

目 录

第 1 章 引言—献给想用数据库而不懂数据库的工程师.....	1
第 2 章 边干边学数据库基础.....	2
2.1 数据库简史.....	2
2.2 建立数据源.....	2
2.2.1 在Access中建立一个数据库.....	2
2.2.2 建立与数据库的连接.....	3
2.2.3 数据库连接的可移植性问题（高级话题）.....	7
2.3 数据库基本操作.....	8
2.3.1 创建一个表格.....	8
2.3.2 删除一个表格.....	10
2.3.3 添加一条记录.....	10
2.3.4 查询一条记录.....	11
2.4 数据库高级操作.....	12
2.4.1 在LabVIEW中执行SQL语言案例研究.....	12
2.4.2 用SQL实现数据查询操作.....	13
2.4.3 用SQL实现删除一条记录.....	13
2.4.4 压缩数据库，释放多余空间.....	14
2.4.5 用SQL实现修改数据操作.....	15
2.5 本章总结.....	16
第 3 章 一个完整的数据库工程范例.....	17
3.1 工程项目要求.....	17
3.2 生成可执行文件(*.exe).....	18
3.3 生成安装文件(Installer).....	19
第 4 章 后记.....	22

第1章 引言—献给想用数据库而不懂数据库的工程师

曾经在一个产品检测项目中，客户要求：当产品检测不合格时，记下该产品对应的序列号，测试时间和各项测试指标，并能对这些数据进行管理和查询。由于自己没有系统的学习过数据库，所以第一时间想到的解决方案是用文件的方式(也只能把数据存成文件了)。在使用文件进行数据储存与管理时，遇到了一个巨大的问题：如何查询数据？基本的文件 IO 函数中，并没有提供现成的查询函数，所以必须自己编程实现。实现的过程是先将数据读入内存，然后再根据关键字进行线性查找，线性查找的时间复杂度为 $O(N)$ ，所以当数据量逐渐增大时，这将是一个非常可怕的过程。这个不可逾越的障碍迫使我不得不再次考虑使用数据库。

想到这儿，我立即到天河书城买了两本网上评价为数据库经典的书《数据库系统概念》和《轻松掌握 SQL》，回到办公室后立即开始学习起来。陌生的术语，难懂的理论；看了后一章便忘了前一章——非常痛苦但还是硬着头皮坚持到了下班。

这种痛苦再加上越来越近的项目交付日期，使我非常焦躁，心里终于有个声音爆发了出来“我不就是想要实现数据的保存，修改，删除和查询吗？我需要把那众多的数据库类型、复杂的关系模型、抽象的关系代数...都搞懂吗？？？”实践后的答案是，不需要，一点都不需要。

我放弃了刚买的新书，打开了 LabVIEW 数据库工具包的用户手册和范例程序，寻找着我期望的数据保存，修改，删除和查询功能。到下班的时候，Everything goes well，基本掌握了用 LabVIEW 数据库工具包进行数据保存，修改，删除和查询的方法。

回想起这段历程，突然有种想与大家一起分享的冲动——不懂数据库的工程师也可以玩转数据库，因为从应用的角度来看，我们的实际需求仅仅是数据的保存、修改、删除和查询，根本不需要去研究复杂的关系模型、抽象的关系代数、艰深的数据库设计...那基本与我们的初始目标南辕北辙。借助 LabVIEW [数据库链接工具包](#)(Database Connectivity toolkit)可以站在应用的层次，很方便的操作数据库，实现数据的保存、修改、删除和查询等功能。

“学以致用，边学边用，急用先学，立竿见影”，在后续的章节中，我们先概览一下必需的与数据库相关的基本概念，然后在 LabVIEW 平台上一边学习，一边实践如何储存、管理和查询数据。

第2章 边干边学数据库基础

2.1 数据库简史

在 20 世纪 60 年代，第一个数据库管理系统(DBMS)发明以前，数据记录主要是通过磁盘或穿孔卡片，那时候，无论是数据的管理、查询或是存储都是一件非常痛苦的事情。随着计算机开始广泛地应用于数据管理，数据共享要求也越来越高，传统的文件系统已经不能满足人们的需要，能够统一管理和共享数据的数据库管理系统应运而生。第一个数据库是美国通用电气公司 Bachman 等人在 1961 年开发成功的 IDS(Integrated DataStore)，它奠定了数据库的基础，并在当时得到了广泛的发行和应用。

随后，在 1970 年，IBM 的研究员 E.F.Codd 博士在刊物《Communication of the ACM》上发表了一篇名为“A Relational Model of Data for Large Shared Data Banks”的论文，提出了关系模型的概念，奠定了关系模型的理论基础。这篇论文被普遍认为是数据库系统历史上具有划时代意义的里程碑。Codd 的心愿是为数据库建立一个优美的数据模型，后来 Codd 又陆续发表多篇文章，论述了范式理论和衡量关系系统的 12 条标准，用数学理论奠定了关系数据库的基础。

1974 年，IBM 的 Ray Boyce 和 Don Chamberlin 将 Codd 关系数据库的 12 条准则的数学定义以简单的关键字语法表现出来，提出了具有里程碑意义的 SQL(Structured Query Language)语言。SQL 语言的功能包括查询、操纵、定义和控制，是一个综合的、通用的关系数据库语言，同时又是一种高度非过程化的语言，只要求用户指出做什么而不需要指出怎么做。SQL 语言的这个特点使之成为了一种真正的跨平台和跨产品的语言。

现今，数据库技术已经发展的比较成熟了，著名的数据库管理系统有 SQL Server、Oracle、DB2、Sybase ASE、Visual ForPro、Microsoft Access 等。Microsoft Access 是在 Windows 环境下非常流行的桌面型数据库管理系统，它作为 Microsoft office 组件之一，安装和使用都非常方便，并且支持 SQL 语言，所以本文将基于 Access 来介绍数据库的操作。

2.2 建立数据源

实现数据库功能的第一步便是建立数据源，下面将详述整个过程。

2.2.1 在 Access 中建立一个数据库

LabVIEW 数据库工具包只能操作而不能创建数据库，所以必须借助第三方数据库管理系统，比如 Access，来创建数据库。本文的大型数据库范例程序是 iPhone 测试，所以先建立一个名为 iPhoneData.mdb 的数据库文件，如图 2.1 所示。

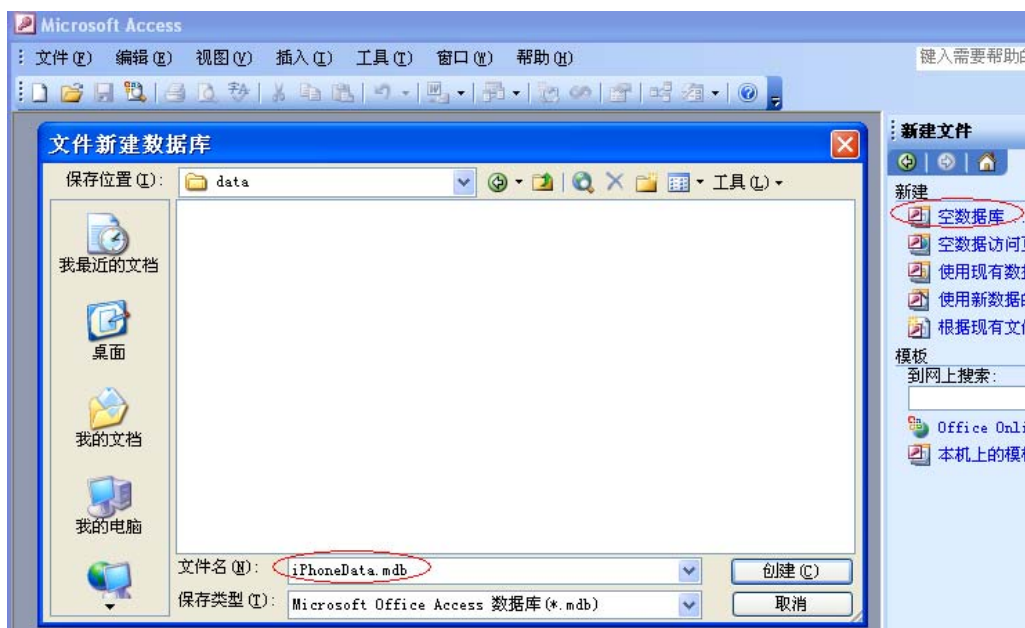


图 2.1 创建 Access 数据库

2.2.2 建立与数据库的连接

在利用 LabVIEW 数据库工具包操作数据库之前，需要先连接数据库，这就像操作文件之前，先要打开文件一样。连接数据库有两种方法：

1. 利用 DSN 连接数据库

LabVIEW 数据库工具包基于 ODBC(Open Database Connectivity)技术，如图 2.2 所示，在使用 ODBC API 函数时，需要提供数据源名 DSN(Data Source Names)才能连接到实际数据库，所以我们需要首先创建 DSN。

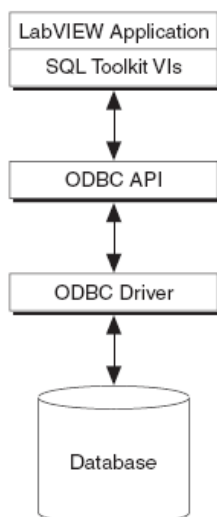




图 2.2 LabVIEW 数据库工具包基于 ODBC 技术

【背景知识】 ODBC(Open Database Connectivity，开放数据库互连)是微软公司开放服务结构(WOSA,Windows Open Services Architecture)中有关数据库的一个组成部分，它建立了一组规范，并提供了一组对数据库访问的标准 API（应用程序编程接口）。这些 API 利用 SQL 来完成其大部分任务。ODBC 本身也提供了对 SQL 语言的支持，用户可以直接将 SQL 语句送给 ODBC。

在“Windows控制面板”中双击“管理工具”，然后双击“数据源”，进入ODBC数据源管理器，如图 2.3所示。

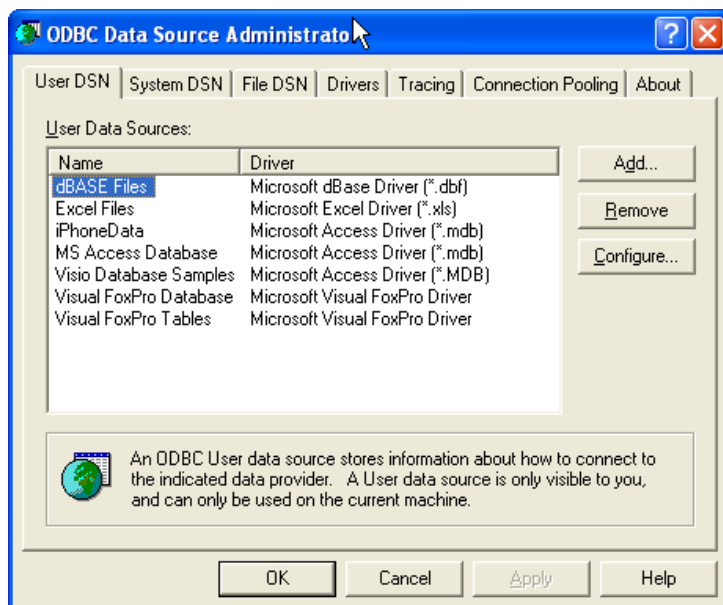


图 2.3 ODBC 数据源管理器

User DSN(用户数据源名)选项卡下建立的数据源名只有本用户才能访问，System DSN(系统数据源名)选项卡下建立的数据源名在该系统下的所有用户都可以访问。User DSN选项卡下点击 **Add...** 按钮，会弹出数据源驱动选择对话框，然后选择Microsoft Access Driver(*.mdb)，如图 2.4所示。

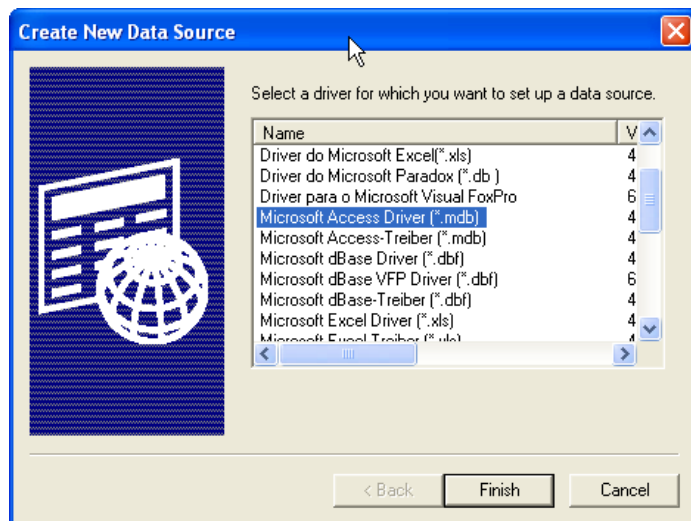


图 2.4 数据源驱动选择对话框

点击“**Finish**”按钮后，会弹出**ODBC Microsoft Access Setup**窗口，在**Data Source Name**填入一个名字，比如iPhoneData，然后在Database栏中单击**Select**按钮选择先前已经建立好的iPhoneData.mdb数据库文件，其它参数保持默认，单击OK按钮，如图 2.5所示。

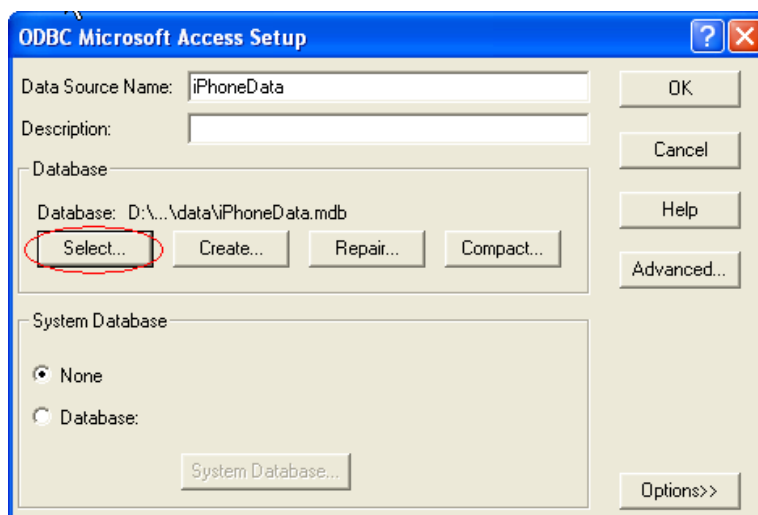


图 2.5 ODBC Microsoft Access Setup 窗口

完成上述设置后，就可以在User DSN选项卡下看到新建的DSN了。单击OK按钮完成DNS的建立。打开随本文的程序：ConnectionExample.vi，在DSN Name中填入刚建好的DSN名并运行，如图 2.6所示。

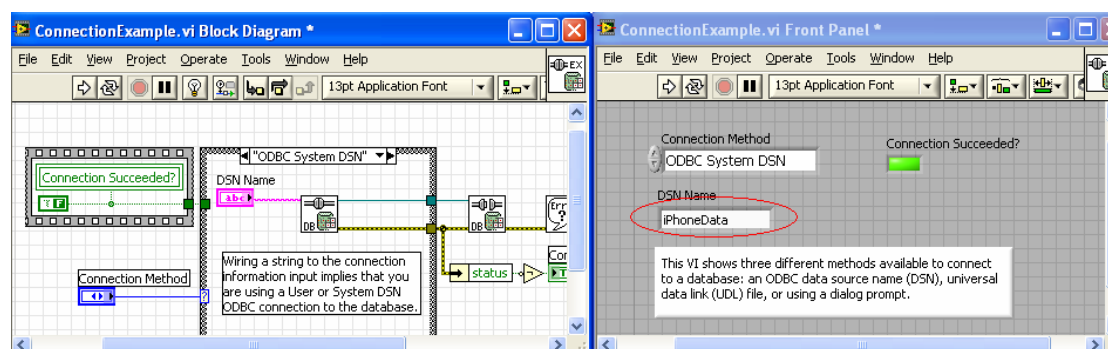


图 2.6 利用 DSN 连接数据库

【注意】使用 DSN 连接数据库需要考虑移植问题，当你把代码发布到其它机器上时，需要手动为其重新建立一个 DSN。

2. 利用 UDL 连接数据库

Microsoft设计的ODBC标准只能访问关系型数据库，对非关系型数据库则无能为力。为解决这个问题，Microsoft还提供了另一种技术：Active数据对象ADO（ActiveX Data Objects）技术。ADO是Microsoft提出的应用程序接口（API）用以实现访问关系或非关系数据库中的数据。ADO使用通用数据连接UDL(Universal Data Link)来获得数据库信息以实现数据库连接。

在iPhoneData.mdb所在的文件夹下点击鼠标右键->新建->Microsoft Data Link，如图 2.7所示，并把文件命名为“iPhoneData.udl”。

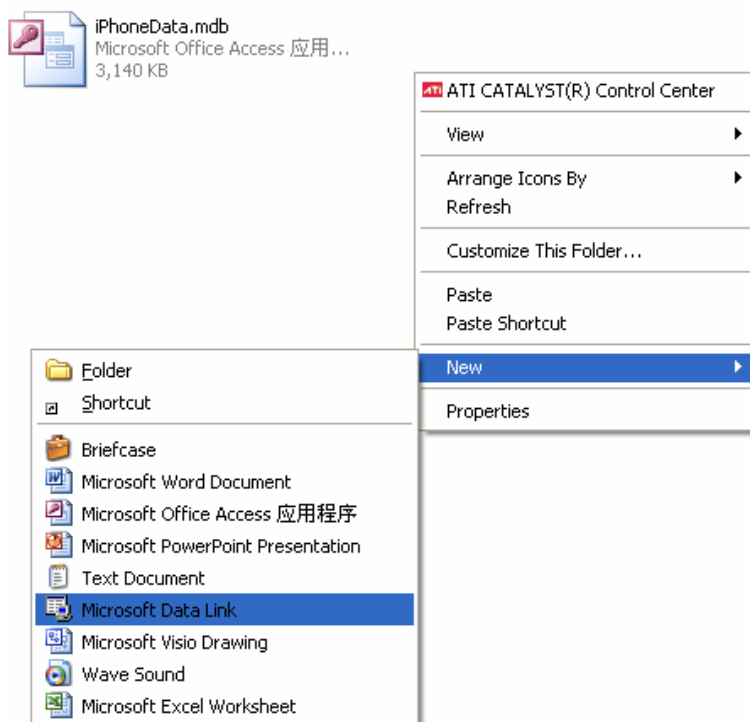


图 2.7 新建 UDL 文件

双击 “iPhoneData.udl”，打开**Data Link Properties**对话框，在**Provider**选项卡中选择 Microsoft Jet 4.0 OLE DB Provider，如图 2.8所示，并点击**Next>>**按钮。

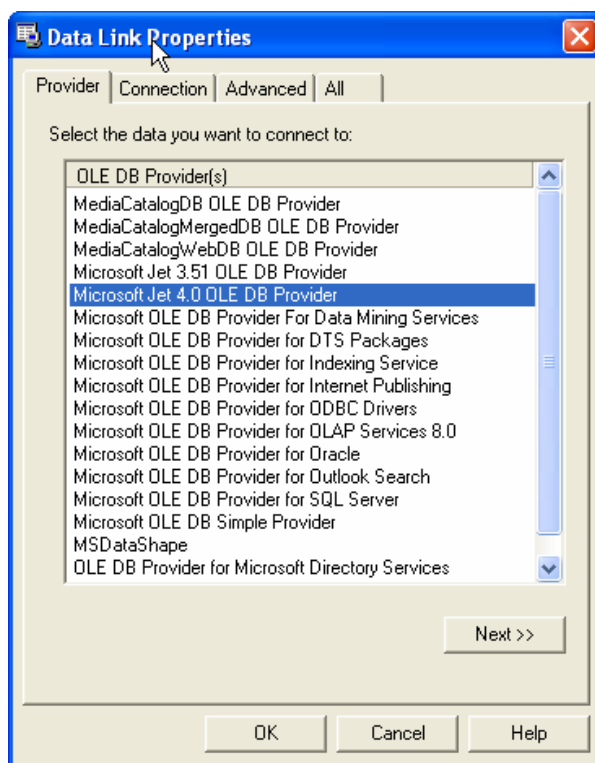


图 2.8 选择 Provider

在**Connection**选项卡中，选择已建立好的数据库文件，然后点击**Test Connection**按钮，

如果没有什么问题的话，会弹出Test Connection succeeded的对话框，如图 2.9所示。

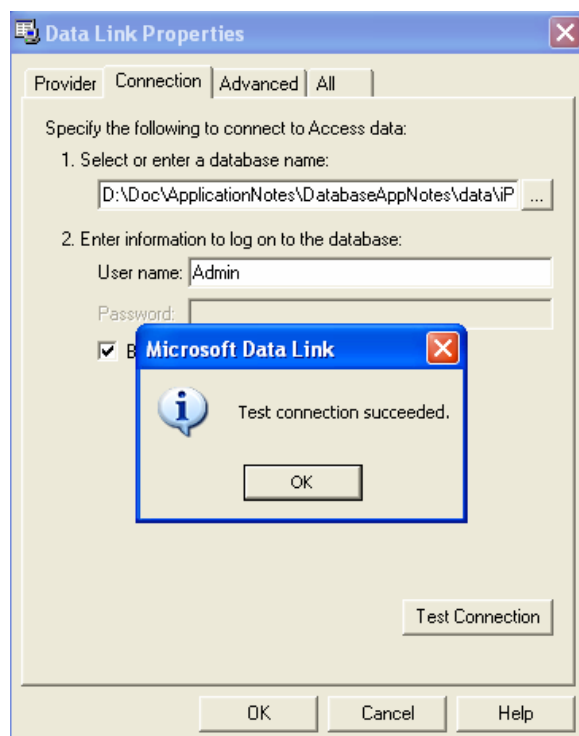


图 2.9 选择数据库源

创建好UDL后，打开随本文的程序：ConnectionExample.vi 在DSN Name中填入刚建好的DSN名并运行，如果成功的话，会如图 2.10所示。

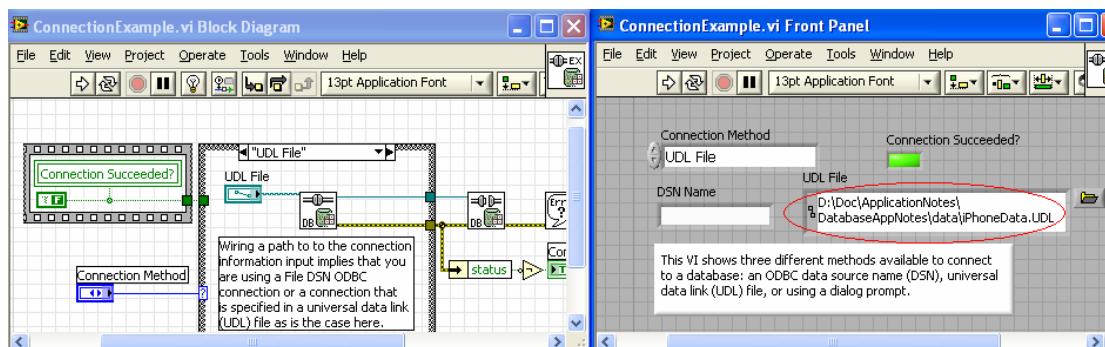


图 2.10 利用 UDL 连接数据库

2.2.3 数据库连接的可移植性问题（高级话题）

用文本编辑器打开刚新建的“iPhoneData.udl”文件，其内容如下所示：

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\Doc\ApplicationNotes\DatabaseAppNotes\data\iPhoneData.mdb;Persist Security Info=False
```

分号(;)后面的内容是注释可以不用搭理。Data Source 后面是数据库的绝对路径，所以，当数据库文件位置变化后，程序就会出错。

我们可以直接把“iPhoneData.udl”文件中的信息重新生成，然后传给DB Tools Open

Connection.vi, 如图 2.11 所示。具体的实现可以参看本文附带的范例

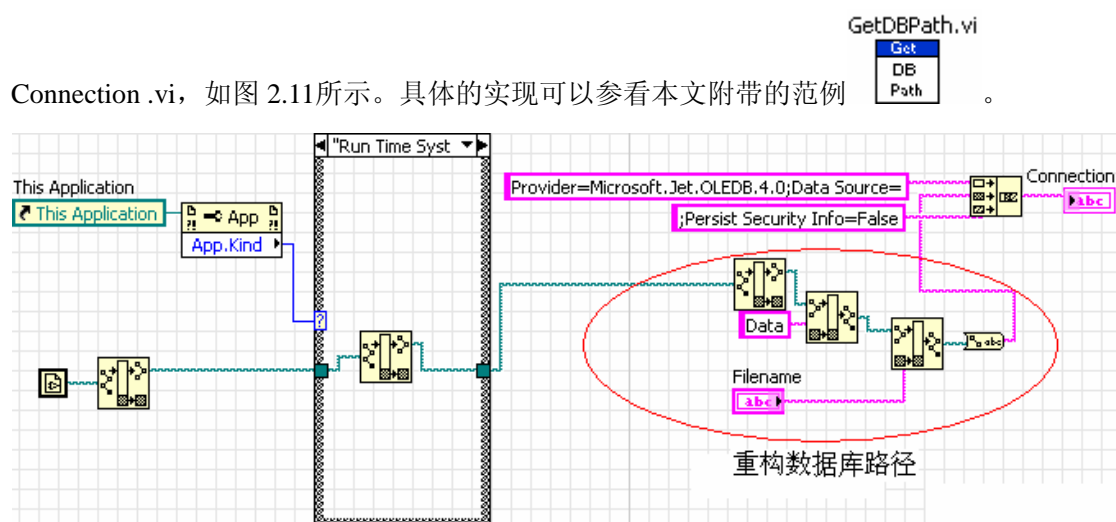


图 2.11 GetDBPath.vi

在后面的程序中，本文都将使用  GetDBPath.vi 来获得数据库的位置信息，然后传给 DB Tools Open Connection .vi。

2.3 数据库基本操作

连接上数据库之后，就可以对数据库进行操作了。本节主要介绍数据库最常用的几种基本操作，包括创建表格、删除表格、添加记录、查询记录。

2.3.1 创建一个表格

数据库是以表的形式来记录数据的，如图 2.12 所示，大家可以用 Access 打开一个数据库文件来体验一下。


■ TestResult : 表							
	TestTime	Operator	UUTSerialNum	PowerPass	EarphoneTMD	TouchPanelPass	MicphonePass
▶	2007-9-1 15:17	Tester	OPU-24736I-33	true	长二进制数据	true	true
	2007-9-1 15:17	Tester	JVU-40246M-63	true	长二进制数据	false	false
	2007-9-1 15:17	Tester	CZO-52187P-77	false	长二进制数据	true	true
	2007-9-1 16:27	Tester	JAL-24479J-62	true	长二进制数据	true	false
	2007-9-1 16:27	Tester	EIT-38585G-56	true	长二进制数据	false	true

图 2.12 数据库里的表格

数据表的每一行，表示一条记录(Record)；每一列，表示记录中的字段(Field)，说的通俗点就是记录中的一项内容，比如测试时间。能够唯一标识表中某一行的属性或属性组，叫主键(Primary Key)，一个表只能有一个主键，但可以有多个候选索引。因为主键可以唯一标识某一行记录，所以可以确保执行数据更新、删除的时候不会出现张冠李戴的错误。

DB Tools Create Table.vi



创建数据表由LabVIEW 数据库工具包中的  实现。参数中的**Table**，为被创建的数据表表名，**Column Information**指定表格每一列的属性，如图 2.13所示。

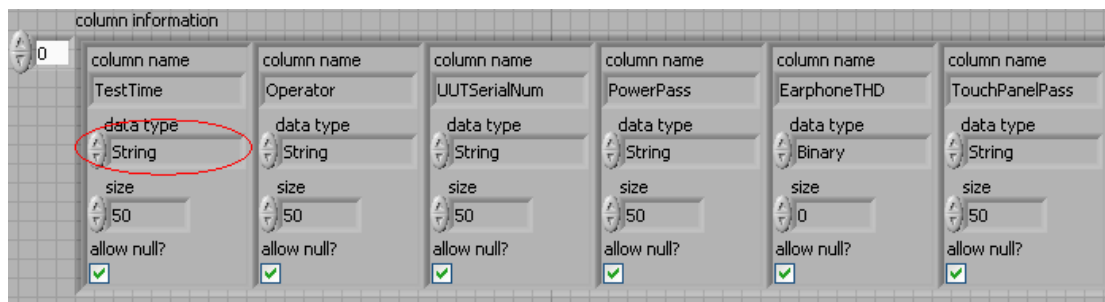


图 2.13 column information

需要注意的是column information中的**Data type**设置部分，LabVIEW的数据类型有许多种，而这里的Data type选项只有几种，所以必须清楚LabVIEW数据类型和Column information中的Data type的对应关系，如表 2.1所示。

表 2.1 对应关系

LabVIEW 数据类型	数据库数据类型
Number	Number
String/Path	String
Array	Binary
Cluster	Binary
Boolean	String or Number
Enum	Number
Variant	Binary
Picture control	Binary
WDT	Binary
Refnum	Not supported
I/O Channel	String
Complex Numbers	Binary

知道了这个对应关系后，创建数据库表格便是一件很容易的事情，范例程序如图 2.14所示，具体代码请参考**CreateTable.vi**。

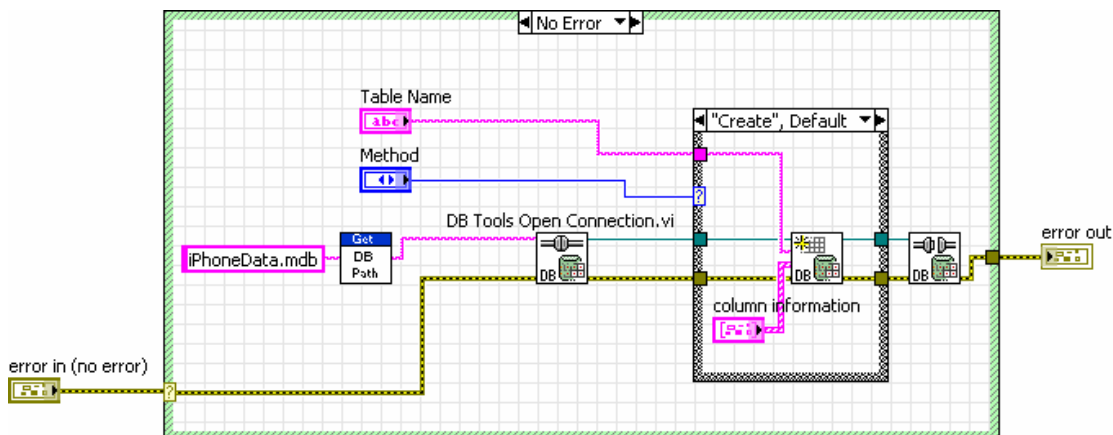



图 2.14 创建数据库表格代码

如果数据库表格创建成功，大家可以用 Access 打开数据库，看到自己创建的表格。

2.3.2 删除一个表格

DB Tools Drop Table.vi



与数据库表格创建相对应的是数据库表格删除，由  实现。将需要删除的数据库的名字告诉 **DB Tools Drop Table.vi** 即可完成数据库表格的删除操作。具体实现如图 2.15 所示。

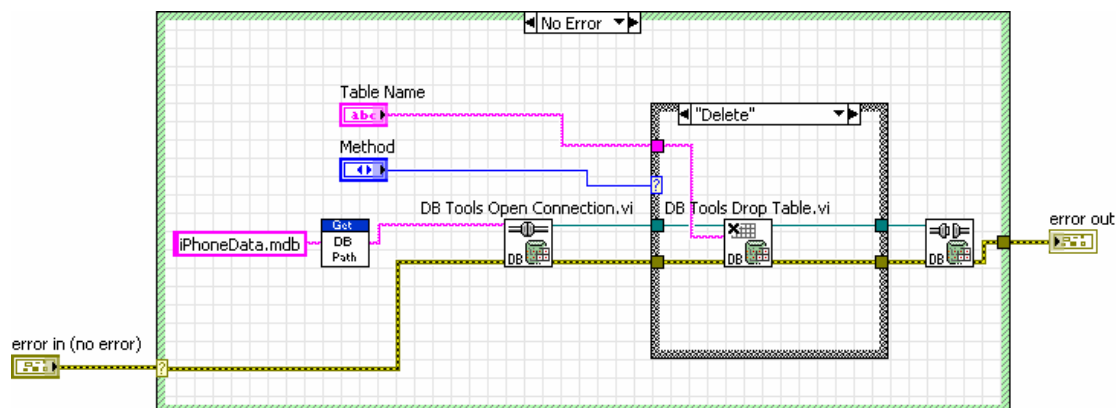



图 2.15 删除数据表格

【分享】表格创建和删除的参数非常相似，可以很方便的在一个 VI 里面同时实现表格的创建和删除操作。

2.3.3 添加一条记录

DB Tools Insert Data.vi



添加一条记录由  来实现，**DB Tools Insert Data.vi** 需要三个主要的参数：**table**(数据表名)告诉 **DB Tools Insert Data.vi** 往数据库里的哪个表格插入数据；**data**，告诉 **DB Tools Insert Data.vi** 插入什么数据；**columns** 对应插入的列的名字，其数据类型是一个字符串数组。添加记录的范例程序如图 2.16 所示。

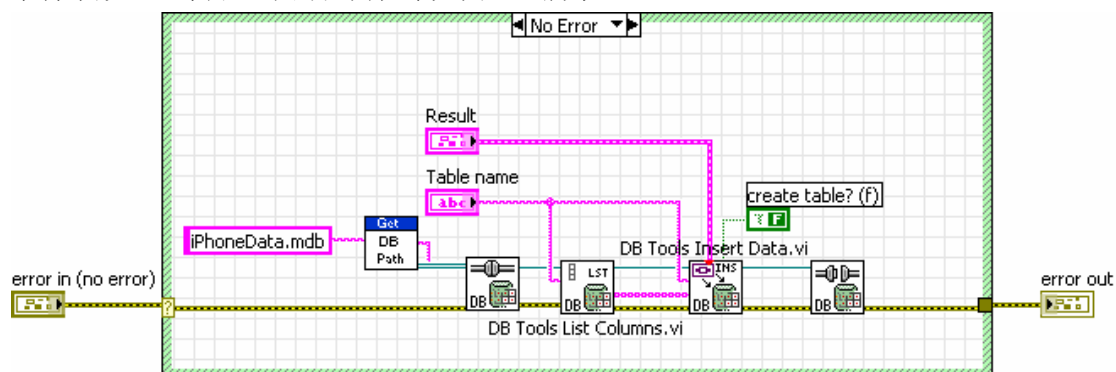



图 2.16 添加一条记录


DB Tools List Columns.vi



LabVIEW 数据库工具包提供了一个工具 VI ，可以把指定数据表的 column 名字读出来传给 **DB Tools Insert Data.vi**，这样可以省去手动输入 column 名字的工作。

2.3.4 查询一条记录

数据能存储到数据库之后，下一个要考虑的操作即是如何把数据读出。把数据从数据库

中读出的VI是 ，我们只需要告诉**DB Tools Select Data.vi**读取哪个数据表格，**DB Tools Select Data.vi**就会把该表格中的所有数据读出来，如图 2.17所示。

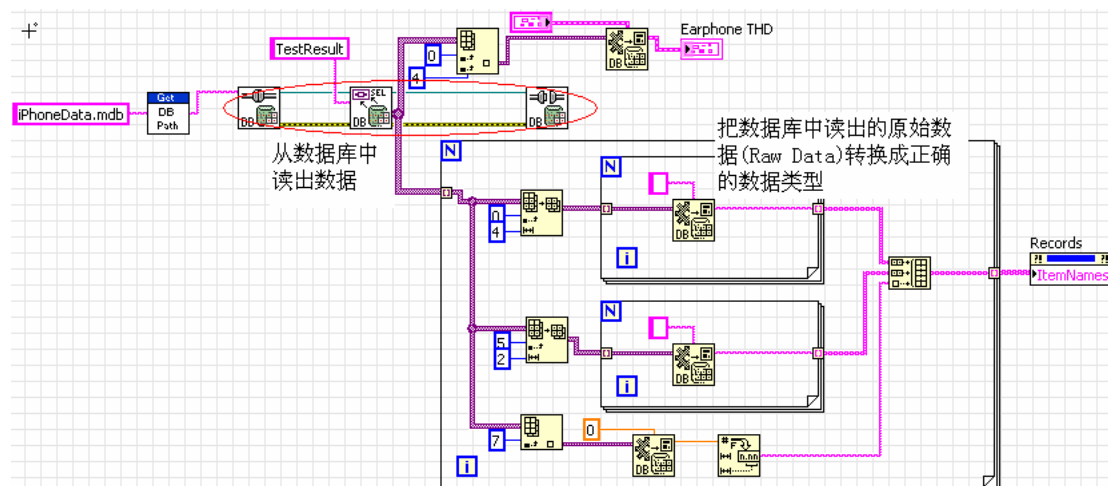





图 2.17 从数据库中读出数据

从  读出的数据是动态数据类型，需要通过  函数把动态数据类型转换成正确的数据类型。

大多数情况下，我们并不需要把数据库中的数据全部读出来。比如，数据库中已经有 1000 条记录了，每次都要把 1000 条记录读出来，费时费力。

LabVIEW 数据库工具包完全支持SQL语言，在  的**optional clause**输入端按照SQL语法输入条件语句，即可读出感兴趣的数据。SQL语言非常简单，如果从来没有接触过SQL语言，可以花一点时间学习一下。LabVIEW 数据库工具包的用户手册附录A上有SQL的快速参考手册。SQL的条件查询语法格式如图 2.18所示。

SELECT	SELECT [DISTINCT] { * col_expr, col_expr, ... } FROM {from_clause} [WHERE {where_clause}] [GROUP BY {group_clause, ...}] [HAVING {having_clause, ...}] [ORDER BY {order_clause, ...}] [FOR UPDATE OF {col_expr, ...}]	SELECT is used to query specified columns FROM database tables. A WHERE clause is used to restrict the selection; and ORDER BY and GROUP BY clauses are used to organize the resulting data. SELECT col1, col2, col3 FROM tabl WHERE col1 >= (col3 * col2) ORDER BY col3 ASC
--------	--	--

图 2.18 SQL 的条件查询语法格式

DB Tools Select Data.vi



已经把**SELECT**语句预先集成好了，我们只需要在**optional clause**输入端加入**WHERE**语句部分即可，如图 2.19所示。范例程序从**TestResult**这张表里把字段**TestTime**等于字符串“2008-8-27 16:24:34”的记录读出来。

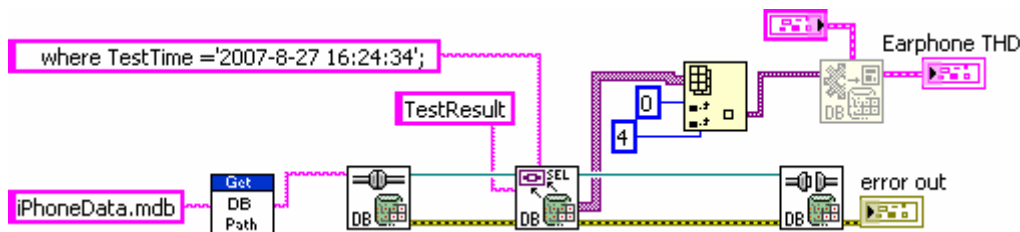


图 2.19 读出感兴趣的数据

【参考知识】WHERE 语句可包括各种条件运算符：

比较运算符(大小比较): >、>=、=、<、<=、<>、!>、!<

范围运算符(表达式值是否在指定的范围): BETWEEN...AND... NOT BETWEEN...AND...

列表运算符(判断表达式是否为列表中的指定项): IN (项 1,项 2.....) NOT IN (项 1,项 2.....)

模式匹配符(判断值是否与指定的字符通配格式相符):LIKE、NOT LIKE

空值判断符(判断表达式是否为空): IS NULL、NOT IS NULL

逻辑运算符(用于多条件的逻辑连接): NOT、AND、OR

2.4 数据库高级操作

数据库操作除了创建表格、删除表格、添加记录、查询记录外，常用的还有删除记录、更新现有数据等操作。由于这些操作并没有现成的 VI，所以需要借助 SQL 语言来实现。

2.4.1 在 LabVIEW 中执行 SQL 语言案例研究

如前所述，绝大部分DBMS都支持SQL语言，LabVIEW 数据库工具包实现的实质也是基于SQL语言，它为不熟悉SQL语言的用户把SQL语言封装了起来，以方便他们使用。2.3节的操作都是基于已封装好的VI。

DB Tools Select Data.vi



双击

，打开它的程序框图，如图 2.20所示，让我们一起来研究一下它的实现过程。

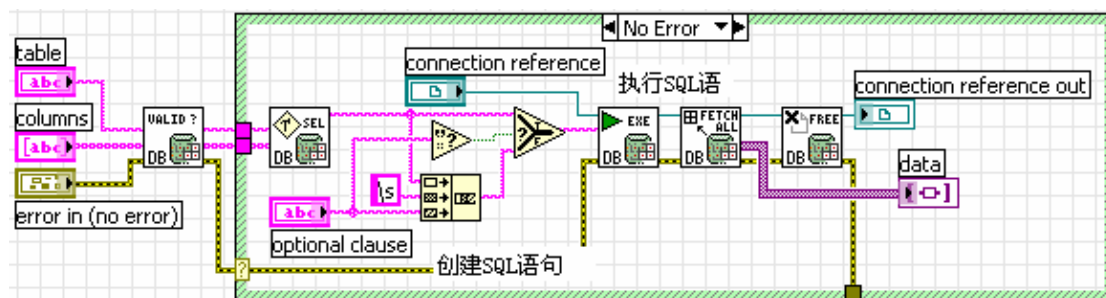


图 2.20 DB Tools Select Data 的代码实现



如图 2.20所示，**DB Tools Select Data.vi**先创建了SQL语句，然后用
来执行SQL语句，以此实现对数据库的操作。

在本节中，我们也沿用这种模式，直接用 SQL 来实现 LabVIEW 数据库工具包中没有提供的功能。

2.4.2 用 SQL 实现数据查询操作

了解了LabVIEW如何执行SQL语言后，我们可以重新实现2.3.4查询一条记录的功能，范例程序如图 2.21所示。

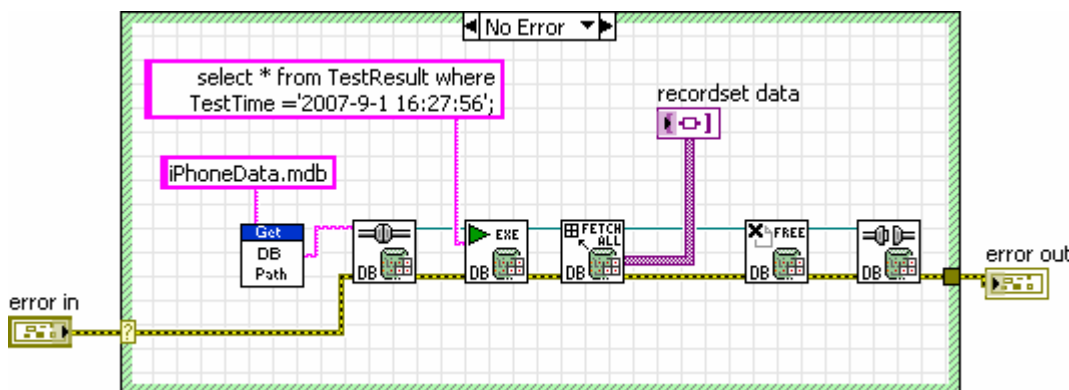


图 2.21 用 SQL 查询记录

2.4.3 用 SQL 实现删除一条记录

通过查阅 SQL 参考手册，我们知道删除一条记录的语句是 “detele”，其语法为：

```
DELETE FROM table_name
WHERE column_name = some_value
```

参考图 2.21，我们很容易实现删除一条记录的操作，如图 2.22所示。

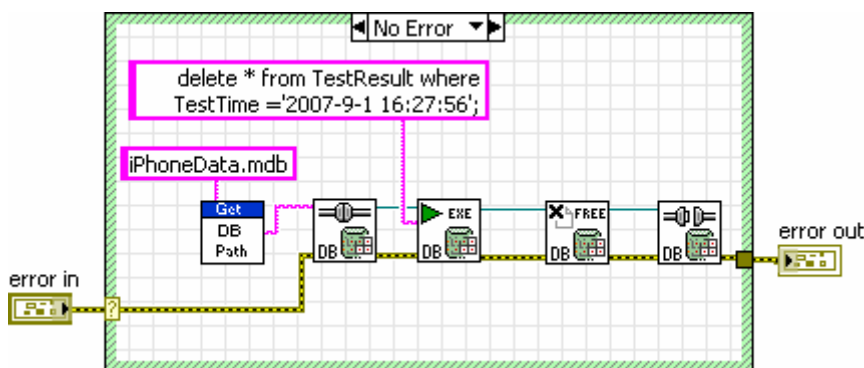


图 2.22 删除一条记录

在使用 “detele” 语句时，需要注意：

1. “detele” 语句不能删除单个字段的值，只能删除一行、多行、所有行或一行也不删除；
2. “detele” 语句仅能删除记录，不能删除表本身；删除表要用 “drop” 语句，这也是为什么 LabVIEW 数据库工具包中删除表的 VI 叫 Drop Table。

2.4.4 压缩数据库，释放多余空间

删除了记录后，你会惊奇的发现数据库文件并没有减小，就算把所有的数据全部都删除掉，结果也一样。这是因为数据库在使用一段时间后，时常会出现因数据删除而造成数据库中空闲空间太多的情况，这时就需要减少分配给数据库文件和事务日志文件的磁盘空间，以免浪费磁盘空间。

Microsoft提供了一个压缩数据库的方法，请参考[How To Compact Microsoft Access Database Through ADO](#)。其基本思想是，使用ADO的扩展：Microsoft Jet OLE DB Provider and Replication Objects (JRO)中的方法：CompactDatabase。Jet对象在文件Msjro.dll里面，如图 2.23 所示。

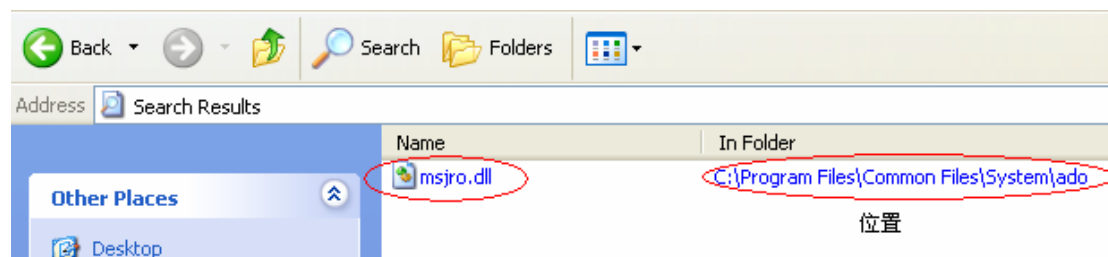



图 2.23 Msjro.dll 文件

下面将简单介绍一下在 LabVIEW 中访问 Jet 对象的方法：

在前面板上放一个 ，在右键菜单中选择“**Select Active Class**”->“**Browse...**”，如图 2.24所示。

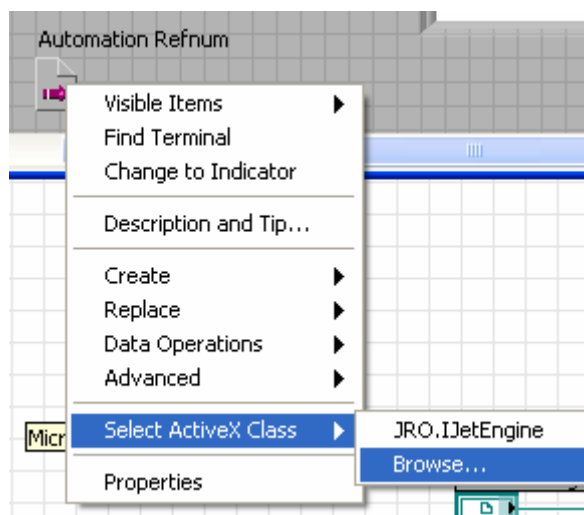


图 2.24 选择 Jet Engine

在弹出的**Select Object From Type Library**窗口中点击**Browse**按钮，按照图 2.24所示的路径选择Msjro.dll。然后在**Objects**选择框中选择**JetEngine(JRO.JetEngine.2.6)**，点击**OK**按钮，如图 2.25所示。

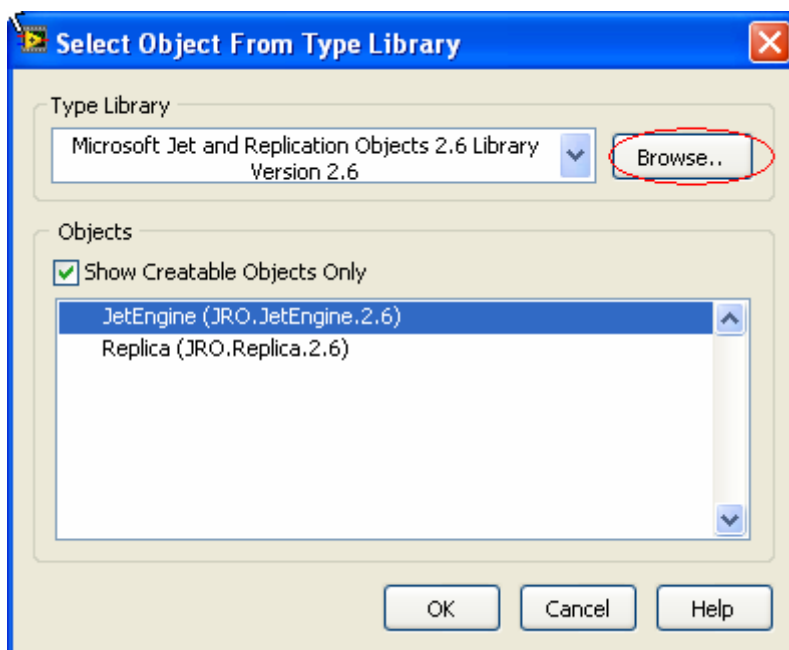


图 2.25 选择 Jet 对象

在程序框图中放置一个方法节点，并选择方法 `CompressDatabase`，如图 2.26 所示。到此为止，就实现了在 LabVIEW 中调用 Jet 对象的 `CompressDatabase` 方法。

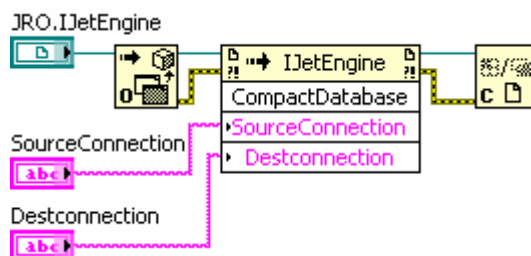


图 2.26 调用 `CompressDatabase` 方法

由于这个方法仅仅对原数据库做了一个压缩后的备份，所以，还需要把原文件移除，用这个备份来替代原数据库文件。详细实现请参考 `Access File Compress Database.vi`。

2.4.5 用 SQL 实现修改数据操作

SQL 中，修改一条记录的语句是 “**update**”，其语法为：

```
UPDATE table_name
SET column_name = new_value
WHERE column_name = some_value
```

修改一条记录的范例程序，如图 2.27 所示：

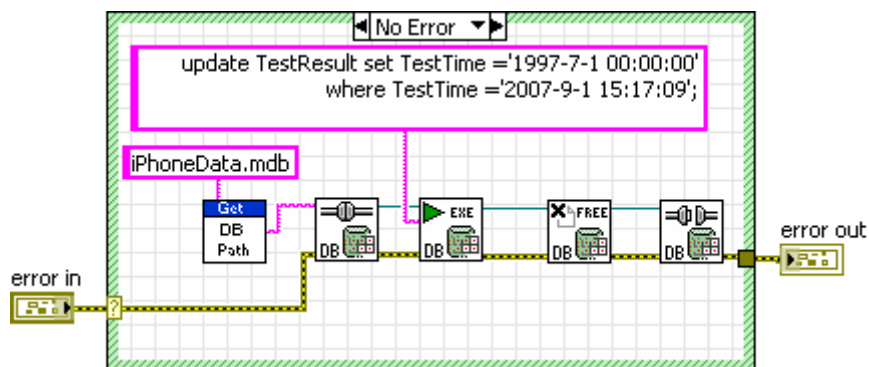


图 2.27 修改一条记录

2.5 本章总结

本章介绍了大多数最常用的数据库操作及在 LabVIEW 平台下的实现方法。SQL 语言是一种数据库操作的通用语言，不仅在 LabVIEW 数据库工具包中可以用到，在其它语言(如 VB,VC)平台下也可以使用。希望大家通过上面的操作不仅学习了 LabVIEW 数据库工具包，也熟悉了 SQL 语言。

第3章 一个完整的数据库工程范例

本章将给出一个工程项目，用于对本文介绍的知识进行消化和总结。

3.1 工程项目要求

读者学完 labVIEW 数据库工具包后，可以按照如下要求，实现一个测试项目工程：

1. 测试 iPhone 的 Power 是否合格；
2. 测试 iPhone 耳机的 THD 曲线；
3. 测试 iPhone 的触摸屏；
4. 测试 iPhone 麦克风的声压级(SPL)；
5. 完成测试后，把这些测试结果记录到数据库中；
6. 能够读入数据库，并能按要求检索数据；
7. 能按照测试时间等内容对结果进行升序或降序排列。

该工程项目的界面如图 3.1和图 3.2所示。

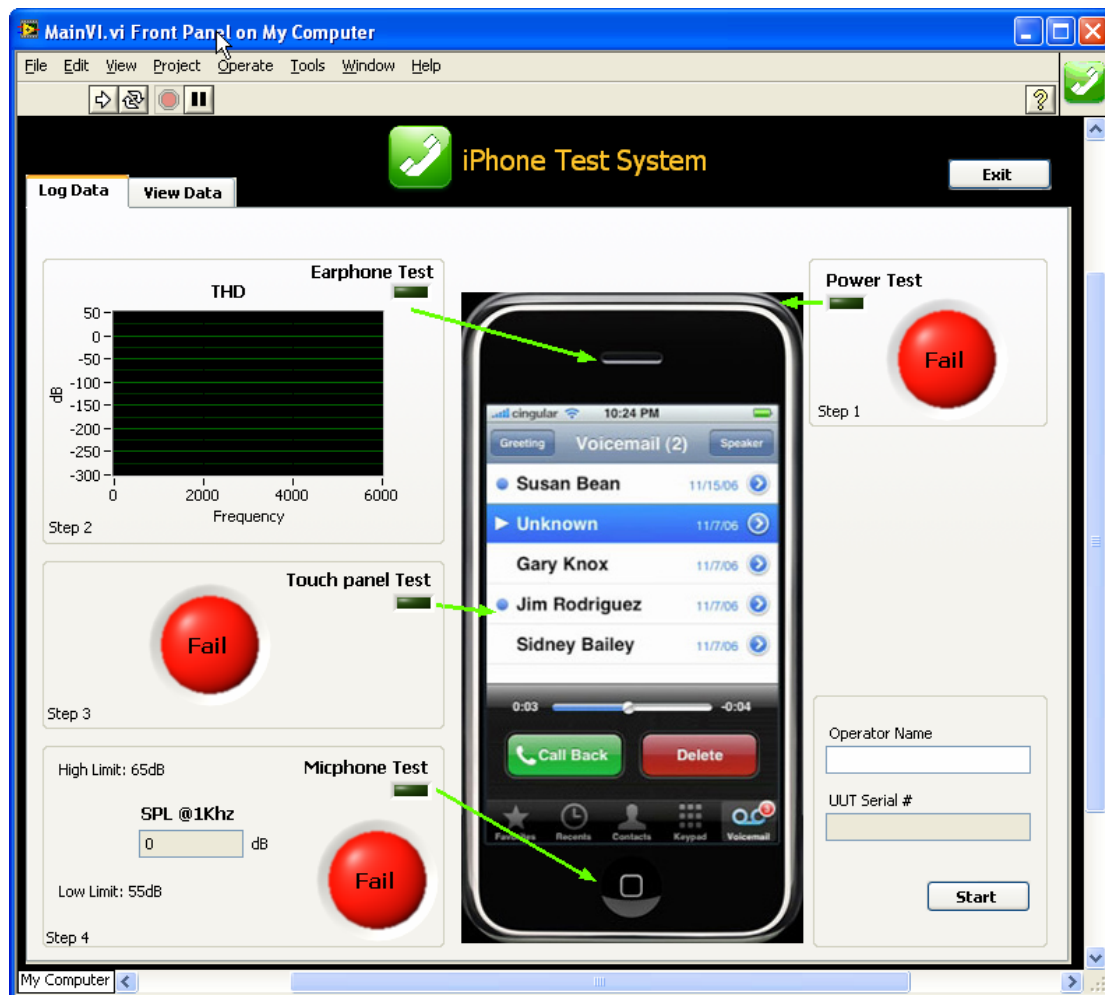


图 3.1 iPhone 测试界面

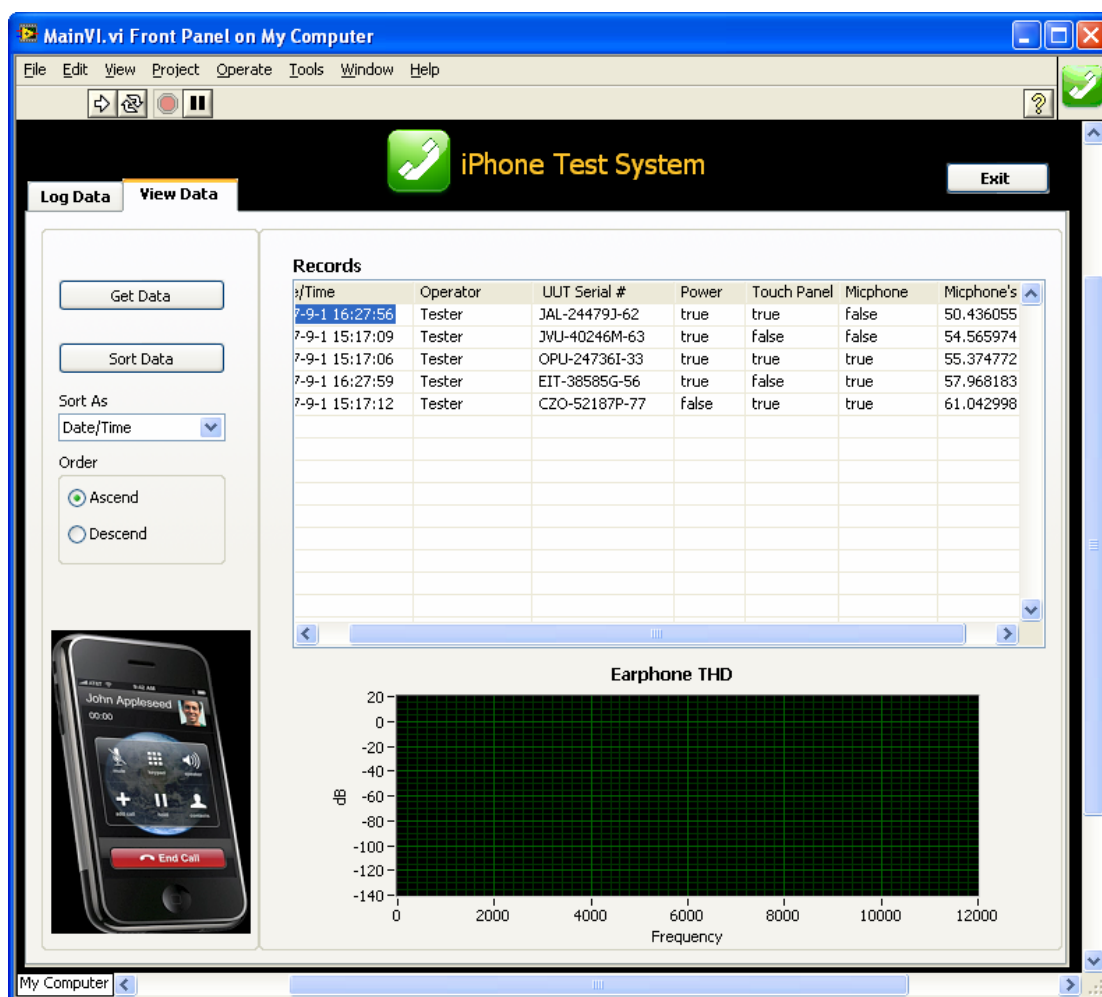


图 3.2 iPhone 测试数据分析界面

具体代码可以参考随本文的范例：DatabaseExample.lvprj。

3.2 生成可执行文件(*.exe)

在2.2.3节中，我们讨论了数据库连接的可移植性问题，如果使用本文介绍的方法，则在生成可执行文件的过程中不用关心UDL文件等问题。在生成可执行文件的过程中唯一需要注意的是把所用到的数据库文件添到到可执行文件的支持文件(Support files)中去。

步骤一：把数据库文件添加到工程中去，如图 3.3所示。

步骤二：在Build specification的Source file选项中，把数据库文件iPhoneData.mdb添加到Support file中，这样iPhoneData.mdb会被加入**Support Directory**。如图 3.4所示。

其它设置与通常生成可执行文件的基本步骤一样，点击”**Build**”按钮即可生成可执行文件。

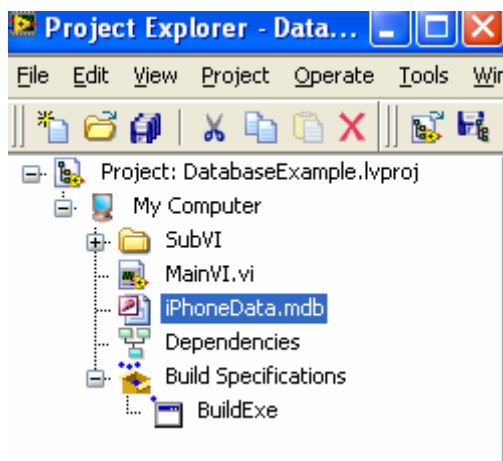


图 3.3 把数据库文件添加到工程

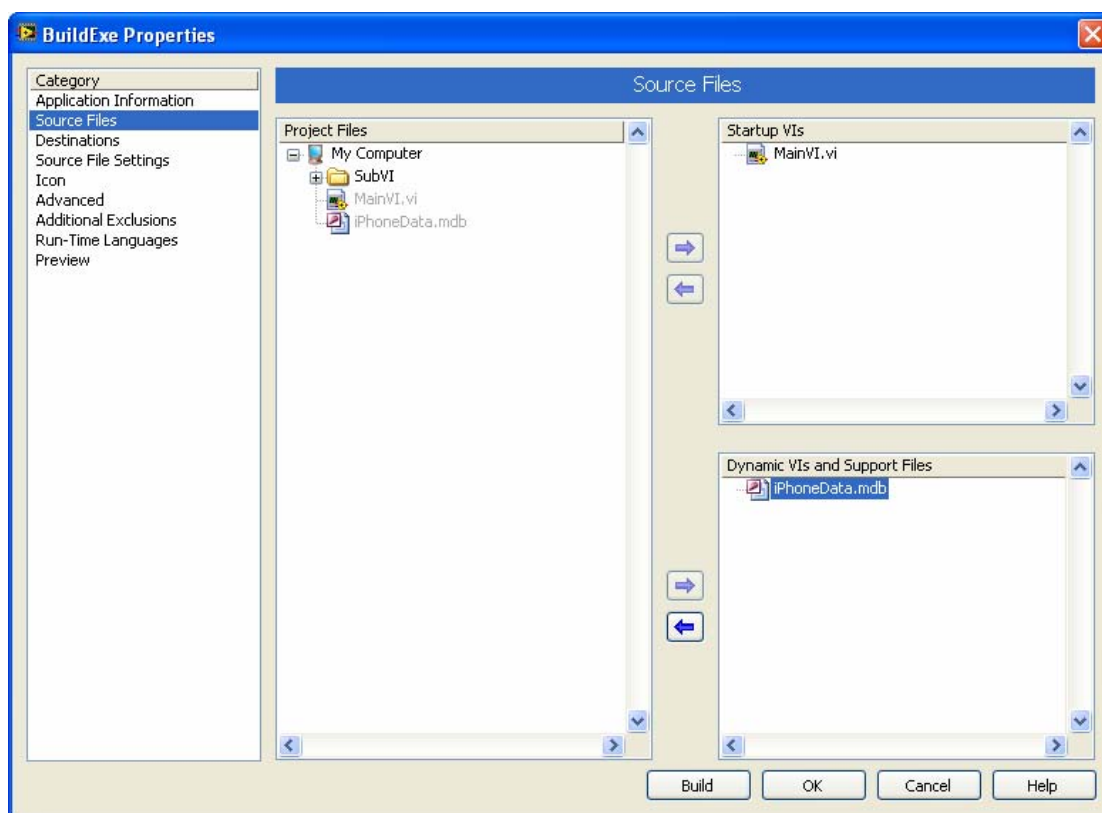


图 3.4 把数据库文件添加到 Support files 中

如果出现生成的可执行文件无法执行数据保存和检索的操作，请检查执行文件和数据库文件的相对路径是否正确。

3.3 生成安装文件(Installer)

生成安装文件，我们所要考虑的问题是，目标机上是否有对 LabVIEW 数据库工具包的支持——MDAC2.5。

LabVIEW 数据库工具包已经自带了 MDAC2.5 的安装文件——mdac_typ.exe，所以，我们需要把 mdac_typ.exe 添加到安装文件中。让安装文件在安装过程中把 MDAC2.5 替我们装到目标机上。

步骤一： 把mdac_typ.exe添加到工程中，如图 3.5所示。mdac_typ.exe在LabVIEW安装路径

\\National Instruments\\LabVIEW 8.2\\Database\\MDAC中。

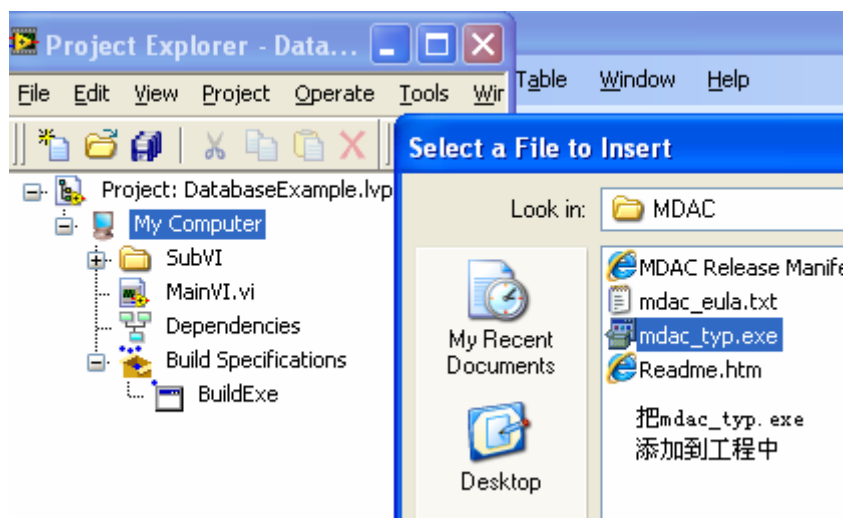


图 3.5 把 mdac_typ.exe 添加到工程中

步骤二：在Source File里面先把已编译好的可执行文件添加到ProgramfileFolder中去，然后把mdac_typ.exe添加到data文件夹中去，如图 3.6所示。

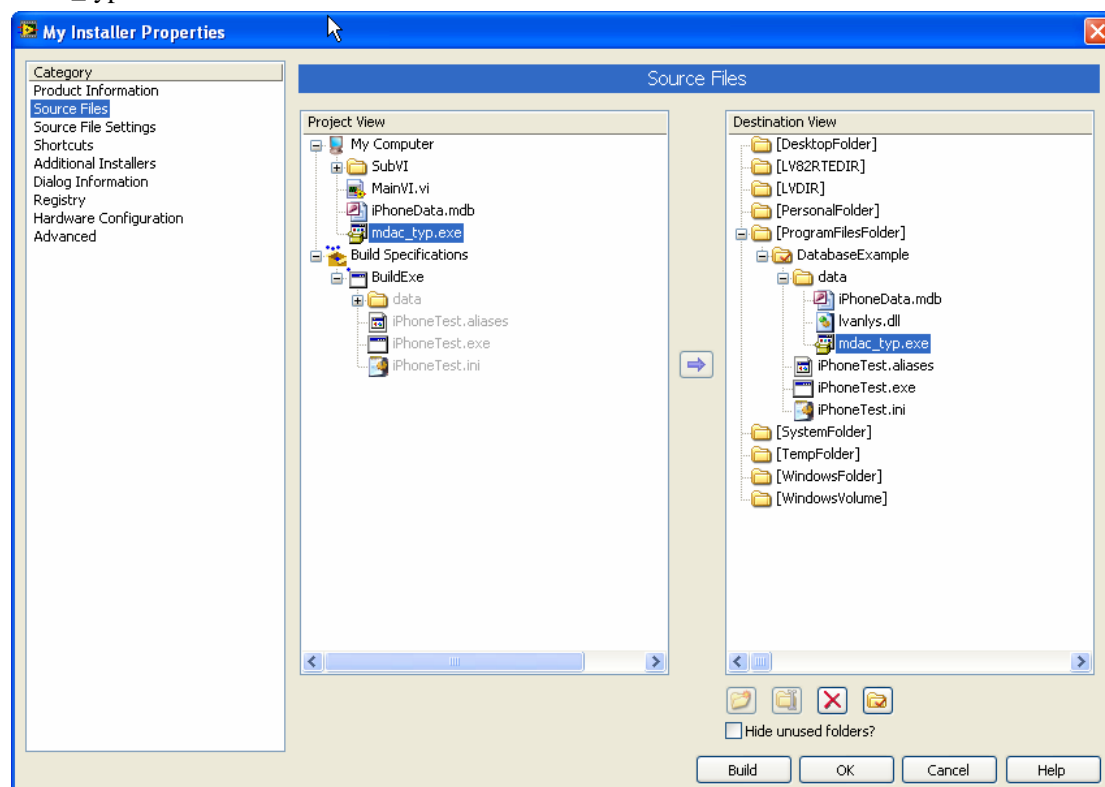


图 3.6 把 mdac_typ.exe 添加到 data 文件夹中

步骤三：在Advanced设置窗口中，把mdac_typ.exe添加到Run Executable after installation中，如图 3.7所示。

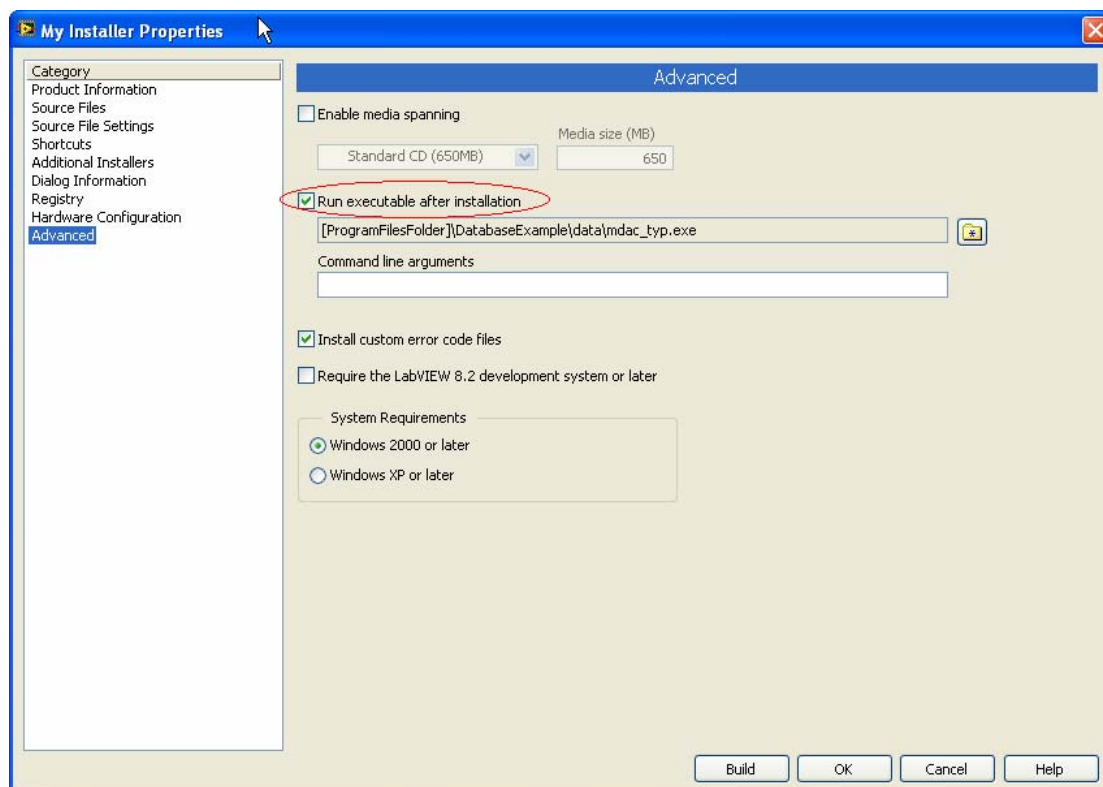


图 3.7 添加 mdac_typ.exe 到 Run Executable after installation

其余步骤与通常制作安装文件的步骤一致，最后点击 **Build** 即可生成安装文件了。

到此，大功终于告成了，还等什么，赶快亲手去试试☺。

第4章 后记

"I hear and I forget;
I see and I remember;
I do and I understand."

眼见为实，耳听为虚，任何事都要亲自付以行动才能真真正正的理解。在 LabVIEW 这样一个非常适合工程师和科学家的平台上，多尝试，多实践是工程能力增长的不二法门。

拙作肯定存在不少问题，有任何问题，可以给我Email: jing.zhang.zju@gmail.com，真心希望能与大家一起分享一起讨论。