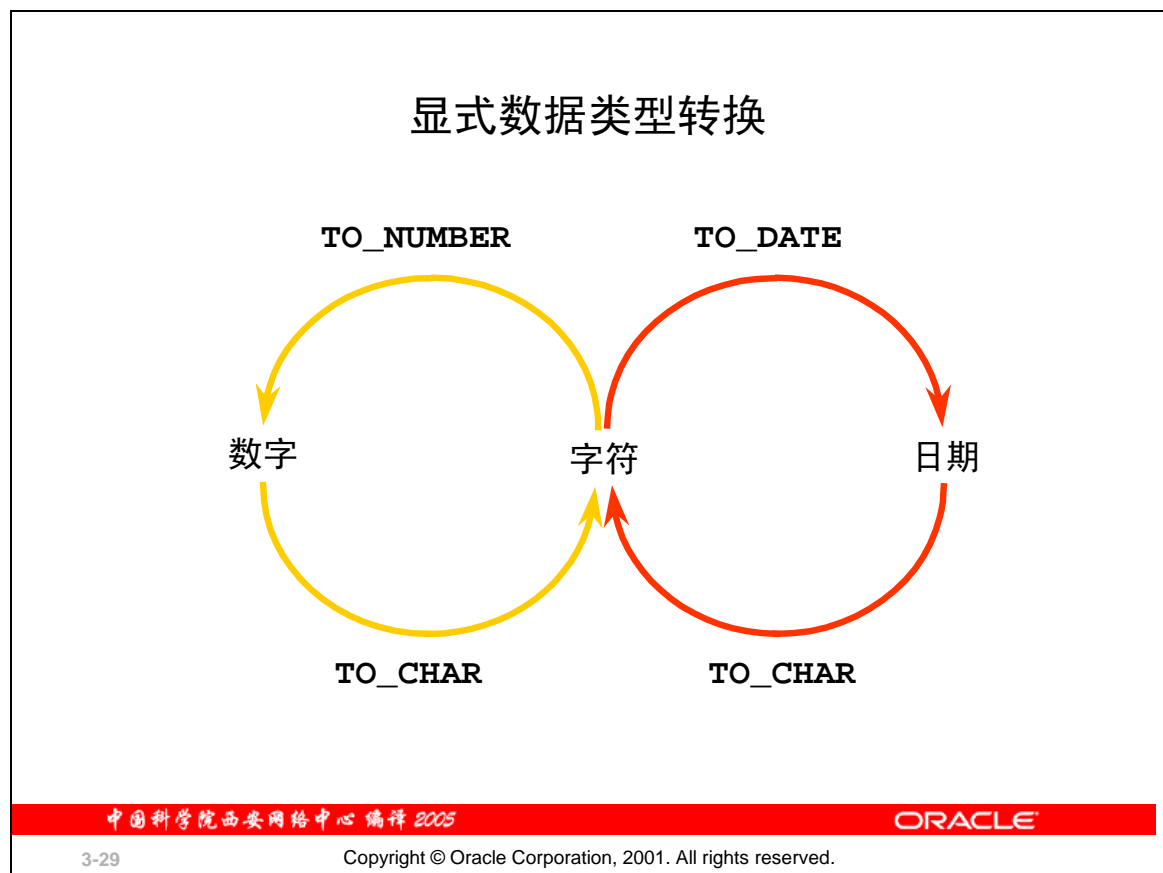
SQL 提供三种函数来从一种数据类型转换值到另一种：

函数	用途
<code>TO_CHAR(<i>number</i> <i>date</i>, [<i>fmt</i>], [<i>nlsparms</i>])</code>	转换一个数字或日期值为一个 VARCHAR2 字符串，带格式化样式 <i>fmt</i> 。 数字转换： <i>nlsparms</i> 参数指定下面的字符，它由数字格式化元素返回： <ul style="list-style-type: none">▪ 小数字符▪ 分组符▪ 本地货币符号▪ 国际货币符号 如果忽略 <i>nlsparms</i> 或其它参数，该函数在会话中使用默认参数值。
<code>TO_CHAR(<i>number</i> <i>date</i>, [<i>fmt</i>], [<i>nlsparms</i>])</code>	指定返回的月和日名字及其缩写的语言。如果忽略该参数，该函数在会话中使用默认日期语言
<code>TO_NUMBER(<i>char</i>, [<i>fmt</i>], [<i>nlsparms</i>])</code>	用由可选格式化样式 <i>fmt</i> 指定的格式转换包含数字的字符串为一个数字。 <i>Nlsparms</i> 参数在该函数中的目的与 <code>TO_CHAR</code> 函数用于数字转换的目的相同
<code>TO_DATE(<i>char</i>, [<i>fmt</i>], [<i>nlsparms</i>])</code>	按照 <i>fmt</i> 指定的格式转换表示日期的字符串为日期值。如果忽略 <i>fmt</i> ，格式是 DD-MON-YY。 <i>Nlsparms</i> 参数的目的与 <code>TO_CHAR</code> 函数用于日期转换时的目的相同。



显式数据类型转换数据（续）

注：该函数列表只提到了在本课中所用到的转换函数。
更多信息，见 *Oracle9i SQL Reference*, “Conversion Functions”。



显式数据类型转换数据（续）

注：在本课中提到的函数列表中只包括部分可用的转换函数。

更多信息，见 *Oracle9i SQL Reference*, “Conversion Functions”。

对日期使用 TO_CHAR 函数

```
TO_CHAR(date, 'format_model')
```

格式模板

- 必须加单引号，并且区分大小写
- 能够包含任一有效的日期格式元素
- 有一个 *fm* 元素用来删除填补的空，或者前导零
- 用一个逗号与日期值分开

中国科学院西安网络中心 编译 2005

ORACLE

3-30

Copyright © Oracle Corporation, 2001. All rights reserved.

以特殊格式显示日期

以前，所有 Oracle 日期值都以 DD-MON-YY 格式显示，你可以用 TO_CHAR 函数将日期从默认格式转换为指定的格式。

原则

- 格式模板必须放在单引号中，并且是大小写敏感的。
- 格式模板可以包括任意日期格式元素。必须用逗号将日期值和格式模板分隔开。
- 在输出中的天和月的名字被自动用空格填充。
- 为了去除已填充的空格或者消去前导零，使用填充方式 *fm* 元素。
- 你可以用后面课程将要介绍的 *iSQL*Plus* COLUMN 命令格式化结果字符域。

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM   employees
WHERE  last_name = 'Higgins';
```

EMPLOYEE_ID	MONTH
205	06/94

日期格式模板的元素

YYYY	数字全写年
YEAR	年的拼写
MM	月的两数字值
MONTH	月的全名
MON	月的三字母缩写
DY	周中天的三字母缩写
DAY	周中天的全名
DD	月的数字天

中国科学院西安网络中心 编译 2005

ORACLE

3-31

Copyright © Oracle Corporation, 2001. All rights reserved.

正确的日期格式化的格式化元素例子

元素	说明
SCC 或 CC	世纪, 带 - 服务器前缀 B.C. 日期
日期中的年 YYYY 或 SYYYY	年, 带 - 服务器前缀 B.C. 日期
YYY 或 YY 或 Y	年的最后 3、2 或 1 个数字
Y,YYY	年, 在这个位置带逗号
IYYY, IYY, IY, I	基于 ISO 标准的 4、3、2 或 1 位数字年
SYEAR 或 YEAR	拼写年; 带 - 服务器前缀 B.C. 日期
BC 或 AD	B.C.A.D.指示器
B.C.或 A.D.	带周期的 B.C./A.D.指示器
Q	四分之一年
MM	月: 两位数字值
MONTH	9 位字符长度的带空格填充的月的名字
MON	3 字母缩写的月的名字
RM	罗马数字月
WW 或 W	年或月的周
DDD 或 DD 或 D	年、月或周的天
DAY	9 位字符长度的带空格填充的天名字
DY	3 字母缩写的天的名字
J	儒略日, 从公元前 4713 年 12 月 31 日开始的天数

中国科学院西安网络中心 编译 2005	ORACLE
3-32	Copyright © Oracle Corporation, 2001. All rights reserved.

教师注释

由于学生在练习 10 中需要用强调格式 D,它格式化返回一个从 1 到 7 的值表示一周的某一天。与 NLS 日期设置选项有关,值 1 可能表示周日或周一,在美国值 1 表示周日。

例子:

```
select TO_CHAR(SYSDATE,'HH24:MI:SS AM') from dual
select TO_CHAR(TO_DATE('21-1 月-95'),'YYYY-MM-DD HH24:MI:SS AM')
from dual
```

```
select TO_CHAR(SYSDATE,'DD "of" MONTH') from dual
select TO_CHAR(TO_DATE('25-JUL-95'),'DD "of" MONTH') from dual
select TO_CHAR(TO_DATE('25-7 月-95'),'DD "of" MONTH') from dual
```

```
select to_char(sysdate, 'ddspth') from dual
select TRUNC(TO_DATE('25-JUL-95'),'ddspth') from dual -- NLS_LANG
是英文设置
select TRUNC(TO_DATE('25-7 月-95'),'ddspth') from dual -- NLS_LANG
是中文设置
```

日期格式模板元素

- 日期的时间部分，时间元素格式

HH24:MI:SS AM	15:45:32 PM
---------------	-------------

- 加字符串，将它们括在双引号中

DD "of" MONTH	12 of OCTOBER
---------------	---------------

- 数字前缀拼出数字

ddspth	fourteenth
--------	------------

例：select to_char(sysdate, 'ddspth') from dual

日期格式模板元素：时间格式化

使用在下表中列出的格式，显示时间信息和文字，并且改变数字为拼写的数字。

元素	说明
AM 或 PM	正午指示
A.M.或 P.M.	带句点的正午指示
HH 或 HH12 或 HH24	天的小时，或小时 (1-12)，或小时 (0-23)
MI	分钟 (0-59)
SS	秒 (0-59)
SSSSS	午夜之后的秒 (0-86399)

其它格式

元素	说明
/.,	在结果中使用标点符号
"of the"	在结果中使用引文串

指定后缀来影响数字显示

元素	说明
TH	序数 (例如，DDTH 显示为 4TH)
SP	拼写出数字 (例如，DDSP 显示为 FOUR)
SPTH or THSP	拼写出序数 (例如，DDSPTH 显示为 FOURTH)

TO_CHAR 函数用于日期转换

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
       AS HIREDATE  
FROM   employees;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999
...	

20 rows selected.

TO_CHAR 函数用于日期

在幻灯片中 SQL 语句显示所有雇员的名字和受雇日期，受雇日期以 17 June 1987 格式显示。

例

修改幻灯片的例子以格式 Seventh of June 1994 12:00:00 AM 显示日期。

```
SELECT last_name,  
       TO_CHAR(hire_date,  
               'fmDdspth "of" Month YYYY fmHH:MI:SS AM')  
       AS HIREDATE  
FROM   employees;
```

LAST_NAME	HIREDATE
King	Seventeenth of June 1987 12:00:00 AM
Kochhar	Twenty-First of September 1989 12:00:00 AM
De Haan	Thirteenth of January 1993 12:00:00 AM
Gietz	Seventh of June 1994 12:00:00 AM

20 rows selected.

注意，月以指定的格式模式显示：换句话说，第一个字母是大写其余字母是小写。

对数字使用 TO_CHAR 函数

`TO_CHAR(number, 'format_model')`

下面是一些你能够和 TO_CHAR 一起使用的格式化元素，用于显示字符形式的数字值：

9	表示一个数
0	强制显示为零
\$	放置一个浮动美元符号
L	使用浮动本地货币符号
.	打印一个小数点
,	打印一个千位指示

中国科学院西安网络中心 编译 2005

ORACLE

3-35

Copyright © Oracle Corporation, 2001. All rights reserved.

TO_CHAR 函数用于数字

在与数字值混用时，例如字符串，你应该用 TO_CHAR 函数转换那些数字为字符数据类型，该函数将 NUMBER 数据类型转换为 VARCHAR2 数据类型。该方法在用于串联时特别有用。

数字格式元素

如果你正在转换一个数字到字符数据类型，你可以用下面的格式元素：

元素	说明	举例	结果
9	数字位置 (9 的个数决定显示宽度)	999999	1234
0	显示前导 0	099999	001234
\$	浮动美圆符号	\$999999	\$1234
L	浮动本地货币符号	L999999	FF1234
.	小数点位置指定	999999.99	1234.00
,	逗号位置指定	999,999	1,234
MI	右边减号 (负值)	999999MI	1234-
PR	将负数加上括号	999999PR	<1234>
EEEE	科学计数法 (格式化必须指定四个 E)	99.999EEEE	1.234E+03
V	乘 10, n 次 (n = V 后面 9 的个数)	9999V99	123400
B	将 0 显示为空格	B9999.99	1234.00

对数字使用 TO_CHAR 函数

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM   employees  
WHERE  last_name = 'Ernst';
```

SALARY
\$6,000.00

原则

Oracle 服务器显示井号 (#) 串代替超过格式化样式提供的数字数目的整数数字。

Oracle 服务器四舍五入存储的小数值到格式化样式提供的小数数字空间。

教师注释 (3-39 页)

你可以示范使用文件 3_39n 中的 fx 修饰符的代码。先带 fx 修饰符运行该文件，然后除去 fx 修饰符再运行该文件。

使用 TO_NUMBER 和 TO_DATE 函数

- 转换字符串到数字，用 TO_NUMBER 函数格式化：

```
TO_NUMBER(char[, 'format_model'])
```

- 转换字符串到日期，用 TO_DATE 函数格式化：

```
TO_DATE(char[, 'format_model'])
```

- 这些函数有一个 **fx** 修饰符，该修饰符指示对字符参数和一个 TO_DATE 函数模板的数据格式的精确匹配

TO_NUMBER 和 TO_DATE 函数

你可能想要转换一个字符串为一个数字或一个日期。为了完成该任务，可以使用 TO_NUMBER 或 TO_DATE 函数。你选择的格式化样式是基于先前所示范的格式化元素的。

“fx” 修饰符指定精确地匹配一个 TO_DATE 函数的字符参数和日期格式化样式：

- 在字符参数中的标点符号和引用文本必须严格地匹配格式化样式的相应部分。
- 字符参数不能有额外的空白。如果没有 fx，Oracle 将忽略额外的空白。
- 在字符参数中的数字的数据必须有相同的数字数目作为在格式化样式中的相应元素。如果没有 fx，字符参数中的数字可以忽略前导 0。

使用 TO_NUMBER 和 TO_DATE 函数

- 转换字符串到数字，用 TO_NUMBER 函数格式化：

```
TO_NUMBER(char[, 'format_model'])
```

- 转换字符串到日期，用 TO_DATE 函数格式化：

```
TO_DATE(char[, 'format_model'])
```

- 这些函数有一个 **fx** 修饰符，该修饰符指示对字符参数和一个 TO_DATE 函数模板的数据格式的精确匹配

TO_NUMBER 和 TO_DATE 函数 (续)

你可能想要转换一个字符串为数字或日期，为了完成这个工作，你可以使用 TO_NUMBER 或 TO_DATE 函数。可以基于前面示范的格式化元素选择格式化样式。

Fx 修饰符指定 TO_DATE 函数对于字符参数和日期格式化样式的额外匹配：

- 在字符参数中，标点符号和引号中的文本必须完全匹配格式化样式中相应的部分。
- 字符参数不能有额外的空格。如果无fx，Oracle 服务器将忽略额外的空格。
- 在字符参数中数字数据必须与格式化样式中的元素有相应的数字个数。如果无fx，在字符参数中的可以忽略前导 0。

例

显示所有在 May 24, 1999 参加工作的雇员的名字和受雇日期。因为使用了 fx 修饰符，需要精确的匹配，并且在单词 ‘May’ 之后的空格不认可。

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');
```

RR 日期格式

当前年	指定的日期	RR 格式	YY 格式
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		如果指定的两位数字年是：	
		0-49	50-99
如果当前年是两位数字表示：	0-49	返回的日期是在当前世纪中	返回的日期是在当前世纪的上个世纪中
	50-99	返回的日期是在当前世纪的下个世纪中	返回的日期是在当前世纪中

中国科学院西安网络中心 编译 2005

ORACLE

3-39

Copyright © Oracle Corporation, 2001. All rights reserved.

RR 数据格式元素

RR 数据格式元素与 YY 元素相似，但你可以用它来指定不同的世纪。你可以用 RR 数据格式元素代替 YY，所以返回的世纪值是根据指定的两位数年和当前年的最后两位数字变化的。幻灯片中的表总结了 RR 元素的行为。

Current Year	Given Date	Interpreted (RR)	Interpreted (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017

RR 日期格式的例子

为了找出1990年以前受雇的雇员，使用 **RR** 格式，不管该命令运行在1999年还是现在，都会得到同样的结果：

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM   employees
WHERE  hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIR
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

RR 日期格式的例子

为了找到 1990 年以前工作的雇员，要使用 **RR** 格式表示年，因为现在的年大于 1999，**RR** 格式将日期中年的部分解释为从 1950 到 1999。

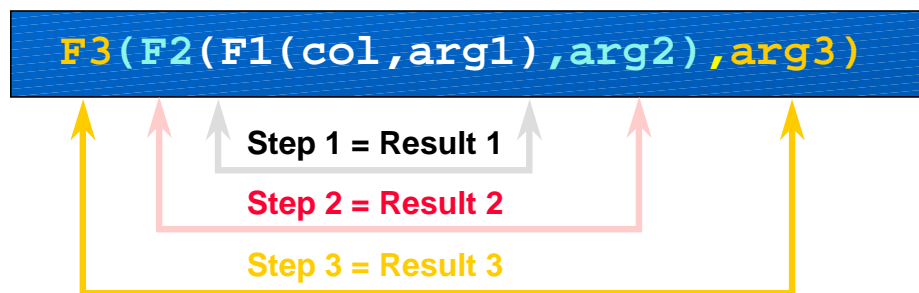
另一方面，下面的命令导致无选择行返回，因为 **YY** 格式将日中年的部分解释为当前世纪 (2090)。

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM   employees
WHERE  TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

no rows selected

嵌套函数

- 单行函数能够被嵌套任意层次
- 嵌套函数的计算是从最里层到最外层



嵌套函数

单行函数可以嵌套任意深度。嵌套的函数按从最里层到最外层的顺序求值。下面的一些例子显示展示了这些函数的灵活性。

嵌套函数

```
SELECT last_name,
       NVL(TO_CHAR(manager_id), 'No Manager')
FROM   employees
WHERE  manager_id IS NULL;
```

LAST_NAME	NVL(TO_CHAR(MANAGER_ID), 'NOMANAGER')
King	No Manager

嵌套函数 (续)

幻灯片中的例子显示公司的领导，他上面没有经理。SQL 语句的赋值包括两个步骤：

1. 求内函数的值，转换一个数为一个字符串。

```
Result1 = TO_CHAR(manager_id)
```

2. 求外函数的值，用文字串代替空值。

```
NVL(Result1, 'No Manager')
```

因为没有给列别名，整个表达式将作为列头。

例子

显示受雇日期 6 个月后的下一个星期五的日期。结果日期将应该是 Friday, August 13th, 1999。结果按受雇日期排序。

```
SELECT      TO_CHAR(NEXT_DAY(ADD_MONTHS
                           (hire_date, 6), 'FRIDAY'),
              'fmDay, Month DDth, YYYY')
            "Next 6 Month Review"
FROM        employees
ORDER BY    hire_date;
```

教师注释

演示：3_nest.sql

目的：举例说明几个单行函数的嵌套。

通用函数

这些函数可用于任意数据类型，并且适用于空值

- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

通用函数

这些函数可用于任何数据类型，并且适用于表达式列表中的空值。

函数	说明
NVL	转换空值为一个实际值
NVL2	如果 expr1 非空，NVL2 返回 expr2；如果 expr1 为空，NVL2 返回 expr3。参数 expr1 可以是任意数据类型。
NULLIF	比较两个表达式，如果相等返回空；如果不相等，返回第一个表达式
COALESCE	返回表达式列表中的第一个非空表达式

注：为了得到有关数百个可用函数的信息，请看 *Oracle9i SQL Reference*，“函数”

NVL 函数

转换一个空值到一个实际的值

- 可用的数据类型可以是日期、字符和数字
- 数据类型必须匹配：
 - `NVL(commission_pct,0)`
 - `NVL(hire_date,'01-JAN-97')`
 - `NVL(job_id,'No Job Yet')`

中国科学院西安网络中心 编译 2005

ORACLE

3-44

Copyright © Oracle Corporation, 2001. All rights reserved.

NVL 函数

用 NVL 函数转换一个空值为一个实际的值。

语法

`NVL (expr1, expr2)`

在语法中：

`expr1` 是包含空值的源值或者表达式

`expr2` 是用于转换空值的目的值

你可以使用 NVL 函数来转换任何数据类型，但返回值通常总是与 `expr1` 的数据类型相同。

NVL 用于转换各种数据类型

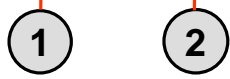
数据类型	转换例子
NUMBER	<code>NVL(number_column,9)</code>
DATE	<code>NVL(date_column, '01-JAN-95')</code>
CHAR or VARCHAR2	<code>NVL(character_column, 'Unavailable')</code>

使用 NVL 函数

```
SELECT last_name, salary, NVL(commission_pct, 0),  
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL  
FROM employees;
```

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000

...
20 rows selected.



NVL 函数

计算雇员的年报酬，你需要用 12 乘以月薪，再加上它的佣金（等于年薪乘以佣金百分比）。

```
SELECT last_name, salary, commission_pct,  
       (salary*12) + (salary*12*commission_pct) AN_SAL  
FROM employees;
```

LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
Vargas	2500		
Zlotkey	10500	.2	151200
Abel	11000	.3	171600
Taylor	8600	.2	123840

20 rows selected.

注意，只有对那些挣有佣金的雇员才计算其年佣金。如果在表达式中的任意列值为空，其结果为空。为了计算所有雇员的值，在进行算术运算之前你必须转换空值为一个数。在幻灯片的例子中，NEL 函数用来转换空值为零。

使用 NVL2 函数

```
SELECT last_name, salary, commission_pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

LAST_NAME	SALARY	COMMISSION_PCT	INCOME
Zlotkey	10500	.2	SAL+COMM
Abel	11000	.3	SAL+COMM
Taylor	8600	.2	SAL+COMM
Mourgos	5800		SAL
Rajs	3500		SAL
Davies	3100		SAL
Matos	2600		SAL
Vargas	2500		SAL

8 rows selected.

中国科学院西安网络中心 编译 2005

ORACLE

3-46

Copyright © Oracle Corporation, 2001. All rights reserved.

NVL2 函数

NVL2 函数检查第一个表达式，如果第一个表达式不为空，那么 NVL2 函数返回第二个表达式；如果第一个表达式为空，那么第三个表达式被返回。

语法

NVL(*expr1*, *expr2*, *expr3*)

在语法中：

expr1 是可能包含空的源值或表达式

expr2 *expr1* 非空时的返回值

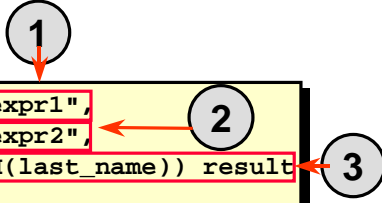
expr3 *expr1* 为空时的返回值

在例子中显示，COMMISSION_PCT 列被检查，如果值被发现，既不为空，第二个表达式 SAL+COMM 被返回；如果 COMMISSION_PCT 为空值，第三个参数 SAL 被返回。

参数 *expr1* 可以是任何数据类型，参数 *expr2* 和 *expr3* 可以是除 LONG 之外的任何数据类型。如果 *expr2* 和 *expr3* 的数据类型不同，Oracle 服务器在比较它们之前将转换 *expr3* 为 *expr2* 的数据类型，除非 *expr3* 是一个 null 常数，在这种情况下，不需要数据类型转换。

返回值的数据类型总是与 *expr2* 的数据类型相同，除非 *expr2* 是字符数据，在这种情况下，返回值的数据类型是 VARCHAR2。

使用 NULLIF 函数



```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name,  LENGTH(last_name)  "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM   employees;
```

FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
Steven	6	King	4	6
Neena	5	Kochhar	7	5
Lex	3	De Haan	7	3
Alexander	9	Hunold	6	9
Bruce	5	Ernst	5	5
Diana	5	Lorentz	7	5
Kevin	5	Mourgos	7	5
Trenna	6	Rajs	4	6
Curtis	6	Davies	6	

...

20 rows selected.

中国科学院西安网络中心 编译 2005

ORACLE

3-47

Copyright © Oracle Corporation, 2001. All rights reserved.

NULLIF 函数

NULLIF 函数比较两个表达式，如果相等，函数返回空，如果不相等，函数返回第一个表达式。第一个表达式不能为 NULL。

语法

```
NULLIF (expr1, expr2)
```

在语法中：

expr1 是对于 *expr2* 的被比较原值

expr2 是对于 *expr1* 的被比较原值。(如果它不等于 *expr1*, *expr1* 被返回)。

注：NULLIF 函数在逻辑上等同于下面的 CASE 表达式。CASE 表达式将在后面讨论：

```
CASE WHEN expr1 = expr 2 THEN NULL ELSE expr1 END
```


使用 COALESCE 函数

- COALESCE 函数超过 NVL 函数的优点是 COALESCE 函数能够接受多个交替的值
- 如果第一个表达式非空，它返回该表达式；否则，它做一个保留表达式的结合

使用 COALESCE 函数

COALESCE 函数返回列表中的第一个非空表达式。

语法

COALESCE (*expr1*, *expr2*, ... *exprn*)

在语法中：

expr1 如果它非空，返回该表达式

expr2 如果第一个表达式为空并且该表达式非空，返回该表达式

exprn 如果前面的表达式都为空，返回该表达式

使用 COALESCE 函数

```
SELECT last_name,  
       COALESCE(commission_pct, salary, 10) comm  
FROM   employees  
ORDER BY commission_pct;
```

LAST_NAME	COMM
Grant	.15
Zlotkey	.2
Taylor	.2
Abel	.3
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000

...
20 rows selected.

使用 COALESCE 函数

在例子中显示，如果 COMMISSION_PCT 值是非空，显示它。如果 COMMISSION_PCT 值是空，则显示 SALARY。如果 COMMISSION_PCT 和 SALARY 值都是空，那么显示 10。

条件表达式

- 在 SQL 语句中提供 IF-THEN-ELSE 逻辑的使用
- 两种用法：
 - **CASE** 表达式
 - **DECODE** 函数

条件表达式

有两种方法用于在 SQL 语句中实现条件处理 (IF-THEN-ELSE 逻辑)，即 CASE 表达式和 DECODE 函数。

注：CASE 表达式是 Oracle9i 服务器新发布的。CASE 表达式与 ANSI SQL 兼容；DECODE 是特殊的 Oracle 语法。

CASE 表达式

使得 IF-THEN-ELSE 条件判断容易实现

```
CASE expr WHEN comparison_expr1 THEN return_expr1
        [WHEN comparison_expr2 THEN return_expr2
        WHEN comparison_exprn THEN return_exprn
        ELSE else_expr]
END
```

CASE 表达式

CASE 表达式可以让你在 SQL 语句中使用 IF-THEN-ELSE 逻辑，而不必调用过程。

在简单的 CASE 表达式中，Oracle 查找每一个 WHEN ... THEN 对，如果 expr 等于 comparison_expr，返回相应的 return_expr。如果没有 WHEN ... THEN 对满足条件，并且 ELSE 子句存在，Oracle 返回 else_expr。否则，Oracle 返回 null。你不能对所有的 return_exprs 和 else_expr 指定文字 NULL。

所有的表达式 (expr、comparison_expr 和 return_expr) 必须是相同的数据类型，可以是 CHAR、VARCHAR2、NCHAR 或 NVARCHAR2。

教师注释

还有一种 CASE 查找表达式，Oracle 从左到右查找直到找到一个为真的条件，然后返回 return_expr。如果没有找到为真的条件，而且 ELSE 子句存在，Oracle 返回 else_expr。否则 Oracle 返回空。更多信息，见 *Oracle9i SQL Reference*, “Expressions”。

使用 CASE 表达式

使得 IF-THEN-ELSE 条件判断容易实现：

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                  WHEN 'ST_CLERK' THEN 1.15*salary  
                  WHEN 'SA_REP' THEN 1.20*salary  
                  ELSE salary END "REVISED_SALARY"  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

中国科学院西安网络中心 编译 2005

ORACLE

使用 CASE 表达式

在前面的 SQL 语句中，JOB_ID 的值被解码，如果 JOB_ID 是 IT_PROG，薪水增加 10%；如果 JOB_ID 是 ST_CLERK，薪水增加 15%；如果 JOB_ID 是 SA_REP，薪水增加 20%。对于所有其他的工作角色，不增加薪水。

可以用 DECODE 函数写相同功能的语句。

DECODE 函数

使得 **CASE** 或者 IF-THEN-ELSE 条件判断容易实现:

```
DECODE(col/expression, search1, result1  
      [, search2, result2,...,]  
      [, default])
```

DECODE 函数

DECODE 函数以一种类似于在多种语言中使用的 IF-THEN-ELSE 逻辑的方法解码一个表达式。DECODE 函数在比较表达式 (*expression*) 和每个查找 (*search*) 值后解码表达式, 如果表达式与查找相同, 返回结果。

如果省略默认值, 当没有查找值与表达式相匹配时返回一个空值。

使用 DECODE 函数

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG', 1.10*salary,
               'ST_CLERK', 1.15*salary,
               'SA_REP', 1.20*salary,
               salary)
       REVISED_SALARY
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

使用 DECODE 函数

在前面的 SQL 语句中，JOB_ID 的值被测试。如果 JOB_ID 是 IT_PROG，薪水增加 10%；如果 JOB_ID 是 ST_CLERK，薪水增加 15%；如果 JOB_ID 是 SA_REP，薪水增加 20%。对于所有其他的工作角色，不增加薪水。

同样的语句可以用伪代码作为一个 IF-THEN-ELSE 语句来表达：

```
IF job_id = 'IT_PROG'      THEN salary = salary*1.10
IF job_id = 'ST_CLERK'    THEN salary = salary*1.15
IF job_id = 'SA_REP'      THEN salary = salary*1.20
ELSE salary = salary
```

使用 DECODE 函数

对每一个在部门80的雇员显示可用的税率

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
               0, 0.00,
               1, 0.09,
               2, 0.20,
               3, 0.30,
               4, 0.40,
               5, 0.42,
               6, 0.44,
               0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

例子

这张幻灯片显示了使用 DECODE 函数的另一个例子。在这个例子中，我们为部门 80 中的每一个雇员决定基于月工资的税率。该税率按照下面的数据取值。

月工资范围	比率
\$0.00 - 1999.99	00%
\$2,000.00 - 3,999.99	09%
\$4,000.00 - 5,999.99	20%
\$6,000.00 - 7,999.99	30%
\$8,000.00 - 9,999.99	40%
\$10,000.00 - 11,999.99	42%
\$12,200.00 - 13,999.99	44%
\$14,000.00 or greater	45%

LAST_NAME	SALARY	TAX_RATE
Zlotkey	10500	.42
Abel	11000	.42
Taylor	8600	.4

小结

在本课中，您应该已经学会如何：

- 用函数执行对数据的计算
- 用函数修饰不同的数据项
- 用函数操纵行组的输出
- 用函数改变数据格式
- 用函数转换列数据类型
- 使用 NVL 函数
- 使用 IF-THEN-ELSE 逻辑

单行函数

单行函数可以被嵌套任意级别，单行函数可以进行下面的操作：

- 字符数据：LOWER、UPPER、INITCAP、CONCAT、SUBSTR、INSTR、LENGTH
- 数字数据：ROUND、TRUNC、MOD
- 日期数据：MONTHS_BETWEEN、ADD_MONTHS、NEXT_DAY、LAST_DAY、ROUND、TRUNC
- 日期值也可以用算术运算。
- 转换函数可以转换字符、日期和数字值：TO_CHAR、TO_DATE、TO_NUMBER
- 有几个函数处理空值，它们包括 NVL、NVL2、NULLIF 和 COALESCE。
- IF-THEN-ELSE 逻辑可以在 SQL 语句中用 CASE 表达式或 DECODE 函数来代替。

SYSDATE 和 DUAL

SYSDATE 是一个日期函数，它返回当前日期和时间。从一个称为 DUAL 的虚拟表中选择 SYSDATE 是惯例。

练习 3, 第二部分：概览

本章练习包括下面的主题：

- 使用数字、字符和日期函数创建查询
- 写大小写敏感的查询以测试字符函数的效果
- 演示一个雇员服务的年和月的计算
- 确定一个雇员的受雇日期

练习 3, 第二部分：概览

该练习中设计了使用字符、数字和日期数据类型的不同函数的各种习题。

不要忘记嵌套函数，其结果的计算是从最里层到最外层的函数。

教师注释

该练习被分为两部分。第一部分包括习题 1-5，含盖 1-23 页的内容。第二部分包括习题 6-14 和剩余课程的相应内容。

对于练习题 6：明确告诉学生，虽然来自一个提供者，但他们的结果可能是不同的，因为在习题中使用了 `SYSDATE`。

给练习 10 的教师提示：在答案中，在 `ORDER BY` 子句对 `TO_CHAR(hiredate-1, 'd')` 的分类中，格式化元素 'd' 返回一个 '1' 表示星期日，返回一个 '2' 表示星期一，依次类推。表达式 `hiredate-1` 有效地“替换”每一个受雇日期为前一天，所以一个雇员如果受雇于星期一，显示的受雇日期就是星期日。`TO_CHAR` 函数返回一个 '1' 用于雇员和被分类的结果集开始于星期一受雇的那些雇员。

练习 3 - 第一部分 (续)

1. 写一个查询显示当前日期，列标签显示为 Date。

Date
08-MAR-01

```
SELECT sysdate "Date"
FROM dual;
```

2. 对每一个雇员，显示 employee number、last_name、salary 和 salary 增加 15%，并且表示成整数，列标签显示为 New Salary。将你的 SQL 语句存到名为 lab3_2.sql 的文本文件中。

```
SELECT employee_id, last_name, salary,
ROUND(salary * 1.15, 0) "New Salary"
FROM employees;
```

3. 运行你的在文件 lab3_2.sql 中的查询。

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
100	King	24000	27600
101	Kochhar	17000	19550
102	De Haan	17000	19550
103	Hunold	9000	10350
104	Ernst	6000	6900
206	Gietz	8300	9545

20 rows selected.

```
SELECT employee_id, last_name, salary,
ROUND(salary * 1.15, 0) "New Salary"
FROM employees;
```

4. 修改查询 lab3_2.sql 添加一个列，该列从新薪水 New Salary 列中减去旧薪水，列标签为 Increase。保存内容到文件 lab3_4.sql 中。运行修订的查询。

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
100	King	24000	27600	3600
101	Kochhar	17000	19550	2550
102	De Haan	17000	19550	2550
103	Hunold	9000	10350	1350
104	Ernst	6000	6900	900
107	Lorentz	4200	4830	630
124	Mourgos	5800	6670	870
141	Rajs	3500	4025	525
142	Davies	3100	3565	465
143	Matos	2600	2990	390
144	Vargas	2500	2875	375
149	Zlotkey	10500	12075	1575
206	Gietz	8300	9545	1245

20 rows selected.

```
SELECT employee_id, last_name, salary,
ROUND(salary * 1.15, 0) "New Salary",
ROUND(salary * 1.15, 0) - salary "Increase"
FROM employees;
```

- 写一个查询用首字母大写，其它字母小写显示雇员的 last names，显示名字的长度，对所有名字开始字母是 J、A 或 M 的雇员，给每列一个适当的标签。用雇员的 last names 排序结果。

Name	Length
Abel	4
Matos	5
Mourgos	7

```
SELECT INITCAP(last_name) "Name",
LENGTH(last_name) "Length"
FROM employees
WHERE last_name LIKE 'J%'
OR last_name LIKE 'M%'
OR last_name LIKE 'A%'
ORDER BY last_name;
```

练习 3 - 第二部分

- 对每一个雇员，显示其的 last name，并且计算从雇员受雇日期到今天的月数，列标签 MONTHS_WORKED。按受雇月数排序结果，四舍五入月数到最靠近的整数月。
注：你的结果将不同。

LAST_NAME	MONTHS_WORKED
Zlotkey	13
Mourgos	16
Grant	22
Lorentz	25
Vargas	32
Taylor	36
Matos	36
Fay	43
Davies	49
Abel	58
Hartstein	61
Rajs	65
Higgins	81
Gietz	81
LAST_NAME	MONTHS_WORKED
De Haan	98
Ernst	118
Hunold	134
Kochhar	138
Whalen	162
King	165

20 rows selected.

```

SELECT last_name, ROUND(MONTHS_BETWEEN
(SYSDATE, hire_date)) MONTHS_WORKED
FROM employees
ORDER BY MONTHS_BETWEEN(SYSDATE, hire_date);

```

7. 写一个查询对每个雇员做计算:

<雇员的 last name> earns <salary> monthly but wants <3 倍 salary>。列标签 Dream Salaries。

Dream Salaries
King earns \$24,000.00 monthly but wants \$72,000.00.
Kochhar earns \$17,000.00 monthly but wants \$51,000.00.
De Haan earns \$17,000.00 monthly but wants \$51,000.00.
Hunold earns \$9,000.00 monthly but wants \$27,000.00.
Ernst earns \$6,000.00 monthly but wants \$18,000.00.
Lorentz earns \$4,200.00 monthly but wants \$12,600.00.
Mourgos earns \$5,800.00 monthly but wants \$17,400.00.
Rajs earns \$3,500.00 monthly but wants \$10,500.00.
Davies earns \$3,100.00 monthly but wants \$9,300.00.

Gietz earns \$8,300.00 monthly but wants \$24,900.00.

20 rows selected.

```
SELECT last_name || ' earns '  
|| TO_CHAR(salary, 'fm$99,999.00')  
|| ' monthly but wants '  
|| TO_CHAR(salary * 3, 'fm$99,999.00')  
|| '.' "Dream Salaries"  
FROM employees;
```

如果有时间，完成下面的练习：

8. 创建一个查询显示所有雇员的 last name 和 salary。格式化为 15 个字符长度，用 \$ 左填充，列标签 SALARY。

LAST_NAME	SALARY
King	\$\$\$\$\$\$\$\$\$24000
Kochhar	\$\$\$\$\$\$\$\$\$17000
De Haan	\$\$\$\$\$\$\$\$\$17000
Hunold	\$\$\$\$\$\$\$\$\$9000
Ernst	\$\$\$\$\$\$\$\$\$6000
Lorentz	\$\$\$\$\$\$\$\$\$4200
Mourgos	\$\$\$\$\$\$\$\$\$5800
Rajs	\$\$\$\$\$\$\$\$\$3500
Davies	\$\$\$\$\$\$\$\$\$3100
Matos	\$\$\$\$\$\$\$\$\$2600
Vargas	\$\$\$\$\$\$\$\$\$2500
Zlotkey	\$\$\$\$\$\$\$\$\$10500
Abel	\$\$\$\$\$\$\$\$\$11000
Taylor	\$\$\$\$\$\$\$\$\$8600
Gietz	\$\$\$\$\$\$\$\$\$8300

20 rows selected.

```
SELECT last_name,  
LPAD(salary, 15, '$') SALARY  
FROM employees;
```

9. 显示每一个雇员的 last name、hire date 和 salary 检查日期，该日期是服务六个月后的第一个星期一，列标签 REVIEW。格式化日期显示看起来象 “Monday, the Thirty-First of July, 2000” 的样子。

LAST_NAME	HIRE_DATE	REVIEW
King	17-JUN-87	Monday, the Twenty-First of December, 1987
Kochhar	21-SEP-89	Monday, the Twenty-Sixth of March, 1990
De Haan	13-JAN-93	Monday, the Nineteenth of July, 1993
Hunold	03-JAN-90	Monday, the Ninth of July, 1990
Ernst	21-MAY-91	Monday, the Twenty-Fifth of November, 1991
Lorentz	07-FEB-99	Monday, the Ninth of August, 1999
Mourgos	16-NOV-99	Monday, the Twenty-Second of May, 2000
Gietz	07-JUN-94	Monday, the Twelfth of December, 1994

20 rows selected.

```
SELECT last_name, hire_date,
TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),
'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW
FROM employees;
```

10. 显示 last name、hire date 和 雇员开始工作的周日，列标签 DAY，用星期一作为周的起始日排序结果。

LAST_NAME	HIRE_DATE	DAY
Grant	24-MAY-99	MONDAY
Ernst	21-MAY-91	TUESDAY
Mourgos	16-NOV-99	TUESDAY
Taylor	24-MAR-98	TUESDAY
Rajs	17-OCT-95	TUESDAY
Gietz	07-JUN-94	TUESDAY
Higgins	07-JUN-94	TUESDAY
King	17-JUN-87	WEDNESDAY
De Haan	13-JAN-93	WEDNESDAY
Davies	29-JAN-97	WEDNESDAY
Hunold	03-JAN-90	WEDNESDAY
Kochhar	21-SEP-89	THURSDAY
Whalen	17-SEP-87	THURSDAY
Vargas	09-JUL-98	THURSDAY
Matos	15-MAR-98	SUNDAY

20 rows selected.

```
SELECT last_name, hire_date,
TO_CHAR(hire_date, 'DAY') DAY
FROM employees
ORDER BY TO_CHAR(hire_date - 1, 'd');
```

如果你想要接受更大的挑战，完成下面的练习：

11. 创建一个查询显示雇员的 last names 和 commission (佣金) 比率。如果雇员没有佣

金，显示 “No Commission”，列标签 COMM。

LAST_NAME	COMM
King	No Commission
Kochhar	No Commission
De Haan	No Commission
Hunold	No Commission
Ernst	No Commission
Lorentz	No Commission
Mourgos	No Commission
Rajs	No Commission
Davies	No Commission
Matos	No Commission
Vargas	No Commission
Zlotkey	.2
Abel	.3
Taylor	.2
Gietz	No Commission

20 rows selected.

```
SELECT last_name,
       NVL(TO_CHAR(commission_pct), 'No Commission') COMM
FROM employees;
```

12. 创建一个查询显示雇员的 last names 并带星号显示他们的年薪，每个星号表示 1000 美圆。按薪水降序排序数据。列标签为 EMPLOYEES_AND_THEIR_SALARIES。

EMPLOYEE_AND_THEIR_SALARIES
King *****
Kochhar *****
De Haan *****
Hartstei *****
Higgins *****
Abel *****
Zlotkey *****
Hunold *****
Taylor *****
Gietz *****
Grant *****
Vargas **

20 rows selected.

```
SELECT rpad(last_name, 8) || ' ' || rpad(' ', (salary*12)/1000+1, '*')
EMPLOYEES_AND_THEIR_SALARIES
FROM employees
```


ORDER BY salary DESC;

13. 用 DECODE 函数，写一个查询，按照下面的数据显示所有雇员的基于 JOB_ID 列值的级别。

工作	级别
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
不在上面的	0

JOB_ID	G
AD_PRES	A
AD_VP	D
AD_VP	D
IT_PROG	C
IT_PROG	C
IT_PROG	C
ST_MAN	B
ST_CLERK	E
AC_ACCOUNT	0

20 rows selected.

```

SELECT job_id, decode (job_id,
                        'ST_CLERK', 'E',
                        'SA_REP', 'D',
                        'IT_PROG', 'C',
                        'ST_MAN', 'B',
                        'AD_PRES', 'A',
                        '0') GRADE
FROM employees;
```

14. 用 CASE 语法，以前面的问题写语句。

```

SELECT    job_id, CASE job_id
            WHEN 'ST_CLERK' THEN 'E'
            WHEN 'SA_REP'   THEN 'D'
            WHEN 'IT_PROG'  THEN 'C'
            WHEN 'ST_MAN'   THEN 'B'
            WHEN 'AD_PRES'  THEN 'A'
            ELSE '0' END    GRADE
FROM employees;
```

