Wonderware® FactorySuite™

SQL Access Manager 用户指南

修订版 A

最后修订日期：2002 年 11 月 20 日

Invensys Systems, Inc.

保留所有权利。未经 Invensys Systems, Inc. 公司事先书面许可，不得以电子、机械、影印、录制或其它任何方式复制、传输本文任何部分或将其储存至检索系统。使用本文所含信息不应承担任何版权或专利责任。虽然在本文的编制过程中已采取一切预防措施，但错误或疏漏在所难免，出版商和作者对此概不承担任何责任。对于因使用本文所含信息而导致的任何损害，亦不承担任何赔偿责任。

本文所含信息如有变更，恕不另行通知，亦不代表 Invensys Systems, Inc. 公司一方的承诺。本文所述软件系在遵守许可证或保密协议的前提下提供。本软件的使用或复制须遵从这些协议规定的各项条款。

© 2002 Invensys Systems,Inc. 版权所有。保留所有权利。

Invensys Systems, Inc.
33 Commercial Street
Foxboro, MA 02035
(949) 727-3200
http://www.wonderware.com

商标

本书所有作为商标或服务标志的专门名词均采用大写字母印刷。Invensys Systems, Inc. 公司并不保证此类信息的准确性。使用本书中的任何术语不应视为会影响任何商标或服务标志的有效性。

Alarm Logger、 ActiveFactory、 ArchestrA、 Avantis、 DBDump、 DBLoad、 DTAnalyst、 FactoryFocus、 FactoryOffice、 FactorySuite、 hotlinks、 InBatch、 InControl、 IndustrialRAD、 IndustrialSQL Server、 InTouch、 InTrack、 MaintenanceSuite、 MuniSuite、 QI Analyst、 SCADAlarm、 SCADASuite、 SuiteLink、 SuiteVoyager、 WindowMaker、 WindowViewer、 Wonderware 以及 Wonderware Logger 均为 Invensys plc、其分支及加盟机构的商标。所有其它品牌可能是其相应所有者的商标。

目录

第 1 章： SQL Access Manager.....5

简介 5
关于本手册 6
技术支持 7
ODBC 兼容性 8

第 2 章： 配置与连接数据库.....9

使用 Oracle 8.0 9
SQLConnect() 格式 9
在 Oracle Date 字段中记录日期与时间 10
使用 Microsoft SQL Server 11
配置客户端 11
支持的数据类型 12
使用 Microsoft Access 12
字符串长度 12
支持的数据库的数据类型值 13

第 3 章： 配置 SQL Access Manager 15

SQL Access Manager 综述 15
配置绑定列表 16
使用特殊分隔符 19
配置表格模板 20
SQL.DEF 文件 23

第 4 章： 使用 SQL 函数.....25

SQL 函数 25
函数 25
SQL 参数 34
在 InTouch QuickScript 中使用 SQL 函数 37
指定复合查询 37
将值放入 InTouch 标记 41
让 InTouch 标记与数据库字段值保持一致 41
数据更新规则的含义 42

第 5 章： 疑难排解 43

函数疑难排解 43
结果码错误消息 43
特定的数据库错误消息 45
调试 SQL Access 46

附录 A: 保留关键字 47

SQL Access 与 ODBC 47
InTouch 49

索引 51

第 1 章

SQL Access Manager

Wonderware® FactorySuite™ SQL Access Manager 可供您访问、修改、创建及删除数据库中的表格。数据库在表格中存储共享公共属性或字段的信息。“结构化查询语言”（Structured Query Language，简称 SQL）是用于访问该类信息的语言。

目录

- 简介
- 关于本手册
- 技术支持
- ODBC 兼容性

简介

InTouch SQL Access Manager 附加程序设计用于方便地传输数据，例如将批次配方从 SQL 数据库传递给 InTouch 应用程序。它还可以简化运行时数据、报警状态或历史数据从 InTouch 到 SQL 数据库的传递过程。例如，在一个机械循环完成之后，公司可能需要保存几套数据，每套对应一个不同的应用程序。 SQL 数据库提供了相应的功能，使您可以在一个或多个第三方应用程序之间方便地传递信息。 SQL Access Manager 允许在任何 InTouch 应用程序中访问并显示这些数据。

InTouch SQL Access 产品由 SQL Access Manager 程序与 “SQL 函数”组成。 SQL Access Manager 程序用于创建数据库列，并将它们和 InTouch 标记名字典中的标记名相关联。将数据库列与 InTouch 数据库标记名关联起来的过程被称作 “绑定”。通过将 InTouch 数据库标记名与数据库列绑定起来， SQL Access Manager 便可直接操作数据库中的数据。 SQL Access Manager 可将数据库字段名及其关联保存到一个逗号分隔变量 (.CSV) 格式的文件 "SQL.DEF"。（此文件位于 InTouch 应用程序目录，并可使用 SQL Access Manager 或 “记事本”之类的任何文本编辑器进行查看或修改）。 SQL Access Manager 还可创建 “表格模板”来定义数据库的结构与格式。

如需有关 “绑定列表”与 “表格模板”的详细信息，请参阅 µ⁄3 ’¬£¨°× 配置 SQL Access Manager”。

“SQL 函数”可用在任何 InTouch 动作脚本中。这些函数可根据操作员的输入、标记名值的更改或特定条件集合的存在而自动执行。例如，若存在报警条件，则应用程序会执行 **SQLInsert()** 或 **SQLUpdate()** 命令，以保存所有适用的数据点与报警状态。“SQL 函数”可用于新建表格、在表格中插入新记录、编辑现有的表格记录、清空表格、删除表格、选择并滚动浏览记录等。

备注 本用户指南未提及的数据库系统不受支持。

关于本手册

本手册由逻辑上相互连贯的多个章节组成，介绍使用 SQL Access Manager 的方方面面。它采用 “过程化”的格式编写而成，以带有序号的步骤阐述如何执行大多数功能或任务。

如果您联机阅读本手册，看到绿色文本时，可单击该文本以 “跳转”到所引用的部分或章节。跳转到另一节或另一章后，如果您想要返回原来的章节，请使用所提供的 “后退”选项。

提示 这些 “提示”告诉您如何更简便快捷地完成某项功能或任务。

*InTouch 用户指南*将帮助您熟悉 WindowMaker 开发环境及其工具，如需有关详细信息，请参阅第 1 章 “WindowMaker 程序元素”。若要了解如何使用窗口、图形对象、向导及 ActiveX 控件等，请阅读第 2 章 “使用 WindowMaker”。

如需有关 InTouch 运行时环境 (WindowViewer) 的详细信息，请参阅联机 *InTouch Runtime 用户指南*。

此外， *InTouch 参考指南*为您提供了更为深入的、有关 InTouch 脚本语言、系统标记名以及标记名**点域**的参考资料。

如需有关附加程序 SPC Pro 的详细信息，请参阅 *SPC Pro 用户指南*。

若需有关附加程序 Recipe Manager 的详细信息，请参阅 *Recipe Manager 用户指南*。

FactorySuite 软件包还包含全部 FactorySuite 组件的联机手册。

备注 若要查看或打印联机手册，须安装 Adobe Acrobat Reader （4.0 或更高版本）。

假定

本手册假定您：

- 熟悉 Windows 2000、Windows XP 和（或）Windows NT 操作系统工作环境。
- 了解如何使用鼠标、 Windows 菜单、选择选项及访问联机 “帮助”。
- 具备使用编程语言或宏语言的经验。若要取得最佳效果，最好应理解一些编程概念，如变量、语句、函数及方法。

技术支持

“Wonderware Technical 技术支持”提供多种支持选项，来回答有关 Wonderware 产品及其实施的各种问题。

在与技术支持部门联系之前，请参阅 *SQL Access Manager 用户指南*中相关的章节，以寻求您在使用系统时所遇问题的可能解决方案。如果您认为有必要求助于技术支持，请提供以下信息：

1. 软件序列号。
2. 运行的 InTouch 版本。
3. 所用操作系统的类型与版本。例如， Microsoft Windows NT 4.0 版工作站。
4. 所遇系统错误消息的准确描述。
5. Wonderware Logger、Microsoft Diagnostic utility (MSD) 或任何其它诊断应用程序提供的所有相关输出列表。
6. 尝试解决该问题的方法细节及其结果。
7. 有关如何重现问题的细节。
8. 如果所遇问题是老问题，且该问题仍然存在，请提供指定的 “Wonderware 技术支持”案例号。

如需有关 “技术支持”的详细信息，请参阅联机 *FactorySuite 系统管理员指南*。

ODBC 兼容性

SQL Access Manager 是一套兼容 ODBC 的应用程序，只要提供了相应的 ODBC 驱动程序，它可以同任何数据库系统进行通信。在使用 ODBC 驱动程序之前，必须通过 “Microsoft ODBC 管理器”程序对它进行配置，以便设置兼容 ODBC 的应用程序与数据库之间的链接。

要配置 ODBC 驱动程序

1. 运行 “Microsoft ODBC 管理器”程序。
2. 选择驱动程序或数据源，然后单击**添加新名称**、**设置缺省值**或**配置**。 **ODBC 驱动程序设置** 对话框。

选项	描述
数据源名	用于识别数据源的用户自定义名称。
描述	此数据源的用户自定义描述。
数据库目录	确定包含数据库文件的目录。如果未指定，则使用当前工作目录。

提示 输入配置所选驱动程序所需的任何其它信息。

3. 单击**确定**。

提示 驱动程序将每个字段的值写入 ODBC.INI 文件。这些值是连接数据源的缺省值。通过修改数据源字段，可以更改这些缺省值。对于 “ODBC 驱动程序设置”对话框不支持的任何属性，可在 ODBC.INI 文件相应的数据源部分手工插入这些项目。

第 2 章

配置与连接数据库

SQL Access Manager 支持在 Oracle、 Microsoft SQL Server 及 Microsoft Access 中开发的数据库。每个数据库的要求均会有所不同。本章用不同的章节分别介绍每种数据库，向您说明如何配置特定的数据库，以便同 SQL Access Manager 通讯。

目录

- 使用 Oracle 8.0
- 使用 Microsoft SQL Server
- 使用 Microsoft Access
- 支持的数据库的数据类型值

使用 Oracle 8.0

要与 Oracle 8.0 进行通讯

1. 确认 InTouch 客户端机器上是否存在 Oracle OLEDB Provider (MSDAORA.DLL)。此文件是在安装 InTouch 时由 MDAC 安装的。
2. 通过执行 InTouch 动作脚本中的 SQLConnect() 函数连接到 Oracle。

   如需有关 SQLConnect() 用法的详细信息，请参阅 第 4 章 “使用 SQL 函数”。

SQLConnect() 格式

SQLConnect() 函数用于连接 Oracle 数据库。SQLConnect() 函数所用的连接字符串格式如下：

```
SQLConnect(ConnectionId,"<attribute>=<value>;
   <attribute>=<value>;...");
```

下面介绍 Oracle 使用的各个属性：

属性	值
Provider	MSDAORA
User ID	用户名。
Password	口令。
Data Source	Oracle Server 名称

示例

```
SQLConnect(ConnectionId,"Provider=MSDAORA; Data
    Source=OracleServer; User ID=SCOTT;
Password=TIGER;");
```

在 Oracle Date 字段中记录日期与时间

要在 Oracle 日期字段中记录日期与时间，必须使用 delim 函数配置 “绑定列表”。

要在 Oracle 日期字段中记录日期与时间

1. 在 **SQL Access Manager** 下的 “应用程序浏览器”中，双击**绑定列表**。此时会出现**绑定列表配置**对话框。



2. 在 **Tagname.FieldName** 框中，输入要使用的标记名。
3. 在**列名**框中，输入 **DATE_TIME delim()** 函数。
4. 在 InTouch 应用程序中，创建一个 QuickScript，以准备从当前日期与时间获取的输入数据。例如：

```
DATE_TIME_TAG = "TO_DATE('" + $DateString + "" +
    StringMid($TimeString,1,8) + "','mm/dd/yy
    hh24:mi:ss')";
```

提示 Date_Time_Tag 在运行时会显示如下：

```
TO_DATE('08/22/97 23:32:18' ,'mm/dd/yy
hh24:mi:ss')
```

使用 Microsoft SQL Server

要与 Microsoft SQL Server 通讯

1. 配置 Windows 数据库客户端。
2. 通过在 InTouch QuickScript 中执行 **SQLConnect()** 函数，连接 Microsoft® SQL Server。

   如需有关 SQLConnect() 用法的详细信息，请参阅 第 4 章 “使用 SQL 函数”。

配置客户端

SQLConnect() 格式

SQLConnect() 函数用于连接 Microsoft SQL Server。通过执行此函数，您可以登录到数据库服务器上，并打开一个连接，以便执行其它的 SQL 函数。 **SQLConnect()** 函数使用的连接字符串格式如下：

```
SQLConnect(ConnectionId,"<attribute>=<value>;
    <attribute>=<value>;...");
```

下文介绍 Microsoft SQL Server 使用的各个属性：

属性	值
Provider	SQLOLEDB
DSN	在 “Microsoft ODBC 管理器”中配置的数据源名称。
UID	登录 ID，区分大小写。
PWD	口令，区分大小写。
SRVR	要访问的数据库表格所在服务器计算机的名称。
DB	要访问的数据库的名称。

示例

```
SQLConnect(ConnectionId,"DSN=SQL_Data;UID=OPERATOR;PWD=XYZ
    Z");
```

支持的数据类型

SQL Access Manager 将 InTouch 中的四种数据类型 （离散型、整型、实型及消息型）与数据库系统中的其它数据类型关联起来。 char 数据类型包含固定长度的字符串。 InTouch 消息标记名要求使用 char 数据类型。同时必须指定字段长度。 Microsoft SQL Server 数据库支持长达 8,000 个字符的 char 字段。不过， InTouch “消息”标记名则限制最多使用 131 个字符。如果消息变量包含的字符超过数据库字段中指定的长度，在插入数据库时该字符串会被截短。

int 数据类型表示 InTouch “整型”标记名。如果未指定字段长度，则该长度会设置为数据库的缺省值。如果指定了长度，则其形式为 “宽度”。“宽度”确定了列的最大位数。

float 数据类型表示 InTouch “实型”标记名。字段长度设置是由数据库固定下来的。此数据类型不要求指定字段长度。

使用 Microsoft Access

要与 Microsoft Access 通讯，必须通过在 InTouch QuickScript 中执行 **SQLConnect()** 函数来连接它。

SQLConnect() 格式

SQLConnect() 函数用于连接 Microsoft Access 数据库。通过执行此函数，您可以登录到数据库服务器，并打开一个连接，以便执行其它的 SQL 函数。 **SQLConnect()** 使用的连接字符串格式如下：

```
SQLConnect(ConnectionId,"<attribute>=<value>;
   <attribute>=<value>;...");
```

DSN 是 Microsoft Access 使用的一个属性，它是在 “Microsoft ODBC 管理器”中配置的数据源的名称。

示例

```
SQLConnect(ConnectionId,"DSN=MSACC");
```

字符串长度

SQL Access Manager 支持的有效数据类型取决于所使用的 ODBC 驱动程序的版本。 text 数据类型包含固定长度的字符串，并且会结合 InTouch “消息”型标记名一起使用。必须指定长度。 Microsoft Access 数据库支持 text 字段，其长度最高可达 255 个字符。 InTouch “消息”型标记名则限制为最长 131 个字符。如果消息变量包含的字符超过给数据库字段指定的长度，则插入数据库时该字符串会被截短。 Microsoft Access ODBC 驱动程序支持每个列名最多使用 17 个字符。使用 SQLSetStatement(Select Col1, Col2, ...) 时，支持最多使用 40 个列。

支持的数据库的数据类型值

Oracle

数据类型	长度	缺省值	范围	标记类型
char	2,000 个字符	1 个字符		消息型
number	38 位	38 位		整数

Microsoft SQL Server

数据类型	长度	缺省值	范围	标记类型
char	8,000 个字符			消息型
int			-2,147,483,648 到 2,147,483,647	整型
float	15 位		$-1.79E^{+308}$ 至 $1.79E^{+308}$	实型

Microsoft Access 2000

数据类型	长度	缺省值	范围	标记类型
text	255 个字符			消息型
number				整型
number				实数

第 3 章

配置 SQL Access Manager

SQL Access Manager 实用程序可以创建 “绑定列表”和 “表格模板”。“绑定列表”关联数据库列与 “InTouch 标记名数据字典”中的标记名。“表格模板”定义数据库中新表格的结构与格式。

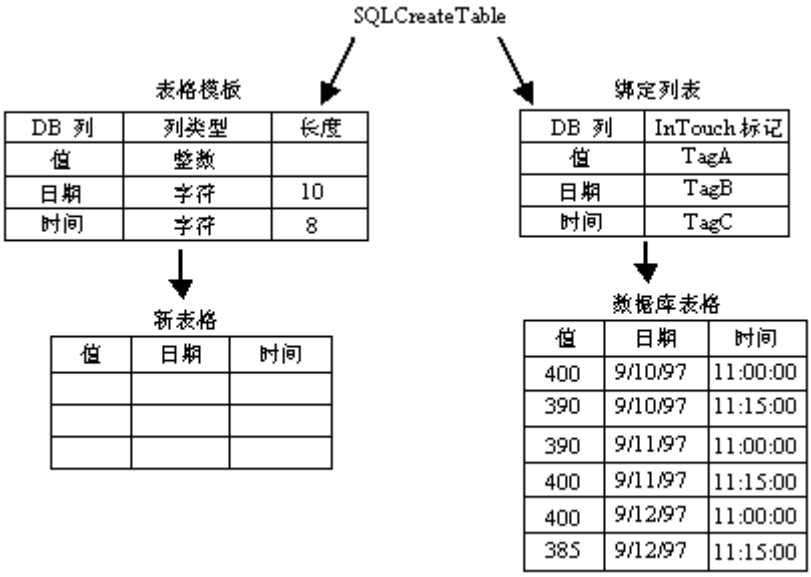
目录

- SQL Access Manager 综述
- 配置绑定列表
- 使用特殊分隔符
- 配置表格模板
- SQL.DEF 文件

SQL Access Manager 综述

InTouch 应用程序执行 **SQLCreateTable()** 命令时，“表格模板”参数会定义新数据库文件的结构。

执行 **SQLInsert()**、 **SQLSelect()** 或 **SQLUpdate()** 时，“绑定列表”参数定义所使用的 InTouch 标记名及关联的数据库列。

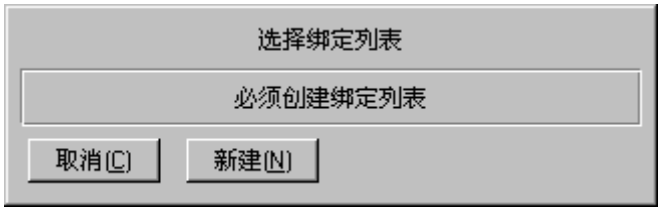


配置绑定列表

“绑定列表”关联数据库列与 “InTouch 数据字典”中的标记名。

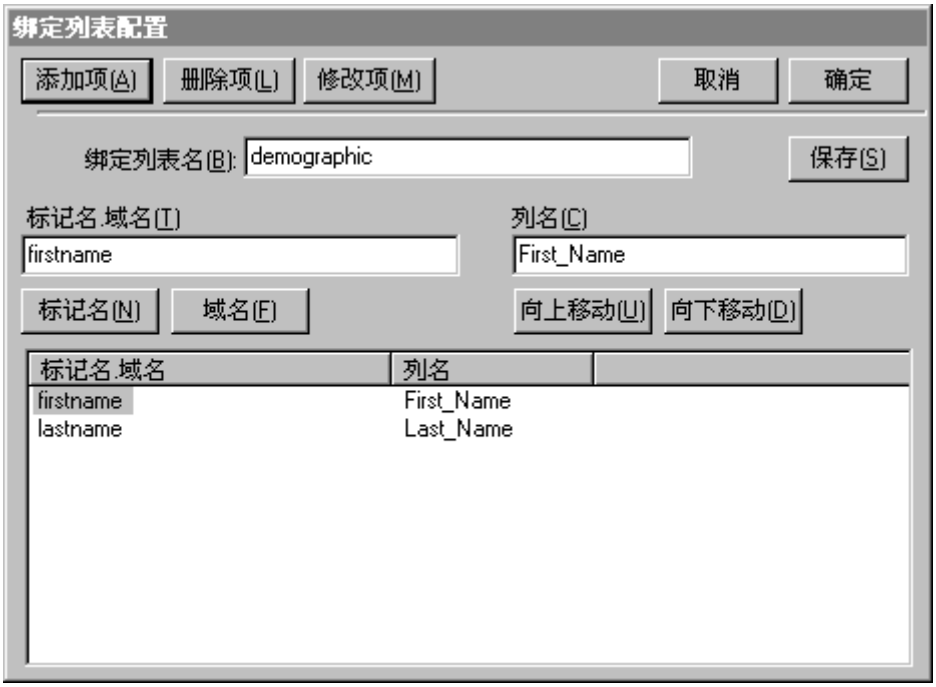
要新建绑定列表

1. 在**特别**菜单上，指向 **SQL Access Manager**，然后单击**绑定列表**，或在 **SQL Access Manager** 下的 “应用程序浏览器”中，双击**绑定列表**。



2. 单击**新建**。

3. 此时会出现**绑定列表配置**对话框。



提示 如果您右击任意文本输入框，则会出现一个菜单，显示出您可以给所选文本应用的各个命令。

4. 在**绑定列表名称**框中，输入 “绑定列表名称”。

提示 “绑定列表名称”最长可达 32 个字符。新 “绑定列表”会将数据库的各个列链接到 InTouch 标记名。例如，如果要创建一个员工人数统计列表，您可以输入 “绑定列表名称”，使其与此处的员工信息相关联。

备注 **SQLInsert()**、**SQLSelect()** 及 **SQLUpdate()** 函数使用 “绑定列表”参数。

5. 在 **Tagname.FieldName** 框中，输入 InTouch **Tagname.FieldName** 名称。

提示 “标记名字典”将 **tagname.field** 与数据库中的 **列名** 相关联。如果目前未在 “标记名字典”中定义此标记名，请双击它以打开**标记名字典**对话框，并立即定义它。

6. 单击**标记名**以选择现有的标记名。此时会出现 “标记浏览器”。

提示 “标记浏览器”会显示当前所选标记源的标记名。要选择标记名，请双击它，或者选择它然后单击**确定**。要选择标记名的**点域**，请单击**点域**箭头，并选择要使用的**点域**，然后单击**确定**。

备注 WindowViewer 一经启动，未在应用程序中使用但已在 SQLAccess 绑定列表中指定的 I/O 型标记名便会被激活（提示给 I/O 服务器）。不必连接数据库即可观察到此操作。

如需有关 “标记浏览器”的详细信息，请参阅联机 *InTouch 用户指南*。

7. 单击 **FieldName** 以便将**点域**追加到标记名。此时会出现**选择字段名**对话框。
8. 单击要使用的**点域**。该对话框将关闭，且**点域**会自动追加到 **Tagname.FieldNam**e 字段中的标记名。

   如需有关标记名**点域**的详细信息，请参阅 *InTouch 用户指南*第 4 章。
9. 在**列名**对话框中，输入列名。

提示 “列名”最长可达 30 个字符。列名直接与数据库列名关联。如果 “列名”包含空格，请在 “绑定列表”及在脚本中使用时用方括号括起 “列名”。例如：

```
WHERE EXPR= "[Pipe Flow} = " text (tagname,"#");
```

提示 您也可以使用特殊的 “分隔符”关联列名与数据库。

如需有关特殊分隔符的详细信息，请参阅 “使用特殊分隔符”。

10. 单击**向上移动**将列表中选定的标记名上移一级。
11. 单击**向下移动**将列表中选定的标记名下移一级。
12. 单击**添加项**将新的 **Tagname.FieldName** 与**列名**添加到 “绑定列表”。
13. 单击**删除项**从 “绑定列表”删除所选的 **Tagname.FieldName** 与**列名**。
14. 单击**修改项**以修改为此 “绑定列表”选定的 **Tagname.FieldName** 或**列名**。
15. 单击**确定**以保存新的 “绑定列表”配置并关闭对话框。

提示 您可以单击**保存**以保存设置而不关闭对话框。

要修改绑定列表

1. 在**特别**菜单中，指向 **SQL Access Manager**，然后单击**绑定列表**，或在 **SQL Access Manager** 下的 “应用程序浏览器”中，双击**绑定列表**。

2. 此时会出现**选择绑定列表**对话框。



3. 选择要更改的 “绑定列表”名称，然后单击**修改**。此时会出现**绑定列表配置**对话框。
4. 修改所需项目。
5. 单击**确定**以保存更改并关闭对话框。

如需有关配置 “绑定列表”的详细信息，请参阅 “要新建绑定列表”。

要删除绑定列表

1. 在**特别**菜单中，指向 **SQL Access Manager**，然后单击**绑定列表**，或在 **SQL Access Manager** 下的 “应用程序浏览器”中，双击**绑定列表**。
2. 此时会出现**选择绑定列表**对话框。



3. 选择要删除的 “绑定列表”名称。
4. 单击**删除**。此时会出现一个消息框，要求您确认删除。单击**是**以删除所选的 “绑定列表”，或单击**否**以取消删除操作。此时会重新出现**绑定列表配置**对话框。
5. 单击**确定**以关闭对话框。

使用特殊分隔符

SQLInsert() 与 **SQLUpdate()** 函数使用缺省格式，用英文单引号括起消息字符串。有些 SQL 数据库希望收到由另一类型的分隔符括起的消息字符串。例如， Oracle 希望收到由方括号括起的日期字符串。出现这种情况时，必须使用 **Delim()** 函数，具体用法如下：

在**绑定列表配置**对话框的**列名**字段中，在列名后输入关键字 "delim"（不区分大小写）。输入关键字 "delim" 时必须后跟：

- 左括号
- 左分隔符
- 逗号

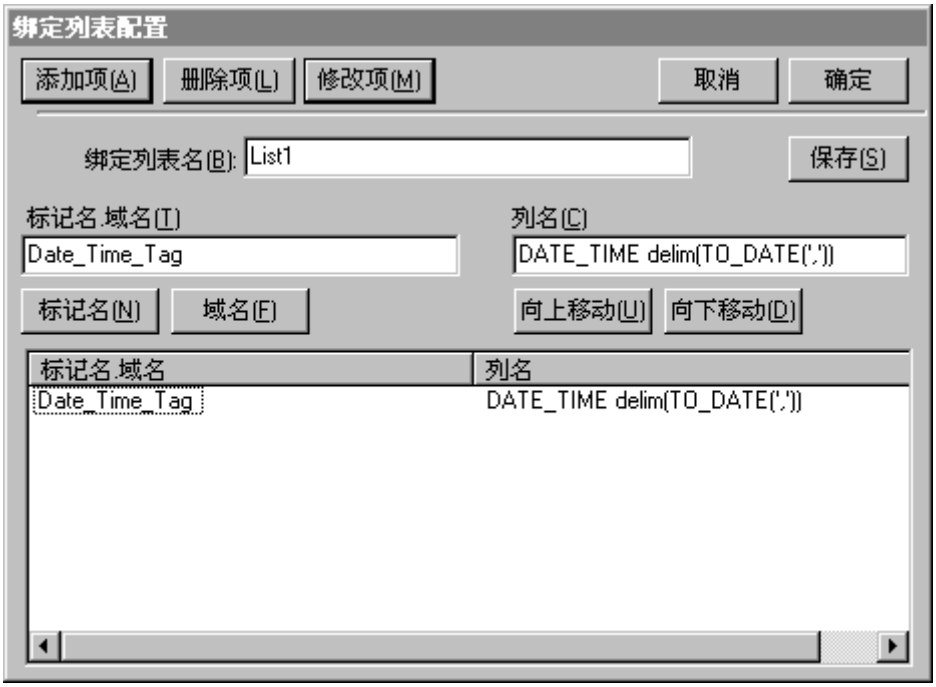
- 右分隔符
- 右括号

示例：**datestring delim (‘,‘)**

要左、右使用相同的分隔符，只需在括号内指定分隔符而不使用逗号。

示例：**datestring delim (‘,‘)**

下例使用不同的左、右分隔符。请注意**日期分隔符 (‘,’)** 是在**列名**字段中的何处输入的。



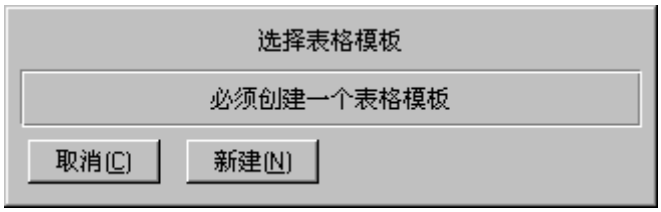
如需有关使用 Oracle 日期字段记录日期与时间的详细信息，请参阅 第 2 章 “配置与连接数据库”。

配置表格模板

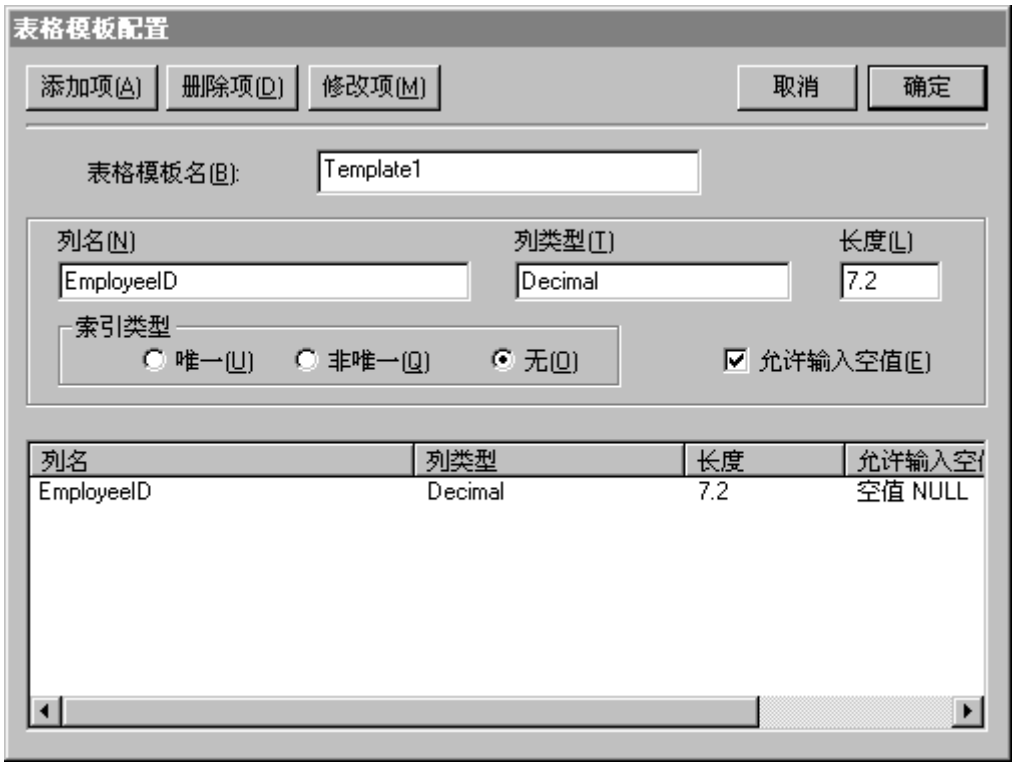
此命令创建 “表格模板”，以定义数据库中新表格的结构与格式。

要新建表格模板

1. 在**特别**菜单上，指向 **SQL Access Manager**，然后单击**表格模板**，或在 **SQL Access Manager** 下的 “应用程序浏览器”中，双击 **表格模板**。



2. 单击**新建**。
3. 此时会出现**表格模板配置**对话框。



提示 如果您右击任意文本输入框，则会出现一个菜单，显示出您可以给所选文本应用的各个命令。

4. 在**表格模板名称**框中，输入 “表格模板”的名称。

备注 “表格模板名称”最长可达 32 个字符。如果您要创建索引，不管是否唯一的，“表格模板名称”均不得超过 **24 个字符**。“表格模板”名称用于确定 **SQLCreateTable()** 函数的数据库结构。

5. 在**列名**框中，输入 “表格模板”的列名。“列名”最长可达 30 个字符。

6. 在**列类型**框中，输入该列的数据类型。数据类型选项会根据所使用的数据库而变化。

   如需有关特定数据库的数据类型的详细信息，请参阅第 2 章 “配置与连接数据库”。

7. 选择**索引类型**，具体如下：

   唯一索引
   可以确保索引列不包含重复的值。

   非唯一索引
   索引列中每个值的组合可以重复。

   无
   无索引。

   提示 执行 **SQLCreateTable()** 时，会自动创建一个索引文件。

8. 选择**允许空值输入**以便允许在此列中输入空数据。

   备注 InTouch 不支持空数据。

   插入数据时，如果尚未给标记名中输入值，则值会是：

数据类型	值
Discrete	0
整数	0
Message	不含字符的字符串

   选择数据时，空值将根据上述数据类型进行转换。

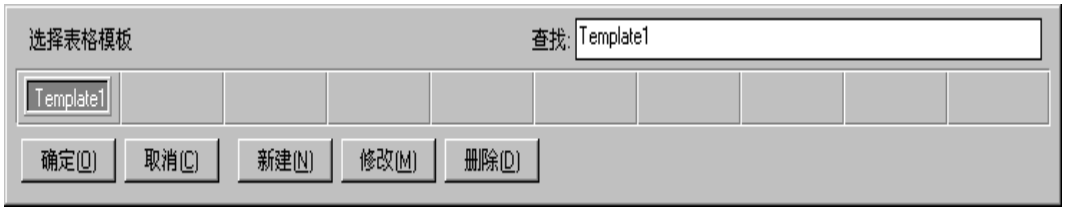
9. 单击**添加项**以便将新的 “列名”、“列类型”、“长度”及 “索引类型”添加到 “表格模板”中。
10. 单击**删除项**以便从 “表格模板”删除所选的 “列名”、“列类型”、“长度”及 “索引类型”。
11. 单击**修改项**以便修改 “表格模板”列表中所选的 “列名”、“列类型”、“长度”及 “索引类型”。
12. 单击**确定**，以保存新的 “表格模板”配置并关闭对话框。

    提示 您可以单击**保存**以保存设置而不关闭对话框。

要修改表格模板

1. 在**特别**菜单中，指向 **SQL Access Manager**，然后单击**表格模板**，或在 **SQL Access Manager** 下的 “应用程序浏览器”中，双击**表格模板**。

2. 此时会出现**选择表格模板**对话框。

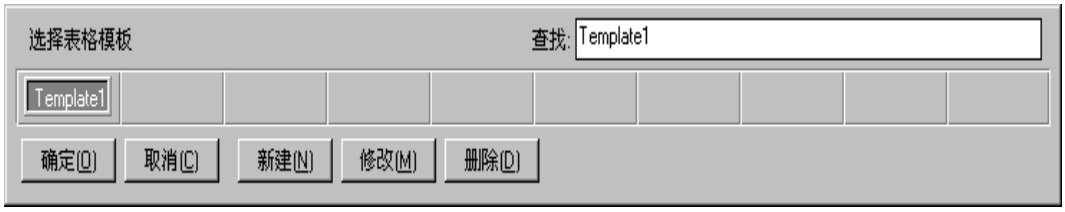


3. 选择要修改的 “表格模板”名称，然后单击**修改**。此时会出现**表格模板配置**对话框。
4. 修改所需项目。
5. 单击**确定**以保存更改并关闭对话框。

   如需有关配置 “表格模板”的详细信息，请参阅 “要新建表格模板”。

要删除表格模板

1. 在**特别**菜单中，指向 **SQL Access Manager**，然后单击**表格模板**，或在 **SQL Access Manager** 下的 “应用程序浏览器”中，双击**表格模板**。
2. 此时会出现**选择表格模板**对话框。



3. 选择要删除的 “表格模板”名称。
4. 单击**删除**。此时会出现一个消息框，要求您确认删除。单击**是**以删除所选的 “绑定列表”，或单击**否**以取消删除操作。此时会再次出现**表格模板配置**对话框。
5. 单击**确定**以关闭对话框。

SQL.DEF 文件

SQL Access Manager 将 “绑定列表”与 “表格模板”的配置信息保存到文件 " SQL.DEF"。此文件是逗号分隔变量 (.CSV) 类型的文件。您可以使用 SQL Access Manager 或任何文本编辑器 （如 “记事本”）来查看或修改 SQL.DEF 文件。该文件中出现的数据如下：

:BindListName,*BindListName*

Tagname1.FieldName,ColumnName1

Tagname2.FieldName,ColumnName2

Tagname3.FieldName,ColumnName3

:TableTemplateName,*TableTemplateName*

ColumnName1,ColumnType,[ColumnLength],Null,Index

ColumnName2,ColumnType,[ColumnLength],Null,Index

ColumnName3,ColumnType,[ColumnLength],Null,Index

第 4 章

使用 SQL 函数

InTouch 使用 SQL 函数与数据库中的信息进行交互。这些函数是标准 InTouch QuickScript 函数的扩展，可用在任何脚本中。它们可供您选择、修改、插入或删除选择访问的表格中的记录。

目录

- SQL 函数
- SQL 参数
- 在 InTouch QuickScript 中使用 SQL 函数

SQL 函数

本节列出每种 SQL 函数。请注意， SQL 动作是同步的。在 SQL 操作完成之前，控制权不会返回给 InTouch （InTouch 的趋势分析、轮询等操作会被挂起）。

所有 SQL 函数 （**SQLNumRows()** 除外）均会返回一个 *ResultCode*。如果 *ResultCode* 非零，则表示函数执行不成功，应采取其它操作。*ResultCode* 可由 **SQLErrorMsg()** 函数使用。

SQL 函数的一般格式如下：

```
SQLFunction(Parameter1, Parameter2,...)
```

如需有关每个 SQL 函数的详细信息及其用法示例，请参阅 *InTouch 参考指南*。

函数

SQLAppendStatement(ConnectionId, SQLStatement)

将 SQLStatement 语句追加到 *ConnectionId* 的缺省 SQL 语句。

SQLClearParam(StatementId, ParameterNumber)

根据参数是数字型还是字符串型，将与 StatementId 关联 *ParameterNumber* 值设置为 0，或长度为 0 的字符串。

SQLClearStatement(ConnectionId, StatementId)

清理与 StatementId 关联的资源。不过，与 *ConnectionId* 关联的缺省语句仍保持不变。

SQLClearTable(ConnectionId, TableName)

删除 TableName 表中所有的行。

SQLCommit(ConnectionId)

提交上一个 SQLTransact 创建的事务。

SQLConnect(ConnectionId, ConnectString)

ConnectString 参数同大多数 ADO 文档 （最详尽的可能就是 Microsoft ADO API Reference）中介绍的 ConnectionString 一样。此参数在任何 InTouch 应用程序中均须修改，以利用特定数据库管理系统自身 OLE DB 供应器的强大威力。

ConnectString 参数的一般形式是由分号隔开的不同组件组成。第一个组件通常是 Provider=ProviderName，其中 ProviderName 是特定数据库系统的 OLE DB 供应器。现有 InTouch 应用程序中的 SQLConnect 函数在 *ConnectString* 参数中没有 “供应器”关键字，因此 ADO 会使用缺省的供应器，即 Microsoft OLE DB Provider for ODBC，也就是 MSDASQL.DLL。尽管现有的 InTouch 应用程序仍可继续工作，但还是强烈建议您将 *ConnectString* 参数更改为使用自身的 OLE DB 供应器。ConnectString 的示例包括下列内容：

示例 1

Microsoft OLE DB Provider for Microsoft Jet （建议使用）

"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:\DBName.mdb;User ID=UserIDStr;Password=PasswordStr;"

Microsoft.Jet.OLEDB.4.0 是 Microsoft Jet （Microsoft Access 数据库引擎）自身的 OLE DB Provider。

示例 2

Microsoft OLE DB Provider for ODBC （使用缺省供应器 MSDASQL for MS Access）：

"Provider=MSDASQL;DSN=DSNStr;UID=UserName;PWD=PasswordStr;"

备注 User ID 与 uid 可互换使用， Password 与 pwd 也可互换使用。不过，如上所述，建议 ConnectString 参数使用 Microsoft.Jet.OLEDB.4.0。

示例 3

Microsoft OLE DB Provider for SQL Server （建议使用）

"provider=sqloledb;Data Source=MyServer;Initial Catalog=MyDB;User Id=sa;Password=;"

SQL Server 的 OLE DB 供应器为 sqloledb。

示例 4

Microsoft OLE DB Provider for SQL Server （建议使用）

"Provider=SQLOLEDB;uid=sa;pwd=;Database=MyDB"

示例 5

Microsoft OLE DB Provider for ODBC （使用缺省供应器 MSDASQL for SQL Server）：

"DSN=Pubs;UID=sa;PWD=;"

示例 6

Microsoft OLE DB Provider for ODBC （使用缺省供应器 MSDASQL for SQL Server）：

"Data Source=Pubs;User ID=sa;" "Password=;"

备注 Data Source 与 Server 可以互换使用， Initial Catalog 与 Database 可以互换使用。

示例 7

Microsoft OLE DB Provider for Oracle （推荐使用）

"Provider=MSDAORA;Data Source=ServerName;User ID=UserIDStr; Password=PasswordStr;"

如果 win.ini 文件 [InTouch] 节下定义了 SQLTrace=1，则每次成功执行 SQLConnect 均会将 ADO 的版本信息、供应器及数据库系统记录到 Wonderware Logger。

SQLCreateTable(ConnectionId, TableName, TemplateName)

使用 TemplateName 创建一个名称为 TableName 的表。

SQLDelete(ConnectionId, TableName, WhereExpr)

从 *TableName* 中删除符合 *WhereExpr* 子句的行。

SQLDisconnect(ConnectionId)

从数据库断开连接，并清理 SQLPrepareStatement 与 SQLInsertPrepare 已创建但尚未释放 （通过执行 SQLClearStatement 与 SQLInsertEnd）的所有资源。

SQLDropTable(ConnectionId, TableName)

从数据库删除名称为 TableName 的表。

SQLEnd(ConnectionId)

清理使用 ConnectionId 与逻辑表关联的资源。

SQLErrorMsg(ResultCode)

只要数据库供应器生成错误，便会返回值为 -1 的 *ResultCode*。返回的 *ResultCode* 总是 -1，但消息则完全从供应器复制而来。

如需 “结果码”列表及错误消息的描述，请参阅第 5 章 “疑难排解”。

SQLExecute(ConnectionId, BindList, StatementId)

执行与 *StatementId* （MS Access 查询、 MS SQL Server 存储过程，或文本型 SQL 语句）关联的语句。 *BindList* 参数可以是长度为零的字符串。如果 *StatementId* 与返回行的查询关联，则使用 SQLExecute 返回的结果更新逻辑表。若指定了实数型绑定列表，则结果会与 *BindList* 关联。预先知道 *StatementId* 与返回行的查询无关时，长度为零的 BindList 就非常有用。

SQLFirst(ConnectionId)

转至逻辑表的第一行，获取该行的值并将其放入 InTouch 标记。

SQLGetRecord(ConnectionId, RecordNumber)

转至逻辑表中行号为 *RecordNumber* 的行，获取该行的值并将其放入 InTouch 标记。

SQLInsert(ConnectionId, TableName, BindList)

使用 InTouch 标记的当前值将一行插入 *TableName*。

SQLInsertEnd(ConnectionId, StatementId)

清理与 SQLInsertPrepare 创建的 *StatementId* 关联的资源。

SQLInsertExecute(ConnectionId, BindList, StatementId)

使用 InTouch 标记的当前值在前一个 SQLInsertPrepare 确定的表格中插入一行。如果 *BindList* 包含 MS SQL Server 表的 “识别”关键字段，则有必要在运行 SQLInsertExecute 之前设置 IDENTITY_INSERT 选项。

示例

使用属 BindList 一部分的识别码插入一行：

```
ResultCode = SQLSetStatement(ConnectionId, "SET
   IDENTITY_INSERT Products ON");
ResultCode = SQLExecute(ConnectionId, "", 0);
ResultCode = SQLInsertPrepare(ConnectionId, TableName,
   Bindlist, StatementId);
ResultCode = SQLInsertExecute(ConnectionId, Bindlist,
   StatementId);
ResultCode = SQLInsertEnd(ConnectionId, StatementId);
```

SQLInsertPrepare(ConnectionId, TableName, BindList, StatementId)

返回要在 SQLInsertExecute 与 SQLInsertEnd 中使用的 *StatementId*。

SQLLast(ConnectionId)

转至逻辑表最后一行，获取该行的值并将其放入 InTouch 标记。

SQLLoadStatement(ConnectionId, FileName)

将文件 *FileName* 中包含的语句加载到 *ConnectionId* 的缺省语句。

SQLManageDSN(ConnectionId)

ConnectionId 未使用。保留它是为了与旧版 SQL Access 保持后向兼容性。因此，可以给该函数传递任何数字。在调用此函数之前，不需要建立数据库连接。

示例

```
SQLManageDSN( 0 )
```

SQLNext(ConnectionId)

转至逻辑表的下一行，获取该行的值并将其放入 InTouch 标记。

SQLNumRows(ConnectionId)

返回逻辑表的行数。因为此函数会返回错误码，此函数的建议用法如下：

```
DIM TEMP AS INTEGER;

TEMP = SQLNumRows(ConnectionId);

IF (TEMP >= 0) THEN
   RowCount = TEMP;

ELSE
   ResultCode = TEMP;

ENDIF;
```

定义

缺省语句是与连接 ID 关联的语句。它可以是文本型 SQL 语句（SELECT、INSERT、DELETE 或 UPDATE）、MS Access（不论有无参数）中队列的名称，或 MS SQL Server （不论有无参数）中存储过程的名称。缺省语句可由 SQLLoadStatement、 SQLSetStatement 及 SQLAppendStatement 修改，且由 SQLExecute 使用，只要指定了 StatementId = 0。

SQLPrepareStatement(ConnectionId, StatementId)

预备缺省语句，并返回 *StatementId* （1、 2、 3 等等）。对于其参数须使用 SQLSetParam*{Type}* 函数进行设置的语句，此种预备工作非常有用。在较早版本的 SQL Access 中，*SQLHandle* 被指定为此函数的第二参数；不过，对于所有函数，目前版本的 SQL Access 已将 SQLHandle 重命名为 StatementId。此函数的行为仍保持不变。

SQLPrev(ConnectionId)

转至逻辑表的上一行，获取该行的值并将其放入 InTouch 标记。

SQLRollback(ConnectionId)

回滚上一个 SQLTransact 创建的事务。

SQLSelect(ConnectionId, TableName, BindList, WhereExpr, OrderByExpr)

指示数据库从表格中检索信息。执行 **SQLSelect()** 函数时，内存中会创建一个临时 “结果表”，其中包含的记录可使用 **SQLFirst()**、**SQLLast()**、 **SQLNext()**、 **SQLNumRows** 及 **SQLPrev()** 进行浏览。

执行语句：

```
SELECT FROM TableName WHERE WhereExpr ORDER BY OrderByExpr
```

如果语句执行成功，则会创建一个临时记录集 （亦称作逻辑表），在预备 SQLFirst、 SQLPrev、 SQLNext、 SQLLast 及 SQLNumRows 时，*BindList* 会被用于关联 InTouch 标记与此表的各个列。即便不包含任何行，此逻辑表仍然有效。例如，在对于所有记录 *WhereExpr* 均为 False 时。

SQLSetParamChar(StatementId, ParameterNumber, Value, Length)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为字符串值 （字符串可以是单个字符）。此函数的最后一个参数指定参数的最大长度。若 *Value* 的长度超过指定的长度， *Value* 将会被截短为指定长度。若指定长度为 0，则可以使用 *Value* 的整个长度。

SQLSetParamDate(StatementId, ParameterNumber, Value)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为日期值。时间被视为 12:00:00 AM （指定日期的开始时间）。

SQLSetParamDateTime(StatementId, ParameterNumber, Value, Precision)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为日期 / 时间值。

SQLSetParamDecimal(StatementId, ParameterNumber, Value, Precision, Scale)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为十进制值。 *Value* 可以是代表十进制数 (123.456) 的字符串 （或 InTouch 消息标记），也可以是数值 （或 InTouch 内存实型标记）。建议使用消息型标记而不是实型标记，以确保参数的精度。不过，如果 *Value* 必须是一个浮点数 （例如，从 I/O 服务器收到的实数值），则该函数仍会继续运行，但由于浮点表示法的限制，无法确保取得很高精度。 *Precision* 是数字中数值的位数， *Scale* 是小数点右边的位数。

SQLSetParamFloat(StatementId, ParameterNumber, Value)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为 64 位带正负号的浮点值。

SQLSetParamInt(StatementId, ParameterNumber, Value)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为 16 位带正负号的整数值。

SQLSetParamLong(StatementId, ParameterNumber, Value)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为 32 位带正负号的整数值。

SQLSetParamNull(StatementId, ParameterNumber, Type, Precision, Scale)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为 NULL。

Type 参数可使用下列值：

0: 字符串

1: 日期 / 时间

2: 整数

3: 浮点

4: 十进制数

与 NULL 值的比较是由 MS SQL Server 中的 ANSI_NULLS 选项控制的。解译此选项的时间取决于数据库系统。在 SQL Server 7.0 中，创建对象时 （不是执行查询时）会解译此选项。在 SQL Server 7.0 中创建存储过程时，缺省情况下，此选项为 ON，因此 "WHERE MyField = NULL" 之类的子句总是返回 NULL (FALSE)，并且使用此子句的 SELECT 语句不会返回任何行。要让 comparison = 或 <> 返回 TRUE 或 FALSE，在创建存储过程时，必须将该选项设置为 OFF。如果 ANSI_NULLS 未设置为 OFF，则 SQLSetParamNull 无法按预期情况执行。在这种情况下，与 NULL 值进行比较应使用语法 "WHERE MyField IS NULL" 或 "WHERE MyField IS NOT NULL"。

示例

使用 SQLSetParamNull 返回 Products 表中 ProductName 不为 NULL 的所有行。

假设使用下段文字在 SQL Server 中创建了一个存储过程。

```
SET ANSI_NULLS  OFF
GO
CREATE PROCEDURE sp_TestNotNull @ProductParam varchar(255)
AS SELECT * FROM Products WHERE ProductName <>
   @ProductParam
GO
SET ANSI_NULLS  ON
GO
```

InTouch 可执行下列 SQL Access 脚本。

```
ResultCode = SQLSetStatement(ConnectionId,
    "sp_TestNotNull");
ResultCode = SQLPrepareStatement(ConnectionId,
    StatementId);
ResultCode = SQLSetParamNull(StatementId, 1, 0, 0, 0);
ResultCode = SQLExecute(ConnectionId, BindList,
    StatementId);
ResultCode = SQLFirst(ConnectionId);
ResultCode = SQLClearStatement(ConnectionId, StatementId);
```

SQLSetParamTime(StatementId, ParameterNumber, Value)

将与 *StatementId* 关联的参数 *ParameterNumber* 设置为时间值。系统当前日期将与指定的时间一起使用。

SQLSetStatement(ConnectionId, SQLStatement)

将语句 SQLStatement 设置到 ConnectionId 的缺省 SQL 语句中。

SQLTransact(ConnectionId)

开始一项数据库事务。只要数据库系统底层的 OLE DB 供应器支持，事务可以嵌套。例如， Microsoft Jet 自身的 OLE DB 供应器支持多达五层的事务嵌套，包括第一层和最后一层事务。

SQLUpdate(ConnectionId, TableName, BindList, WhereExpr)

使用 InTouch 标记的当前值更新 *TableName* 表中符合 *WhereExpr* 子句的所有行。

SQLUpdateCurrent(ConnectionId)

使用通过绑定列表 （在 SQLSelect 或 SQLExecute 中指定的）映射到表字段的 InTouch 标记更新逻辑表的当前行。如果有其它行与当前行相同，则它们全部都会更新。如果 SQL Access 中有太多相同的行需要更新，则此函数可能会在更新一定数量的行之后返回一个错误。该错误消息会类似于 “Microsoft 游标引擎：关键列信息不足或不正确。受更新影响的行太多”。一次最多可修改 54 个相同的行。

要避免发生此种情形，请在表中创建一个独特的关键字段，这样就不会再有相同的行。强烈建议您让 SQL Access 使用的所有表均有一个唯一的键。对于没有键的表，建议将类型为 AutoNumber (MS Access) 的字段或 Identity (SQL Server) 行的整数字段用作主键，这样 SQLUpdateCurrent 仅会影响一行。此主键字段并非必须包含到 BindList 中。

SQL 参数

下面介绍每个 SQL 函数所需的参数。在脚本中输入用英文引号括起的参数时 ("Parameter1") 时，会使用整个字符串。如果未使用英文引号，则 Parameter1 会被认定为标记名，系统会从 Intouch 标记名字典访问值为 Parameter1 的标记名。

示例

```
"c:\main\file" vs. Location
```

其中： location 是 InTouch 消息标记名。

"c:\main\file" 是字符串

大多数 SQL 函数的参数会是下列一项或多项：

BindList

对应 SQL.DEF 文件中的一个 “绑定列表”名称。

ConnectionID

用户创建的内存整型标记名，用以存储由 SQLConnect 函数指定给每个数据库连接的编号 (ID)。

ConnectString

确定在 **SQLConnect()** 中使用的数据库及任何附加登录信息的字符串。

ErrorMsg

包含错误消息文本描述的消息变量。

如需有关错误消息描述的详细信息，请参阅第 5 章 “疑难排解”。

FileName

包含信息的文件的名称。

MaxLen

此参数关联的列的最大尺寸。此设置确定参数是可变字符型还是长可变字符型。如果 *MaxLen* 小于或等于数据库允许的最大字符串，则此参数为可变字符型。如果大于该数值，则长可变字符型。

OrderByExpression

定义列和排序方向。只能使用列名进行排序。表达式的格式必须为：

ColumnName [ASC|DESC]

要使用列名 （如 manager）按升序方式给所选表格排序：

"manager ASC"

要按多个列进行排序，则表达式的格式为：

ColumnName [ASC|DESC],

ColumnName [ASC|DESC]

要按某个列名 （如 temperature）的升序、另一个列名 （如 time）的降序给所选表格排序：

"temperature ASC, time DESC"

ParameterNumber

语句中实际参数的数目。

ParameterType

指定参数的数据类型。有效值：

类型	描述
Char	可填空格的固定长度字符串
Var Char	可变长度的字符串
Decimal	BCD 数目
Integer	4 字节带符号整数
Small integer	2 字节带符号整数
Float	4 字节浮点数
Double Precision Float	8 字节浮点数
DateTime	8 字节日期时间值
Date	4 字节日期时间值

类型	描述
Time	4 字节日期时间值
无类型	无数据类型

ParameterValue

要设置的实际值。

Precision

小数值的精度、字符的最大长度，或日期时间值的字节数。

RecordNumber

要检索的实际记录号。

ResultCode

大多数 SQL 函数返回的整型变量。如果函数执行成功，返回的 *ResultCode* 为零 (0)；如果失败，则返回负整数。

如需有关详细信息，请参阅第 5 章 “疑难排解”。

Scale

十进制值的小数位数。仅当适用于被设置为空的参数时，才需要使用此值。

StatementId

使用高级功能语句时， SQL 返回供内部使用的 *StatementId*。

SQLStatement

实际的语句，例如：

```
ResultCode=SQLSetStatement(ConnectionID,"Select LotNo,
   LotName from LotInfo");
```

TableName

要访问的数据库表格的名称。

TemplateName

要使用的模板定义的名称。

WhereExpression

给表的任意一行定义既可为真、亦可为假的条件。此命令只从表中提取该条件为真的行。表达式必须采用下列格式：
ColumnName *comparison_operator* expression

备注 如果列的数据类型为字符，则必须用英文单引号括起表达式。

如下示例将选择所有列名包含 EmployeeID 值的行：

```
name='EmployeeID'
```

下例将选择包含部件号 100 到 199 的所有行：

```
partno>=100 and partno<200
```

下例将选择其 temperature 列包含的值大于 350 的所有行：

```
temperature>350
```

在 InTouch QuickScript 中使用 SQL 函数

通过在 QuickScript 编辑器对话框中单击**附件**按钮，可自动在 InTouch QuickScripts 中插入 SQL 函数。 SQL 函数会自动插入脚本中当前的光标位置。

如需有关 InTouch QuickScript 的详细信息，请参阅 *InTouch 用户指南*第 6 章 “在 InTouch 中创建 QuickScript”。

指定复合查询

通过使用 SQL Access Manager，您可以指定自己设计的复合查询与 SQL 语句。这些查询既可动态建立，亦可包含到一个外部文件。此外，这些查询可包含运行时需要 “传递”给查询的参数。随后，必须执行这些查询并可能会返回结果集。 SQL Access Manager API 可供执行数据库能处理的任何 SQL 语句，并检索该查询的结果。您还可执行存储过程这种衍生产品。（对存储过程的支持并不完整）。
如需有关存储过程的详细信息，请参阅 “支持存储过程”。

动态建立查询

要动态建立查询，需要使用两个附加函数：**SQLSetStatement()** 与 **SQLAppendStatement()**。 **SQLSetStatement()** 用于开始一条新的 SQL 语句。这可以是任何有效的 SQL 语句，包括存储过程的名称。由于 InTouch 最多只支持 131 个字符的字符串，因而提供了 **SQLAppendStatement()**，用以将附加字符串串连至语句。

备注 **粗体**文本指的是 SQL 查询语言的命令。

示例

```
ResultCode = SQLSetStatement (ConnectionID, "Select LotNo,
   LotName, LotDescription, LotQuantity from LotInfo,
   ProductionInfo");

ResultCode = SQLAppendStatement (ConnectionID, " where
   LotInfo.LotNo = ProductionInfo.LotNo");

ResultCode = SQLAppendStatement (ConnectionID, " order by
   LotNo,NotName,LotQuantity");
```

现在便可以执行该语句了。

备注 许多数据库列名与表名区分大小写。为了让以上脚本可正常工作，输入的列名与数据库名必须与数据库表格中的完全相同。

从文件读取 SQL 语句

您可以在其它软件包 （如 Microsoft Access 及其它第三方数据库工具）中建立查询，然后使用 InTouch 的 SQL Access 执行查询。因为这些软件包中有多个可生成 SQL 语句，您可以通过使用 **SQLLoadStatement()** 调用该 SQL 语句，并将其存储到文件，处理非常方便。

示例

```
ResultCode = SQLLoadStatement (ConnectionID,
   "c:\myappdir\lotquery.sql");
```

现在便可以执行该语句了。

修改扩展 SQL 语句

为提供完整的 SQL 功能， SQL Access Manager 允许您指定包含 InTouch 标记名值的 where 子句。为了能够在运行时指定 SQL 参数，系统提供了下列函数：

- SQLPrepareStatement()
- SQLSetParam*Type*()
- SQLClearStatement()
- SQLClearParam()

要在 SQL 语句上执行参数替换，请在需要以后指定参数的 SQL 语句位置放一个 "?" 号。在 “预备”语句并在其中 “设置”参数之后，便可以执行该语句了。

SQLPrepareStatement() 可预备语句以便执行。它并不执行语句，而只是激活语句以便您设置参数值。 **SQLSetParamType()** 是供您设置 SQL 语句参数值的函数集。

示例

```
ResultCode = SQLSetStatement (ConnectionID, "Select LotNo,
   LotName, LotDescription, LotQuantity from LotInfo,
   ProductionInfo");

ResultCode = SQLAppendStatement (ConnectionID, " where
   LotInfo.LotNo = ?");

ResultCode = SQLAppendStatement (ConnectionID, " order by
   LotNo,NotName,LotQuantity");

ResultCode = SQLPrepareStatement (ConnectionID,
   StatementId);

{ 将语句的句柄返回给标记 'StatementId'}

ResultCode = SQLSetParamInt (StatementId, 1,
   tagLotNumber);

{ 将 tagLotNumber 值放入 param}
```

由于该语句只有一个参数，因而现在便可以执行了。

执行语句并完成语句的预备工作之后，即可调用 **SQLClearStatement()** 来释放与该语句关联的资源。

备注 调用 **SQLEnd()** 释放 “未命名的” SQL 语句（由现有 SQL Access 函数生成的那些语句），以及由 **SQLSetStatement()** 和 **SQLLoadStatement()** 创建但尚未预备的语句。

执行扩展 SQL 语句

在动态建立语句或从文件读取语句，并且已预备并修改它 （属可选项）之后，现在便可以执行语句了。 SQL Access Manager API 使用 **SQLExecute()** 函数来完成此项工作。 **SQLExecute()** 将执行当前的活动语句 （即为 **SQLSetStatement()** 或 **SQLLoadStatement()** 创建的语句），或执行以前预备且由语句句柄参数指定的语句。

示例 1

```
ResultCode = SQLLoadStatement (ConnectionID,
   "c:\myappdir\lotquery.sql");

ResultCode = SQLExecute (ConnectionID, "BindList", 0);

{ 将 select 的结果放入 BindList 中指定的标记。预备的语句的句柄为零
   }

ResultCode = SQLNext (ConnectionID);

{ 获取 Select 的结果 }
```

示例 2

```
ResultCode = SQLSetStatement (ConnectionID, "Select LotNo,
   LotName, LotDescription, LotQuantity from LotInfo,
   ProductionInfo");

ResultCode = SQLAppendStatement (ConnectionID, " where
   LotInfo.LotNo = ?");

{ 问号表示以后再处理 }

ResultCode = SQLAppendStatement (ConnectionID, " order by
   LotNo,NotName,LotQuantity");

ResultCode = SQLPrepareStatement (ConnectionID,
   StatementId);

{ 将语句的句柄返回给标记 'StatementId'}

ResultCode = SQLSetParamInt (StatementId, 1,
   tagLotNumber);

{ 将 tagLotNumber 值放入 param}

ResultCode = SQLExecute (ConnectionID, "BindList",
   StatementId); { 将 Select 的结果置入标记，

该标记在 BindList 中指定，准备好的语句句柄在 StatementId 中 }

ResultCode = SQLNext (ConnectionID);

{ 获取 Select 的结果 }
```

示例 3

SQLSetStatement – 此语句必须用于复合查询及超过 131 个字符的字符串表达式。字符串表达式超过 131 个字符时，请使用 SQLAppend

```
SQLSetStatement(ConnectionID, "Select  Speed, Ser_No from
   tablename where Ser_No ='" +   Serial_input  + "'");

SQLExecute(ConnectionID, "BindList", 0);
```

在上例中， StatementId 设置为零，这样在执行语句之前就不必调用 SQLPepare(Connection_Id, StatementId)。因为 StatementId 并非由 SQLPepare 建立，若要正确结束此 select 语句，请使用 SQLEnd 函数，而不是 SQLClearStatement() 函数。

```
SQLSetStatement(Connection_Id, "Select  Speed, Ser_No from
   tablename where Ser_No ='" +   Serial_input  + "'");

SQLPrepareStatement(Connection_Id, StatementId);

SQLExecute(Connection_Id, StatementId);
```

在上例中， StatementId 是由 SqlPrepareStatement 创建的，并在 SQLExecute 函数中使用。要结束此 select 语句，请使用 SQLClearStatement 释放资源并释放 StatementId。

支持存储过程

SQLExecute() 函数支持执行有些存储过程。例如，假设您在名称为 "LotInfoProc" 的数据库服务器上创建了一个存储过程，它包含下列 Select 语句："Select LotNo,LotName from LotInfo"。您需要编写下列 InTouch QuickScript 以执行该过程并获取结果：

使用 Microsoft SQL Server

```
ResultCode = SQLSetStatement (ConnectionID,
   "LotInfoProc");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
{ 获取 Select 的结果 }
```

使用 Oracle 或 Microsoft Access

```
ResultCode = SQLSetStatement (ConnectionID, "{CALL
   LotInfoProc}");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
{ 获取 Select 的结果 }
```

将值放入 InTouch 标记

通过使用 SQLFirst、SQLPrev、SQLNext、SQLLast 及 SQLGetRecord 这五个脚本函数，您可以浏览逻辑表中的各行，获取字段值并将其放入 InTouch 标记。如果字段为 NULL，则根据该标记是模拟型还是消息型，相关的 InTouch 标记值会为 0，或长度为 0 的字符串。如果数据库中的字符串长度大于 131 个字符，则只有头 131 个字符会被复制到相关的 InTouch 消息标记中。

让 InTouch 标记与数据库字段值保持一致

通过使用 SQLUpdate、 SQLUpdateCurrent、 SQLInsert 及 SQLInsertExecute 这四个脚本函数，您可以使用 InTouch 标记值更新表，或将它插入表中。如果 InTouch 消息标记超过表中相应文本字段所定义的大小，则消息标记中使用的字符数会被用于定义该字段的大小。因为 InTouch 标记不能为 NULL，若 BindList 包含字段，则不可能使用这些函数执行更新，或将 NULL 值插入数据库。要将 NULL 值插入某个字段，必须在不包含此字段的 INSERT 语句上使用 SQLExecute，该语句应定义为可接受 NULL 值。

数据更新规则的含义

获取值并将其放入 InTouch 标记且保持表字段中的数据与标记值一致的规则意味着，在下列情况下，可能会不经意修改表中的值。

无意中将 NULL 值转换为零或空字符串

执行浏览函数之一可将 NULL 值当作 0 或零长度字符串 （例如， Tag1）放入 InTouch 标记。在更新 BindList 中的一些标记之后，执行 SQLUpdateCurrent 可将 0 或零长度字符串返回表中，覆盖与 Tag1 关联的 NULL 值。执行 SQLUpdate 会更新使用来自 Tag1 （非 NULL 值）的 0 或零长度字符串的行。

无意中将 0 或空字符串插入表中

执行浏览函数之一可将 NULL 值当作 0 或零长度字符串 （例如， Tag1）放入 InTouch 标记。在更新 BindList 中的一些其他之后，执行 SQLInsert 或 SQLInsertExecute 可将 0 或零长度字符串 （Tag1 中）返回表中 （非 NULL 值）。

第 5 章

疑难排解

本章介绍如何使用 SQL 函数返回的 “结果码” 来排解 SQL 应用程序中的疑难杂症。第一节介绍 **SQLErrorMsg()** 函数，并列出 “SQL 结果码” 表及其对应的 “错误消息”。第二节包含列出特定数据库 “错误消息” 的表格。

目录

- 函数疑难排解
- 特定的数据库错误消息
- 调试 SQL Access

函数疑难排解

所有的 “SQL 函数” 均会返回一个可用于排解疑难的*结果码*。**SQLErrorMsg()** 函数会返回与*结果码*关联的 “错误消息”。

示例

```
ErrorMsg=SQLErrorMsg(ResultCode);
```

其中：

ErrorMsg 是内存消息标记。

ResultCode 是先前 SQL 函数返回的整型值。

结果码错误消息

对于此处未记载的 “结果码”，请参阅特定数据库的文档，并务必请查看 Wonderware Logger 以获取所有附加信息。

SQLErrorMsg() 函数将设置 InTouch 消息标记名 *ErrorMsg* 的值。下面列出可能出现的一些 SQL “结果码”及其相应的错误消息与描述：

结果码	错误消息	描述
0	未发生错误	命令成功执行
-1	< 来自 DB 供应器的消息 >	< 来自 DB 供应器的特定错误消息 >
-2	无法执行空语句	SQLExecute(ConnectionId, BindList, 0) 执行前未调用含非空语句的 SQLSetStatement 或 SQLLoadStatement。
-4	返回值为 Null	从数据库读取的整型或实型值为 null。这只是发送给 Wonderware Logger 的一个警告。
-5	无更多可获取的行	已到达表中最后一条记录
-7	无效的参数 ID	调用了 SQLSetParamI{Type}，但参数 ID 无效。
-8	无效的参数列表	无效参数列表的示例：1, 2, 3, 5 （缺 4）。
-9	NULL 参数的类型无效	调用了 SQLSetParamNull，但类型无效。
-10	不允许更改参数的数据类型	调用了 SQLSetParam *{Type}*，但使用的类型与现有参数不同。
-11	不允许在成功执行语句之后添加参数。	在成功执行语句之后，使用新参数 ID 调用 SQLSetParam *{Type}*。
-12	无效的日期时间格式	遇到无效的日期时间格式，例如在执行 SQLSetParamTime、 SQLInsertExecute 或 SQLUpdateCurrent 时。
-13	无效的十进制格式	遇到无效的十进制格式，例如在执行 SQLSetParamDecimal、 SQLInsertExecute 或 SQLUpdateCurrent 时。
-14	无效的货币格式	遇到无效的货币格式，例如在执行 SQLInsertExecute 或 SQLUpdateCurrent 时。
-15	此操作的语句类型无效	使用 SQLPrepareStatement 创建的语句 ID 调用 SQLInsertEnd，或使用 SQLInsertPrepare 创建的语句 ID 调用 SQLClearStatement。

结果码	错误消息	描述
-1001	内存不足	没有足够的内存执行此函数。
-1002	无效的连接	传递给函数的 ConnectionId 无效。
-1003	未找到绑定列表	指定的 “绑定列表”名称不存在。
-1004	未找到模板	指定的 “表格模板”不存在。
-1005	内部错误	出现内部错误。请致电 “技术支持部门”。
-1006	字符串为空	警告 - 从数据库读取的字符串为空。这只是发送给 Wonderware Logger 的一个警告。
-1007	字符串被截短	警告 - 从数据库读取的字符串超过 131 个字符，它在 select 语句中会被截短。此警告会发送给 Wonderware Logger。
-1008	无 Where 子句	Delete 语句中没有 Where 子句。
-1009	连接失败	如需有关连接失败的详细信息，请查看 Wonderware Logger。
-1010	连接字符串的 DB= 部分指定的数据库不存在	指定的数据库不存在。
-1011	未选定行	未先执行 SQLSelect() 或 SQLExecute 命令，就试图调用 SQLNumRows()、 SQLFirst()、SQLNext()、 SQLLast 或 SQLPrev() 命令。
-1013	未找到要加载的文件	无法找到调用 SQLLoadStatement 时使用的文件名。

特定的数据库错误消息

Oracle

查看 Oracle Server 文档中关于特定错误消息及解决方案的详细信息。

Microsoft SQL Server

错误消息	解决方案
一次不能有多条语句处于活动状态	在执行 **SQLSelect()** 之后，尝试执行 SQL 命令。执行 **SQLEnd()** 以便从 **SQLSelect()** 释放系统资源，或在第二条语句中使用单独的 ConnectionId。
没有足够的内存来处理命令	尝试重新启动客户端工作站。
无效的对象名、表名	目前使用的数据库中没有该表名。请尝试使用 DB=database name。

查看 Microsoft SQL Server 文档中关于特定错误消息及其解决方案的详细信息。

调试 SQL Access

在 win.ini 文件 [InTouch] 部分下定义的 SQLTrace=1 标志对于调试 SQL Access 脚本非常有效。新的 SQL Access 模块不使用跟踪文件 sqltrace.txt。

附 录 A

保留关键字

SQL Access 与 ODBC

本附录列出一些关键字，这些关键字不得在 “SQL Access 绑定列表”、“表格模板”及 “开放式数据库连接”（Open Database Connectivity，简称 ODBC）接口中使用。

如果某个保留关键字被用作 “绑定列表”或 “表格模板”中的 “列名”，则 Wonderware Logger 中会产生一条错误消息。所产生的错误的类型取决于使用的 ODBC 驱动程序及发现该关键字的位置。例如，最常见的错误之一是在 “绑定列表”与 “表格模板”中将 DATE 与 TIME 用作 “列名”。为避免出现此类错误，请使用稍微有些不同的名称，例如， "aDATE" 与 "aTIME"。

这些保留关键字定义 InTouch SQL Access 所使用的 “结构化查询语言”（Structured Query Language，简称 SQL）。使用的特定 ODBC 驱动程序也可以识别这些关键字。 SQL Access 将包含一个或多个保留关键字的 SQL 命令传递给 ODBC.DLL 文件。如果无法正确解译 SQL 命令， SQL Access 会在 Wonderware Logger 中生成一条错误消息。

SQL Access 与 ODBC 的保留关键字按字母顺序排列如下：

ABSOLUTE
ADA
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BEGIN
BETWEEN
BIT
BIT_LENGTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHAR_LENGTH
CHARACTER
CHARACTER_LENGTH
CHECK
CLOSE COALESCE
COBOL
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURSOR
DATE
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFERRABLE

DEFERRED
ENTF
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DICTIONARY
DISCONNECT
DISPLACEMENT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
ENDEESC
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
EXTRACT
FALSE
FETCH
FIRST
FLOAT
FOR FOREIGN
FORTRAN
FOUND
FROM FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IGNORE
IMMEDIATE
IN
INCLUDE
INDEX
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
EINFUEN
INTEGER
INTERSECT
INTERVALL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LEFT
LEVEL
LIKE
LOCAL
LOWER
MATCH
MAX
MIN
MINUTE
MODULE
MONTH
MUMPS
NAMES
NATIONAL
NCHAR
NEXT
NONE
NOT
NULL
NULLIF
NUMERIC
OCTET_LENGTH
OF
OFF
ON
ONLY
OPEN
OPRN
OPTION
OR
ORDER
OUTER
OUTPUT
OVERLAPS
PARTIAL
PASCAL
PLI
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROWS
SCHEMA
SCROLL
SECOND
SECTION
SELECT
SEQUENCE
SET
SIZE
SMALLINT
SOME
SQL
SQLCA
SQLCODE
SQLERROR
SQLSTATE
SQLWARNING
SUBSTRING
SUM
SYSTEM
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINU
TO
TRANSACTION
TRANSLATE
TRANSLATION
TRUE
UNION
UNIQUE
UNKNOWN
UPDATE
UPPER
USAGE
USING
WERT
VALUES
VARCHAR
VARING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
YEAR

InTouch

下面是 InTouch 的保留关键字：

As
Call
Dim
Discrete
Integer
Message
Real
Return
RetVal

索引

B

BindListName 23, 34
绑定列表 16
  标记浏览器 17
  删除 19
  新建 16
  修改 18
保留关键字 47
表格模板 20
  删除 23
  新建 21
  修改 22
表格模板名称 21, 24
表格模板命令 22
标记浏览器 17

C

ConnectionID 34
ConnectString 34
CSV 5, 23
参数 34
  BindListName 34
  ConnectionID 34
  ConnectString 34
  ErrorMsg 34
  MaxLen 35
  OrderByExpression 35
  ParameterNumber 35
  ParameterType 35
  ParameterValue 36
  Precision 36
  RecordNumber 36
  ResultCode 36
  Scale 36
  SQLStatement 36
  StatementId 36
  TableName 36
  TemplateName 36
  WhereExpression 37
  文件名 34
查询
  动态建立 38
  复合 38
从文件读取 SQL 语句 38

D

Delim 函数 19
动态建立查询 38

E

ErrorMsg 34

F

分隔符 19

G

关于本手册 6

H

函数
  SQLAppendStatement 25
  SQLClearParam 26
  SQLClearStatement 26
  SQLClearTable 26
  SQLCommit 26
  SQLConnect 11
  SQLCreateTable 27
  SQLDelete 27
  SQLDisconnect 28
  SQLDropTable 28
  SQLEnd 28
  SQLErrorMsg 28
  SQLExecute 28
  SQLFirst 28
  SQLGetRecord 28
  SQLInsert 28
  SQLInsertEnd 28
  SQLInsertExecute 29
  SQLInsertPrepare 29
  SQLLast 29
  SQLLoadStatement 29
  SQLManageDSN 29
  SQLNumRows 30
  SQLPrepareStatement 30
  SQLPrev 30
  SQLRollback 30
  SQLSelect 30
  SQLSetParamChar 31
  SQLSetParamDate 31
  SQLSetParamDecimal 31
  SQLSetParamFloat 31
  SQLSetParamInt 32
  SQLSetParamLong 32
  SQLSetParamNull 32
  SQLSetParamTime 33
  SQLSetStatement 33
  SQLTransact 33
  SQLUpdate 33
  SQLUpdateCurrent 33
  使用 25

J

结构化查询语言 5
结果码错误消息 43

L

联机手册 6

列名 18

M

MaxLen 35
Microsoft Access
  连接要求 12
  支持的数据类型 12, 13
Microsoft SQL Server
  连接要求 11
  支持的数据类型 12
Mircrosoft SQL Server
  支持的数据类型 13
命令
  表格模板 22

O

ODBC 管理器程序 8
ODBC 兼容性 8
ODBC.INI 8
Oracle
  支持的数据类型 13
OrderByExpression 35

P

ParameterNumber 35
ParameterType 35
ParameterValue 36
Precision 36
配置 SQL Access Manager 15
配置绑定列表 16
配置表格模板 20

Q

QuickScripts 37

R

RecordNumber 36
ResultCode 36, 43

S

Scale 36
SQL Access Manager 简介 5
SQL Access Manager 综述 15
SQL 参数 34
SQL 函数格式 25
SQL 函数疑难排解 43
SQL.DEF 5, 23
SQLConnect 11
SQLErrorMsg 43
SQLInsert 19
SQLSelect 30
SQLStatement 36
SQLUpdate 19, 33
StatementId 36
删除绑定列表 19
删除表格模板 23
使用 Microsoft Access 12
使用 Microsoft SQL Server 11
使用 SQL 函数 25
使用特殊分隔符 19
数据库 5

T

TableName 36
Tagname.FieldName 18
TemplateName 36
特定的数据库错误消息
  Microsoft SQL Server 45

W

WhereExpression 37
Wonderware 技术支持 7
文件名 34

X

修改绑定列表 18
修改表格模板 22
修改扩展 SQL 语句 38

Y

疑难排解 43

Z

在 InTouch 中使用 SQL 函数 37
在 Oracle Date 字段中记录日期与时间 10
支持存储过程 41
支持的数据库 11
  Microsoft Access 12
  Microsoft SQL Server 11
支持的数据类型 13
指定复合查询 37
执行扩展 SQL 语句 39