Wonderware® FactorySuite™

SQL Access Manager

Руководство пользователя

Редакция A

Август 2002

Invensys Systems, Inc.

www.klinkmann.com

KLINKMANN

Finland
P.O. Box 38, FI-00371 Helsinki
Ph. +358 9 540 4940
Fax +358 9 5413 541
automation@klinkmann.fi

Санкт-Петербург	тел. +7 812 327 3752; klinkmann@klinkmann.spb.ru
Москва	тел. +7 095 461 3623; moscow@klinkmann.spb.ru
Tallinn	tel. + 372 668 4500; klinkmann.est@klinkmann.ee
Rīga	tel. +371 738 1617; klinkmann@klinkmann.lv
Vilnius	tel. +370 5 215 1646; post@klinkmann.lt
Київ	тел. +38 044 239 1250; klinkmann@klinkmann.kiev.ua

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

Торговые марки

Содержание

ГЛАВА 1

SQL Access Manager

Wonderware® FactorySuite™ SQL Access Manager (Менеджер SQL-доступа) позволяет создавать, модифицировать, манипулировать содержимым и удалять пользовательские таблицы в базе данных. Информация в базе данных хранится в виде таблиц, которые могут иметь общие атрибуты или поля. SQL (Structured Query Language – язык структурированных запросов) – это язык, позволяющий выполнять необходимые действия над хранимой информацией.

Содержание

Введение

Программа InTouch SQL Access Manager – это средство, облегчающее передачу различной информации, такой как рецепты смесей и т.д. из SQL-базы данных в InTouch-приложение. С её помощью упрощается передача оперативных данных, состояния алармов или архивной информации из InTouch в SQL-базу. Например, после окончания технологического цикла какого-либо оборудования может возникнуть необходимость сохранения определённых наборов данных для различных приложений. SQL-базы обеспечивают возможность обмена информацией между одним или несколькими приложениями. SQL Access Manager обеспечивает выдачу хранимой информации в любое InTouch-приложение.

Продукт InTouch SQL Access Manager состоит из собственно программы SQL Access Manager и SQL-функций. Программа SQL Access Manager используется для создания и определения соответствия столбцов таблиц базы данных переменным Словаря InTouch-приложения. Этот процесс называется "связывание" (binding). Связывание InTouch-переменных со столбцами таблицы позволяет SQL Access Manager напрямую манипулировать информацией в базе данных. Наименования полей базы данных и соответствующих переменных хранятся программой в файле "SQL.DEF" в CSV-формате (Comma Separated Variable – переменные, разделённые запятыми). Этот файл находится в том же каталоге, что и InTouch-приложение, и может просматриваться и модифицироваться как программой SQL Access Manager, так и любым текстовым редактором типа Notepad и т.д. SQL Access Manager создаёт также шаблоны таблиц (Table Templates), определяющие формат и структуру базы данных.

Дополнительную информацию о связывании переменных со столбцами таблиц и Шаблонах таблиц можно найти в Главе 3 настоящего руководства "Конфигурирование SQL Access Manager".

SQL-функции можно использовать в любом исполнительном скрипте InTouch. Эти функции могут автоматически запускаться в момент операторского ввода, при изменении значения переменной или возникновении определённых условий. В частности, при возникновении аварийной ситуации возможен автоматический запуск функций **SQLInsert()** и **SQLUpdate()**, сохраняющих в таблице все необходимые прикладные данные и сведения об аларме. Кроме того, с помощью SQL-функций можно создавать новые таблицы, осуществлять вставку новых строк в таблицу, редактировать существующие строки, чистить и уничтожать таблицы, выбирать и просматривать требуемые строки и т.д.

Примечание. Не упоминаемые в настоящем руководстве базы данных системой не поддерживаются.

О руководстве

Данное руководство состоит из нескольких глав, описывающих определённые аспекты использования SQL Access Manager. Руководство написано в "процедурном" формате – решение каждой задачи приведено в виде последовательности выполнения шагов.

При просмотре данного руководства в онлайновом режиме в некоторых местах будут видны перекрёстные ссылки, отмеченные зелёным цветом текста. Это ссылки на другие разделы и главы руководства, щёлкнув на которых, можно сразу перейти к соответствующей информации. После перехода вы сможете так же просто вернуться обратно, нажав кнопку Back (назад).

Подсказка. "Подсказки" информируют вас о более простом и быстром способе выполнения поставленной задачи.

Подробные сведения о среде разработки InTouch, её инструментах и требованиях можно найти в *Руководства пользователя InTouch*.

В Главе 1 "Программные элементы WindowMaker" *Руководства пользователя InTouch* содержится информация о среде разработки WindowMaker и её инструментальных средствах. Способы работы с окнами, графическими объектами, мастерами, ActiveX-объектами и т.д. приведены в Главе 2 "Использование WindowMaker" этого же документа.

Информацию о среде исполнения InTouch (WindowViewer) вы можете найти в онлайновой справочной системе *Руководство пользователя InTouch Runtime*.

Более подробную информацию о языке скриптов InTouch, о системных переменных, о полях переменных (**.fields**) можно найти в *InTouch Reference Guide*.

Более подробную информацию о дополнительной программе SPC Pro можно найти в *Руководстве пользователя по SPC*.

Более подробную информацию о дополнительной программе Recipe Manager можно найти в *Руководстве пользователя по Recipe Manager (Менеджер рецептов)*.

Пакет FactorySuite содержит справочную информацию в электронном виде для всех компонент пакета.

Примечание. Чтобы просматривать или распечатывать электронные руководства, вы должны установить Adobe Acrobat Reader (версии 4.0 или выше).

Основные предположения

Предполагается, что пользователь имеет некоторые знания в следующих областях:

- Операционные системы Windows 95 и/или Windows NT.
- Принципы использования мыши, меню Windows, выбора вариантов и вызова справки.
- Программирование и макроязык. Желательно разбираться в таких вопросах, как переменные, операторы, функции и методы.

Техническая поддержка

Служба Технической Поддержки компании Wonderware предлагает самые разные варианты помощи по вопросам, связанным с продуктами Wonderware и их использованием.

Прежде чем обращаться в Техническую службу, прочтите соответствующие главы своего *Руководства пользователя SQL Access Manager*, которые могут помочь вам в решении возникшей проблемы. Если оно вам не поможет и вы вынуждены будете обратиться в службу технической поддержки, предоставьте ей следующую информацию:

1. Серийный номер программного обеспечения.
2. Номер версии InTouch.
3. Тип и номер версии операционной системы (например Microsoft Windows NT версии 4.0 для рабочей станции).
4. Точный текст сообщений об ошибках, выданных системой.
5. Все соответствующие распечатки программ Wonderware Logger, Microsoft Diagnostic MSD и других диагностических утилит.
6. Подробное описание попыток разрешения возникших проблем и полученных результатов.
7. Описание, почему возникла данная проблема.
8. При наличии – номер, присвоенный ранее вашей проблеме группой Технической Поддержки Wonderware (если данная проблема ещё не решена).

Дополнительную информацию по Технической Поддержке можно найти в *Руководстве администратора системы FactorySuite.*

Совместимость с ODBC

SQL Access Manager совместим со стандартом ODBC и может взаимодействовать с любой базой данных (при условии, что в системе имеется соответствующий ODBC-драйвер). До использования ODBC-

драйвера его необходимо сконфигурировать при помощи программы Microsoft ODBC Administrator, для того чтобы установить связи между ODBC-приложением и базой данных.

Чтобы сконфигурировать ODBC-драйвер:

1. Запустите программу Microsoft ODBC Administrator.
2. Выберите драйвер или источник данных, нажмите кнопку **Add New Name (добавить новое имя)**, **Set Default (установить параметры по умолчанию)** или **Configure (конфигурировать)**. Появится диалоговое окно **ODBC Driver Setup (параметры ODBC-драйвера).**

Настройка	**Описание**
Data Source Name (имя источника данных)	Описание пользовательского имени, идентифицирующего источник данных.
Description (описание)	Описание источника данных.
Database Directory (каталог базы данных)	Каталог, в котором хранятся файлы базы данных. Если каталог не указан, то подразумевается текущий рабочий каталог.

Подсказка. Введите также всю остальную информацию, необходимую для конфигурирования драйвера.

3. Нажмите **ОК**.

Подсказка. Драйвер записывает введённые значения в файл ODBC.INI. Все эти величины являются параметрами по умолчанию при подключении к определённому источнику данных. Значения по умолчанию могут быть изменены путём модификации полей источника данных. Если какой-либо атрибут отсутствует в диалоговом окне ODBC Driver Setup (определение параметров драйвера), его можно вручную ввести в соответствующую секцию файла ODBC.INI.

ГЛАВА 2

Конфигурирование и подключение к базам данных

SQL Access Manager поддерживает доступ к базам данных, разработанных в Oracle, Microsoft SQL Server и Microsoft Access. Требования со стороны каждой базы конкретны и уникальны. Каждой базе данных посвящён отдельный параграф, в котором содержится информация о её конфигурировании для обеспечения взаимодействия с SQL Access Manager.

Содержание

Поддержка Oracle 8.0

Для взаимодействия с БД Oracle 8.0:

1. Проверьте, что Oracle OLEDB Provider (MSDAORA.DLL) установлен на вашем компьютере. Этот файл устанавливается утилитой MDAC при установке InTouch.
2. Подключитесь к Oracle, выполнив функцию SQLConnect() в InTouch-скрипте действия.

Дополнительную информацию о функции SQLConnect() можно найти в Главе 4 "SQL-функции".

Формат SQLConnect()

Подключение к базам данных Oracle выполняется с помощью функции **SQLConnect()**, имеющей следующий формат:

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

В следующей таблице приведено описание всех атрибутов, используемых Oracle.

Атрибут	Значение
Provider	MSDAORA
User ID	Имя пользователя.

Password	Пароль.
Data Source	Имя машины сервера Oracle.

Пример:

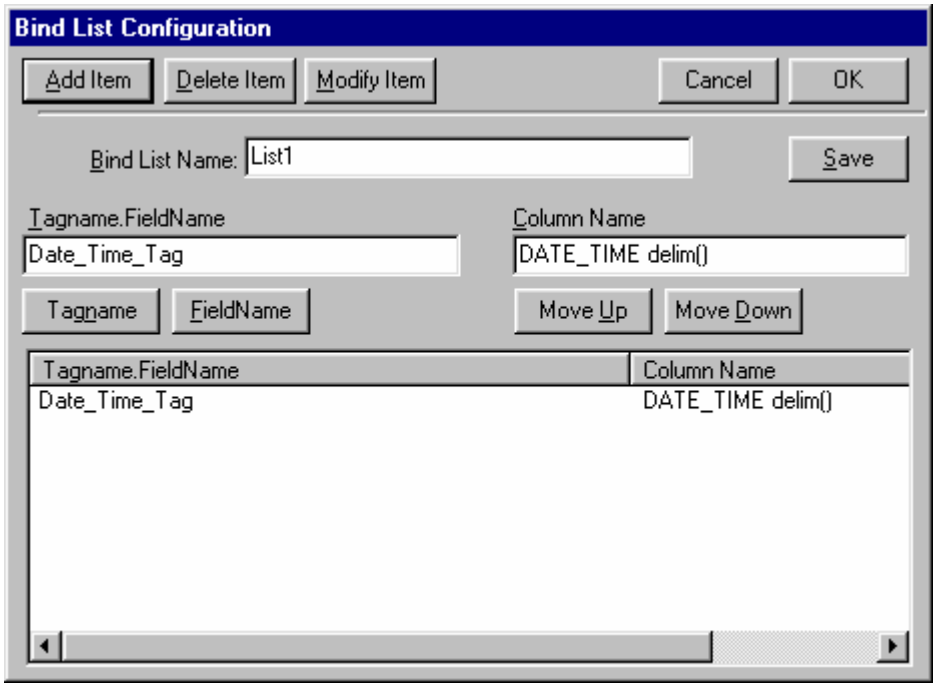
```
SQLConnect(ConnectionId, "Provider=MSDAORA; Data
   Source=OracleServer; User ID=SCOTT;
Password=TIGER;");
```

Запись времени и даты в поля Oracle

Для регистрации времени и даты в полях данных Oracle необходимо сформировать Bind List (список соответствия), используя функцию delim.

Для регистрации даты и времени в поле данных Oracle:

1. В окне Application Explorer в группе **SQL Access Manager** щёлкните дважды на строке **Bind List (список соответствия)**. Появится диалоговое окно **Bind List Configuration (конфигурирование списка соответствия)**:



2. В поле **TagName.FieldName** введите имя используемой переменной.
3. В поле **Column Name (название столбца)** введите обращение к функции **DATE_TIME delim()**.
4. В вашем InTouch-приложении для получения текущей даты и времени суток используйте скрипт. Например.

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

Подсказка. Дата_время_переменная отображаются в следующем формате:

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

Поддержка Microsoft SQL Server

Для взаимодействия с Microsoft SQL Server:

1. Сконфигурируйте клиента базы данных Windows.
2. Подключитесь к Microsoft® SQL Server при помощи функции **SQLConnect() из скрипта InTouch**.

   Дополнительную информацию о функции **SQLConnect()** можно найти в Главе 4 "SQL-функции".

Формат функции SQLConnect()

Подключение к базам данных Microsoft SQL Server выполняется с помощью функции **SQLConnect().** При выполнении этой функции происходит регистрация приложения на сервере базы данных и установление соединения с базой данных, позволяющее выполнять и другие SQL-функции. Формат обращения к **SQLConnect()** следующий:

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

В следующей таблице приведено описание всех атрибутов, используемых Microsoft SQL Server.

Атрибут	**Значение**
Provider	SQLOLEDB
DSN	Название источника данных, указанное в окне Microsoft ODBC Administrator.
UID	Имя пользователя. Регистр вводимых символов имеет значение.
PWD	Пароль. Регистр вводимых символов имеет значение.
SRVR	Имя компьютера-сервера с таблицами базы данных.
DB	Имя базы данных.

Пример:

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

Поддерживаемые типы данных

SQL Access Manager поддерживает четыре типа данных в InTouch (дискретный, целый, вещественный, символьный) с соответствующими типами базах данных. Тип данных **char** соответствует символьным строкам фиксированной длины. InTouch-переменные типа "message" должны иметь тип char. Указание длины поля обязательно. Максимальная длина поля типа char в базах данных Microsoft SQL Server равна 8000 символам. Максимальная длина InTouch-переменных типа "message" равна 131 символу. Если длина InTouch-переменной типа "message" превышает размер поля базы данных, то при занесении символьной строки в базу она будет усечена до указанного значения.

Тип данных int предназначен для InTouch-переменных типа Integer. Если длина поля не указана, то её значение будет равно длине поля по умолчанию для данного типа полей. Если длина указана, то она должна иметь формат Width, где Width – это общее число позиций для столбца.

Вещественные числа соответствуют InTouch-переменным типа Real. Длина поля определяется базой данных. Указание длины поля для данных этого типа не обязательно.

Поддержка Microsoft Access

Для взаимодействия с Microsoft Access необходимо подключиться к ней при помощи функции **SQLConnect()**.

Формат функции SQLConnect()

Подключение к базам данных Microsoft Access выполняется с помощью функции **SQLConnect().** При исполнении этой функции происходит регистрация приложения на сервере базы данных и установление соединения с базой, позволяющее выполнять и другие SQL-функции. Формат обращения к **SQLConnect()** следующий:

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

DSN является атрибутом, используемым Microsoft Access и именем источника данных, определённом в Microsoft ODBC Administrator.

Пример:

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

Длина строки

Конкретные типы, которые SQL Access Manager поддерживает, зависят от версии, используемого ODBC-драйвера. Текстовые данные соответствуют символьным строкам фиксированной длины (для InTouch-переменных типа "message"), при этом указание длины обязательно. Максимальная длина текстовых полей в базах данных Microsoft Access равна 255 символам, длина же InTouch-переменных типа "message" не может превышать 131 символ. Если длина InTouch-переменной типа "message" превышает размер поля базы данных, то при занесении символьной строки в базу она будет усечена до указанного значения. ODBC-драйвер для Microsoft Access поддерживает названия столбцов длиной до 17 символов. Максимально допустимое число столбцов при использовании операторов типа SQLSetStatement(Select Col1, Col2,…) – 40.

Типы данных, поддерживаемые базой данных

Oracle

Тип данных	Длина, символов	Длина по умолчанию, символов	Диапазон значений	Тип InTouch-переменных
char	2 000 символов	1		Message

number	38 цифр	38 цифр		Integer

Microsoft SQL Server

Тип данных	Длина, символов	Длина по умолчанию, символов	Диапазон значений	Тип переменных
char	8000 символов			Message
Int			от –2 147 483 648 до 2 147 483 647	Integer
float	15 цифр		от $-1{,}79E^{+308}$ до $1{,}79E^{+308}$	Real

Microsoft Access 2000

Тип данных	Длина, символов	Длина по умолчанию, символов	Диапазон значений	Тип InTouch-переменных
text	255 символов			Message
number				Integer
number				Real

ГЛАВА 3

Конфигурирование SQL Access Manager

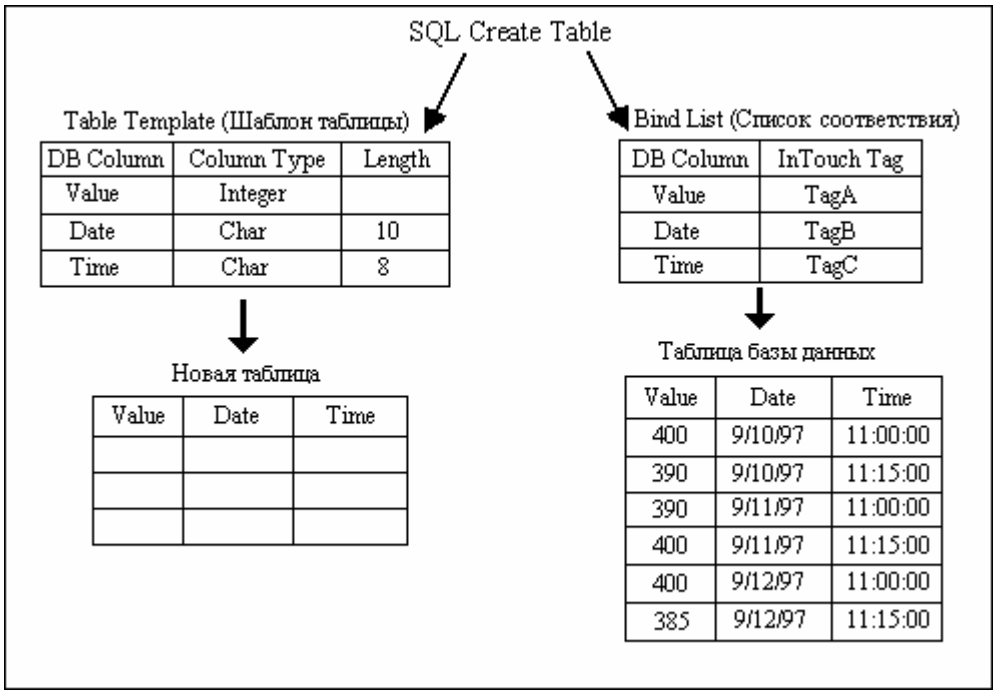
Вспомогательная программа SQL Access Manager создаёт Bind Lists (списки соответствия) и Table Templates (шаблоны таблиц). В Bind Lists определены соответствия между столбцами таблиц базы данных и переменными Словаря InTouch-приложения. Table Templates определяют структуру и формат новых таблиц в базе.

Содержание

Общие сведения об SQL Access Manager

Определение структуры нового файла базы данных происходит в момент выполнения команды **SQLCreateTable()** в InTouch-приложении на основании аргументов Шаблона таблицы.

При выполнении команд **SQLInsert(), SQLSelect()** и **SQLUpdate()** выбор InTouch-тегов и соответствующих столбцов таблиц осуществляется в зависимости от содержания Списка соответствия (Bind List).



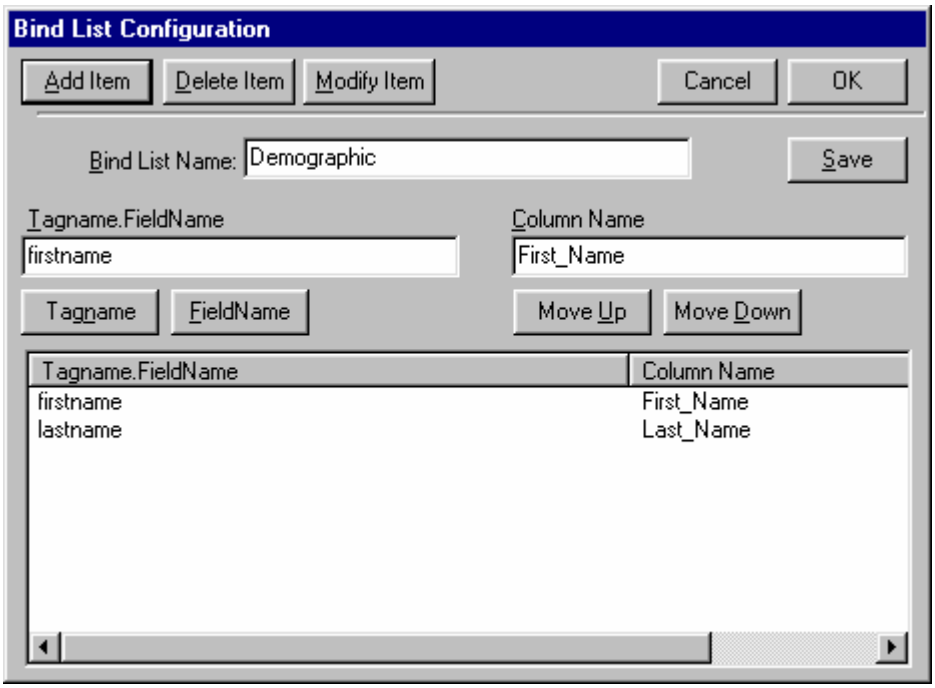
Создание Списка соответствия (Bind List)

В Bind List определено соответствие между тегами InTouch-приложения и столбцами таблицы базы данных.

Для создания нового списка:

1. В меню **Special (специальные)** укажите на строку **SQL Access Manager** и щёлкните команду **Bind List (список соответствия)**, либо в окне Application Explorer в группе **SQL Access Manager** дважды щёлкните строку **Bind List (список соответствия)**.

2. Нажмите кнопку **New (новый)**.
3. Появится диалоговое окно **Bind List Configuration (конфигурация Списка соответствия)**:



Подсказка. При щелчке правой кнопкой мыши на любом текстовом поле диалогового окна появится меню команд, применимых к выделенному сегменту текста.

4. Введите имя Списка соответствия в поле **Bind List Name (имя Списка соответствия)**.

Подсказка. Имя Списка не должно превышать в длину 32 символа. Созданный Список связывает между собой столбцы таблиц базы данных с тегами InTouch. Например, если пользователь собирает сведения о составе семьи своих работников, в данное поле можно ввести какой-либо текст, поясняющий назначение информации.

Примечание. Функции SQLInsert(), SQLSelect() и SQLUpdate() при выполнении используют значения из Списка Соответствия. Кроме того, при выполнении SQLExecute() необходимо обращать особое внимание на порядок переменных, поскольку эта функция присваивает переменным новые значения в указанной последовательности.

5. В поле **TagName.FieldName** введите имя поля (**.field**) тега InTouch.

Подсказка. Словарь тегов связывает теги и их поля с названиями столбцов базы данных. Если указанный тег ещё не определен в Словаре, для раскрытия окна Словаря дважды щёлкните на его имени.

6. Нажмите кнопку **TagName** для указания требуемого тега. Появится диалоговое окно браузера тегов.

Подсказка. В окне браузера будут показаны все теги, определённые в текущий момент для выбранного источника переменных. Для выбора какого-либо тега щёлкните на нем дважды либо выделите её и нажмите кнопку **ОК**. Для указания какого-либо поля тега нажмите на кнопку со стрелкой в поле **Dot Field**, выберите требуемое поле и нажмите **ОК**.

Примечание. Тип тега ввода/вывода, не используемые в вашем приложении, но указанные в Списке соответствия, SQLAccess, будут активизироваться (запрашиваться у сервера ввода/вывода), как только

запустится WindowViewer. Чтобы увидеть такое поведение, не требуется никакого подключения к базе данных.

Дополнительную информацию о браузере тегов можно найти в *Руководстве Пользователя InTouch*.

7. Нажмите на кнопку **FieldName**, чтобы добавить имя поля к переменной. Появится диалоговое окно **Choose field Name** (**выбор имени поля)**.
8. Щёлкните на требуемом поле. Диалоговое окно закроется, и выбранное поле автоматически будет добавлено к имени тега **TagName.FieldName**.

   Дополнительную информацию о полях тегов можно найти в Главе 4 *Руководства Пользователя InTouch*.
9. В поле **Column Name** введите название соответствующего столбца.

   Подсказка. Длина названия столбца не должна превышать 30 символов. Введённое название непосредственно связывается с названием столбца в базе данных. Если во введённом названии содержится хотя бы один пробел, при его использовании в скриптах и указании в Списке соответствия это название должно заключаться в квадратные скобки. Например:

   ```
   WHERE EXPR= "[Pipe Flow] = "text (tagName, "#");
   ```

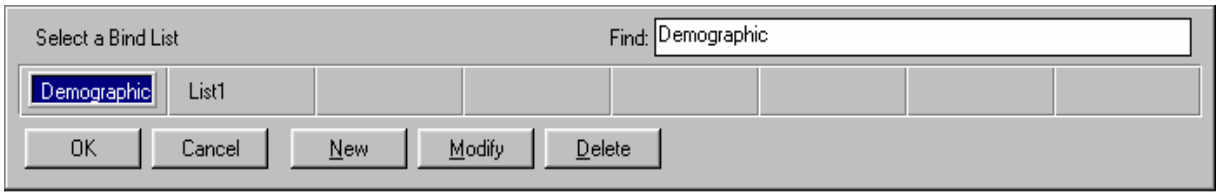
   Подсказка. Для связывания названия столбца с базой данных могут применяться также Специальные ограничители.

   Дополнительную информацию о специальных ограничителях можно найти в параграфе "Использование специальных ограничителей".
10. Нажмите кнопку **Move Up (переместить вверх)** для перемещения выбранного тега по списку на один уровень вверх.
11. Нажмите кнопку **Move Down (переместить вниз)** для перемещения выбранного тега по списку на один уровень вниз.
12. Нажмите кнопку **Add Item (добавить элемент)**, чтобы внести в Список выбранные **TagName.FieldName** и **Column Name (название столбца)**.
13. Нажмите кнопку **Delete Item (удалить элемент)**, чтобы удалить из Списка выбранные **TagName.FieldName** и **Column Name (название столбца)**.
14. Нажмите кнопку **Modify Item (модифицировать элемент)**, чтобы изменить в данном Списке выбранные **TagName.FieldName** и **Column Name (название столбца)**.
15. Нажмите **ОК** для сохранения созданного Списка соответствия и закрытия диалогового окна.

    Подсказка. При нажатии на кнопку **Save (сохранить)** сохранение информации происходит без закрытия окна.

Для изменения Списка соответствия:

1. В меню **Special (специальные)** укажите на строку **SQL Access Manager** и щёлкните команду **Bind List (список соответствия)**, либо в окне Application Explorer в группе **SQL Access Manager** дважды щёлкните строку **Bind List (список соответствия)**.
2. Появится диалоговое окно **Select a Bind List (выбор Списка соответствия)**:

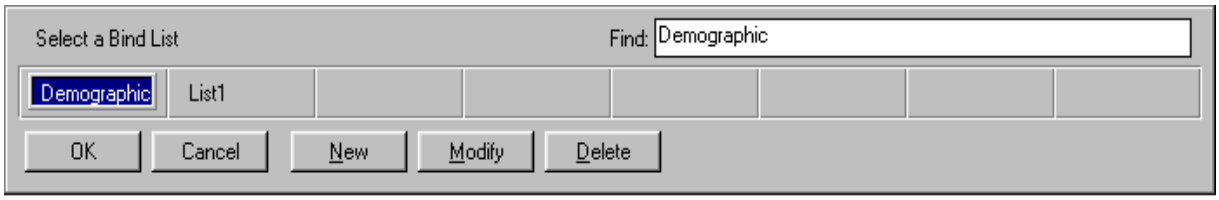


3. Укажите имя модифицируемого Списка Соответствия и нажмите кнопку **Modify (изменить)**. Появится диалоговое окно **Bind List Configuration (конфигурация Списка соответствия)**.
4. Измените необходимые элементы.
5. Нажмите **ОК** для сохранения внесённых изменений и закрытия диалогового окна.

   Дополнительную информацию о конфигурировании Списков Соответствия можно найти в пункте "Для создания Списка Соответствия".

Для удаления Списка соответствия:

1. В меню **Special (специальные)** укажите на строку **SQL Access Manager** и щёлкните команду **Bind List (список соответствия)**, либо в окне Application Explorer в группе **SQL Access Manager** дважды щёлкните строку **Bind List (список соответствия)**.
2. Появится диалоговое окно **Select a Bind List (выбор Списка соответствия)**:



3. Укажите имя удаляемого Списка соответствия.
4. Нажмите кнопку **Delete (удалить)**. Появится запрос на подтверждение операции удаления. Нажмите кнопку **Yes (да)** для удаления Списка либо **No (нет)** для отмены команды**.** Вновь появится диалоговое окно **Bind List Configuration (конфигурация Списка соответствия)**.
5. Нажмите **ОК** для закрытия диалогового окна.

Использование специальных ограничителей

Обычно символьные строки в операторах **SQLInsert()** и **SQLUpdate()** заключаются в одинарные кавычки. В некоторых SQL-базах данных входные строки должны иметь в качестве ограничителей другие символы. Например, в Oracle символьные строки должны заключаться в квадратные скобки. В этих случаях необходимо пользоваться функцией **Delim()**:

В диалоговом окне **Bind List Configuration (конфигурация Списка соответствия)** в поле **Column Name (название столбца)** после введённого наименования необходимо ввести ключевое слово "delim" (в любом регистре). После ключевого слова должны следовать следующие символы:

- левая скобка;
- левый ограничитель;
- запятая;

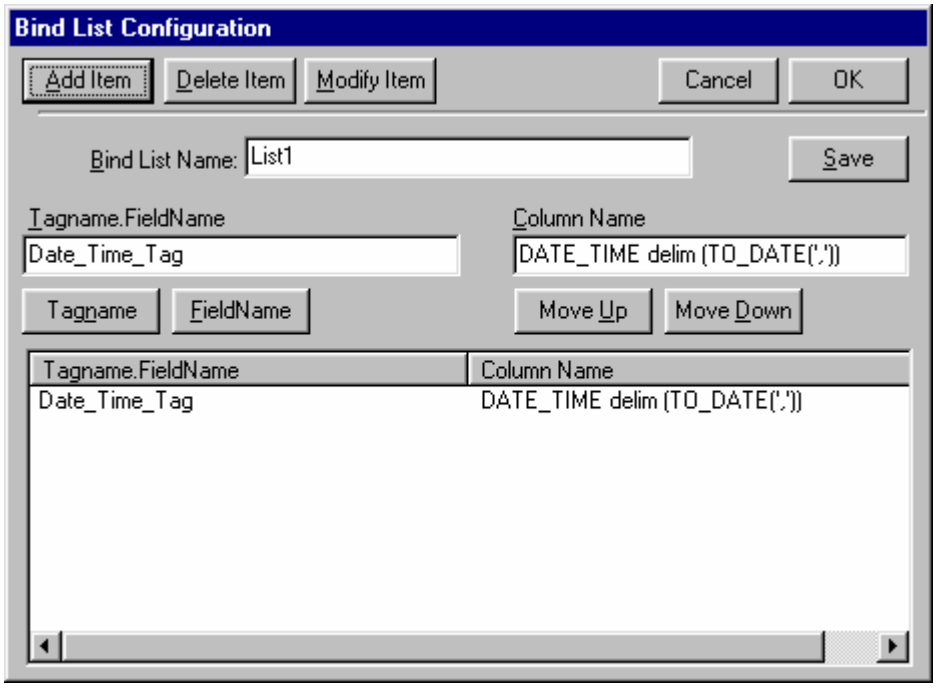
- правый ограничитель;
- правая скобка.

Пример: **datestring delim (',')**

Чтобы использовать один и тот же символ-ограничитель и слева, и справа, укажите его в скобках без запятой.

Пример: **datestring delim (' ')**

В следующем примере используются различные правый и левый ограничители. Отметьте, в каком месте поля **Column Name (имя столбца)** вводится строка **date delim (',')**:



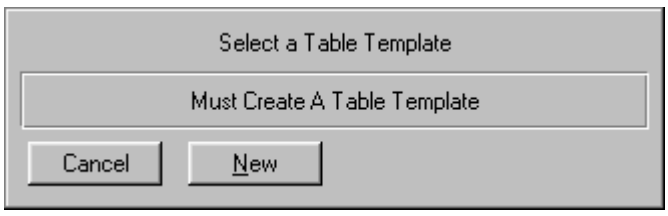
Дополнительную информацию о записи текущих показаний времени и даты в поля БД Oracle можно найти в Главе 2 "Конфигурирование и подключение к базам данных".

Создание Шаблона таблицы

С помощью данной команды создаётся Шаблон Таблицы, определяющий структуру и формат новой таблицы базы данных.

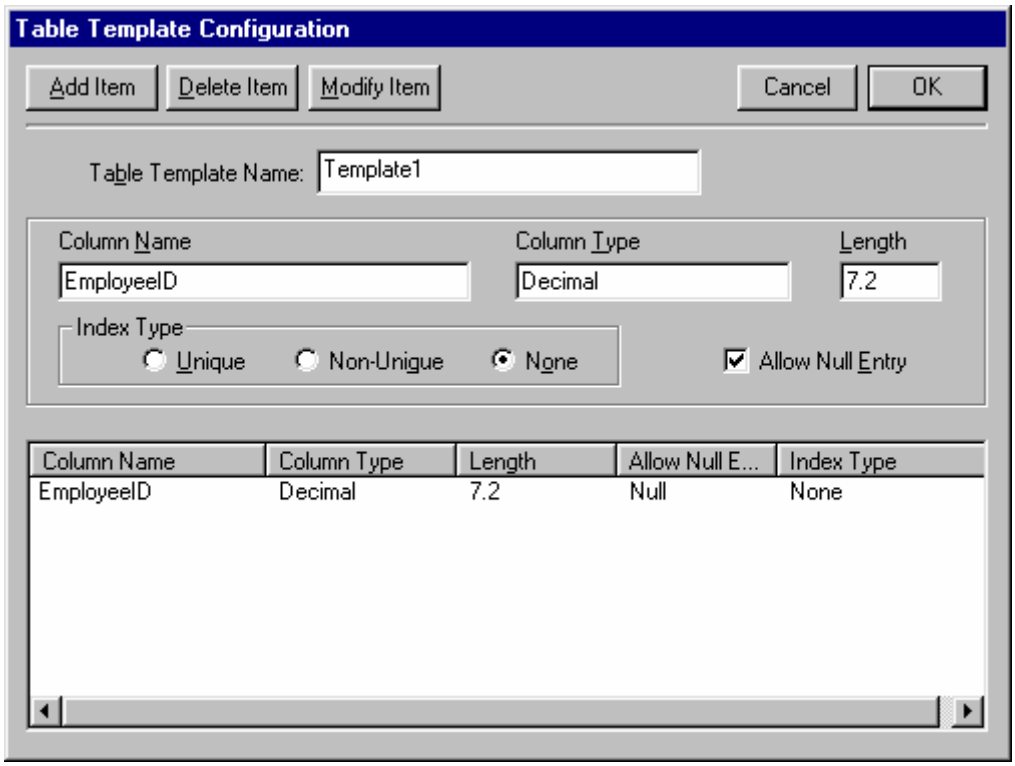
Для создания нового Шаблона:

1. В меню **Special (специальные)** укажите на строку **SQL Access Manager** и щёлкните команду **Table Template (Шаблон таблицы)**, либо в окне Application Explorer в группе **SQL Access Manager** дважды щёлкните строку **Table Template (Шаблон таблицы)**.



2. Нажмите кнопку **New (новый)**.

3. Появится диалоговое окно **Table Template Configuration (конфигурация Шаблона таблицы)**:



Подсказка. При щелчке правой кнопкой мыши на любом текстовом поле диалогового окна появится меню команд, применимых к выделенному сегменту текста.

4. Введите имя Шаблона в поле **Table Template Name (имя Шаблона таблицы)**.

Примечание. Имя Шаблона не должно превышать в длину 32 символа. Имя Шаблона используется системой для определения структуры базы данных в момент выполнения функции **SQLCreateTable()**.

5. Введите имя столбца в поле **Column Name (имя столбца)**. Длина имени не должна превышать 30 символов.

6. Введите тип данных столбца в поле **Column Type**. Варианты типов данных зависят от вида используемой базы данных.

   Дополнительную информацию о типах данных различных баз данных можно найти в Главе 2, в параграфе "Конфигурирование и подсоединение к базам данных".

7. Укажите **Index Type (тип индекса)** следующим образом:

   Unique

   Каждое значение в этом столбце должно быть уникальным.

   Non-Unique

   Уникальность всех значений в данном столбце не требуется

   None

   Индекс отсутствует.

Подсказка. Индексный файл автоматически создаётся при выполнении оператора **SQLCreateTable()**.

8. Установите флажок **Allow Null Entry (разрешить нулевой ввод)** для ввода NULL-значений в строки этого столбца

Подсказка. InTouch не поддерживает NULL-значения.

Если для какой-либо переменной при вводе значение указано не было, то ей будет присвоена следующее значение (в зависимости от типа).

Тип	Значение
Discrete	0
Integer	0
Message	Пустая строка

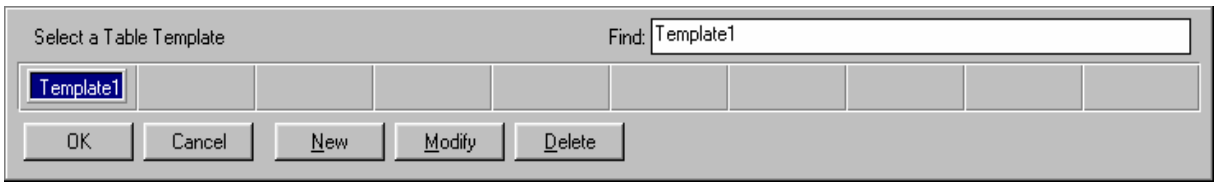
При вводе NULL-значения будут преобразовываться в вышеуказанные величины в зависимости от типа.

9. Нажмите кнопку **Add Item (добавить элемент)** для внесения названия, типа столбца, длины и типа индекса в Шаблон Таблицы.
10. Нажмите кнопку **Delete Item (удалить элемент)** для удаления названия, типа столбца, длины и типа индекса из Шаблона таблицы.
11. Нажмите кнопку **Modify Item (изменить элемент)** для модификации названия, типа столбца, длины и типа индекса в Шаблоне таблицы.
12. Нажмите **ОК** для сохранения Шаблона и закрытия диалогового окна.

Подсказка. Нажатие на кнопку **Save (сохранить)** приведёт к сохранению введённой информации без закрытия окна.

Для изменения Шаблона таблицы:

1. В меню **Special (специальные)** укажите на строку **SQL Access Manager** и щёлкните команду **Table Template (Шаблон таблицы)**, либо в окне Application Explorer в группе **SQL Access Manager** дважды щёлкните строку **Table Template (Шаблон таблицы)**.
2. Появится диалоговое окно **Select a Table Template (выбор Шаблона таблицы)**:

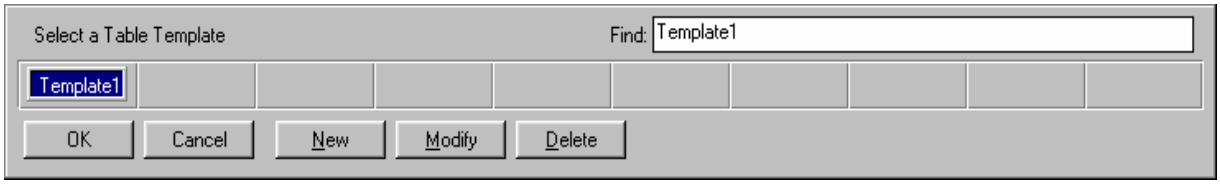


3. Укажите имя модифицируемого Шаблона и нажмите кнопку **Modify (изменить)**. Появится диалоговое окно **Table Template Configuration (конфигурация Шаблона таблицы)**.
4. Измените необходимые элементы.
5. Нажмите **ОК** для сохранения внесённых изменений и закрытия диалогового окна.

Дополнительную информацию о конфигурировании Шаблона таблицы можно найти в пункте "Для создания Шаблона таблицы".

Для удаления Шаблона таблицы:

1. В меню **Special (специальные)** укажите на строку **SQL Access Manager** и щёлкните команду **Table Template (Шаблон таблицы)**, либо в окне Application Explorer в группе **SQL Access Manager** дважды щёлкните строку **Table Template (Шаблон таблицы)**.
2. Появится диалоговое окно **Select a Table Template (выбор Шаблона таблицы)**:



3. Укажите имя удаляемого Шаблона таблицы.
4. Нажмите кнопку **Delete (удалить)**. Появится запрос на подтверждение операции удаления. Нажмите кнопку **Yes (да)** для удаления Шаблона либо **No (нет)** для отмены команды**.** Вновь появится диалоговое окно **Table Template Configuration (конфигурация Шаблона таблицы)**.
5. Нажмите **ОК** для закрытия диалогового окна.

Файл SQL.DEF

Информация о Списках соответствия и Шаблонах таблиц хранится в файле с именем "SQL.DEF". Формат этого файла – CSV (Comma-Separated Variable – переменные, разделённые запятыми). Файл "SQL.DEF" можно просматривать и редактировать, используя SQL Access Manager или средствами любого доступного текстового редактора (например Notepad). Информация записана в файле в следующем формате:

:BindListName,*Имя_списка_соответствия*

ИмяПеременной1. ИмяПоля,ИмяСтолбца1

ИмяПеременной 2 ИмяПоля, ИмяСтолбца 2

ИмяПеременной 3.ИмяПоля, ИмяСтолбца 3

:TableTemplateName,*Название_шаблона_таблицы*

ИмяСтолбца 1, ТипСтолбца,[ДлинаСтолбца],Null,Индекс

ИмяСтолбца 2, ТипСтолбца, [ДлинаСтолбца],Null, Индекс

ИмяСтолбца 3, ТипСтолбца, [ДлинаСтолбца],Null, Индекс

ГЛАВА 4

Использование SQL-функции

SQL-функции используются в InTouch для обмена информацией с базами данных. Все они являются расширением набора стандартных QuickScript-функций InTouch и могут вызываться из любого скрипта. С их помощью выполняется выбор, модификация, добавление или уничтожение строк в таблицах, к которым обращается пользователь.

Содержание

SQL-функции

В данном параграфе перечислены доступные SQL-функции. Необходимо помнить, что все они являются синхронными, то есть управление в InTouch-приложение не возвращается до тех пор, пока SQL-запрос не будет выполнен (таким образом, все опросы, построения графиков и т.д. приостанавливаются на неопределённый интервал времени до возвращения результата).

Все SQL-функции (за исключением **SQLNumRows()**) возвращают некоторый результат *ResultCode*. Ненулевое возвращаемое значение означает ошибку при выполнении функции и может использоваться в функции **SQLErrorMsg()**.

Стандартный формат обращения к SQL-функциям следующий:

SQLFunction(Parameter1, Parameter2, ...)

Подробные сведения об SQL-функциях и примерах их использования можно найти в *Справочном Руководстве по InTouch*

Функция

SQLAppendStatement(ConnectionId, SQLStatement)

Продолжение SQL-оператора, созданного с помощью *SQLStatement* для *ConnectionId*.

SQLClearParam(StatementId, ParameterNumber)

Присваивает параметру *ParameterNumber* значение 0 или строку нулевой длины в зависимости от типа параметра (числовой или символьный).

SQLClearStatement(ConnectionId, StatementId)

Освобождает ресурсы, связанные со StatementId, определяемым оператором *SQLHandle.*

SQLClearTable(ConnectionId, TableName)

Удаляет все строки в таблице с именем TableName.

SQLCommit(ConnectionId)

Определяет конец последовательности команд, входящих в одну транзакцию.

SQLConnect(ConnectionId, ConnectString)

Параметр *ConnectString* является тем же, что и ConnectionString, определённым в документации по ADO. Этот параметр может быть модифицирован в любом приложении InTouch для подключения через провайдер OLE DB к любой системе управления базами данных.

Параметр *ConnectString* содержит несколько компонентов, разделённых точкой с запятой. Первый компонент обычно является именем провайдера Provider=ProviderName, где ProviderName – это имя OLE DB провайдера для выбранной базы данных. Если в функция SQLConnect() в существующем приложении InTouch не используется ключевое слово Provider в строке подключения *ConnectString*, то будет применяться Microsoft OLE DB Provider для ODBC. Несмотря на то, что существующее приложение InTouch будет работать, рекомендуется использовать соответствующий провайдер OLE DB в строке подключения *ConnectString*. Ниже приведены примеры строки подключения:

Пример 1

Рекомендуется для использования Microsoft OLE DB Provider for Microsoft Jet:

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

Microsoft.Jet.OLEDB.4.0 является актуальным и рекомендуемым OLE DB Provider для Microsoft Jet (Microsoft Access Database engine).

Пример 2

Microsoft OLE DB Provider для ODBC используется для MS Access как провайдер по умолчанию:

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

Примечание. Идентификаторы пользовательского ID и uid являются взаимозаменяемыми, как и пароль Password и pwd.

Пример 3

Рекомендуемый формат строки подключения для Microsoft SQL Server:

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

Имя провайдера OLE DB для SQL Server – sqloledb.

Пример 4

Рекомендуемый формат строки подключения для Microsoft SQL Server:

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

Пример 5

Рекомендуемый формат строки подключения для Microsoft SQL Server:

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

Пример 6

Рекомендуемый формат строки подключения для Microsoft SQL Server:Microsoft OLE DB:

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

Примечание Имя источника данных и сервера могут быть взаимозаменяемыми.

Пример 7

Рекомендуемый формат строки подключения для Oracle:

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

Если в разделе [InTouch] файла win.ini определяется SQLTrace=1, каждое успешное выполнение функции SQLConnect будет регистрироваться в файле регистрации событий Wonderware Logger.

SQLCreateTable(ConnectionId, TableName, TemplateName)

Создаёт таблицу с именем TableName, используя шаблон TemplateName.

SQLDelete(ConnectionId, TableName, WhereExpr)

Удаляет строки в таблице TableName, для которых выполняется условие WhereExpr.

SQLDisconnect(ConnectionId)

Отключает соединение с базой данных и освобождает все ресурсы, которые были созданы функциями SQLPrepareStatement и SQLInsertPrepare, и которые не были до этого освобождены (при выполнении SQLClearStatement и SQLInsertEnd).

SQLDropTable(ConnectionId, TableName)

Удаляет из базы данных таблицу с именем TableName.

SQLEnd(ConnectionId)

Освобождает ресурсы, связанные с операциями по каналу ConnectionId.

SQLErrorMsg(ResultCode)

Возвращает в переменной *ResultCode* значение –1, если ошибку сгенерировал провайдер базы данных.

Перечень значений Result Codes и описание сообщений об ошибках представлен в Главе 5 "Проблемы и методы устранения"

SQLExecute(ConnectionId, BindList, StatementId)

Запускает выполнение SQL-оператора с определённым *StatementId* (запрос MS Access, хранимая процедура MS SQL Server или SQL-оператор в

текстовом формате). Параметр *BindList* может и меть длину 0. Если *StatementId* ассоциируется с запросом, возвращающим строку, последовательная таблица обновляется результатом SQLExecute. Если указан Real Bind List (список вещественных связей), результат ассоциируется с *BindList*. BindList с нулевой длиной полезен в случаях, когда заранее известно, что *StatementId* не ассоциируется с запросом, возвращающим строку.

SQLFirst(ConnectionId)

Переход на первую строку таблицы и считывание значений из этой строки в соответствующие теги InTouch.

SQLGetRecord(ConnectionId, RecordNumber)

Переход на строку с номером *RecordNumber* в таблице и считывание значений в соответствующие теги InTouch.

SQLInsert(ConnectionId, TableName, BindList)

Производит вставку новой строки в таблице *TableName*, используя значения тегов InTouch.

SQLInsertEnd(ConnectionId, StatementId)

Освобождает ресурсы, связанные с *StatementId* и созданные с помощью функции SQLInsertPrepare.

SQLInsertExecute(ConnectionId, BindList, StatementId)

Использует текущие значения тегов InTouch, чтобы вставить запись в таблицу, идентифицируемую функцией SQLInsertPrepare. Если Список Соответствия (*BindList)* определяет поля таблицы MS SQL Server, необходимо до запуска функции SQLInsertExecute установить режим IDENTITY_INSERT.

Пример

Вставка записи с ключом идентификации, который является частью Списка соответствия (BindList):

```
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**)**

Возвращает идентификатор для дальнейшего использования в функциях SQLInsertExecute и SQLInsertEnd.

SQLLast(ConnectionId)

Переход к последней записи в таблице и считывание значений в теги InTouch.

SQLLoadStatement(ConnectionId, FileName)

Чтение оператора, хранящегося в файле *FileName*. После чтения этот оператор полностью аналогичен оператору, созданному с помощью функции **SQLSetStatement().**

SQLManageDSN(ConnectionId)

ConnectionId не используется, Он введён для обратной совместимости со старыми версиями SQL Access. Следовательно, в функцию может быть передано любое значение. Не требуется устанавливать никакое соединение с базой данных до вызова этой функции.

Пример

```
SQLManageDSN( 0 )
```

SQLNext(ConnectionId)

Переход к следующей записи в последовательной таблице и извлечение значений этой записи в теги InTouch.

SQLNumRows(ConnectionId)

Возвращает количество записей в последовательной таблице. Поскольку эта функция может возвращать код ошибки, рекомендуется использовать её следующим образом:

```
DIM TEMP AS INTEGER;

TEMP = SQLNumRows(ConnectionId);

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

ELSE
   ResultCode = TEMP;

ENDIF;
```

Определение

Оператором по умолчанию является оператор, ассоциированный с ID соединением. Он может быть текстовым SQL-оператором (SELECT, INSERT, DELETE или UPDATE), именем запроса к MS Access (с или без параметров) или именем хранимой процедуры MS SQL Server (с или без параметров). Оператор по умолчанию изменяется с помощью SQLLoadStatement, SQLSetStatement и SQLAppendStatement и используется в SQLExecute каждый раз, когда устанавливается StatementId = 0.

SQLPrepareStatement(ConnectionId, StatementId)

Подготавливает оператор по умолчанию и возвращает *StatementId* (1, 2, 3 и т.д.) Эта подготовка полезна для операторов с параметрами, которые устанавливаются с помощью функций SQLSetParam*{Type}*. В старых версиях SQL Access в качестве второго параметра для этой функции указывается *SQLHandle*, однако в текущей версии SQL Access во всех функциях SQLHandle переименован в StatementId. Поведение функции осталось без изменений.

SQLPrev(ConnectionId)

Переход к предыдещей записи в последовательной таблице и извлечение значений этой записи в InTouch-переменные.

SQLRollback(ConnectionId)

Откат назад транзакции, которая была создана последним вызовом SQLTransact.

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

Предписывает базе данных извлечь информацию из таблицы. Когда функция **SQLSelect()** выполняется, в памяти создаётся временная таблица результатов (Results Table), содержащая записи, которые могут быть просмотрены с помощью using **SQLFirst()**, **SQLLast()**, **SQLNext(), SQLNumRows** и **SQLPrev()**.

Выполните оператор:

SELECT FROM TableName WHERE WhereExpr ORDER BY OrderByExpr

Если оператор выполняется успешно, создаётся временный ряд записей (оформленный как последовательная таблица) и, чтобы связать InTouch-переменные со столбцами этой таблицы, используется *BindList*, подготовленный для SQLFirst, SQLPrev, SQLNext, SQLLast и SQLNumRows. Эта последовательная таблица остаётся доступной, даже если она не имеет ни одной записи. Например, когда *WhereExpr* для всех записей равно False.

SQLSetParamChar(StatementId, ParameterNumber, Value, Length)

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, значение символьной строки (строка может состоять из одного символа). Последний параметр функции задаёт максимальную длину параметра. Если длина *Value* больше, чем указанная длина, *Value* усекается до заданной длины. Если указана длина 0, используется полная длина *Value*.

SQLSetParamDate(StatementId, ParameterNumber, Value)

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, значение даты. Время считается равным 12:00:00 полночи (начало указанной даты).

SQLSetParamDateTime(StatementId, ParameterNumber, Value, Precision)

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, значение даты/времени.

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

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, десятичное число. *Value* может быть либо строкой (или тег InTouch-типа сообщение), которая представляет десятичное число (123.456), или числовым значением (или вещественным внутренним тегом InTouch). Рекомендуется использовать тег-сообщение взамен вещественного тега, чтобы гарантировать точность параметра. Тем не менее, если *Value* должна быть числом с плавающей запятой (например вещественным значением,

возвращаемым из сервера ввода/вывода), функция будет продолжать работать, но точность может быть не гарантированной вследствие ограничений на представление чисел с плавающей запятой. Параметр *Precision* является количеством цифр в числе, а *Scale* – количеством цифр справа от десятичной запятой.

SQLSetParamFloat(StatementId, ParameterNumber, Value)

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, значение 64-битное знаковое значение с плавающей запятой.

SQLSetParamInt(StatementId, ParameterNumber, Value)

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, 16-битное знаковое целое значение.

SQLSetParamLong(StatementId, ParameterNumber, Value)

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, 32-битное знаковое целое значение.

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

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, значение NULL.

Параметр *Type* может иметь следующие значения:

0. срока
1. время/дата
2. Integer (целое)
3. float (с плавающей запятой)
4. decimal (десятичное)

Сравнение с NULL-значением управляется установкой ANSI_NULLS в MS SQL Server. Время распознавания этого режима зависит от системы базы данных. В SQL Server 7.0 этот режим распознаётся во время создания объекта (но не во время выполнения запроса). Когда в SQL Server 7.0 создаётся хранимая процедура, этот режим по умолчанию установлен в ON (включено), и, следовательно, выражение типа "WHERE MyField = NULL" всегда возвращает NULL (FALSE), и никакие записи не возвращаются из оператора SELECT, использующего это выражение. Чтобы сравнение = или <> возвращало TRUE или FALSE, необходимо при создании хранимой процедуры установить режим в OFF (выключено). Если переменная ANSI_NULLS не установлена в OFF, SQLSetParamNull не будет работать так, как ему положено. В этом случае для сравнения с NULL-значением необходимо использовать синтаксис "WHERE MyField IS NULL" или "WHERE MyField IS NOT NULL".

Пример

Использование SQLSetParamNull, чтобы возвратить все записи в таблице Продуктов, для которых ProductName не равно NULL.

Предположим, что хранимая процедура создаётся в SQL Server с помощью следующего текста:

```
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 может выполнить следующий скрипт SQL Access.

```
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)

Записывает в параметр *ParameterNumber*, ассоциированный с *StatementId*, значение времени. Используется текущая дата совместно с указанным временем.

SQLSetStatement(ConnectionId, SQLStatement)

Устанавливает SQLStatement на SQL-оператор по умолчанию для ConnectionId.

SQLTransact(ConnectionId)

Начинает транзакцию базы данных. Транзакции могут быть вложенными в той степени, как это поддерживается основным провайдером OLE DB для системы базы данных. Например, чистый провайдер OLE DB для Microsoft Jet поддерживает транзакции до пятого уровня вложенности, включая первую и последнюю транзакции.

SQLUpdate(ConnectionId, TableName, BindList, WhereExpr)

Использует текущее значение InTouch-тегов для обновления всех записей в таблице с именем *TableName*, удовлетворяющих условию *WhereExpr*.

SQLUpdateCurrent(ConnectionId)

Обновляет текущую запись последовательной таблицы, используя InTouch-теги, привязанные к полям таблицы с помощью Списка соответствия (Bind List), заданного в SQLSelect или SQLExecute. Если имеются записи,

которые идентичны данной записи, все они будут обновлены. Если имеется очень много идентичных записей, которые должны быть обновлены в SQL Access, эта функция может возвратить ошибку после обновления части записей. Сообщение об ошибке может быть подобным следующему: " Microsoft Cursor Engine: Key Column information is insufficient or incorrect. Too many rows were affected by update." (Microsoft Cursor Engine: информация ключевого столбца недостаточна или неверна. Очень много записей, которые подпадают под обновление.)

Чтобы исключить эту ситуацию, создайте в таблице уникальное ключевое поле, так чтобы никакие записи не были идентичными. Настоятельно рекомендуется, чтобы все используемые таблицы SQL Access имели уникальный ключ. Для таблиц без ключа рекомендуется, чтобы в качестве первичного ключа использовалось поле типа AutoNumber (MS Access) или целое поле, используемое для обозначения идентичности (Identity) записей (SQL Server), – так чтобы функция SQLUpdateCurrent воздействовала только на одну запись. Этот первичный ключ не должен включаться в BindList.

Параметры SQL-функций

В данном параграфе описаны все параметры, необходимые для SQL-функций. Если параметр записан в скрипте в парных кавычках (типа "Parameter1"), используется именно указанная символьная строка. Если кавычек нет, то предполагается, что Parameter1 – это некоторая переменная, и система обращается за значением этой переменной в Словарь InTouch-приложения.

Пример:

"c:\main\file" vs. Location

Здесь: Location – тег InTouch типа "message", "c:\main\file" – литерал.

Возможные параметры SQL-функций следующие:

BindList

Одно из имён Списка соответствия, хранящихся в файле SQL.DEF.

ConnectionID

Внутренний целый тег, созданный пользователем для хранения числа (ID), присваиваемого функцией SQLConnect каждому соединению с базой данных.

ConnectString

Символьная строка, определяющая базу данных и всю регистрационную информацию, требуемую для выполнения SQLConnect().

ErrorMsg

Переменная типа "message", содержащая текст сообщения об ошибке.

Дополнительную информацию по сообщениям об ошибках можно найти в Главе 5 "Проблемы и методы устранения".

FileName

Имя файла, который содержит некоторую информацию.

MaxLen

Максимальная ширина столбца, с которым связан этот параметр. Данная величина определяет, имеет ли этот параметр тип "переменной длины" (varying character) или тип "длинной переменной длины" (long varying character). Если *MaxLen* меньше или равно наибольшей строке символов, допустимых для базы данных, этот параметр является "переменной длины". Если больше, – "длинной переменной длины".

OrderByExpression

Выбор столбцов для сортировки и порядка сортировки. Указываются только наименования столбцов. Формат выражения следующий:

ColumnName [ASC|DESC]

Чтобы отсортировать какой либо столбец (например manager) в возрастающем порядке значений, необходимо записать:

"manager ASC"

Сортировка по значениям нескольких столбцов задаётся следующим образом:

ColumnName [ASC|DESC],

ColumnName [ASC|DESC]

Чтобы отсортировать таблицу по возрастающим значениям одного столбца (например temperature) и убывающим значениям другого (например time), необходимо записать:

"temperature ASC, time DESC"

ParameterNumber

Фактический номер параметра в операторе.

ParameterType

Тип указанного параметра. Допустимые значения:

Тип	**Описание**
Char	Строка символов фиксированной длины, заполненная пробелами.
VarChar	Строка символов переменной длины.
Decimal	Число в BCD-формате.
Integer	4-байтное целое со знаком.
Small Integer	2-байтное целое со знаком.
Float	4-байтное в формате с плавающей запятой.
Double Precision Float	8-байтное в формате с плавающей запятой.
DateTime	8 байтное со значениями даты и времени.
Date	4 байтное со значениями даты и времени.
Time	4 байтное со значениями даты и времени.
No Type	Без конкретного типа данных.

ParameterValue

Фактическое присваиваемое значение.

Precision

Количество значащих десятичных разрядов, или максимальное значение, или размер строки символов, или длина в байтах для значений даты и времени.

RecordNumber

Номер извлекаемой записи или строки.

ResultCode

Целая переменная, возвращаемая большинством SQL-функций. При успешном завершении *ResultCode* равен 0, в случае какой-либо ошибки имеет отрицательное значение.

Дополнительную информацию можно найти в Главе 5 "Проблемы и методы устранения".

Scale

Масштаб десятичной величины. Данная величина требуется только в том случае, если она применима к устанавливаемому в NULL параметру.

SQLHandle

При выполнении операторов с расширенными функциональными возможностями SQL возвращает *SQLHandle*, которое является внутренним значением.

SQLStatement

Реальный оператор, например:

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

TableName

Название таблицы базы данных.

TemplateName

Название используемого Шаблона.

WhereExpression

Определяет некоторое условие, которое для каждой строки таблицы может принимать значения либо "истина", либо "ложь". Соответствующий оператор извлекает только те строки, для которых значением условного выражения является "истина". Выражение должно иметь следующий формат:

ColumnName *оператор_сравнения* выражение

Примечание. Если тип столбца – символьный, выражение должно заключаться в одинарные кавычки.

В следующем примере выбираются все строки, в которых в столбце Name хранится значение **EmployeeID:**

```
Name='EmployeeID'
```

В следующем примере выбираются строки со значениями partno от 100 до 199:

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

В следующем примере выбираются все строки, в которых в столбце temperature хранятся величины более 350:

```
temperature>350
```

Применение SQL-функций в скриптах QuickScript

Все SQL-функции автоматически вставляются в текст любого скрипта после нажатия на кнопку **Add-ons (расширения)** в диалоговом окне редактора скриптов QuickScript. SQL-функция вставляется в текст в текущем положении курсора.

Дополнительную информацию о Скриптах QuickScript InTouch можно найти в *Руководстве Пользователя InTouch*, в Главе 6 "Создание QuickScript-скриптов в InTouch".

Построение сложных запросов

SQL Access Manager позволяет строить SQL-операторы и запросы сложной конструкции. Такие запросы могут создаваться как динамически, так и храниться в отдельном файле. Кроме того, они могут содержать параметры, которые должны "передаваться" запросу во время исполнения приложения. После передачи параметров эти запросы должны исполниться и, возможно, возвратить результаты. API-интерфейс SQL Access Manager позволяет исполнять любые запросы, которые можно выдать базе данных, и возвращать все результаты таких запросов. Кроме того, возможно применение хранимых процедур (механизм хранимых процедур пока ещё поддерживается не полностью).

Дополнительную информацию о хранимых процедурах можно найти в параграфе "Поддержка хранимых процедур".

Динамическое построение запросов

Для динамического построения запросов необходимы следующие две функции: **SQLSetStatement()** и **SQLAppendStatement()**. Первая начинает новый SQL-оператор, который может быть любым допустимым SQL-выражением, включая и имя хранимой процедуры. Поскольку в InTouch длина символьных строк не может превышать 131 символ, то для расширения текста SQL-выражения необходимо обращаться к **SQLAppendStatement()**.

Примечание. Выделенный текст соответствует командам языка SQL Query.

Пример

```
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,
    LotName, LotQuantity");
```

После этих команд SQL-оператор готов к исполнению.

Примечание. Многие базы данных чувствительны к регистру ввода символов для имен столбцов и таблиц. Чтобы приведённый выше пример работал, необходимо указывать имена таблиц и столбцов точно в том же виде, в каком они используются в базе данных.

Считывание SQL-операторов из файла

Создавать сложные SQL-запросы можно и с помощью других средств, таких как Microsoft Access и т.д., и затем использовать SQL Access для передачи запроса в InTouch для исполнения. Созданные с помощью этих средств SQL-операторы можно записать в обычный файл и впоследствии загружать их в InTouch-приложение путём вызова функции **SQLLoadStatement()**.

Пример:

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

После этого загруженный оператор готов к исполнению.

Модификация расширенных SQL-выражений

Полностью все функциональные возможности SQL-языка обеспечиваются путём указания оператора *where*, содержащего некоторое значение тега InTouch. Для указания SQL-параметров во время исполнения приложения имеются следующие функции:

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

Чтобы впоследствии выполнить подстановку параметров в SQL-оператор, необходимо в нужном месте поставить символ "?". Затем SQL-оператор "подготавливается", параметры "вставляются" в нужное место и оператор исполняется.

SQLPrepareStatement() подготавливает оператор к исполнению. Оператор не исполняется, а только переводится в "активное" состояние так, что вам остается только подставить значения параметров.

SQLSetParamType() – это на самом деле набор функций, с помощью которых осуществляется подстановка значений параметров в SQL-оператор.

Пример:

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

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

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

ResultCode = SQLPrepareStatement(ConnectionID, StatementId);

{возвращает внутренний указатель в StatementId}

ResultCode = SQLSetParamInt(StatementId, 1, tagLotNumber);

{записывает значение переменной tagLotNumber в нужный параметр запроса}

Поскольку в запросе был только один параметр, который нуждался в определении, то теперь запрос готов к исполнению.

Для высвобождения выделяемых запросу ресурсов после его исполнения можно воспользоваться функцией **SQLClearStatement()**.

Примечание. SQLEnd() вызывается для освобождения "неименованных" SQL-операторов (то есть создаваемых существующими функциями SQL Access), а также тех, которые создаются функциями **SQLSetStatement()** и **SQLLoadStatement()** и неподготовленных.

Исполнение расширенных SQL-операторов

Теперь, когда SQL-оператор либо динамически создан, либо прочитан из файла и, возможно, подготовлен и модифицирован, настало время его исполнить. Эта задача решается при помощи функции **SQLExecute()** API-интерфейса SQL Access Manager. **SQLExecute()** исполняет либо текущий активный SQL-оператор – то есть созданный функциями **SQLSetStatement()** или **SQLLoadStatement()**, – либо уже подготовленный оператор, определяемый внутренним указателем.

Пример № 1

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

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

{заносим результаты поиска в переменные, определяемые списком соответствия BindList. Указатель подготовленного оператора равен 0}

ResultCode = SQLNext(ConnectionID);

{Читаем результаты выборки}

Пример №2

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

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

{знак вопроса означает, что конкретное значение будет указано позднее}

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

ResultCode = SQLPrepareStatement(ConnectionID, ConnectionID);

```
{возвращает внутренний указатель в ConnectionID }
ResultCode = SQLSetParamInt(ConnectionID, 1, tagLotNumber);
{подставляем значение переменной tagLotNumber в нужный
   параметр запроса}
ResultCode = SQLExecute(ConnectionID, "BindList", SQLHandle);
{записываем результаты выборки в переменные,
   определяемые списком соответствия BindList, указатель
   оператора записывается в ConnectionID }
ResultCode = SQLNext(ConnectionID);
{Читаем результаты оператора Select}
```

Пример №3

SQLSetStatement – этот оператор необходим для построения сложных запросов и строковых выражений, длина которых превышает 131 символ. Для расширения строковых выражений необходимо пользоваться функцией SQLAppend.

```
SQLSetStatement(Connect_ID, "Select Speed, Set_No from
   tableName where Ser_No ='" + Serial_input + "'");
SQLExecute(ConnectionID, 0);
```

В приведённом выше примере *SQLHandle* сброшен в 0, поэтому оператор не должен обращаться к SQLPrepare(*Connect_ID, SQLHandle*) до исполнения запроса. Поскольку *SQLHandle* создавался не оператором SQLPrepare(), то для корректного завершения команды выбора необходимо вместо SQLClearStatement() вызвать функцию SQLEnd():

```
SQLSetStatement(Connect_Id, "Select Speed, Set_No from
   tableName where Ser_No ='" + Serial_input + "'");
SQLPrepareStatement(Connect_Id, SQLHandle);
SQLExecute(Connect_Id, SQLHandle);
```

В этом примере *SQLHandle* создаётся оператором SQLPrepareStatement() и используется в функции SQLExecute(). Для завершения этой команды выбора используйте оператор SQLClearStatement() для высвобождения ресурсов и освобождения *SQLHandle*.

Поддержка хранимых процедур

Оператор SQLExecute() позволяет также запускать и хранимые процедуры. Например, предположим, что на сервере базы данных хранится некоторая процедура с названием "LotInfoProc", в которой есть следующий оператор выбора: "Select LotNo, LotName from LotInfo". Запустить эту процедуру на исполнение и прочитать результаты можно с помощью следующего скрипта:

Использование Microsoft SQL Server

```
ResultCode = SQLSetStatement(ConnectionID, "LotInfoProc");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext(ConnectionID);
```

```
{Читаем результаты оператора Select}
```

Использование Oracle или Microsoft Access.

```
ResultCode = SQLSetStatement(ConnectionID, "{CALL
   LotInfoProc}");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext(ConnectionID);
{Читаем результаты оператора Select}
```

Извлечение значений в InTouch-теги

Для пяти скриптовых функций: SQLFirst, SQLPrev, SQLNext, SQLLast и SQLGetRecord – разрешена навигация между записями послежовательной таблицы и извлечение значений в InTouch-теги. Если в поля записано NULL, значения связанных InTouch-тегов будут нулевыми или строками нулевой длины, в зависимости от того, имеет ли тег аналоговый или строковый тип. Если строка базе данных больше, чем 131 символ, только первые 131 символов копируются в связанный с InTouch-сообщением тег.

Сохранение InTouch-тегов в полях базы данных

Четыре скриптовых функции: SQLUpdate, SQLUpdateCurrent, SQLInsert, И SQLInsertExecute – позволяют обновлять или делать ввод в таблицу, используя значения InTouch-тегов. Если тег InTouch-сообщения длиннее, чем заданный размер для соответствующего текстового поля таблицы, количество символов, которые берутся из тега сообщения, будет зависеть от поля. Так как теги InTouch не могут иметь значение NULL, невозможно, используя эти функции, обновить или ввести в базу данных значение NULL, если BindList содержит поле. Выход заключается в том, чтобы использовать SQLExecute в операторе INSERT, не содержащий полей, которые определены так, что позволяют вводить значение NULL.

Подключение правил обновления данных

Правила извлечения значений в InTouch-теги и существующих данных в поля таблицы учитывают, что возможно невольное преобразование значений в таблицу по следующему сценарию.

Неумышленное преобразование NULL-значений в нули или пустые строки

Исполнение одной навигационной функции извлекает NULL-значения в InTouch-теги в виде нулей или нулевых строк (то есть Tag1). После того как обновятся какие-либо другие теги в BindList, исполнение SQLUpdateCurrent сохраняет нули или нулевые строки обратно в таблицу, переписывая NULL-значение, связанное с Tag1. Исполнение SQLUpdate обновит записи, используя эти нули или строки нулевой длины из Tag1 (а не значение NULL).

Невольный ввод в таблицу нулей или пустых строк

Исполнение навигационных функций извлекает NULL-значения в InTouch-теги как нули или строки нулевой длины (то есть Tag1). После того как обновятся какие-либо другие теги в BindList, выполнение SQLInsert или

SQLInsertExecute сохранит нули или строки нулевой длины (из Tag1) в таблице (а не значение NULL).

ГЛАВА 5

Проблемы и их методы устранения

Данная глава содержит описание функции **SQLErrorMsg()** и перечень кодов завершения, возвращаемых SQL-функциями. В первой таблице главы перечислены все коды функций и соответствующие сообщения об ошибках. Вторая таблица хранит коды возврата некоторых баз данных.

Содержание

- Функции обработки ошибок
- Сообщения об ошибках баз данных
- Отладка SQL Access

Функции обработки ошибок

Все SQL-функции по завершению выдают код возврата *ResultCode*, который может быть использован для поиска ошибок. Соответствующее этому коду текстовое сообщение об ошибке может быть получено с помощью функции **SQLErrorMsg()**.

Пример:

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

где:

ErrorMsg – внутренняя переменная типа "message",

ResultCode – целая переменная, в которую был записан код возврата предшествующей SQL-функции.

Коды возврата и соответствующие сообщения

Если какой-либо код возврата не упомянут в следующей таблице, то вам необходимо обратиться к документации используемой базы данных и просмотреть сообщения программы Wonderware Logger.

Функция **SQLErrorMsg()** устанавливает значение тега InTouch типа сообщение *ErrorMsg*. Ниже приведён список некоторых возможных кодов SQL-результатов и их соответствующих сообщений об ошибках и описаний.

Код возврата	Сообщение	Причина
0	No errors occurred (ошибок нет)	Команда была выполнена успешно

–1	<Message from DB Provider> (сообщение от провайдера БД)	<Специфичное для провайдера базы данных сообщение об ошибке>
–2	An empty statement cannot be executed (Пустой оператор не может быть выполнен)	SQLExecute(ConnectionId, BindList, 0) выполнена без предварительного вызова SQLSetStatement или SQLLoadStatement с непустым параметром.
–4	Value returned was Null (Возвращённое значение – Null)	Прочитанное из базы данных целое или вещественное значение равно нулю. Это только сообщение, посылаемое в This is only a warning sent to Logger Wonderware Logger.
–5	No more rows to fetch (строк для извлечения больше нет)	Была прочитана последняя строка таблицы
–7	Invalid parameter ID (недопустимый идентификатор параметра)	Была вызвана SQLSetParamI*{Type}* с недопустимыми параметрами.
–8	Invalid parameter list (недопустимый параметр списка)	Пример недопустимого списка параметров: 1, 2, 3, 5 (Missing 4).
–9	Invalid type for NULL parameter (недопустимый тип для NULL-параметра)	Была вызвана SQLSetParamNull с недопустимым типом.
–10	Changing data type of parameter is not allowed (Изменение типа данных параметра на разрешено)	Была вызвана SQLSetParam *{Type}* с другим типом для существующего параметра.
–12	Adding parameter after the statement has been executed successfully is not allowed (Добавление параметра после того как оператор успешно выполнен, не разрешено)	Была вызвана SQLSetParam *{Type}* для нового идентификатора ID, после того как оператор был успешно выполнен.
–13	Invalid date time format (недопустимый формат даты и времени)	Встретился неверный временной формат даты, например, когда исполнялась SQLSetParamTime, SQLInsertExecute или SQLUpdateCurrent.
–14	Invalid decimal format (недопустимый десятичный формат)	Встретился неверный десятичный формат, например, когда исполнялась SQLSetParamTime, SQLInsertExecute или SQLUpdateCurrent.

–15	Invalid currency format (Недопустимый денежный формат)	Встретился неверный денежый формат, например, когда исполнялась SQLInsertExecute или SQLUpdateCurrent.
–16	Invalid statement type for this operation (Недопустимый для этой операции тип оператора)	SQLInsertEnd вызвана для идентификатора оператора, созданного с помощью SQLPrepareStatement, или SQLClearStatement вызвана для идентификатора оператора, созданного с помощью SQLInsertPrepare.
–1001	Out of memory (Не хватает памяти)	Недостаточно памяти для выполнения указанной функции.
–1002	Invalid connection (Недействительное соединение)	Функции было передано неверное значение ConnectionID.
–1003	No Bind List found (Список соответствия не найден)	Список соответствия с указанным названием не существует.
–1004	No template found (Шаблон не найден)	Шаблон таблицы с указанным именем не существует.
–1005	Internal Error (Внутренняя ошибка)	Возникла внутренняя ошибка. Обратитесь в Службу Технической Поддержки.
–1006	String is null (Null-строка)	Предупреждение: из базы данных была прочитана строка со значением NULL.
–1007	String is truncated (Строка усечена)	Предупреждение: прочитанная из базы данных строка была слишком длинная и потому усечена до 131 символа
–1008	No Where clause (нет оператора Where)	В операторе Delete (удалить) отсутствует предложение Where.
–1009	Connection failed (Соединение нарушено)	Более подробную информацию о причинах нарушения соединения с базой данных необходимо искать в записях Logger (журнала).
–1010	The database specified on the DB= portion of the connect string does not exist (Указанная в части "DB=" строки подключения база данных не существует)	Указанная база данных не существует.
–1011	No rows were Selected (Строки не выбраны)	Была осуществлена попытка выполнить SQLNumRows(), SQLFirst(), SQLNext() или SQLPrev() до исполнения команды SQLSelect().

–4013	The connection, statement, or query handle you provided is not valid (Указанные соединение, оператор или указатель запроса неверны)	Возможно, неверно определён тип столбца. В частности, если в шаблоне таблицы для dBASE столбец имеет тип *character* вместо *char*, то будет возвращена именно эта ошибка.

Сообщения об ошибках, выдаваемые базами данных

Oracle

Обратитесь к вашей документации на Oracle Server относительно специфических сообщений об ошибках и решений.

Microsoft SQL Server

Сообщение	**Решение проблемы**
You cannot have more than one statement active at a time (Более одного активного оператора в каждый момент времени запускать нельзя)	Была предпринята попытка выполнить какую-либо SQL-функцию после запуска SQLSelect(). Необходимо сначала для высвобождения выделенных для SQLSelect системных ресурсов выполнить SQLEnd() либо для второго оператора воспользоваться другим соединением (ConnectionID).
There is not enough memory available to process the command (Для исполнения команды недостаточно памяти)	Попробуйте перезагрузить рабочую станцию клиента.
Invalid object Name table Name (Неверное имя таблицы объектов)	В рабочей базе данных отсутствует таблица с указанным названием. Смените базу командой “DB=*database_Name*”.

Обратитесь к вашей документации на Microsoft SQL Server относительно специфических сообщений об ошибках и решений.

Отладка SQL-доступа

Для отладки скриптов SQL Access полезен флаг SQLTrace=1, определённый в разделе [InTouch] в файле win.ini. Новый модуль SQL Access не использует файл трассировки sqltrace.txt.

ПРИЛОЖЕНИЕ А

Зарезервированные слова

SQL Access и ODBC

В данном приложении приведены все слова, зарезервированные SQL Access и ODBC.

Если какое-либо зарезервированное слово будет использовано в качестве названия столбца в Списке соответствия или Шаблоне таблицы, то в Logger (журнал) будет передано соответствующее сообщение об ошибке. Тип генерируемой ошибки зависит от вида конкретного ODBC-драйвера и местоположения обнаруженного зарезервированного слова. В частности, одной из самых распространЁнных ошибок является использование ключевых слов DATE и TIME в качестве названий каких-либо столбцов в Списке Соответствия или Шаблоне Таблицы. Чтобы этого не происходило, измените названия столбцов, например на "aDATE" и "aTIME".

Зарезервированные ключевые слова определяют SQL-язык, используемый в InTouch SQL Access. Эти слова также распознаются и используемым ODBC-драйвером. SQL Access передает SQL-команды с одним или несколькими ключевыми словами подпрограммам файла ODBC.DLL. Если SQL-команда не может быть корректно интерпретирована, SQL Access передает программе Logger соответствующее сообщение об ошибке.

Ниже в алфавитном порядке приведены все зарезервированные ключевые слова:

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_LENGT
H
CHECK
CLOSE COALESCE
COBOL
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAM
P
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
EINFUGEN
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

Следующие ключевые слова зарезервированы для InTouch:

As
Call
Dim
Discrete
Integer
Message
Real
Return
RetVal

Предметный указатель