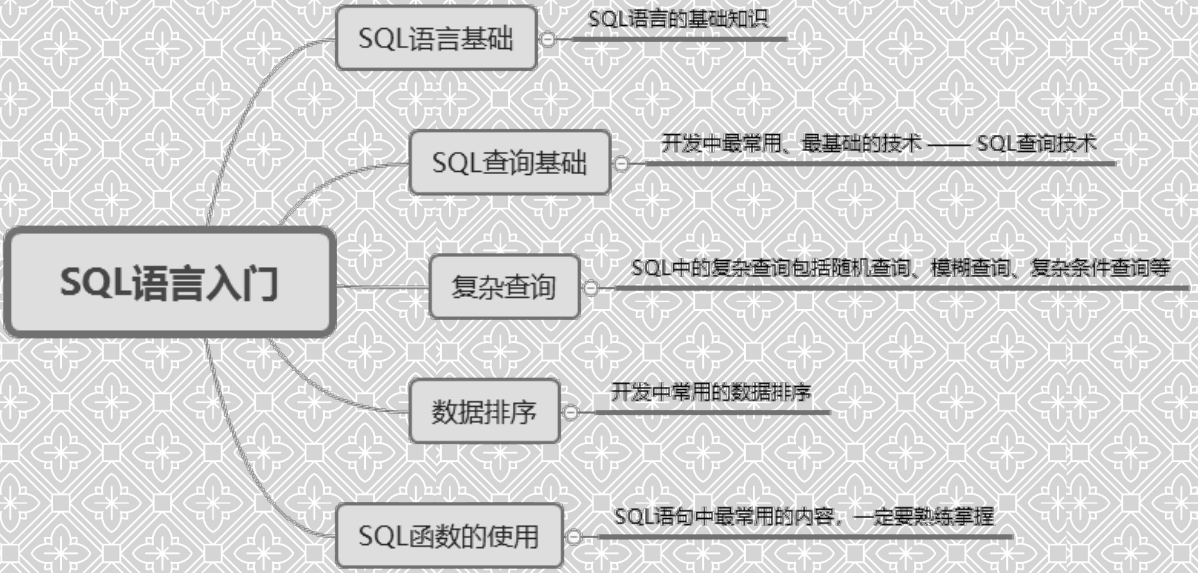
第 1 篇

SQL 语言入门

本篇介绍了 SQL 语言基础、SQL 查询基础、复杂查询、数据排序、SQL 函数的使用等 SQL 基础知识，并结合大量的图示、举例、视频等，帮助读者快速掌握 SQL 语言基础，为进一步学习奠定坚实的基础。

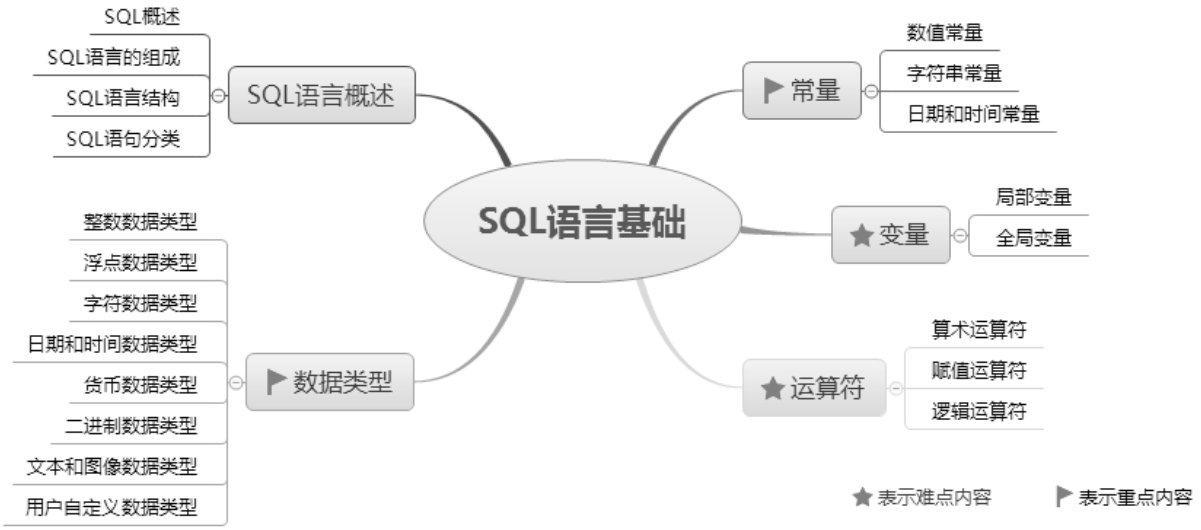


第 1 章

SQL 语言基础

本章主要介绍 SQL 语言的相关概念，包括 SQL 语言简介、常见的数据类型、常量、变量和运算符。通过本章的学习，读者应该掌握 SQL 语言的基础概念，以及常见的关系数据库。

本章知识架构及重难点如下：



1.1 SQL 语言概述

1.1.1 SQL 概述

SQL（Structured Query Language 结构化查询语言）语言在 1974 年由 Boyce 和 Chamberlin 提出。1975—1979 年，IBM 公司 San Jose Research Laboratory 研制的关系数据库管理系统（原形系统 System R）实现了这种语言。

SQL 是一种组织、管理和检索计算机数据库存储数据的工具。SQL 是一种计算机语言，可以用来与数据库交互。事实上，SQL 使用的是一种特殊类型的数据库，即关系数据库。

SQL 本身不是一个数据库管理系统，也不是一个独立的产品，而是数据库管理系统不可缺少的组成部分，它是与 DBMS 通信的一种语言和工具。由于其功能丰富，语言简洁，使用方法灵活，因此倍

受用户和计算机业界的青睐，被众多计算机公司和软件公司采用。经过多年的发展，SQL 语言已成为关系数据库的标准语言。

1.1.2　SQL 语言的组成

SQL 语言是具有强大查询功能的数据库语言。除此以外，SQL 还可以控制 DBMS 为其用户提供的所有功能，介绍如下。

- ☑ 数据定义语言（DDL，Data Definition Language）：SQL 允许用户定义存储数据的结构和组织，以及存储数据项之间的关系。
- ☑ 数据检索语言：SQL 允许用户或应用程序从数据库中检索存储的数据并使用它。
- ☑ 数据操纵语言（DML，Data Manipulation Language）：SQL 允许用户或应用程序通过添加新数据、删除旧数据和修改以前存储的数据对数据库进行更新。
- ☑ 数据控制语言（DCL，Data Control Language）：可以使用 SQL 来限制用户检索、添加和修改数据的能力，保护存储的数据不被未授权的用户所访问。
- ☑ 数据共享：可以使用 SQL 来协调多个并发用户共享数据，确保用户不会相互干扰。
- ☑ 数据完整性：SQL 在数据库中定义完整性约束条件，使它不会因不一致的更新或系统失败而遭到破坏。

因此，SQL 是一种综合性语言，用来控制数据库并与数据库管理系统进行交互。SQL 是数据库子语言，包含大约 40 条专用于数据库管理任务的语句。

数据操作类 SQL 语句如表 1.1 所示。

表 1.1　数据操作类 SQL 语句

运　算　符	行　　为
SELECT	从数据库表中检索数据行和列
INSERT	把新的数据记录添加到数据库中
DELETE	从数据库中删除数据记录
UPDATE	修改现有的数据库中的数据

数据定义类 SQL 语句如表 1.2 所示。

表 1.2　数据定义类 SQL 语句

运　算　符	行　　为
CREATE TABLE	在一个数据库中创建一个数据库表
DROP TABLE	从数据库中删除一个表
ALTER TABLE	修改一个现存表的结构
CREATE VIEW	把一个新的视图添加到数据库中
DROP VIEW	从数据库中删除视图
CREATE INDEX	为数据库表中的一个字段创建索引
DROP INDEX	从数据库表的一个字段中删除索引
CREATE PROCEDURE	在一个数据库中创建一个存储过程

续表

运 算 符	行 为
DROP PROCEDURE	从数据库中删除存储过程
CREATE TRIGGER	创建一个触发器
DROP TRIGGER	从数据库中删除触发器
CREATE SCHEMA	向数据库添加一个新模式
DROP SCHEMA	从数据库中删除一个模式
CREATE DOMAIN	创建一个数据值域
ALTER DOMAIN	改变域定义
DROP DOMAIN	从数据库中删除一个域

数据控制类 SQL 语句如表 1.3 所示。

表 1.3 数据控制类 SQL 语句

运 算 符	行 为
GRANT	授予用户访问权限
DENY	拒绝用户访问
REVOKE	删除用户访问权限

事务控制类 SQL 语句如表 1.4 所示。

表 1.4 事务控制类 SQL 语句

运 算 符	行 为
COMMIT	结束当前事务
ROLLBACK	终止当前事务
SET TRANSACTION	定义当前事务数据访问特征

程序化 SQL 语句如表 1.5 所示。

表 1.5 程序化 SQL 语句

运 算 符	行 为
DECLARE	定义查询游标
EXPLAN	描述查询数据访问计划
OPEN	检索查询结果打开一个游标
FETCH	检索一条查询结果记录
CLOSE	关闭游标
PREPARE	为动态执行准备 SQL 语句
EXECUTE	动态执行 SQL 语句
DESCRIBE	描述准备好的查询

1.1.3 SQL 语句结构

每条 SQL 语句均由一个谓词（Verb）开始，该谓词描述这条语句要产生的动作，如 SELECT 或

UPDATE 关键字。谓词后紧接着一条或多条子句（Clause），子句中给出了被谓词作用的数据或提供谓词动作的详细信息。每一条子句由一个关键字开始，SELECT 谓词后为 FROM 关键字。下面介绍 SELECT 语句的主要结构。语法格式如下：

```
SELECT  子句
[INTO 子句]
FROM 子句
[WHERE 子句]
[GROUP  BY 子句]
[HAVING  子句]
[ORDER BY 子句]
```

【例 1.1】在 db_mrsql 数据库中，使用 SELECT 关键字查询药品销售表 tb_sell，并且使用 ORDER BY 关键字按照“药品编号”的降序排列来显示该表中的相关信息。运行结果如图 1.1 所示。**（实例位置：资源包\TM\sl\1\1）**

SQL 语句如下：

```
--使用 SELECT 关键字查询药品销售表 tb_sell，并且使用 order by 关键字按照“药品编号”的降序排列显示该表中的相关信息
SELECT * FROM tb_sell ORDER BY  药品编号  DESC
```

结果　消息

	药品编号	药品名称	药品数量	价格	状态	金额
1	1003	退烧药	10	50	1	500
2	1002	骨伤膏药	10	16	1	160
3	1001	感冒药	20	3	0	60

图 1.1　查询药品销售表中的信息

1.1.4　SQL 语句分类

SQL 语句分为 7 类，具体如下。

- ☑ 变量说明语句：用来说明变量的命令。
- ☑ 数据定义语言：用来建立数据库、数据库对象和定义列。大部分是以 CREATE 或 DROP 开头的命令，如 CREATE TABLE、CREATE VIEW 和 DROP TABLE 等。
- ☑ 数据操纵语言：用来操纵数据库中数据的命令，如 SELECT、INSERT、UPDATE 和 DELETE 等。
- ☑ 数据控制语言：用来控制数据库组件的存取许可、存取权限等命令，如 GRANT、REVOKE 等。
- ☑ 流程控制语言：用于控制应用程序流程的语句，如 IF WHILE 和 CASE 等。
- ☑ 内嵌函数：说明变量的命令。
- ☑ 其他命令：嵌于命令中使用的标准函数。

1.2　数 据 类 型

1.2.1　整数数据类型

整数数据类型是 SQL 语言中最常用的数据类型之一，包括 int、smallint、tinyint 和 bigint 多种数据

类型，它可以存储一定范围的整数。

1．int（integer）

int 数据类型可存储-2^{31}～$2^{31}-1$（−2 147 483 648～2 147 483 647）的所有正负整型数据，存储空间为 4 个字节。32 位的存储空间其中一位表示整型数据值的正负号，其他 31 位表示整型数据值的长度和大小。

2．smallint

smallint 数据类型可存储-2^{15}～$2^{15}-1$（−32 768～32 767）的所有正负整型数据，存储空间为 2 个字节，是比 int 数据类型存储容量小的数据类型。

3．tinyint

tinyint 数据类型可存储 0～255 的所有正整型数据，存储空间为 1 个字节。

4．bigint

bigint 数据类型可存储-2^{63}～$2^{63}-1$（−9 223 372 036 854 775 808～9 223 372 036 854 775 807）的所有正负整型数据，存储空间为 8 个字节。

注意

bigint 数据类型可以应用在任何一个可以应用 int 数据类型的地方。通常，当需要用整数表示，但又超越 int 数据类型范围的情况下，可以使用 bigint。但是，系统并不会自动在超越 int 数据类型范围的情况下，把 int 数据类型自动转化为 bigint 数据类型。

1.2.2 浮点数据类型

浮点数据类型用来存储必须精确计算的正负小数。浮点数据类型的优点是能够存储大范围的数字。缺点是容易发生舍入误差。例如，如果某列的精度是 30，大于 30 的位数可以存储，但却不能保证其精度。舍入误差只能影响一个数超过精度的右边各位，所以在精度范围内数据是准确的。

1．real

real 数据类型可存储−3.40e38～3.40e38 的所有精度正负小数。一个 real 类型的数据占用 4 个字节的存储空间。

2．float

float 数据类型可存储−1.79e308～1.79e308 的浮点精度数字。一个 float 类型的数据占用 8 个字节的存储空间。float 数据类型定义的形式为：

```
float [(n)]
```

其中，n 用于存储科学记数法为 float 型数据尾数的位数，其大小为 1～53。

3．decimal（numeric）

decimal 数据类型的存储数据范围是−e38～e38−1 的固定精度和小数位的数字数据。一个 decimal 类型的数据占用 2～17 个字节。decimal 数据类型定义的形式为：

```
decimal[(p[，s])]
```

其中，p 是指精度，指定小数点左边和右边可以存储的十进制数字的最大个数。精度必须是 1～38 的值。s 是小数位数，小数位数必须介于 0～精度 p。

4．numeric

numeric 数据类型与 decimal 数据类型完全相同，表示数据的范围、所占的存储空间及定义的形式都一样。

1.2.3　字符数据类型

字符数据类型用来存储数字符号、字母以及特殊符号。使用字符型数据时，一般要给数据加上单引号或双引号。

1．char

char 数据类型使用固定长度来存储字符，最大长度为 8000 个字符。char 数据类型的定义形式为：

```
char[(n)]
```

其中，n 表示所有字符占用的存储空间，以字节为单位。n 必须是一个 1～8000 的数值。若不指定 n 值，则系统默认为 1。

利用 char 数据类型来定义表列或者定义变量时，应该给定数据的最大长度。如果实际数据的字符长度小于给定的最大长度，则多余的字节会以空格填充；如果实际数据的字符长度超过给定的最大长度，则超过的字符将会被截断。在使用字符型常量为字符数据类型赋值时，必须使用双引号或单引号将字符型常量括起来，如'Happy' '乐'。

注意

使用 char 数据类型的最大好处在于可以精确计算数据占用的空间。计算机占用的空间非常重要。在一些庞大的系统里定义一些表列长度时几个字节的差距，也许意味着上百兆数据空间的节省或浪费。

2．nchar

nchar 用来定义固定长度的 Unicode 数据，最大长度为 4000 个字符。与 char 类型相似，nchar 数据类型的定义形式为：

```
nchar{[n]}
```

其中，n 表示所有字符占用的存储空间，以字节为单位。n 必须是一个 1～4000 的数值。

nchar 类型采用 Unicode 标准的数据类型，多占用一倍的存储空间。使用 Unicode 标准的好处是因

其使用 2 个字节做存储单位，故其中一个存储单位的容量就大大增加了，可以将多种语言文字囊括在内，如在一个数据列中可以同时出现中文、英文、法文和德文等，而不会出现编码冲突。

3．varchar

varchar 用来存储最长 8000 个字符的变长字符数据。与 char 数据类型不同，varchar 数据类型的存储空间随存储在表列中每一个数据的字符数的不同而变化。

例如，如果定义表列为 varchar(20)，那么存储在该列的数据最长为 20 个字节。但是，在每列数据没有达到 20 个字节时，并不会在多余的字节上填充空格。

varchar 数据类型的定义形式为：

```
varchar[(n)]
```

其中，n 表示所有字符占用的存储空间，以字节为单位。n 必须是一个 1～8000 的数值。若输入的数据过长，SQL 将会截掉其超出的部分。

说明

当存储在表列中的数据值的大小经常变化时，使用 varchar 数据类型可以有效地节省空间。

4．nvarchar

用来定义可变长度的二进制数据，最大长度为 4000 个字符。nvarchar 数据类型的定义形式为：

```
Nvarchar[(n)]
```

其中，n 表示所有字符占用的存储空间，以字节为单位。n 必须是一个 1～4000 的数据。nvarchar 与 nchar 的区别和 varchar 与 char 的区别类似。

1.2.4 日期和时间数据类型

日期和时间数据类型可以存储日期和时间的组合数据，包括 datetime 和 salldatetime 两类。

1．datetime

datetime 数据类型所占用的存储空间为 8 个字节，其中前 4 个字节用于存储 1900 年 1 月 1 日以前或以后的天数，数值分正负，正数表示在此日期之后的日期，负数表示在此日期之前的日期；后 4 个字节用于存储从此日零时起所指定的时间经过的毫秒数。如果在输入数据时省略了时间部分，则系统将 12:00:00:000AM 作为时间默认值；如果省略了日期部分，则系统将 1900 年 1 月 1 日作为日期默认值。

datetime 数据类型用于存储日期和时间的结合体，它可以存储从公元 1753 年 1 月 1 日零时起到公元 9999 年 12 月 31 日 23 时 59 分 59 秒之间的所有日期和时间，其精确度可达 1.33 毫秒。

2．smalldatetime

smalldatetime 数据类型使用 4 个字节存储数据。其中前 2 个字节存储从 1900 年 1 月 1 日以后的天数，后 2 个字节存储此日零时起所指定的时间经过的分钟数。

smalldatetime 数据类型与 datetime 数据类型相似，但日期时间范围较小，为从 1900 年 1 月 1 日到

2079 年 6 月 6 日。此数据类型精度较低，只能精确到分钟，分钟个位上为根据秒数四舍五入的值，即以 30 秒为界四舍五入。例如，当 datetime 时间为 14:38:30.283 时，smalldatetime 认为是 14:39:00。

1.2.5　货币数据类型

货币数据类型用于存储货币值。在使用货币数据类型时，应在数据前加上货币符号，系统才能辨识其为哪国的货币，如果不加货币符号，则默认为“￥”。

1．money

money 数据类型使用 8 个字节存储。货币数据值为-2^{63}～$2^{63}-1$（−9 223 372 036 854 775 808～9 223 372 036 854 775 807）。数据精度为万分之一货币单位。

2．smallmoney

smallmoney 货币数据值为−2 147 483 648～2 147 483 647，存储空间为 4 个字节。其存储的货币值范围比 money 数据类型小。

1.2.6　二进制数据类型

二进制数据是一些用十六进制来表示的数据。例如，十进制数据 245 表示成十六进制数据应该是 F5。在 SQL 中，使用两种数据类型存储二进制数据，分别是 binary 和 varbinary。

1．binary

binary 数据类型用于存储二进制数据。在使用时必须指定 binary 类型数据的大小，至少应为 1 个字节。binary 数据类型占用 n+4 个字节的存储空间。其定义形式为：

```
binary (n)
```

n 表示数据的长度，取值为 1～8000。

在输入数据时，必须在数据前加上字符“0x”，作为二进制数据类型标识。例如，要输入“very”则应输入“0xvery”。如果输入的数据过长，系统将会自动截掉其超出部分。如果输入的数据位数为奇数，则系统会在起始符号“0x”后添加一个 0，如上述的“0xvery”会被系统自动变为“0x0very”。

2．varbinary

varbinary 数据类型用于存储可变长度的二进制数据，存储长度等于实际数值长度加上 4 个字节。varbinary 数据类型的定义形式如下：

```
varbinary (n)
```

n 的取值也为 1～8000，如果输入的数据过长，系统将会截掉超出部分。

1.2.7　文本和图像数据类型

前面介绍的几种数据类型存储的容量有限，当需要存储大量字符及二进制数据时，就需要使用文

本和图像数据类型。SQL 提供了 text、ntext 和 image 3 种文本和图像数据类型。

1．text

text 数据类型用于存储大量文本数据，其容量理论上为 1～$2^{31}-1$（2 147 483 647）个字节，在实际应用时需要视硬盘的存储空间而定。

在 SQL 中，通常将 text 和 image 类型的数据直接存放到表的数据中，而不是存放到不同的数据页中。这就减少了存储 text 和 image 数据的空间，并相应减少了磁盘处理这类数据的 I/O 数量。

2．ntext

ntext 数据类型与 text 类型相似。不同的是，ntext 类型采用 UNICODE 标准字符集。因此，其理论容量为 $2^{30}-1$（1 073 741 823）个字节。

3．image

image 数据类型是可变长度的二进制数据类型，最大长度为 $2^{31}-1$ 个字符。通常用来存储图形等对象连接与嵌入 OLE（Object Linking and Embedding）对象。在输入数据时与 binary 数据类型一样，必须在数据前加上字符“0x”作为二进制标识。

1.2.8 用户自定义数据类型

用户自定义数据类型并不是真正的数据类型，它只是提供了一种加强数据库内部元素和基本数据类型之间一致性的机制。通过使用用户自定义数据类型能够简化对常用规则和默认值的管理。

在 SQL Server 数据库中，可以使用系统数据类型 sp_addtype 创建用户自定义数据类型。语法格式如下：

```
sp_addtype[@typename=]type，
[@phystype=]system_data_type
[，[@nulltype=]'null_type']
[，[@owner=]'owner_name']
```

参数说明如下。

- ☑ [@typename=]type：指定待创建的用户自定义数据类型的名称。用户自定义数据类型名称必须遵循标识符的命名规则，而且在数据库中唯一。
- ☑ [@phystype=]system_data_type：指定用户自定义数据类型所依赖的系统数据类型。
- ☑ [@nulltype=]'null_type'：指定用户自定义数据类型的可空属性，即用户自定义数据类型处理空值的方式。取值为 NULL，NOT NULL 或 NONULL。

【例 1.2】 使用系统数据类型 sp_addtype 创建用户自定义数据类型。**（实例位置：资源包\TM\sl\1\2）**

SQL 语句如下：

```
use db_mrsql --使用 db_mrsql 数据库
--使用 sp_addtype 创建用来存储邮政编码信息的 postalcode 用户的定义数据类型
EXEC sp_addtype mrVarChar,'varchar(20)','not null'
```

执行此 SQL 语句，将创建自定义数据类型“mrVarChar”。

注意

如果已经使用 Management Studio 创建了同名的用户自定义类型，那么执行此 SQL 代码时会出现如图 1.2 所示的错误。

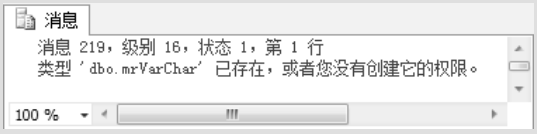


图 1.2　创建同名自定义数据类型时报错

创建了用户自定义数据类型后，就可以像系统数据类型一样使用它。例如，在 db_mrsoft 数据库的表中创建新的字段，为字段“学生姓名”指定数据类型时，可以在下拉列表框中选择刚刚创建的数据类型 mrVarChar，如图 1.3 所示。

图 1.3　创建字段时选择 mrVarChar 数据类型

另外，根据需要，还可以修改和删除用户自定义的数据类型。使用系统存储过程 sp_droptype 可删除用户定义的数据类型。

1.3　常　　量

内存中存储的始终恒定的量叫作常量。SQL 同样规定了数字、字符串、日期时间和符号常量的格式。

1.3.1　数值常量

（1）整数和小数常量在 SQL 中被写成普通的小数数字，前面可加正负号。

例如：500，−45，11.23。

（2）在数字常量的各数位之间不能加逗号。

例如：456 456 这个数字不能表示为 456,456。

（3）浮点常量使用符号 e 来指定。

例如：5.5e3，−6.23e2，7.8e−3。其中，e 被读作“乘 10 的几次幂”。

1.3.2 字符串常量

（1）SQL 规定字符串常量要包含在单引号内。

例如：'HELLO'。

（2）如果一个字符串常量文本中包含了一个单引号，则在这个常量内写作两个连续的单引号。

例如：“' COME ON' ' 0.5 千米'”表示“COME ON ' 0.5 千米”。

1.3.3 日期和时间常量

（1）表示日期、时间和时间间隔的常量值被指定为字符串常量。下面的书写是合法的。

例如：'2008-03-10 '，'05/08/2010'。

（2）日期和时间根据国家不同，书写方式也不同。

例如：美国为 mm/dd/yyyy；欧洲为 dd.mm.yyyy；日本为 yyyy-mm-dd 等。

1.4 变　　量 

内存中存储的可以变化的量叫作变量。为了在内存中存储信息，用户必须指定存储信息的单元，并为该存储单元命名，以方便获取信息，这就是变量的功能。SQL 语句可以使用两种变量：一种是局部变量（Local Variable），另外一种是全局变量（Global Variable）。局部变量和全局变量的主要区别在于存储的数据作用范围不同。

1.4.1 局部变量

局部变量是用户可以自定义的变量，它的作用范围仅限于程序内部。局部变量的名称由用户自行定义，符合标识符命名规则，以@开头。

1．声明局部变量

局部变量的声明需要使用 DECLARE 语句。语法格式如下：

```
DECLARE
{
@varible_name    datatype    [ ,... n   ]
}
```

参数说明如下。

- ☑ @varible_name：局部变量的名称，必须以@开头，变量名形式符合 SQL 标识符的命名方式。
- ☑ datatype：局部变量的数据类型，可以是除 text、ntext 或者 image 类型以外所有的系统数据类型和用户自定义数据类型。一般来说，如果没有特殊的用途，建议使用系统提供的数据类型，这样做可以减少维护应用程序的工作量。