

IBM[®] DB2[®] 通用数据库



应用程序构建指南

版本 7

IBM[®] DB2[®] 通用数据库



应用程序构建指南

版本 7

在使用本资料 and 它支持的产品之前，请参阅第371页的『附录E. 注意事项』中的一般信息。

本文档包含 IBM 的专利信息。它在许可协议下提供，并受版权法保护。本出版物包含的信息不包括任何产品保证，且本手册提供的任何声明不应作如此解释。

通过您当地的 IBM 代表或 IBM 分部可订购出版物，或者，通过致电 1-800-879-2755（在美国）或 1-800-IBM-4YOU（在加拿大）来订购出版物。

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 对于您所提供的任何信息，有权利以任何它认为适当的方式使用或散发，而不必对您负任何责任。

© Copyright International Business Machines Corporation 1993, 2000. All rights reserved.

目录

欢迎开发 DB2 应用程序	vii	用户定义函数样本	22
DB2 开发人员版	vii	DB2 调用层接口样本	22
安装信息	viii	Java 样本	24
DB2 应用程序开发书籍	ix	SQL 过程样本	25
DB2 程序设计接口	x	ADO、RDO 和 MTS 样本	27
使用嵌入式 SQL 语句	x	“对象链接和嵌入”样本	28
使用“DB2 调用层接口”	xii	命令行处理器样本	28
比较 DB2 CLI 与嵌入式动态 SQL	xii	记录管理用户出口样本	29
使用“Java 数据库链接”(JDBC)	xiv	第2章 设置	31
使用 DB2 API	xv	设置 OS/2 环境	32
使用“ActiveX 数据对象”(ADO) 和“远程 数据对象”(RDO)	xv	设置 UNIX 环境	33
使用 IBM、第三方和 ODBC 最终用户工具	xv	设置 Windows 32 位操作系统环境	34
DB2 功能部件	xvi	在服务器上启用通信	36
约束	xvii	Windows NT 和 Windows 2000	36
用户定义类型 (UDT) 和大对象 (LOB)	xvii	创建、编目和联编样本数据库	37
存储过程	xviii	创建	38
用户定义函数 (UDF)	xix	编目	39
OLE 自动化 UDF 和存储过程	xx	联编	40
触发器	xx	下一步	41
“DB2 通用数据库”的工具	xxi	第3章 构建 DB2 应用程序的一般信息	43
第1章 介绍	1	构建文件、Makefile 和错误检查实用程序	44
谁应使用本书	3	构建文件	44
如何使用本书	3	Makefile	46
突出显示的约定	3	错误检查实用程序	48
关于 DB2 应用程序开发客户机	4	Java Applet 和应用程序	49
支持的服务器	6	DB2 API 应用程序	50
按平台分类的受支持软件	6	DB2 调用层接口 (CLI) 应用程序	51
AIX	7	嵌入式 SQL 应用程序	51
HP-UX	8	存储过程	53
Linux	9	用户定义函数 (UDF)	54
OS/2	9	多线程应用程序	54
PTX	9	用 C++ 编写 UDF 和存储过程的考虑事项	54
Silicon Graphics IRIX	10	第4章 构建 Java Applet 和应用程序	57
Solaris	10	设置环境	58
Windows 32 位操作系统	11	AIX	58
样本程序	11	HP-UX	59
DB2 API 非嵌入式 SQL 样本	16	Linux	60
DB2 API 嵌入式 SQL 样本	19	OS/2	61
不含 DB2 API 的嵌入式 SQL 样本	21		

PTX	62	多线程应用程序	119
Silicon Graphics IRIX	63	VisualAge C++ 版本 4.0.	120
Solaris	65	DB2 CLI 应用程序	121
Windows 32 位操作系统	66	使用 DB2 API 的 DB2 CLI 应用程序	123
Java 样本程序	68	DB2 CLI 存储过程	124
JDBC 程序	68	DB2 API 应用程序	127
Applet	68	嵌入式 SQL 应用程序	128
应用程序	69	嵌入式 SQL 存储过程	130
存储过程	70	用户定义函数 (UDF)	133
SQLJ 程序	71	IBM COBOL Set AIX 版	135
Applet	74	使用编译程序	135
应用程序	75	DB2 API 和嵌入式 SQL 应用程序	136
存储过程	76	嵌入式 SQL 存储过程	138
用户定义函数 (UDF)	79	Micro Focus COBOL	141
DB2 Java Applet 的一般要点	80	使用编译程序	141
第5章 构建 SQL 过程	83	DB2 API 和嵌入式 SQL 应用程序	142
设置 SQL 过程环境	83	嵌入式 SQL 存储过程	144
创建 SQL 过程	87	REXX	148
调用 SQL 过程	87	第7章 构建 HP-UX 应用程序	151
使用 CALL 命令	87	HP-UX C	152
OS/2 DB2 CLI 客户机应用程序	88	DB2 CLI 应用程序	152
OS/2 嵌入式 SQL 客户机应用程序	88	使用 DB2 API 的 DB2 CLI 应用程序	154
UNIX DB2 CLI 客户机应用程序	89	DB2 CLI 存储过程	154
UNIX 嵌入式 SQL 客户机应用程序	89	DB2 API 和嵌入式 SQL 应用程序	157
Windows DB2 CLI 客户机应用程序	90	嵌入式 SQL 存储过程	159
Windows 嵌入式 SQL 客户机应用程序	90	用户定义函数 (UDF)	161
第6章 构建 AIX 应用程序	91	多线程应用程序	163
重要考虑事项	91	HP-UX C++	165
安装和运行 IBM 与 Micro Focus COBOL	92	DB2 API 和嵌入式 SQL 应用程序	165
存储过程和 UDF 的入口点	92	嵌入式 SQL 存储过程	167
存储过程和 CALL 语句	93	用户定义函数 (UDF)	170
UDF 和 CREATE FUNCTION 语句	94	多线程应用程序	172
IBM C	95	Micro Focus COBOL	173
DB2 CLI 应用程序	95	使用编译程序	173
使用 DB2 API 的 DB2 CLI 应用程序	98	DB2 API 和嵌入式 SQL 应用程序	174
DB2 CLI 存储过程	98	嵌入式 SQL 存储过程	176
DB2 API 和嵌入式 SQL 应用程序	101	第8章 构建 Linux 应用程序	179
嵌入式 SQL 存储过程	104	Linux C	179
用户定义函数 (UDF)	107	DB2 CLI 应用程序	179
多线程应用程序	109	使用 DB2 API 的 DB2 CLI 应用程序	182
IBM C Set++	110	DB2 CLI 存储过程	182
DB2 API 和嵌入式 SQL 应用程序	110	DB2 API 和嵌入式 SQL 应用程序	184
嵌入式 SQL 存储过程	113	嵌入式 SQL 存储过程	187
用户定义函数 (UDF)	116	用户定义函数 (UDF)	189

多线程应用程序	191	UDF 的 DB2 CLI 客户机应用程序	252
Linux C++	192	DB2 API 和嵌入式 SQL 应用程序	252
DB2 API 和嵌入式 SQL 应用程序	192	多线程应用程序	256
嵌入式 SQL 存储过程	195	MIPSpro C++	257
用户定义函数 (UDF)	197	DB2 API 和嵌入式 SQL 应用程序	257
多线程应用程序	199	多线程应用程序	260
第9章 构建 OS/2 应用程序	201	第12章 构建 Solaris 应用程序	263
IBM VisualAge C++ OS/2 版的版本 3	201	SPARCompiler C	263
DB2 CLI 应用程序	202	DB2 CLI 应用程序	264
使用 DB2 API 的 DB2 CLI 应用程序	204	使用 DB2 API 的 DB2 CLI 应用程序	266
DB2 CLI 存储过程	204	DB2 CLI 存储过程	267
DB2 API 和嵌入式 SQL 应用程序	206	DB2 API 和嵌入式 SQL 应用程序	269
嵌入式 SQL 存储过程	209	嵌入式 SQL 存储过程	272
用户定义函数 (UDF)	211	用户定义函数 (UDF)	275
IBM VisualAge C++ OS/2 版的版本 4.0	213	多线程应用程序	278
IBM VisualAge COBOL OS/2 版	213	SPARCompiler C++	279
使用编译程序	213	DB2 API 和嵌入式 SQL 应用程序	279
嵌入式 SQL 应用程序	214	嵌入式 SQL 存储过程	282
嵌入式 SQL 存储过程	217	用户定义函数 (UDF)	285
Micro Focus COBOL	219	多线程应用程序	288
使用编译程序	219	Micro Focus COBOL	289
DB2 API 和嵌入式 SQL 应用程序	220	使用编译程序	289
嵌入式 SQL 存储过程	222	DB2 API 和嵌入式 SQL 应用程序	289
REXX	224	嵌入式 SQL 存储过程	292
第10章 构建 PTX 应用程序	225	第13章 构建 Windows 32 位操作系统的应	297
ptx/C	225	用程序	297
DB2 CLI 应用程序	225	Microsoft Visual Basic	299
使用 DB2 API 的 DB2 CLI 应用程序	228	ActiveX 数据对象 (ADO)	299
DB2 CLI 存储过程	228	远程数据对象 (RDO)	300
DB2 API 和嵌入式 SQL 应用程序	231	对象链接和嵌入 (OLE) 自动化	302
嵌入式 SQL 存储过程	233	Microsoft Visual C++	302
用户定义函数 (UDF)	235	ActiveX 数据对象 (ADO)	303
多线程应用程序	237	对象链接和嵌入 (OLE) 自动化	303
ptx/C++	238	DB2 CLI 应用程序	304
DB2 API 和嵌入式 SQL 应用程序	238	使用 DB2 API 的 DB2 CLI 应用程序	307
嵌入式 SQL 存储过程	241	DB2 CLI 存储过程	307
用户定义函数 (UDF)	243	DB2 API 和嵌入式 SQL 应用程序	310
多线程应用程序	245	嵌入式 SQL 存储过程	312
第11章 构建 Silicon Graphics IRIX 应用程	247	用户定义函数 (UDF)	315
序	247	IBM VisualAge C++ 版本 3.5	318
MIPSpro C	248	DB2 CLI 应用程序	318
DB2 CLI 应用程序	248	使用 DB2 API 的 DB2 CLI 应用程序	320
使用 DB2 API 的 DB2 CLI 应用程序	251	DB2 CLI 存储过程	321
存储过程的 DB2 CLI 客户机应用程序	251	DB2 API 和嵌入式 SQL 应用程序	323

嵌入式 SQL 存储过程	326	附录C. 问题确定	351
用户定义函数 (UDF)	329	附录D. 使用 DB2 资料库	353
IBM VisualAge C++ 版本 4.0	331	DB2 PDF 文件和打印的书籍	353
IBM VisualAge COBOL	331	DB2 信息	353
使用编译程序	331	打印 PDF 书籍	361
DB2 API 和嵌入式 SQL 应用程序	332	订购打印书籍	362
嵌入式 SQL 存储过程	334	DB2 联机文档	363
Micro Focus COBOL	336	存取联机帮助	363
使用编译程序	336	查看联机信息	365
DB2 API 和嵌入式 SQL 应用程序	337	使用 DB2 向导	367
嵌入式 SQL 存储过程	339	设置文档服务器	368
Object REXX	341	搜索联机信息	369
附录A. 关于数据库管理程序实例	343	附录E. 注意事项	371
附录B. 迁移应用程序	345	注册商标	373
问题	346	索引	375
条件	348	与 IBM 联系	381
其他迁移考虑事项	349	产品信息	381

欢迎开发 DB2 应用程序

本前言提供开发 DB2 应用程序所需的入门信息，特别介绍了“DB2 开发人员版”产品。

『DB2 开发人员版』一节提供了安装信息，以满足您进行特殊开发的需要。此信息帮助您确定如何从“IBM DB2 通用开发人员版的版本 7.1”或“IBM DB2 个人开发人员版的版本 7.1”安装 DB2。

第ix页的『DB2 应用程序开发书籍』一节描述 DB2 资料库中介绍应用程序开发的主要书籍。

第x页的『DB2 程序设计接口』一节讲述 DB2 应用程序开发所涉及的重要的程序设计接口概念。

第xvi页的『DB2 功能部件』一节描述可用来进行 DB2 应用程序开发的主要功能部件。

DB2 开发人员版

“DB2 通用数据库”为应用程序开发提供了两个产品程序包：“DB2 个人开发人员版”和“DB2 通用开发人员版”。“个人开发人员版”提供在 OS/2、Linux 和 Windows 32 位操作系统上运行的“DB2 通用数据库个人版”和“DB2 Connect 个人版”产品。“DB2 通用开发人员版”提供在这些平台以及在 AIX、HP-UX、PTX、Silicon Graphics IRIX 和 Solaris** 操作环境 **上运行的 DB2 产品。

使用这些产品所提供的软件，可开发和测试在一种操作系统上运行、而存取同一种或另一种操作系统上的数据库的应用程序。例如，您可创建一个应用程序，它在 Windows NT 操作系统上运行，但存取 UNIX 平台如 AIX 上的数据库。参阅“许可证协议”，以了解使用“开发人员版”产品的条款和条件。

“开发人员版”包装盒装有几片 CD-ROM，它们含有开发和测试应用程序需要的所有代码。在每个包装盒中，您将找到：

- 几个操作系统版本的“DB2 通用数据库”产品 CD-ROM。每个 CD-ROM 包含用于受支持操作系统的 DB2 服务器、管理客户机、应用程序开发客户机和运行期客户机。提供这些 CD-ROM，只是供您测试应用程序。如果您的公司需要安装和使用数据库，则需要购买该产品以获得“通用数据库”产品的有效许可

证。“个人开发人员版”包装盒装有 OS/2、Linux 和 Windows 32 位操作系统的“个人版”产品 CD-ROM。“通用开发人员版”包装盒装有 DB2 支持的所有操作系统版本的 CD-ROM。

- 包含几个操作系统的 Netscape 浏览器的 CD-ROM。如果您的机器上没有安装浏览器，可用 CD-ROM 上的浏览器查看 HTML 信息。
- 包含 PDF 格式的 DB2 书籍的 DB2 出版物 CD-ROM。
- 受支持平台的 DB2 Extender。
- 受支持平台的 OLAP Starter Kit。安装 OLAP Starter Kit 之前必须安装 DB2 服务器。
- 此外，您还可得到其他有助于应用程序开发的软件的副本。此软件可能会随时更新，且只有持有许可证协议才能使用。

“通用开发人员版”还包含以下内容：

- 包含所有平台的管理客户机的一套 CD-ROM。这些客户机包含管理数据库的工具，如“控制中心”和“事件分析程序”。还允许您在任何系统上运行应用程序。
- 包含所有平台的应用程序开发客户机的一套 CD-ROM。这些客户机包含应用程序开发工具、样本程序和头文件。每个 DB2 AD 客户机都包括开发应用程序需要的一切。
- 包含所有平台的运行期客户机的一套 CD-ROM。可在任何系统上从运行期客户机运行应用程序。运行期客户机没有管理客户机的某些功能部件，如“DB2 控制中心”和“事件分析程序”，因此占用的空间较少。
- 包含受支持平台的 Net.Data 的 CD-ROM。

安装信息

每个 DB2 产品的 CD-ROM 都包含可直接从 CD-ROM 查看的 HTML 格式的安装信息。为每种受支持平台提供了快速入门一书，它解释如何安装 DB2 服务器、管理客户机、应用程序开发客户机和运行期客户机。

可从 *安装和配置补遗* 一书中获取其他的信息。此书只能从客户机 CD-ROM 中查看。

您需要的书籍位于您所用的语言的子目录中。CD-ROM 中的 README.TXT 文件告诉您可在 CD-ROM 中何处找到该书籍文件。

运行浏览器后，单击该书子目录中的 index.htm 文件。以下是快速入门和安装和配置补遗两书的子目录。

db2i2 *DB2 (OS/2 版) Quick Beginnings*

db2ix *DB2 (UNIX 版) 快速入门*

db2i6 *DB2 Windows 版快速入门*

db2iy *安装和配置补遗*

DB2 出版物 CD-ROM 包含所有与该产品一起交付的 DB2 书籍的 PDF 文件。可使用 Adobe Acrobat Reader 直接从 CD-ROM 查看这些书籍。您所用语言的可用 DB2 书籍位于相应的语言子目录中。有关安装信息，可访问快速入门和安装和配置补遗两书。其文件名以上面列出的字符集开头。

有关打印 PDF 文件的说明，参阅第361页的『打印 PDF 书籍』。

有关如何访问这些书籍的全部细节，参阅该 CD-ROM 中的 README.TXT 文件。

DB2 应用程序开发书籍

第353页的『附录D. 使用 DB2 资料库』中描述了 DB2 资料库。DB2 书籍一般分为两类：一类提供有关管理 DB2 数据库的信息，另一类提供有关 DB2 应用程序开发的信息。某些书籍同时提供这两类信息。作为 DB2 应用程序开发人员，您可能发现会同时参考这两类 DB2 书籍。但您的重点，以及本节的重点是 DB2 资料库中主要的应用程序开发书籍。

主要有两本书介绍应用程序的设计。它们是 *CLI Guide and Reference*，该书讨论如何编写 DB2 CLI 应用程序；*Application Development Guide*，该书讨论除 DB2 CLI 之外的所有其他不同种类的 DB2 程序设计。这两本书也包含参考信息。

在您现在阅读的应用程序构建指南这本书中可找到有关设置开发环境以及编译、连接与运行应用程序的信息。

有关 SQL 语句和函数的语法，可参考 *SQL Reference*。

有两本书属于 DB2 资料库中的管理类别，它们也是进行应用程序设计的重要参考书籍。其中 *Administrative API Reference* 包含用于管理 DB2 数据库的所有管理功能的详情。您可能会觉得在应用程序中使用 DB2 API 来代替 SQL 语句，或一起使用它们很方便。*Command Reference* 包含所有 DB2 命令（除 SQL 语句以外）的详情，并说明如何使用“DB2 命令行处理器”（CLP）。

要解决环境设置或程序开发中的问题，可参考 *Troubleshooting Guide*。它帮助您确定错误来源，校正问题，并有助于在向“DB2 客户服务”机构咨询时使用诊断工具。信息参考包含 DB2 出错信息的完整列表和说明，它是您调试应用程序时非常重要的一本参考书籍。

DB2 资料库中的这些书籍和其他书籍，以及 DB2 环境中可用的联机信息，将提供开发 DB2 应用程序需要的信息。有关非专门介绍 DB2 的开发信息，可参阅供应商提供的有关您所用的编译程序、解释程序或其他开发工具的文档。

DB2 程序设计接口

可使用几种不同的程序设计接口来存取 DB2 数据库。您可以：

1. 使用 DB2 API 执行管理功能，如备份和复原数据库。
2. 将静态和动态 SQL 语句嵌入应用程序。
3. 在应用程序中编写“DB2 调用层接口”(DB2 CLI) 的函数调用，以调用动态 SQL 语句。
4. 开发调用“Java 数据库链接”应用程序设计接口 (JDBC API) 的 Java 应用程序和 Applet。
5. 开发符合“数据存取对象”(DAO) 和“远程数据对象”(RDO) 规范的 Microsoft Visual Basic 和 Visual C++ 应用程序，以及使用“对象链接和嵌入数据库 (OLE DB) 桥接”的“ActiveX 数据对象”(ADO) 应用程序。
6. 使用 IBM 或第三方工具(如 Net.Data、Excel、Perl) 以及“开放式数据库链接”(ODBC) 最终用户工具(如 Lotus Approach 及其程序设计语言 LotusScript) 开发应用程序。

应用程序存取 DB2 数据库的方式将取决于想要开发的应用程序类型。例如，如果想开发数据输入应用程序，可以选择将静态 SQL 语句嵌入应用程序。如果想开发在万维网 (WWW) 上执行查询的应用程序，可能要选择 Net.Data、Perl 或 Java。

使用嵌入式 SQL 语句

“结构化查询语言”(SQL) 是一种数据库接口语言，它用来存取并处理 DB2 数据库中的数据。可以将 SQL 语句嵌入应用程序，使应用程序能执行 SQL 支持的任何任务，如检索或存储数据。通过使用 DB2，可以用 C/C++、COBOL、FORTRAN、Java (SQLJ) 以及 REXX 程序设计语言来编写嵌入式 SQL 应用程序。

嵌入了 SQL 语句的应用程序称为主程序。用于创建主程序的程序设计语言称为主语言。用这种方式定义程序和语言，是因为它们包含了 SQL 语句。

对于静态 SQL 语句，在编译前就应知道 SQL 语句类型以及表名和列名。唯一未知的是语句正搜索或更新的特定数据值。可以用主语言变量表示那些值。在运行应用程序之前，要预编译、联编然后编译静态 SQL 语句。静态 SQL 最好在模式变动不大的数据库中运行。否则，这些语句很快会过时。

相反，动态 SQL 语句是应用程序在运行期构建并执行的那些语句。一个提示最终用户输入 SQL 语句的关键部分（如要搜索的表和列的名称）的交互式应用程序是一个典型的动态 SQL 示例。应用程序在运行时构建 SQL 语句，然后提交这些语句进行处理。

可以编写只有静态 SQL 语句或只有动态 SQL 语句，或者兼有两者的应用程序。

一般来说，静态 SQL 语句最适合用于执行预定义事务的高性能应用程序。预订系统是这种应用程序的一个典型示例。

一般来说，动态 SQL 语句最适合于必须在运行期指定事务的、要快速更改数据库的应用程序。交互式查询界面是这种应用程序一个很好的示例。

将 SQL 语句嵌入应用程序时，必须按以下步骤预编译应用程序并将其与数据库联编：

1. 创建源文件，以包含带嵌入式 SQL 语句的程序。
2. 连接数据库，然后预编译每个源文件。

预编译程序将每个源文件中的 SQL 语句转换成对数据库管理程序的 DB2 运行期 API 调用。预编译程序还在数据库中生成一个存取程序包，并可选择生成一个联编文件（如果您指定要创建一个的话）。

存取程序包包含由 DB2 优化器为应用程序中的静态 SQL 语句选择的存取方案。这些存取方案包含数据库管理程序执行静态 SQL 语句所需的信息，以便该管理程序可以用优化器确定的最有效的方式来执行这些语句。对于动态 SQL 语句，优化器在您运行应用程序时创建存取方案。

联编文件包含创建存取程序包所需要的 SQL 语句和其他数据。可以使用联编文件在以后重新联编应用程序，而不必首先预编译应用程序。重新联编创建针对当前数据库状态的优化存取方案。如果应用程序将存取与预编译时所用数据库不同的数据库，则必须重新联编应用程序。如果数据库统计信息自上次联编后已经更改，您应重新联编应用程序。

3. 使用主语言编译程序编译修改的源文件（以及其他无 SQL 语句的文件）。
4. 将目标文件与 DB2 和主语言库连接，以生成一个可执行程序。
5. 如果在预编译时未对联编文件进行联编；或者准备存取不同的数据库，则应对联编文件进行联编以创建存取程序包。
6. 运行该应用程序。此应用程序使用程序包中的存取方案存取数据库。

Java 嵌入式 SQL (SQLJ)

DB2 AD 客户机提供 DB2 Java 嵌入式 SQL (SQLJ) 支持。利用 DB2 SQLJ 支持和 DB2 JDBC 支持，可构建和运行 SQLJ Applet、应用程序和存储过程。它们包含静态 SQL，且使用与 DB2 数据库联编的嵌入式 SQL 语句。

有关 DB2 SQLJ 支持的详情，访问以下的 Web 页面：

<http://www.ibm.com/software/data/db2/java>

使用“DB2 调用层接口”

DB2 CLI 是 C 或 C++ 应用程序用来存取 DB2 数据库的程序设计接口。DB2 CLI 基于 Microsoft 的“开放式数据库链接”(ODBC) 规范以及 ISO CLI 标准。因为 DB2 CLI 基于业界标准，因此已经熟悉这些数据库接口的应用程序员可在较短时间内掌握它。

使用 DB2 CLI 时，应用程序将动态 SQL 语句作为函数自变量传递给数据库管理程序进行处理。因此，DB2 CLI 可以代替嵌入式动态 SQL。

也可能在 CLI、ODBC 或 JDBC 应用程序中将 SQL 语句作为静态 SQL 运行。CLI/ODBC/JDBC 静态简要功能部件允许应用程序的最终用户在很多场合下使用静态 SQL，而不使用动态 SQL。有关详情，参阅：

<http://www.ibm.com/software/data/db2/udb/staticcli>

可不使用 ODBC 驱动程序管理器构建 ODBC 应用程序，只需要将您的应用程序与 UNIX 上的 libdb2 以及 OS/2 和 Windows 32 位操作系统上的 db2cli.lib 相连，就可在任何平台上使用 DB2 的 ODBC 驱动程序。DB2 CLI 样本程序演示了这种操作。在 UNIX 上这些样本程序位于 sqllib/samples/cli 中，在 OS/2 和 Windows 32 位操作系统上位于 %DB2PATH%\samples\cli 中。

不需要预编译或联编 DB2 CLI 应用程序，因为这些程序使用与 DB2 一起提供的公共存取程序包。您只需简单地编译并连接应用程序。

但在 DB2 CLI 或 ODBC 应用程序可以存取 DB2 数据库之前，DB2 AD 客户机所带的 DB2 CLI 联编文件必须与要存取的每个 DB2 数据库联编。这在首次连接数据库时自动完成，但建议数据库管理员从要存取 DB2 数据库的每个平台上的一个客户机中对联编文件进行联编。有关联编说明，参阅第40页的『联编』。

例如，假设您有 OS/2、AIX 以及 Windows 95 客户机，每个客户机存取两个 DB2 数据库。管理员应从一个 OS/2 客户机对要存取的每个数据库联编一次联编文件。接着，管理员应从一个 AIX 客户机对要存取的每个数据库联编一次联编文件。最后，管理员应在一个 Windows 95 客户机上做同样的操作。

比较 DB2 CLI 与嵌入式动态 SQL

可以使用嵌入式动态 SQL 语句或 DB2 CLI 开发动态应用程序。在这两种情况下，都是在运行期准备 SQL 语句并处理这些语句。每种方法都有其独特的优点，如下所示。

DB2 CLI 的优点

可移植性

DB2 CLI 应用程序使用标准函数集将 SQL 语句传递至数据库。在运行 DB2 CLI 应用程序之前，只需编译并连接它们。相反，在运行嵌入式 SQL 应用程序之前，必须先预编译，然后编译，再将它们与数据库联编。这一过程将应用程序有效地连接至某一特定数据库。

无需联编

不需要将个别 DB2 CLI 应用程序与它们存取的每个数据库进行联编。对于所有 DB2 CLI 应用程序，只需与 DB2 CLI 一起交付的联编文件联编一次就可以了。这可以显著减少管理应用程序所花费的时间。

扩充取装和输入

DB2 CLI 函数使您能够用单个调用检索数据库中的多行，并将这些行放入一个数组中。DB2 CLI 函数也允许您使用一系列的输入变量多次执行一个 SQL 语句。

用于编目的一致界面

数据库系统包含目录表，这些表含有数据库及其用户的信息。这些目录的形式在不同系统中可以各不相同。DB2 CLI 提供了一致的界面来查询关于表、列、外部关键字和主关键字以及用户特权等部件的目录信息。使应用程序不受数据库服务器不同发行版的目录更改影响，并且不受数据库服务器之间的差异的影响。不必编写针对某特定服务器或产品版本的目录查询。

扩充数据转换

DB2 CLI 在 SQL 和 C 数据类型之间自动转换数据类型。例如，将任何 SQL 数据类型取装为 C char 数据类型，即将该 SQL 数据类型转换成字符串表示的类型。这使得 DB2 CLI 非常适合交互式查询应用程序。

无需全局数据区

DB2 CLI 不需要应用程序控制的、常常是复杂的全局数据区，如 SQLDA 和 SQLCA，这些数据区一般与嵌入式 SQL 应用程序相关联。相反，DB2 CLI 自动分配并控制所需的数据结构，并给应用程序提供一个句柄来引用这些数据结构。

通过存储过程检索结果集

DB2 CLI 应用程序可以检索由驻留在服务器上的存储过程生成的多行和多个结果集。

可滚动游标

DB2 CLI 支持服务器端的可滚动游标，该游标可用于与数组输出连接。这对在滚动框中显示数据库信息的 GUI 应用程序是很有用的，这些滚动框利用了 Page Up、Page Down、Home 以及 End 键。您可以声明一个游标是

可滚动的，然后在结果集中前移或后移一行或多行。也可以通过如下方法来读取一些行：指定自结果集的当前行、起始行或结尾行，或者您先前标记的特定行开始的偏移。

嵌入式动态 SQL 的优点

所有的 DB2 CLI 用户共享相同的特权。嵌入式 SQL 通过向程序包的特定用户授予执行特权来提供更细致的安全性。

嵌入式 SQL 不只支持 C 和 C++。如果您喜欢用另一种语言编程，这可算是一个优点。

动态 SQL 与静态 SQL 一般都非常兼容。如果您已知道如何设计静态 SQL 程序，那么改为用动态 SQL 设计程序可能不象改改用 DB2 CLI 设计程序那样困难。

使用“Java 数据库链接”(JDBC)

DB2 的 Java 支持包括 JDBC，一个以开发商为中心的动态 SQL 接口，它通过标准的 Java 方法提供对应用程序的数据存取。JDBC 与 DB2 CLI 相似之处在于您不必预编译或联编 JDBC 程序。由于是以开发商为中心的标准，JDBC 应用程序提供了增强的可移植性。使用 JDBC 编写的应用程序只使用动态 SQL。

在通过 Internet 存取 DB2 数据库方面，JDBC 特别有用。使用 Java 程序设计语言，可以开发 JDBC Applet 和应用程序，这些程序使用网络连接存取并处理远程 DB2 数据库中的数据。也可创建 JDBC 存储过程，它们驻留在服务器上，存取数据库服务器，并将信息返回给调用存储过程的远程客户机应用程序。

JDBC API 类似于 CLI/ODBC API，它提供一种标准的方法从 Java 代码存取数据库。Java 代码将 SQL 语句作为函数自变量传递给 DB2 JDBC 驱动程序。此驱动程序处理客户机 Java 代码对 JDBC API 的调用。

Java 的可移植性使您能够将 DB2 存取权传递给多个平台上的客户机，而只需要一个启用 Java 的 web 浏览器。

Java 应用程序依靠 DB2 客户机与 DB2 连接。象任何其他应用程序一样，您可从桌面或命令行启动应用程序。DB2 JDBC 驱动程序处理应用程序对 JDBC API 的调用，并使用客户机连接将请求传递给服务器并接收结果。

Java Applet 不需要 DB2 客户机连接。通常是将 Applet 嵌入“超文本标记语言”(HTML) Web 页面。

要运行您的 Applet，在客户机上只需要一个可运行 Java 的 web 浏览器或 Applet 浏览器。当装入 HTML 页时，浏览器将 Java Applet 下载至您的机器上，然后下

载 Java 类文件和 DB2 的 JDBC 驱动程序。当 Applet 调用 JDBC API 来连接 DB2 时，JDBC 驱动程序通过驻留在 web 服务器上的 JDBC Applet 服务器来建立与 DB2 数据库的单独的网络连接。

有关 DB2 JDBC 支持的详情，访问以下的 Web 页面：

<http://www.ibm.com/software/data/db2/java>

使用 DB2 API

您的应用程序可能需要执行某些数据库管理任务，如创建、激活、备份或复原数据库。DB2 提供了大量 API，因此可以从您的应用程序（包括嵌入式 SQL 和 DB2 CLI 应用程序）中执行这些任务。这使您能够将相同的管理功能编写到应用程序中，可使用 DB2 服务器管理工具来执行它们，这在第xxi页的『“DB2 通用数据库”的工具』中有讨论。

另外，可能需要执行一些只能使用 DB2 API 执行的特定任务。例如，可能要检索某个错误信息的文本，以便应用程序可以将该文本呈现给最终用户。要检索该信息，必须使用“获取出错信息 API”。

使用“ActiveX 数据对象” (ADO) 和“远程数据对象” (RDO)

可以编写符合“数据存取对象” (DAO) 和“远程数据对象” (RDO) 规范的 Microsoft Visual Basic 和 Microsoft Visual C++ 数据库应用程序。DB2 还支持使用 Microsoft OLE DB 至 ODBC 桥接的“ActiveX 数据对象” (ADO) 应用程序。

“ActiveX 数据对象” (ADO) 可用来编写一个应用程序，以便通过 OLE DB 提供者存取和操纵数据库服务器中的数据。ADO 的主要优点是开发时间短、易于使用以及占用磁盘少。

“远程数据对象” (RDO) 提供一个通过 ODBC 存取远程数据源的信息模型。RDO 提供一个对象集，它使得与数据库连接、执行查询和存储过程、处理结果和落实服务器更改更容易。RDO 是为存取远程 ODBC 关系数据源而专门设计的，它使得使用 ODBC 更容易，而不必进行复杂的应用程序编程。

使用 IBM、第三方和 ODBC 最终用户工具

要执行基本任务，如查询数据库，可使用 Net.Data 或 Perl。

Net.Data 使您可以通过 web 应用程序经 Internet 和企业内部网存取 DB2 数据。它利用 web 服务器接口 (API)，提供比公共网关接口 (CGI) 应用程序更高的性能。Net.Data 用 Java、REXX、Perl 和 C++ 等语言支持客户端处理以及服务器端处理。Net.Data 提供条件逻辑和丰富的宏语言。Net.Data Web 页面位于：

<http://www.ibm.com/software/data/net.data/>

DB2 通过 DBD::DB2 驱动程序支持数据存取的“Perl 数据库接口”(DBI) 规范。
“DB2 通用数据库”的 Perl DBI web 站点位于:

<http://www.ibm.com/software/data/db2/perl/>

它包含最新的 DBD::DB2 驱动程序和相关信息。

也可使用 ODBC 最终用户工具如 Lotus Approach、Microsoft Access 以及 Microsoft Visual Basic 来创建应用程序以执行这些任务。ODBC 工具提供一个比使用高级程序设计语言更简单的方法来开发应用程序。

Lotus Approach 提供两种方式来存取 DB2 数据。可以使用图形界面执行查询、创建报表和分析数据。或者可使用 LotusScript 开发应用程序，LotusScript 是一种功能完备的、面向对象的程序设计语言，它提供大量的对象、事件、方法和特性以及一个内部程序编辑器。

DB2 功能部件

DB2 带有在服务器上运行的各种功能部件，可以使用这些功能部件来补充或扩展应用程序。使用了 DB2 功能部件，就不必编写自己的代码来执行同样的任务。DB2 也允许您将部分代码存储在服务器上，而不必将所有代码都保存在客户机应用程序中。这有利于提高性能并易于维护。

有些功能部件用于保护数据和定义数据之间的关系。还有一些与对象相关的功能部件，用于创建灵活的高级应用程序。可用多种方式使用某些功能部件。例如，约束使您可以保护数据并定义数据值之间的关系。以下是一些关键的 DB2 功能部件:

- 约束
- 用户定义类型 (UDT) 和大对象 (LOB)
- 用户定义函数 (UDF)
- 触发器
- 存储过程

在决定是否使用 DB2 功能部件时，要考虑以下几点:

应用程序独立性

可以使应用程序独立于它所处理的数据。使用在数据库上运行的 DB2 功能部件使您能够维护和更改围绕数据的逻辑结构而不影响应用程序。如果需要对该逻辑结构做更改，则只需要在一个地方做更改；即在服务器中更改而不必在存取数据的每个应用程序中都做更改。

性能 通过在服务器上存储和运行应用程序的各部分，可以使应用程序执行得更

快。这样就将某些处理移交给功能更强大的服务器，从而可以减少客户机应用程序和服务器之间的网络通信量。

应用程序需求

您的应用程序可以有其他应用程序没有的独特逻辑结构。例如，如果您的应用程序以某一特定次序处理数据输入错误，而该次序不适合其他应用程序，则可能要编写自己的代码来处理这种情况。

在某些情况下，您可能决定使用在服务器上运行的 DB2 功能部件，因为这些功能部件可被几个应用程序使用。而在其他情况下，您又可能决定将逻辑结构保存在您的应用程序中，因为只有您的应用程序使用该逻辑结构。

约束

要保护数据并定义数据之间的关系，通常应定义商业规则。这些规则定义哪些数值对于表中的列有效，或者定义一个或多个表中列的相互关联方式。

DB2 提供约束作为使用数据库系统来实施这些规则的一种方式。如果您通过使用数据库系统来实施这些商业规则，就不必在应用程序中编写代码来实施这些规则。然而，如果某个商业规则只适用于某一个应用程序，则应该在这个应用程序中编写此规则，而不应使用全局数据库约束。

DB2 提供下列类型的约束：

1. NOT NULL 约束
2. UNIQUE 约束
3. PRIMARY KEY 约束
4. FOREIGN KEY 约束
5. CHECK 约束

可使用 SQL 语句 CREATE TABLE 和 ALTER TABLE 来定义约束。

用户定义类型 (UDT) 和大对象 (LOB)

数据库中的每个数据元素都存储在表的列中，并且每列都定义有一个数据类型。数据类型对可放入列中的值的类型进行限制，并限制对这些值可执行的操作。例如，一个整型列只能包含一个固定范围内的数。DB2 包括一组具有定义的特性和行为的内部数据类型：字符串、数字、日期时间值、大对象、空值、图形字符串、二进制字符串和数据链路。

然而，有时内部数据类型可能无法满足应用程序的需要。DB2 提供用户定义类型 (UDT)，使您能够定义应用程序需要的不同数据类型。

UDT 基于内部数据类型。定义 UDT 时，也应定义对该 UDT 有效的操作。例如，可以定义基于 DECIMAL 数据类型的 MONEY 数据类型。然而，对于 MONEY 数据类型，可能只允许加法和减法运算，而不允许乘法和除法运算。

大对象 (LOB) 允许您存储并处理数据库中大而复杂的数据对象：如音频、视频、图象和大文档。

UDT 和 LOB 相结合会使您的工作能力大大提高。您不再限于使用由 DB2 提供的内部数据类型来建立商业数据模型和捕捉该数据的语义。您可以使用 UDT 为高级应用程序定义大而复杂的数据结构。

除了扩充内部数据类型之外，UDT 还有以下几个好处：

支持应用程序中面向对象的程序设计

可以将类似对象分组为相关的数据类型。这些类型具有名称、内部表示和特定的行为。通过使用 UDT，可以让 DB2 知道新类型的名称以及该类型在内部如何表示。LOB 是新类型可能的内部表示法之一，并且最适合表示大而复杂的数据结构。

通过强类型转换和封装保证数据完整性

强类型转换保证：只有定义在特殊类型上的函数和运算才可应用于该类型。封装确保 UDT 的行为受可应用于这些行为的函数和运算符的限制。在 DB2 中，可以以用户定义函数 (UDF) 的形式提供 UDT 的行为，为满足广泛的用户需求，可编写 UDF。

集成到数据库管理程序中的性能

因为 UDT 是内部表示的，与内部数据类型的表示方式相同，所以这些 UDT 与内部数据类型共享相同的有效代码来实现内部函数、比较运算符、索引以及其他函数。利用 LOB 的 UDT 则例外，它不能和比较运算符及索引一起使用。

存储过程

应用程序一般通过网络存取数据库。但如果返回大量数据，性能可能较差。存储过程是在数据库服务器上运行的。客户机应用程序可以调用存储过程，然后由存储过程执行数据库存取，这样不必跨网络返回不需要的数据。存储过程只返回客户机应用程序需要的结果。

使用存储过程有以下几个好处：

减小网络通信量

将 SQL 语句分组可减小网络通信量。一般应用程序的每个 SQL 语句通过网络需要两个行程。将 SQL 语句分组使每组语句用两个行程就可穿过网络，从而使应用程序的性能更好。

存取只在服务器上存在的功能部件

存储过程可以存取只在服务器上运行的命令，如 LIST DATABASE DIRECTORY 和 LIST NODE DIRECTORY；可以利用服务器上增加的内存和磁盘空间；并且可以存取安装在服务器上的其他软件。

实施商业规则

可以使用存储过程定义几个应用程序公用的商业规则。这是除了使用约束和触发器之外，另外一种定义商业规则的方法。

当应用程序调用存储过程时，它将按照存储过程中定义的规则，以一致的方式处理数据。如果需要更改规则，则只需在存储过程中做一次更改，而不需要在调用存储过程的每个应用程序中都做更改。

用户定义函数 (UDF)

通过 SQL 提供的内部功能可能未满足应用程序的全部需要。为了允许您扩展这些功能，DB2 支持用户定义函数 (UDF)。可以用 Visual Basic、C/C++ 或 Java 编写自己的代码，来执行返回单个标量值或表的任何 SQL 语句中的操作。

UDF 给您提供了相当大的灵活性。它们可以从数据库中将单个标量值作为选择列表的一部分返回，或者可以从非数据库源（如电子表格）中返回整个表。

UDF 提供一种使应用程序标准化的方法。通过实现一个公共的用户定义函数集，很多应用程序就可以使用同样的方式处理数据，从而确保结果一致。

用户定义函数也支持在应用程序中采用面向对象的程序设计。UDF 提供了两个功能，一个是抽象，它允许您定义对数据对象执行操作的方法。另一个是封装，它允许您控制对对象的基础数据的存取，从而防止直接操作数据和可能的损坏。

OLE DB 表函数

Microsoft OLE DB 是一组 OLE/COM 接口，它们为应用程序提供统一的存取方式来存取各种不同信息源中存储的数据。“DB2 通用数据库”使您能够定义存取 OLE DB 数据源的表函数，从而简化 OLE DB 应用程序的创建。您可以通过 OLE DB 接口存取数据源，以便对它们执行 GROUP BY、JOIN 和 UNION 等操作。例如，可定义一个 OLE DB 表函数，来从 Microsoft Access 数据库或 Microsoft Exchange 通讯录返回一个表，然后创建一个报表，将此 OLE DB 表函数返回的数据与 DB2 数据库中的数据完美地组合在一起。

利用 OLE DB 表函数提供对任何 OLE DB 提供器的内部存取，可减少应用程序开发工作。对于 C、Java 和 OLE 自动化表函数，开发人员需要实现表函数；而对于 OLE DB 表函数，类属的内部 OLE DB 用户可通过任何 OLE DB 提供器来检索数据。您只需要注册语言类型为 OLEDB 的表函数，并将 OLE DB 提供器和相关行集用作数据源。您不必做任何 UDF 程序设计来利用 OLE DB 表函数。

OLE 自动化 UDF 和存储过程

OLE（对象链接和嵌入）自动化是 Microsoft 公司的 OLE 2.0 体系结构的一部分。利用 OLE 自动化，无论应用程序是用何种语言编写的，它们均可以在 OLE 自动化对象中揭示程序的特性和方法。这样，其他应用程序如 Lotus Notes 或 Microsoft Exchange 可通过 OLE 自动化利用这些特性和方法来集成这些对象。

DB2 Windows 32 位操作系统版允许使用 UDF 和存储过程存取 OLE 自动化对象。要存取 OLE 自动化对象并调用其方法，必须将这些对象的方法注册为 UDF 或存储过程。然后 DB2 应用程序可以通过调用 UDF 或存储过程来调用这些方法。UDF 可以是标量函数或表函数。

例如，您可以开发一个应用程序来查询用诸如 Microsoft Excel 的产品创建的电子表格中的数据。为此，应开发一个 OLE 自动化表函数，这个表函数从工作表中检索数据，并将数据返回至 DB2。然后 DB2 可以处理该数据，进行联机分析处理 (OLAP)，并将查询结果返回至应用程序。

触发器

触发器定义一组操作，由对指定的表进行的删除、插入或更新操作来执行。当执行这种 SQL 操作时，就称触发器被激活了。触发器可以在 SQL 操作之前或之后被激活。使用 SQL 语句 CREATE TRIGGER 定义触发器。

可以用几种方式来使用在更新或插入之前运行的触发器：

- 在数据库中更新或插入值之前，要检查或修改值。如果需要将数据从用户查看方式转换为某些内部数据库格式，这是很有用的。
- 运行以用户定义函数编写的其他非数据库操作。

类似地，也可以用几种方式来使用在更新或插入后运行的触发器：

- 更新其他表中的数据。这对维护数据间关系或保存审查记录信息很有用。
- 根据该表或其他表中的其他数据进行检查。当参考完整性约束不合适或对表的检查约束只限于检查当前表时，这一方法对确保数据完整性很有用。
- 运行以用户定义函数编写的非数据库操作。当发出警报或要在数据库外部更新信息时，这是很有用的。

使用触发器有以下几个好处：

更快地开发应用程序

触发器存储在数据库中，并可用于所有应用程序。这使您不必为每个应用程序编写等效的函数。

全局实施商业规则

触发器只需定义一次，然后就可由使用触发器管理的数据的所有应用程序使用。

更易于维护

任何更改只需在数据库中做一次，而不必在使用触发器的每个应用程序中都做更改。

“DB2 通用数据库”的工具

开发应用程序时可以使用各种不同的工具。“DB2 通用数据库”提供下列工具帮助您在应用程序中编写和测试 SQL 语句，并帮助您监控这些语句的性能：

注：并不是所有工具在每个平台上都可用。

控制中心

一种显示数据库对象（如数据库、表以及程序包）及其相互关系的图形界面。使用“控制中心”来执行诸如配置系统、管理目录、备份与恢复系统、调度作业以及管理媒体等管理任务。

“控制中心”包括下列设施：

命令中心

用于在交互式窗口中输入 DB2 命令和 SQL 语句，并在结果窗口中查看执行结果。可以浏览整个结果并将输出保存到文件中。

脚本中心

用于创建脚本，您可以存储它们并在以后调用。这些脚本可包含 DB2 命令、SQL 语句或操作系统命令。可以安排脚本以无人照管方式运行。可以运行这些作业一次，也可以将这些作业设置为按计划重复运行。重复的运行计划表对于备份之类的任务尤其有用。

日志 用于查看下列类型的信息：暂挂执行、正在执行或已完成执行的作业的所有可用信息；恢复历史记录；警报记录以及信息记录。也可以使用“日志”来查看作业以无人照管方式运行的结果。

警报中心

用于监控系统以获得对潜在问题的早期警告，或自动采取措施解决问题。

工具设置

用于更改“控制中心”、“警报中心”和“复本”的设置。

性能监控程序

“控制中心”的一个可安装选项，“性能监控程序”是一个图形界面，它为 DB2 系统提供性能数据收集、查看、报告、分析以及报警等各种功能。使用“性能监控程序”进行性能调整。

可以选择监控瞬象或事件。“瞬象监控程序”允许您按指定的时间间隔捕捉时间点信息。“事件监控程序”允许您记录整个事件（如连接）中的性能信息。

Visual Explain

“控制中心”的一个可安装选项，Visual Explain 是一个图形界面，它允许您分析并调整 SQL 语句，包括查看优化器为 SQL 语句选择的存取方案。

存储过程构建器 (SPB)

支持 DB2 存储过程快速开发的 GUI 工具。它为 DB2 系列（从工作站到 OS/390）提供单一的开发环境。在 Windows 32 位操作系统上，可从以下流行的应用程序开发工具启动它：Microsoft Visual Studio、Microsoft Visual Basic 和 IBM VisualAge for Java，或作为与“IBM DB2 通用数据库”程序组分开的一个独立的应用程序启动。也可通过执行以下文件启动它：

```
%DB2PATH%\bin\DB2SPB.exe
```

其中 %DB2PATH% 指向 DB2 的安装目录。

在 AIX 和 Solaris 上，可用 db2spb 命令启动“存储过程构建器”。

第1章 介绍

谁应使用本书	3	Windows 32 位操作系统	11
如何使用本书	3	样本程序	11
突出显示的约定	3	DB2 API 非嵌入式 SQL 样本	16
关于 DB2 应用程序开发客户机	4	DB2 API 嵌入式 SQL 样本	19
支持的服务器	6	不含 DB2 API 的嵌入式 SQL 样本	21
按平台分类的受支持软件	6	用户定义函数样本	22
AIX	7	DB2 调用层接口样本	22
HP-UX	8	Java 样本	24
Linux	9	SQL 过程样本	25
OS/2	9	ADO、RDO 和 MTS 样本	27
PTX	9	“对象链接和嵌入”样本	28
Silicon Graphics IRIX	10	命令行处理器样本	28
Solaris	10	记录管理用户出口样本	29

本书提供为开发 DB2 应用程序设置环境所需的信息，以及在此环境中编译、连接和运行这些应用程序的详细说明。它说明如何在下列平台上使用 DB2 通用数据库版本 7.1 的 DB2 应用程序 (DB2 SDK) 客户机来构建应用程序：

- AIX
- HP-UX
- Linux
- OS/2
- PTX
- Silicon Graphics IRIX
- Solaris 操作环境
- Windows 32 位操作系统

注：

1. DB2 NUMA-Q 版支持 PTX 操作系统。
2. Windows 32 位操作系统包括 Windows NT、Windows 95、Windows 98 和 Windows 2000。每当本书提到 Windows 32 位操作系统时，是指所有这些操作系统，但在有服务器网络体系结构 (SNA) 支持和 REXX 支持的情况下例外。仅在 Windows NT 和 Windows 2000 上提供这些支持。

要开发应用程序，可使用下列程序设计接口：

DB2 应用程序设计接口 (DB2 API)

提供管理 DB2 数据库的管理函数。

DB2 调用层接口 (DB2 CLI)

是一个基于 X/Open CLI 规范的可调用的 SQL 接口，它与 Microsoft 公司的“开放式数据库链接”(ODBC) 接口兼容。

嵌入式 SQL

将 SQL 语句直接编入程序中，这种程序必须预编译，以便转换为运行期函数调用。

Java 嵌入式 SQL (SQLJ)

在一个生成的简要表中使用 SQL 语句，这些语句要预编译并定制为运行期函数调用，以提供一个连接数据库管理程序的接口。

Java 数据库链接 (JDBC)

是 Java 的一个动态 SQL API。JDBC API 包括在为受支持的平台提供的“Java 开发工具箱”中。

有关各种程序设计接口的详情，参考：

- *Application Development Guide*，讨论如何编写和设计使用嵌入式 SQL、Java 嵌入式 SQL 和“Java 数据库链接”(JDBC) 来存取 DB2 系列服务器的应用程序。还讨论了用户定义函数 (UDF)。
- *CLI Guide and Reference*，说明如何编写和设计使用“DB2 调用层接口”和 ODBC 的应用程序。
- *Administrative API Reference*，讨论如何编写和设计使用“DB2 应用程序设计接口”的应用程序。

可查找下列书籍以找到详细的相关信息，如详细的产品安装和设置过程：

- *DB2 (OS/2 版) Quick Beginnings*，说明如何在 OS/2 服务器和客户机工作站上安装数据库管理程序和 DB2 应用程序开发客户机。
- *DB2 (UNIX 版) 快速入门*，说明如何在 UNIX 服务器和客户机工作站上安装数据库管理程序和 DB2 应用程序开发客户机。
- *DB2 Windows 版快速入门*，说明如何在 Windows 32 位操作系统服务器和客户机工作站上安装数据库管理程序和 DB2 应用程序开发客户机。
- *Command Reference*，说明如何使用“DB2 命令行处理器”(CLP) 和所有非 SQL DB2 命令。
- *Troubleshooting Guide*，帮助您解决与 DB2 客户机和服务器相关的应用程序开发问题，以及执行数据库管理和连接的相关任务时遇到的问题。

有关 DB2 文档库的完整列表，参阅第353页的『附录D. 使用 DB2 资料库』。

注：本书中的示例按原样提供，且不提供任何保证。用户，而不是 IBM，承担着质量、性能和修复任何缺陷的全部风险。

谁应使用本书

如果您要在 DB2 通用数据库版本 7.1 当前支持的一个平台上开发应用程序，应使用本书。本书描述您的程序如何用 DB2 API 管理 DB2 数据库，以及如何用 DB2 CLI、嵌入式 SQL、SQLJ 和 JDBC 存取 DB2 数据库。

为了使用本书，应了解要使用的平台所支持的一种或多种程序设计语言。在第6页的『按平台分类的受支持软件』中列出了这些语言。

如何使用本书

设计本书，是为了可以方便地找到开发应用程序所需的信息。本书包括如下几章：

- 第 1 章至第 3 章：每章包含所有平台的一般介绍性信息。
- 第 4 章和第 5 章：每章包含所有平台的特定程序设计信息。
- 第 6 章至第 13 章：每章包含针对一个平台的程序设计信息。

所有应用程序开发人员应阅读前三章，然后阅读包含他们需要的特定程序设计信息的“构建应用程序”一章，具体哪一章取决于他们将使用的操作系统以及程序设计语言。

附录提供了有关各种主题的重要附加信息。

突出显示的约定

本书使用下列约定：

斜体 指示下面一种情况：

- 新术语的引入
- 用户提供的名称或值
- 对另一个信息源的引用
- 一般性强调

大写字体

指示下面一种情况：

- 数据库管理程序数据类型
- 字段名
- 关键字
- SQL 语句

示例文本

指示下面一种情况:

- 编程示例和代码段
- 命令
- 输出的示例, 类似于系统显示的结果
- 特定数据值的示例
- 系统信息的示例
- 文件名和目录名
- 指示您要输入的信息

黑体 强调某一点。

关于 DB2 应用程序开发客户机

注: 应用程序开发客户机在 DB2 的先前版本中称为 DB2 软件开发工具箱 (DB2 SDK) 客户机。

DB2 应用程序开发 (DB2 AD) 客户机提供工具和环境, 以便您开发可存取 DB2 服务器和实现了“分布式关系数据库体系结构”(DRDA) 的应用程序服务器的应用程序。

可使用安装的 DB2 AD 客户机来构建和运行 DB2 应用程序。也可在以下 DB2 客户机上运行 DB2 应用程序:

- DB2 运行期客户机
- DB2 管理客户机

有关如何设置程序设计环境的信息, 参阅第31页的『第2章 设置』。

适用于本书所描述平台的 DB2 AD 客户机包括:

- **C/C++、Java、COBOL 和 Fortran 语言的预编译程序**, (前提是该平台支持这种语言; 有关详情, 请参阅第6页的『按平台分类的受支持软件』)。
- **嵌入式 SQL 应用程序支持**, 包括程序设计库、包含文件和代码样本。
- **DB2 调用层接口 (DB2 CLI) 应用程序支持**, 包括程序设计库、包含文件和代码样本, 它们用于开发易于移植到 ODBC 并可用 ODBC SDK 编译的应用程序。可从 Microsoft 公司获得用于 Windows 32 位操作系统的 ODBC SDK, 从其他各个供应商那里获得用在其他平台上的 ODBC SDK。对于 Windows 32 位操作系统, DB2 客户机包含一个 ODBC 驱动程序, 它支持用 Microsoft ODBC Software Developer's Kit 开发的应用程序。对于所有其他平台, DB2 客户机包

含一个可选择安装的 ODBC 驱动程序，它支持可用该平台的 ODBC SDK（如果有的话）开发的应用程序。只有用于 Windows 32 位操作系统的 DB2 客户机包含 ODBC 驱动程序管理器。

- **DB2 Java Enablement**，包括用于开发 Java 应用程序和 Applet 的“DB2 Java 数据库链接”（DB2 JDBC）支持，以及用于开发 Java 嵌入式 SQL 应用程序和 Applet 的“DB2 Java 嵌入式 SQL”（DB2 SQLJ）支持。
- **IBM 公司开发的 Java Development Kit (JDK) 1.1.8 和 Java Runtime Environment (JRE) 1.1.8**，与 DB2 AIX 版和 DB2 Windows 32 位操作系统版一起安装，与 DB2 OS/2 版一起交付。
- **REXX 语言支持**，可用在 AIX（仅适用于 32 位应用程序）、OS/2 和 Windows 32 操作系统上。在 DB2 版本 5.2 以上的版本中未更新此支持。
- **“ActiveX 数据对象” (ADO) 和“对象链接和嵌入” (OLE) 自动化 UDF 和存储过程**，用于 Windows 32 位操作系统，包括用 Microsoft Visual Basic 和 Microsoft Visual C++ 语言实现的代码样本。还有，使用“远程数据对象” (RDO) 的代码样本，RDO 是用 Microsoft Visual Basic 实现的。
- **“对象链接和嵌入数据库” (OLE DB) 表函数**，用于 Windows 32 位操作系统。
- **“DB2 存储过程构建器” (SPB)**，可用在 AIX、Solaris 和 Windows 32 位操作系统上。它是一个基于 GUI 的工具，支持 DB2 存储过程的快速开发。它为 DB2 系列（从工作站到 OS/390）提供单一的开发环境。可从以下流行的应用程序开发工具启动它：Microsoft Visual Studio、Microsoft Visual Basic 和 IBM Visual Age for Java，或作为与“IBM DB2 通用数据库”程序组分开的一个独立的应用程序启动。在 AIX 和 Solaris 上，可使用 db2spb 命令启动它。
- **交互式 SQL**，可通过“命令中心”或“命令行处理器” (CLP) 使用它创建 SQL 语句的原型或对数据库执行特定查询。
- **一组已归档的 API**，它允许其他应用程序开发工具直接在其产品内实现 DB2 的预编译程序支持。例如，在 AIX 和 OS/2 上，IBM COBOL 使用此接口。有关这组预编译程序服务 API 的信息，可以匿名方式从 FTP 站点获取：<ftp://ftp.software.ibm.com>。称为 prepapi.psbins 的 PostScript 文件位于目录 /ps/products/db2/info 中。此文件使用二进制格式。如果您不能访问此电子论坛，而又想获得一份此文档，可从“服务信息传单”描述的“IBM 服务”中订购。
- **SQL92 和 MVS 的一致性标志**，它标识应用程序中不符合“ISO/ANSI SQL92 初级”标准或不受 DB2 OS/390 版支持的嵌入式 SQL 语句。如果将在某台工作站上开发的应用程序迁移到另一个平台，则该标志会说明它们的语法是否兼容以节省您的时间。参考 *Command Reference*，以获得关于 PRECOMPILE PROGRAM 命令中的 SQLFLAG 选项的信息。

支持的服务器

可使用 DB2 AD 客户机来开发将在特定平台上运行的应用程序。然而，您的应用程序可存取下列平台服务器上的远程数据库：

- DB2 AIX 版
- DB2 HP-UX 版
- DB2 Linux 版
- DB2 OS/2 版
- DB2 NUMA-Q 版
- DB2 SCO UnixWare 7 版
- DB2 Solaris 版
- DB2 Windows NT 版
- 与“分布式关系数据库体系结构” (DRDA) 兼容的应用程序服务器，如：
 - DB2 OS/390 版
 - DB2 AS/400 版
 - DB2 VSE 版和 VM 版（以前是 SQL/DS VM 版和 VSE 版）
 - 由 IBM 之外的数据库开发商提供的、与 DRDA 兼容的应用程序服务器。

注：

1. DB2 NUMA-Q 版支持 PTX 操作系统。
2. DB2 SCO UnixWare 7 版仅支持 DB2 版本 5.2。

按平台分类的受支持软件

本节列出本书中所描述的各平台上 DB2 支持的编译程序和相关软件。该编译程序信息假定：您使用的是适合该平台的 DB2 预编译程序，而不是可构建到所列出的某个编译程序中的预编译程序支持。参考适合您操作系统的快速入门一书，以获取它所支持的通信产品的信息。

有关最新的 DB2 编译程序信息和相关软件的更新信息，请访问以下 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

注：

1. **DB2 发行版注意事项**可能包含受支持平台的编译程序和操作系统的更新信息。“发行版注意事项”有纯文本格式和 HTML 格式，可从您产品 CD-ROM 上的下列路径找到，其中 `<language_directory>` 是所用语言的目录，`index.htm` 是主要的 HTML 文件：

纯文本文件：

`doc/<language_directory>/release.txt (UNIX)`

`Doc\<language_directory>\release.txt (OS/2 和 Windows)`

HTML 文件:

doc/<language_directory>/db2ir/index.htm (UNIX)

Doc\<language_directory>\db2ir\index.htm (OS/2 和 Windows)

2. **Fortran 和 REXX**。在“DB2 通用数据库版本 5.2”中，DB2 对 Fortran 和 REXX 的功能部件的增强将不超过这两种语言的支持级别。
3. **Fortran**。在 DB2 版本 7.1 中不提供 Fortran 样本程序。有关获取 DB2 版本 6.1 的 Fortran 样本的信息，访问上面给出的 DB2 应用程序开发 Web 页面。
4. **HP-UX**。如果您要将 DB2 从 HP-UX 版本 10 或更早版本迁移至 HP-UX 版本 11，则您的 DB2 程序必须用 HP-UX 版本 11 上的 DB2 重新预编译（如果这些程序包括嵌入式 SQL），且必须重新编译。这包括所有 DB2 应用程序、存储过程、用户定义函数和用户出口程序。另外，在 HP-UX 版本 11 上编译的 DB2 程序可能无法在 HP-UX 版本 10 或更早版本上运行。在 HP-UX 版本 10 上编译和运行的 DB2 程序可以与 HP-UX 版本 11 服务器远程连接。
5. **Micro Focus COBOL**。用 DB2 版本 2.1.1 或更早版本预编译和用 Micro Focus COBOL 编译的任何现存应用程序，都应使用 DB2 的当前版本重新预编译，再用 Micro Focus COBOL 重新编译。如果用先前版本的 IBM 预编译程序构建的这些应用程序未重新预编译，则当发生异常终止时数据库可能被破坏。
6. **Perl**。“Perl 数据库接口” (Perl DBI) 的 DB2 UDB 驱动程序 (DBD::DB2) 发行版 0.71 可用在 AIX、HP-UX、Linux、Solaris 和 Windows NT 上。此驱动程序可从以下站点下载:

<http://www.ibm.com/software/data/db2/perl>

7. **REXX**。IBM Object REXX Windows NT/95 版不再随 DB2 一起交付。有关获取 Object REXX 的信息，请访问以下站点:

<http://www.ibm.com/software/ad/obj-rexx/>

AIX

DB2 AIX 版支持下列操作系统:

AIX/6000

版本 4.2.1 和更高版本

(64 位的版本 4.3.3 和更高版本)

DB2 AIX 版支持下列程序设计语言和编译程序:

C IBM C AIX 版的版本 3.6.6 (64 位的版本 3.6.6.3)

C++ IBM C Set++ AIX 版的版本 3.6.6 (64 位的版本 3.6.6.3)

IBM VisualAge C++ 版本 4.0

COBOL

IBM COBOL Set AIX 版的版本 1.1

Micro Focus COBOL 版本 4.0.20 (PRN 12.03 或更高版本)

Micro Focus COBOL 版本 4.1.10 (PRN 13.04 或更高版本)

Fortran

IBM XL Fortran AIX 版的版本 4.1 (32 位) 和版本 5.1.0 (32 位和 64 位)

Java IBM 公司开发的 Java Development Kit (JDK) 版本 1.1.8 AIX 版和 Java Runtime Environment (JRE) 版本 1.1.8 AIX 版 (与 DB2 一起安装)

IBM 公司开发的 Java Development Kit (JDK) 版本 1.2.2 AIX 版和 Java Runtime Environment (JRE) 版本 1.2.2 AIX 版

Perl “Perl 数据库接口” (Perl DBI) 的 DB2 UDB 驱动程序 (DBD::DB2) 发行版 0.71 (参阅以上注意事项)

REXX IBM AIX REXX/6000 AISPO 产品号: 5764-057

IBM Object REXX AIX 版的版本 1.1

REXXSAA 4.00

注: REXX 支持仅适用于 32 位

HP-UX

DB2 HP-UX 版支持下列操作系统:

HP-UX

版本 11.0, 带有 HP-UX 核心操作系统 2000 年补丁包版本 B.11.00.A1214 (Y2K-1100) 或更高版本的补丁包。

DB2 HP-UX 版支持下列程序设计语言和编译程序:

C HP C 编译程序版本 A.11.00.03

C++ HP-UX C++ 版本 A.12.00

COBOL

Micro Focus COBOL 版本 4.1

Fortran

HP Fortran/9000 版本 10.0

HP-UX F77 B.11.00.01

Java Hewlett-Packard 公司开发的 HP-UX Developer's Kit for Java 发行版 1.1.8

Perl “Perl 数据库接口” (Perl DBI) 的 DB2 UDB 驱动程序 (DBD::DB2) 发行版 0.71 (参阅以上注意事项)

Linux

DB2 Linux 版支持下列操作系统:

Linux 内核版本 2.2.12 或更高版本、glibc 版本 2.1.2 或更高版本、libstdc++ 版本 2.9.0、rpm (必须安装) 以及 pdksh 程序包 (运行 “DB2 命令行处理器” 所必需的)

DB2 Linux 版支持下列程序设计语言和编译程序:

C GNU/Linux gcc 版本 egcs-2.91.66 (egcs-1.1.2 发行版)

C++ GNU/Linux g++ 版本 egcs-2.91.66 (egcs-1.1.2 发行版)

Java IBM Developer Kit Linux 版和 Runtime Environment Linux 版的版本 1.1.8

Perl “Perl 数据库接口” (Perl DBI) 的 DB2 UDB 驱动程序 (DBD::DB2) 发行版 0.71 (参阅以上注意事项)

OS/2

DB2 OS/2 版支持下列操作系统:

OS/2 WARP 3.0、WARP 4.0 和 WARP 4.5

DB2 OS/2 版支持下列程序设计语言:

C/C++ IBM VisualAge C++ OS/2 版的版本 3 和 4.0

COBOL

IBM VisualAge COBOL OS/2 版的版本 2.0

Micro Focus COBOL 版本 4.0.20

FORTRAN

WATCOM FORTRAN 77 32 版本 10.5

Java IBM 公司开发的 Java Development Kit (JDK) 版本 1.1.8 OS/2 版和 Java Runtime Environment (JRE) 版本 1.1.8 OS/2 版 (与 DB2 一起安装)

REXX IBM 过程语言 2/REXX (作为 OS/2 的一部分提供)

PTX

DB2 NUMA-Q 版支持下列操作系统:

PTX 版本 4.5

DB2 NUMA-Q 版支持下列程序设计语言和编译程序:

C ptx/C 版本 4.5

C++ ptx/C++ 版本 5.2

Java ptx/JSE 版本 3.0

Silicon Graphics IRIX

DB2 Silicon Graphics IRIX 版支持下列操作系统:

Silicon Graphics IRIX

版本 6.2 和更高版本

DB2 Silicon Graphics IRIX 版支持下列程序设计语言和编译程序:

C MIPSpro C 编译程序 7.2

C++ MIPSpro C++ 7.2

Fortran

MIPSpro Fortran-77 7.2

Java Silicon Graphics 公司开发的 Java2 Software Development Kit 版本 1.2.1 (JDK 1.2.1)

Solaris

DB2 Solaris 版支持下列操作系统:

Solaris

版本 2.6、Solaris 7 和 Solaris 8

DB2 Solaris 版支持下列程序设计语言和编译程序:

C SPARCompiler C 版本 4.2 (32 位) 和版本 5.0 (32 位和 64 位)

C++ SPARCompiler C++ 版本 4.2 (32 位) 和版本 5.0 (32 位和 64 位)

COBOL

Micro Focus COBOL 版本 4.0

Fortran

SPARCompiler Fortran 版本 4.2 和版本 5.0

Java Sun Microsystems 公司开发的 Java Development Kit (JDK) 版本 1.1.8 和版本 1.2 Solaris 版

Perl “Perl 数据库接口” (Perl DBI) 的 DB2 UDB 驱动程序 (DBD::DB2) 发行版 0.71 (参阅以上注意事项)

Windows 32 位操作系统

DB2 Windows 32 位操作系统版支持下列操作系统:

Microsoft Windows NT

版本 4.0, 带服务程序包版本 4 或更高版本。

Microsoft Windows 2000

Microsoft Windows 98

Microsoft Windows 95

版本 4.00.950 或更高版本

DB2 Windows 32 位操作系统版支持下列程序设计语言:

Basic Microsoft Visual Basic 版本 4.2 和版本 5.0 (没有为此语言提供任何 DB2 预编译程序)

C/C++ Microsoft Visual C++ 版本 5.0 和 6.0

IBM VisualAge C++ Windows 版的版本 4.2 和版本 5.0

COBOL

Micro Focus COBOL 版本 4.0.20

IBM VisualAge COBOL 版本 2.0

REXX IBM Object REXX Windows NT/95 版的版本 1.1 (参阅以上注意事项)

Java IBM 公司开发的 Java Development Kit (JDK) 1.1.8 Win32 版和 Java Runtime Environment (JRE) 1.1.8 Win32 版 (与 DB2 一起安装)

Sun Microsystems 公司开发的 Java Development Kit (JDK) 1.2 Win32 版

Microsoft Software Developer's Kit for Java 版本 3.1

Perl “Perl 数据库接口” (Perl DBI) 的 DB2 UDB 驱动程序 (DBD::DB2) 发行版 0.71 (仅适用于 Windows NT。参阅以上注意事项。)

样本程序

注:

1. 本节描述用 DB2 支持的所有平台上的程序设计语言编写的样本程序。并没有将所有的样本程序都移植到所有平台或移植成受支持的程序设计语言。

2. DB2 样本程序按原样提供，且不提供任何保证。用户，而不是 IBM，承担着质量、性能和修复任何缺陷的全部风险。

样本程序是与 DB2 应用程序开发 (DB2 AD) 客户机一起提供的。可将样本程序用作模板来创建自己的应用程序。

对于每种受支持的语言以及对于每种语言中嵌入式 SQL 和非嵌入式 SQL 程序，样本程序文件扩展名都不相同。对于一种语言中的程序组，文件扩展名也可能不相同。在下表中对这些不同的样本文件扩展名进行分类：

按语言分类的样本文件扩展名

第13页的表1.

按程序组分类的样本文件扩展名

第14页的表2.

下面的表按类型来描述样本程序：

不含嵌入式 SQL 的 DB2 API 样本程序

第16页的表3.

DB2 API 嵌入式 SQL 样本程序

第19页的表4.

不含 DB2 API 的嵌入式 SQL 样本程序

第21页的表5.

用户定义函数样本程序

第22页的表6

DB2 CLI 样本程序

第22页的表7.

Java JDBC 样本程序

第24页的表8.

Java SQLJ 样本程序

第24页的表9.

SQL 过程样本程序

第25页的表10.

“ActiveX 数据对象”、“远程数据对象”和“Microsoft 事务服务器”样本程序

第27页的表11.

对象链接和嵌入 (OLE) 自动化样本程序

第28页的表12.

对象链接和嵌入数据库 (OLE DB) 表函数

第28页的表13.

命令行处理器 (CLP) 样本程序

第28页的表14.

记录管理用户出口程序

第29页的表15.

注:

1. 第19页的表4包含使用 DB2 API 和嵌入式 SQL 语句的程序。有关所有 DB2 API 样本程序, 请参阅第16页的表3和第19页的表4。有关所有嵌入式 SQL 样本程序 (除 Java SQLJ 外), 请参阅第19页的表4和第21页的表5。
2. 第22页的表6的 UDF 样本程序不包含 DB2 CLI UDF 程序。有关这些样本程序, 请参阅第22页的表7。

表 1. 按语言分类的样本文件扩展名

语言	目录	嵌入式 SQL 程序	非嵌入式 SQL 程序
C	samples/c samples/cli (CLI 程序)	.sqc	.c
C++	samples/cpp	.sqc (UNIX) .sqx (Windows & OS/2)	.C (UNIX) .cxx (Windows & OS/2)
COBOL	samples/cobol samples/cobol_mf	.sqb	.cbl
JAVA	samples/java	.sqlj	.java
REXX	samples/rexx	.cmd	.cmd

表 2. 按程序组分类的样本文件扩展名

样本组	目录	文件扩展名
ADO, RDO, MTS	samples\ADO\VB (Visual Basic) samples\ADO\VC (Visual C++) samples\RDO samples\MTS	.bas .frm .vbp (Visual Basic) .cpp .dsp .dsw (Visual C++)
CLP	samples/clp	.db2
OLE	samples\ole\msvb (Visual Basic) samples\ole\msvc (Visual C++)	.bas .vbp (Visual Basic) .cpp (Visual C++)
OLE DB	samples\oledb	.db2
SQL 过程	samples/sqlproc	.db2 .c .sqc (客户机应用程序)
用户出口	samples/c	.cad (OS/2) .cadsm (UNIX & Windows) .cdisk (UNIX & Windows) .ctape (UNIX)

注:

目录定界符

在 UNIX 上是 /。在 OS/2 和 Windows 平台，是 \。在这些表中，除非该目录仅在 Windows 和 / 或 OS/2 上可用，否则，使用 UNIX 定界符。

文件扩展名

为表中只具有一种扩展名的样本提供的。

嵌入式 SQL 程序

需要预编译，但 REXX 嵌入式 SQL 程序除外，因为这种程序的嵌入式 SQL 语句是在程序运行时解释的。

IBM COBOL 样本

仅在 AIX、OS/2 和 Windows 32 位操作系统的 cobol 子目录中提供。

Micro Focus Cobol 样本

仅在 AIX、HP-UX、OS/2、Solaris 操作环境和 Windows 32 位操作系统的 cobol_mf 子目录中提供。

Java 样本

包含“Java 数据库链接”(JDBC) Applet、应用程序和存储过程，Java 嵌入式 SQL (SQLJ) Applet、应用程序和存储过程，以及 Java UDF。提供了所有受支持的 DB2 平台的 Java 样本。

REXX 样本

仅对 AIX、OS/2 和 Windows NT 操作系统提供。

CLP 样本

是执行 SQL 语句的“命令行处理器”脚本。

OLE 样本

表示用 Microsoft Visual Basic 和 Microsoft Visual C++ 编写的“对象链接和嵌入”(OLE)，仅在 Windows 32 位操作系统上提供。

ADO、RDO 和 MTS 样本

包含用 Microsoft Visual Basic 和 Microsoft Visual C++ 编写的“ActiveX 数据对象”样本，用 Microsoft Visual Basic 编写的“远程数据对象”和“Microsoft 事务服务器”样本，仅在 Windows 32 位操作系统上提供。

用户出口样本

是用于归档和检索数据库记录文件的“记录管理用户出口”程序。这类文件必须用 .c 扩展名重新命名，并作为 C 语言程序编译。

可在安装了 DB2 的目录的 samples 子目录中找到这些样本程序。每种受支持的语言都有一个子目录。以下示例显示如何在每个受支持的平台上找到用 C 或 C++ 编写的样本程序。

- 在 UNIX 平台上。

可在数据库实例目录下的 sqllib/samples/c 中，找到嵌入式 SQL 和 DB2 API 程序的 C 源代码；DB2 CLI 程序的 C 源代码在 sqllib/samples/cli 中。有关样本表中那些程序的其他信息，参考 DB2 实例下适当的 samples 子目录中的 README 文件。README 文件将包含本书中未列出的任何附加样本。

- 在 OS/2 和 Windows 32 位操作系统上。

可在 DB2 安装目录下的 %DB2PATH%\samples\c 中找到嵌入式 SQL 和 DB2 API 程序的 C 源代码；DB2 CLI 程序的 C 源代码在 %DB2PATH%\samples\cli 中。变量 %DB2PATH% 确定将 DB2 安装在何处。取决于装有 DB2 的驱动器，%DB2PATH% 将指向 drive:\sqllib。有关样本表中那些样本程序的其他信息，参考适当的 %DB2PATH%\samples 子目录中的 README 文件。README 文件将包含本书中未列出的任何附加样本。

样本程序目录在大多数平台上一一般是只读的。在改变或构建样本程序之前，将这些样本程序复制到工作目录中。

DB2 API 非嵌入式 SQL 样本

表 3. 不含嵌入式 SQL 的 DB2 API 样本程序

样本程序	包括的 API
backrest	<ul style="list-style-type: none">• sqlbftcq - 取装表空间容器查询• sqlbstsc - 设置表空间容器• sqlfudb - 更新数据库配置• sqlubkp - 备份数据库• sqluroll - 前滚数据库• sqlurst - 复原数据库
checkerr	<ul style="list-style-type: none">• sqlaintp - 获取出错信息• sqlogstt - 获取 SQLSTATE 信息
cli_info	<ul style="list-style-type: none">• sqlqryi - 查询客户机信息• sqleseti - 设置客户机信息
client	<ul style="list-style-type: none">• sqlqryc - 查询客户机• sqlesetc - 设置客户机
d_dbconf	<ul style="list-style-type: none">• sqleatin - 连接• sqledtin - 断开• sqlfddb - 获取数据库配置缺省值
d_dbmcon	<ul style="list-style-type: none">• sqleatin - 连接• sqledtin - 断开• sqlfdsys - 获取数据库管理程序配置缺省值
db_udcs	<ul style="list-style-type: none">• sqleatin - 连接• sqlecrea - 创建数据库• sqledrpd - 卸下数据库
db2mon	<ul style="list-style-type: none">• sqleatin - 连接• sqlmon - 获取 / 更新监控开关• sqlmonss - 获取瞬象• sqlmonsz - 估计 sqlmonss() 输出缓冲区必需的大小• sqlmrset - 复位监控程序

表 3. 不含嵌入式 SQL 的 DB2 API 样本程序 (续)

样本程序	包括的 API
dbcat	<ul style="list-style-type: none"> • sqlcadb - 编目数据库 • sqledcls - 关闭数据库目录扫描 • sqledgne - 获取下一个数据库目录项 • sqledosd - 打开数据库目录扫描 • sqleuncd - 取消编目数据库
dbcmnt	<ul style="list-style-type: none"> • sqledcgd - 更改数据库注解 • sqledcls - 关闭数据库目录扫描 • sqledgne - 获取下一个数据库目录项 • sqledosd - 打开数据库目录扫描 • sqleisig - 安装信号处理程序
dbconf	<ul style="list-style-type: none"> • sqleatin - 连接 • sqlcrea - 创建数据库 • sqledrpd - 卸下数据库 • sqlfrdb - 复位数据库配置 • sqlfudb - 更新数据库配置 • sqlfxdb - 获取数据库配置
dbinst	<ul style="list-style-type: none"> • sqleatcp - 连接和更改口令 • sqleatin - 连接 • sqledtin - 断开 • sqlegins - 获取实例
dbmconf	<ul style="list-style-type: none"> • sqleatin - 连接 • sqledtin - 断开 • sqlfrsys - 复位数据库管理程序配置 • sqlfusys - 更新数据库管理程序配置 • sqlfxsys - 获取数据库管理程序配置
dbsnap	<ul style="list-style-type: none"> • sqleatin - 连接 • sqlmonss - 获取瞬象
dbstart	<ul style="list-style-type: none"> • sqlepstart - 启动数据库管理程序
dbstop	<ul style="list-style-type: none"> • sqlefrce - 强制应用程序 • sqlepstp - 停止数据库管理程序

表 3. 不含嵌入式 SQL 的 DB2 API 样本程序 (续)

样本程序	包括的 API
dscscat	<ul style="list-style-type: none"> • sqlgdad - 编目 DCS 数据库 • sqlgdcl - 关闭 DCS 目录扫描 • sqlgdel - 取消编目 DCS 数据库 • sqlgdge - 获取数据库的 DCS 目录项 • sqlgdgt - 获取 DCS 目录项 • sqlgdsc - 打开 DCS 目录扫描
dmscont	<ul style="list-style-type: none"> • sqleatin - 连接 • sqlecrea - 创建数据库 • sqledrpd - 卸下数据库
ebcdicdb	<ul style="list-style-type: none"> • sqleatin - 连接 • sqlecrea - 创建数据库 • sqledrpd - 卸下数据库
migrate	<ul style="list-style-type: none"> • sqlmgdb - 迁移数据库
monreset	<ul style="list-style-type: none"> • sqleatin - 连接 • sqlmrset - 复位监控程序
monsz	<ul style="list-style-type: none"> • sqleatin - 连接 • sqlmonss - 获取瞬象 • sqlmonsz - 估计 sqlmonss() 输出缓冲区必需的大小
nodecat	<ul style="list-style-type: none"> • sqlctnd - 编目节点 • sqlencls - 关闭节点目录扫描 • sqlengne - 获取下一个节点目录项 • sqlenops - 打开节点目录扫描 • sqluncn - 取消编目节点
restart	<ul style="list-style-type: none"> • sqlerstd - 重新启动数据库
setact	<ul style="list-style-type: none"> • sqlesact - 设置记帐字符串
setrundg	<ul style="list-style-type: none"> • sqlesdeg - 设置运行期等级
sws	<ul style="list-style-type: none"> • sqleatin - 连接 • sqlmon - 获取 / 更新监控开关
utilapi	<ul style="list-style-type: none"> • sqlaintp - 获取出错信息 • sqlqstt - 获取 SQLSTATE 信息

DB2 API 嵌入式 SQL 样本

表 4. DB2 API 嵌入式 SQL 样本程序

样本程序	包括的 API
asynrlog	<ul style="list-style-type: none">• sqlurlog - 异步读记录
autocfg	<ul style="list-style-type: none">• db2AutoConfig -- 自动配置• db2AutoConfigMemory -- 自动配置空闲内存• sqlfudb -- 更新数据库配置• sqlfusys -- 更新数据库管理程序配置• sqlesetc -- 设置客户机• sqlaintp -- SQLCA 信息
dbauth	<ul style="list-style-type: none">• sqluadau - 获得权限
dbstat	<ul style="list-style-type: none">• sqlureot - 重组表• sqlustat - 运行统计
expsamp	<ul style="list-style-type: none">• sqluexpr - 调出• sqluimpr - 调入
impexp	<ul style="list-style-type: none">• sqluexpr - 调出• sqluimpr - 调入
loadqry	<ul style="list-style-type: none">• db2LoadQuery - 装入查询
makeapi	<ul style="list-style-type: none">• sqlabndx - 联编• sqlaprep - 预编译程序• sqlpstp - 停止数据库管理程序• sqlpstr - 启动数据库管理程序
rebind	<ul style="list-style-type: none">• sqlarbnd - 重新联编
rechist	<ul style="list-style-type: none">• sqlubkp - 备份数据库• sqluhcls - 关闭恢复历史文件扫描• sqluhgne - 获取下一个恢复历史文件项• sqluhops - 打开恢复历史文件扫描• sqluhprn - 修剪恢复历史文件• sqluhupd - 更新恢复历史文件

表 4. DB2 API 嵌入式 SQL 样本程序 (续)

样本程序	包括的 API
tabscont	<ul style="list-style-type: none"> • sqlbctcq - 关闭表空间容器查询 • sqlbftcq - 取装表空间容器查询 • sqlbotcq - 打开表空间容器查询 • sqlbtcq - 表空间容器查询 • sqlfmem - 空闲内存
tabspace	<ul style="list-style-type: none"> • sqlbctsq - 关闭表空间查询 • sqlbftpq - 取装表空间查询 • sqlbgtss - 获取表空间统计信息 • sqlbmtsq - 表空间查询 • sqlbotsq - 打开表空间查询 • sqlbstpq - 单个表空间查询 • sqlfmem - 空闲内存
tload	<ul style="list-style-type: none"> • sqluexpr - 调出 • sqluload - 装入 • sqluvqdp - 停顿表的表空间
tspace	<ul style="list-style-type: none"> • sqlbctcq - 关闭表空间容器查询 • sqlbctsq - 关闭表空间查询 • sqlbftcq - 取装表空间容器查询 • sqlbftpq - 取装表空间查询 • sqlbgtss - 获取表空间统计信息 • sqlbmtsq - 表空间查询 • sqlbotcq - 打开表空间容器查询 • sqlbotsq - 打开表空间查询 • sqlbstpq - 单个表空间查询 • sqlbstsc - 设置表空间容器 • sqlbtcq - 表空间容器查询 • sqlfmem - 空闲内存
utilemb	<ul style="list-style-type: none"> • sqlaintp - 获取出错信息 • sqllogstt - 获取 SQLSTATE 信息

不含 DB2 API 的嵌入式 SQL 样本

表 5. 不含 DB2 API 的嵌入式 SQL 样本程序

样本程序名	程序说明
adhoc	演示动态 SQL 和 SQLDA 结构如何交互式处理 SQL 命令。用户输入 SQL 命令，然后返回与该 SQL 命令对应的输出。
advsql	演示高级 SQL 表达式如 CASE、CAST 和标量全查询的用法。
blobfile	通过从样本数据库中读取 BLOB 值并将其放入文件中，演示如何处理“二进制大对象”(BLOB)。此文件的内容可以使用外部浏览器来显示。
columns	演示用动态 SQL 处理的游标的用法。本程序列示从在希望的模式名下的 SYSCAT.COLUMNS 得到的结果集。
cursor	演示使用静态 SQL 的游标的用法。
delete	演示从数据库删除项目的静态 SQL。
dynamic	演示使用动态 SQL 的游标的用法。
joinsql	演示如何使用高级 SQL 连接表达式。
largevol	演示在分区环境中的并行查询处理，以及如何使用 NFS 文件系统自动合并结果集。仅在 AIX 上可用。
lobeval	演示 LOB 定位器的用法并延迟对实际 LOB 数据的估计。
lobfile	演示 LOB 文件句柄的用法。
lobloc	演示 LOB 定位器的用法。
lobval	演示 LOB 的用法。
openftch	演示如何使用静态 SQL 取装、更新和删除行。
recursql	演示高级 SQL 递归查询的用法。
sampudf	演示如何用“用户定义类型”(UDT)和“用户定义函数”(UDF)来修改表项。此程序中说明的所有 UDF 都是源 UDF。
spclient	调用 spserver 共享库中存储过程的客户机应用程序。
spcreate.db2	一个 CLP 脚本，包含 CREATE PROCEDURE 语句，以注册由 spserver 程序创建的存储过程。
spdrop.db2	一个 CLP 脚本，包含 DROP PROCEDURE 语句，以取消注册 spserver 程序生成的存储过程。
spserver	演示存储过程的服务器程序。客户机程序是 spclient。
static	演示如何使用静态 SQL 检索信息。
tabsql	演示高级 SQL 表表达式的用法。
tbdefine	演示创建和卸下表。

表 5. 不含 DB2 API 的嵌入式 SQL 样本程序 (续)

样本程序名	程序说明
thdsrver	演示如何使用 POSIX 线程 API 来创建和管理线程。该程序维护着一个场境存储池。一个 generate_work 函数从主程序执行，并创建由工作者线程执行的动态 SQL 语句。当一个场境变得可用时，就会创建一个线程，并指派它执行指定的工作。生成的工作由若干语句组成，用于从 sample 数据库的 STAFF 或 EMPLOYEE 表中删除项目。此程序仅在 UNIX 平台上可用。
trigsq1	演示如何使用高级 SQL 触发器和约束。
udfcli	演示如何调用由 udfsrv 程序创建并存储在服务器上的用户定义函数 (UDF) 来存取 sample 数据库中的表。
updat	演示如何使用静态 SQL 更新数据库。
varinp	演示如何使用参数标记作为嵌入式动态 SQL 语句调用的输入变量。

用户定义函数样本

表 6. 用户定义函数样本程序

样本程序名	程序说明
DB2Udf.java	一个 Java UDF，它演示几种任务，包括整数除法、“字符大对象” (CLOB) 的处理和 Java 实例变量的使用。
udfsrv.c	使用用户定义函数 ScalarUDF 创建库，以存取样本数据库表。
UDFsrv.java	演示 Java “用户定义函数” (UDF) 的用法。

DB2 调用层接口样本

表 7. DB2 通用数据库中的样本 CLI 程序

样本程序名	程序说明
公共实用程序文件	
utilcli.c	用在 CLI 样本中的实用程序函数
utilapi.c	调用 DB2 API 的实用程序函数。
应用程序级别 - 涉及 DB2 和 CLI 应用程序级别的样本。	
apinfo.c	如何获取和设置应用程序级别信息。
aphndls.c	如何分配和释放句柄。
apsqlca.c	如何使用 SQLCA 数据。
安装图象级别 - 涉及 DB2 和 CLI 安装图象级别的样本。	
ilinfo.c	如何获取和设置安装级别信息 (如 CLI 驱动程序的版本)。
实例级别 - 涉及 DB2 和 CLI 实例级别的样本。	
ininfo.c	如何获取和设置实例级别信息。
数据库级别 - 涉及 DB2 数据库对象的样本。	

表 7. DB2 通用数据库中的样本 CLI 程序 (续)

样本程序名	程序说明
dbconn.c	如何连接数据库和断开连接。
dbinfo.c	如何获取和设置数据库级别的信息。
dbmconn.c	如何与多个数据库连接和断开连接（使用 DB2 API 创建和卸下第二个数据库）。
dbmuse.c	如何用多个数据库执行事务（使用 DB2 API 创建和卸下第二个数据库）。
dbnative.c	如何将包含 ODBC 转义子句的语句转换为数据源特定格式。
dbuse.c	如何使用数据库对象。
dbusemx.sqc	如何同时使用单个数据库和嵌入式 SQL。
表级别 - 涉及 DB2 表对象的样本。	
tbconstr.c	如何使用表约束。
tbconstr.c	如何创建、改变和卸下表。
tbinfo.c	如何获取和设置表级别的信息。
tbmod.c	如何修改表中的信息。
tbread.c	如何读取表中的信息。
数据类型级别 - 涉及数据类型的样本。	
dtinfo.c	如何获取数据类型的信息。
dtlob.c	如何读写 LOB 数据。
dtudt.c	如何创建、使用和卸下用户定义的单值类型。
UDF 级别 - 演示用户定义函数的样本。	
udfcli.c	调用 udfsrv.c 中用户定义函数的客户机应用程序。
udfsrv.c	被 udfcli.c 样本调用的用户定义函数 ScalarUDF。
存储过程级别 - 演示调用 CLI 中存储过程的样本。	
spcreate.db2	发出 CREATE PROCEDURE 语句的 CLP 脚本。
spdrop.db2	用于从目录中卸下存储过程的 CLP 脚本。
spclient.c	用于调用在 spserver.c 中声明的服务器函数的客户机程序。
spserver.c	在服务器上构建和运行的存储过程函数。
spcall.c	调用任何存储过程的程序。

注: samples/cli 目录中的其他文件包括:

- README - 列出所有示例文件。
- makefile - 所有文件的 Makefile
- 应用程序和存储过程的构建文件

Java 样本

表 8. “Java 数据库链接” (JDBC) 样本程序

样本程序名	程序说明
DB2App1.java	一个 JDBC 应用程序，它使用调用用户的特权来查询样本数据库。
DB2App1t.java	一个 JDBC Applet，它使用 JDBC Applet 的驱动程序来查询数据库。它使用在 DB2App1t.html 中指定的用户名、口令、服务器和端口号参数。
DB2App1t.html	嵌入 Applet 的样本程序 DB2App1t 的 HTML 文件。它需要用服务器和用户信息进行定制。
DB2UdCli.java	一个 Java 客户机应用程序，它调用 Java 用户定义函数 DB2Udf。
Dynamic.java	演示使用动态 SQL 的游标。
MRSPcli.java	这是调用服务器程序 MRSPsrv 的客户机程序。该程序演示从 Java 存储过程返回多个结果集。
MRSPsrv.java	这是客户机程序 MRSPcli 调用的服务器程序。该程序演示从 Java 存储过程返回多个结果集。
Outcli.java	一个 Java 客户机应用程序，它调用 SQLJ 存储过程 Outsrv。
PluginEx.java	一个 Java 程序，它向“DB2 Web 控制中心”添加新的菜单项和工具栏按钮。
Spclient.java	一个 JDBC 客户机应用程序，它调用 Spserver 存储过程类中的 PARAMETER STYLE JAVA 存储过程。
Spcreate.db2	一个 CLP 脚本，它包含 CREATE PROCEDURE 语句，以将 Spserver 类中包含的方法注册为存储过程。
Spdrop.db2	一个 CLP 脚本，它包含取消注册 Spserver 类中包含的存储过程所需的 DROP PROCEDURE 语句。
Spserver.java	一个演示 PARAMETER STYLE JAVA 存储过程的 JDBC 程序。该客户机程序是 Spclient.java。
UDFcli.java	一个 JDBC 客户机应用程序，它调用 Java 用户定义函数库 UDFsrv 中的函数。
UseThrds.java	显示如何使用线程来异步运行 SQL 语句 (CLI 样本 async.c 的 JDBC 版本)。
V5SpCli.java	一个 Java 应用程序，它调用 DB2GENERAL 存储过程 V5Stp.java。
V5Stp.java	演示一个 DB2GENERAL 存储过程，它在服务器上更新 EMPLOYEE 表，然后将新的工资和工资单信息返回给客户机。该客户机程序是 V5SpCli.java。
Varinp.java	演示如何使用参数标记作为嵌入式动态 SQL 语句调用的输入变量。

表 9. Java 嵌入式 SQL (SQLJ) 样本程序

样本程序名	程序说明
App.sqlj	使用静态 SQL 检索和更新样本数据库的 EMPLOYEE 表中的数据。
App1t.sqlj	使用 JDBC Applet 的驱动程序查询数据库的 Applet。它使用在 App1t.html 中指定的用户名、口令、服务器和端口号参数。

表 9. Java 嵌入式 SQL (SQLJ) 样本程序 (续)

样本程序名	程序说明
Applt.html	嵌入 Applet 的样本程序 Applt 的 HTML 文件。它需要用服务器和用户信息进行定制。
Cursor.sqlj	演示使用静态 SQL 的迭代程序。
OpF_Curs.sqlj	Openftch 程序的类文件。
Openftch.sqlj	演示如何使用静态 SQL 取装、更新和删除行。
Outsrv.sqlj	演示使用 SQLDA 结构的存储过程。它使用 sample 数据库的 STAFF 表中雇员的平均薪水来填写 SQLDA。在完成数据库处理（查找平均值）后，存储过程将已填充的 SQLDA 和 SQLCA 状态返回给 JDBC 客户机程序 Outcli。
Stclient.sqlj	一个 SQLJ 客户机应用程序，它调用由 SQLJ 存储过程程序 Stserver 创建的 PARAMETER STYLE JAVA 存储过程。
Stcreate.db2	一个 CLP 脚本，它包含 CREATE PROCEDURE 语句，以将 Stserver 类中包含的方法注册为存储过程。
Stdrop.db2	一个 CLP 脚本，它包含 DROP PROCEDURE 语句，以取消注册 Stserver 类中包含的存储过程。
Stserver.sqlj	一个演示 PARAMETER STYLE JAVA 存储过程的 SQLJ 程序。客户机程序是 Stclient.sqlj。
Static.sqlj	使用静态 SQL 检索信息。
Stp.sqlj	一个存储过程，它更新服务器上的 EMPLOYEE 表并将新的薪水和工资信息返回给 JDBC 客户机程序 StpCli。
UDFclie.sqlj	一个客户机应用程序，它调用 Java 用户定义函数库 UDFsrv 中的函数。
Updat.sqlj	使用静态 SQL 更新数据库。

SQL 过程样本

表 10. SQL 过程样本程序

样本程序名	程序说明
basecase.db2	UPDATE_SALARY 过程提高 "sample" 数据库的 "staff" 表中由 "empno" IN 参数标识的员工的工资。该过程根据使用 "rating" IN 参数的 CASE 语句确定提高的数量。
basecase.sqc	调用 UPDATE_SALARY 过程。
baseif.db2	UPDATE_SALARY_IF 过程提高 "sample" 数据库的 "staff" 表中由 "empno" IN 参数标识的员工的工资。该过程根据使用 "rating" IN 参数的 IF 语句确定提高的数量。
baseif.sqc	调用 UPDATE_SALARY_IF 过程。
dynamic.db2	CREATE_DEPT_TABLE 过程将使用动态 DDL 来创建新表。该表的名称基于该过程的 IN 参数值。
dynamic.sqc	调用 CREATE_DEPT_TABLE 过程。

表 10. SQL 过程样本程序 (续)

样本程序名	程序说明
iterate.db2	ITERATOR 过程将使用 FETCH 循环检索 "department" 表的数据。如果 "deptno" 列的值为 'D11', 则将修改的数据插入 "department" 表中。如果 "deptno" 列的值为 'D11', 则 ITERATE 语句将控制流传送回 LOOP 语句的开头。
iterate.sqc	调用 ITERATOR 过程。
leave.db2	LEAVE_LOOP 过程统计在一个 LOOP 语句中执行的 FETCH 操作数, 然后 "not_found" 条件处理程序调用 LEAVE 语句。LEAVE 语句使控制流退出循环, 并结束该存储过程。
leave.sqc	调用 LEAVE_LOOP 过程。
loop.db2	LOOP_UNTIL_SPACE 过程统计在一个 LOOP 语句中执行的 FETCH 操作数, 直到游标检索到 "midinit" 列含有空格 (' ') 值的一行。该循环语句使控制流退出循环, 并结束该存储过程。
loop.sqc	调用 LOOP_UNTIL_SPACE 过程。
nestcase.db2	BUMP_SALARY 过程使用嵌套 CASE 语句来提高 "sample" 数据库的 "staff" 表中由 dept IN 参数指示的部门的员工工资。
nestcase.sqc	调用 BUMP_SALARY 过程。
nestif.db2	BUMP_SALARY_IF 过程使用嵌套 IF 语句来提高 "sample" 数据库的 "staff" 表中由 dept IN 参数指示的部门的员工工资。
nestif.sqc	调用 BUMP_SALARY_IF 过程。
repeat.db2	REPEAT_STMT 过程统计在一个重复语句中执行的 FETCH 操作数, 直到游标再也检索不到行。条件处理程序使控制流退出重复循环, 并结束该存储过程。
repeat.sqc	调用 REPEAT_STMT 过程。
rsultset.c	调用 MEDIAN_RESULT_SET 过程, 显示平均工资, 然后显示由 SQL 过程生成的结果集。此客户机是用 CLI API 编写的, 它可接收结果集。
rsultset.db2	MEDIAN_RESULT_SET 过程从 "sample" 数据库的 "staff" 表中获取由 "dept" IN 参数指示的部门的员工的平均工资。将该平均值赋予工资 OUT 参数, 并返回给 "rsultset" 客户机。该过程然后打开一个 WITH RETURN 游标, 以返回工资值大于平均值的员工的结果集。该过程将结果集返回给客户机。
spserver.db2	此 CLP 脚本中的 SQL 过程演示基本的错误处理、嵌套存储过程调用, 然后将结果集返回给客户机应用程序或发出调用的应用程序。可在 CLI 样本目录中使用 "spcall" 应用程序调用这些过程。还可使用位于 C 和 CPP 样本目录中的 "spclient" 应用程序调用不返回结果集的过程。
whiles.db2	DEPT_MEDIAN 过程从 "sample" 数据库的 "staff" 表中获取由 "dept" IN 参数指示的部门的员工的平均工资。将该平均值赋予工资 OUT 参数, 并返回给 "whiles" 客户机。然后 whiles 客户机打印平均工资。
whiles.sqc	调用 DEPT_MEDIAN 过程。

ADO、RDO 和 MTS 样本

表 11. ADO、RDO 和 MTS 样本程序

样本程序名	程序说明
Bank.vbp	一个 RDO 程序，它创建和维护银行支行的数据，能够对客户帐户执行事务处理。该程序可以使用用户指定的任何数据库，因为它包含 DDL，可以创建应用程序存储数据所需的表。
Blob.vbp	此 ADO 程序演示如何检索 BLOB 数据。它从 sample 数据库的 emp_photo 表中检索和显示图形。该程序也可用本地文件中的图象替换 emp_photo 表中的图象。
BLOBAccess.dsw	此样本演示如何用 Microsoft Visual C++ 存取 ADO/Blob。它类似于 Visual Basic 样本 Blob.vbp。该 BLOB 样本有两个主要功能： <ol style="list-style-type: none">1. 从 Sample 数据库读取 BLOB，并将它显示到屏幕上。2. 从文件读取 BLOB，并将它插入到数据库中。（调入）
Connect.vbp	此 ADO 程序将创建一个连接对象，并建立与 sample 数据库的连接。一旦完成，该程序将断开并退出。
Commit.vbp	此应用程序演示如何使用 ADO 的自动落实/人工落实功能部件。该程序查询 sample 数据库的 EMPLOYEE 表，以找到雇员号码和姓名。用户可以选择以自动落实方式或人工落实方式与数据库连接。在自动落实方式下，会在数据库中自动更新用户对一个记录所做的所有更改。在人工落实方式下，用户需要启动一个事务，然后才能进行任何更改。通过执行回滚，可以撤消自事务开始后所做的更改。通过落实该事务，可永久保存这些更改。退出该程序时将自动回滚这些更改。
db2com.vbp	此 Visual Basic 项目演示如何使用“Microsoft 事务服务器”更新数据库。它创建一个由客户机程序 db2mts.vbp 使用的服务器 DLL，并有四个类模块： <ul style="list-style-type: none">• UpdateNumberColumn.cls• UpdateRow.cls• UpdateStringColumn.cls• VerifyUpdate.cls 对于此程序，在 sample 数据库中创建了一个临时表 DB2MTS。
db2mts.vbp	这是一个客户机程序的 Visual Basic 项目，它使用“Microsoft 事务服务器”调用根据 db2com.vbp 创建的服务器 DLL。
Select-Update.vbp	此 ADO 程序执行与 Connect.vbp 相同的功能，但还提供一个 GUI 界面。有了此界面，用户可查看、更新和删除存储在 sample 数据库的 ORG 表中的数据。
Sample.vbp	此 Visual Basic 项目通过 ADO 使用 Keyset 游标，提供一个可存取 sample 数据库中所有数据的图形用户界面。
VarChar.dsp	一个 Visual C++ 程序，它使用 ADO 将 VarChar 数据作为文本字段存取。它提供一个图形用户界面，允许用户查看和更新 sample 数据库的 ORG 表中的数据。

“对象链接和嵌入”样本

表 12. “对象链接和嵌入” (OLE) 样本程序

样本程序名	程序说明
sales	演示对 Microsoft Excel 销售电子表格的上卷查询 (用 Visual Basic 实现)。
names	查询 Lotus Notes 通讯录 (用 Visual Basic 实现)。
inbox	通过 OLE/Messaging 查询 Microsoft Exchange 收件箱的电子邮件信息 (用 Visual Basic 实现)。
invoice	一种 OLE 自动化用户定义函数, 它将 Microsoft Word 票据文档作为电子邮件附件发送 (用 Visual Basic 实现)。
bcounter	一个 OLE 自动化用户定义函数, 使用实例变量演示暂存区 (用 Visual Basic 实现)。
ccounter	计数器 OLE 自动化用户定义函数 (用 Visual C++ 实现)。
salarysrv	一个 OLE 自动化存储过程, 它计算 sample 数据库中 STAFF 表的平均薪水 (用 Visual Basic 实现)。
salarycltvc	一个 Visual C++ 嵌入式 SQL 样本, 它调用 Visual Basic 存储过程 salarysrv。
salarycltvb	一个 Visual Basic DB2 CLI 样本, 它调用 Visual Basic 存储过程 salarysrv。
testcli	一个 OLE 自动化嵌入式 SQL 客户机应用程序, 它调用存储过程 tstsrv (用 Visual C++ 实现)。
tstsrv	一个 OLE 自动化存储过程, 演示在客户机和存储过程之间传递各种类型 (用 Visual Basic 实现)。

表 13. “对象链接和嵌入数据库” (OLE DB) 表函数

样本程序名	程序说明
jet.db2	Microsoft.Jet.OLEDB.3.51 提供器
mapi.db2	用于 MAPI 的 INTERSOLV Connect OLE DB
msdaora.db2	用于 Oracle 的 Microsoft OLE DB 提供器
msdasql.db2	用于 ODBC 驱动程序的 Microsoft OLE DB 提供器
msidxs.db2	Microsoft OLE DB 索引服务器提供器
notes.db2	用于 Notes 的 INTERSOLV Connect OLE DB
sampprov.db2	Microsoft OLE DB 样本提供器
sqloledb.db2	用于 SQL 服务器的 Microsoft OLE DB 提供器

命令行处理器样本

表 14. “命令行处理器” (CLP) 样本程序。

样本文件名	文件说明
const.db2	创建具有 CHECK CONSTRAINT 子句的表。
cte.db2	演示公共表表达式。演示此高级 SQL 语句的等效样本程序是 tabsql。

表 14. “命令行处理器” (CLP) 样本程序。(续)

样本文件名	文件说明
flt.db2	演示递归查询。演示此高级 SQL 语句的等效样本程序是 recursql。
join.db2	演示表的外连接。演示此高级 SQL 语句的等效样本程序是 joinsql。
stock.db2	演示触发器的用法。演示此高级 SQL 语句的等效样本程序是 trigsq1。
testdata.db2	使用 DB2 内部函数，如 RAND() 和 TRANSLATE(), 用随机生成的测试数据填充表。
thaisort.db2	此脚本专用于泰国语用户。泰国语排序是按语音次序进行的，它要求对前导元音及其辅音进行排序前的处理 / 交换，然后进行排序后的处理，以便按正确的排序次序查看数据。该文件实施泰国语排序的方法是这样的：先创建 UDF 函数 presort 和 postsort，然后创建一个表；再对该表调用这些函数以对表数据排序。要运行此程序，您首先必须根据 C 源文件 udf.c 构建用户定义函数程序 udf。

记录管理用户出口样本

表 15. “记录管理用户出口” 样本程序。

样本文件名	文件说明
db2uext2.cadsm	这是一个样本“用户出口”，它利用 ADSTAR DSM (ADSM) API 来归档和检索数据库记录文件。该样本提供调用的审查记录（有关每个选项的记录存储在单独的文件中），包括时间戳记和接收到的参数。它还提供出错调用的错误记录，包括时间戳记和用于确定问题的错误隔离字符串。可禁用这些选项。该文件必须重新命名为 db2uext2.c，并作为 C 程序编译。在 UNIX 和 Windows 32 位操作系统上可用。其 OS/2 版本是 db2uexit.cad。
db2uexit.cad	这是 db2uext2.cadsm 的 OS/2 版本。该文件必须重新命名为 db2uext2.c，并作为 C 程序编译。
db2uext2.cdisk	这是一个样本“用户出口”，它利用交付时所在的特定平台的系统复制命令。该程序归档和检索数据库记录文件，并提供调用的审查记录（每个选项的记录存储在单独的文件中），包括时间戳记和接收到的参数。它还提供出错调用的错误记录，包含时间戳记和用于确定问题的错误隔离字符串。可禁用这些选项。该文件必须重新命名为 db2uext2.c，并作为 C 程序编译。在 UNIX 和 Windows 32 位操作系统上可用。
db2uext2.ctape	这是一个样本“用户出口”，它利用交付时所在的特定 UNIX 平台的系统磁带命令。该程序归档和检索数据库记录文件。对系统磁带命令的所有限制也适用于此用户出口。该样本提供调用的审查记录（有关每个选项的记录存储在单独的文件中），包含时间戳记和接收到的参数。它还提供出错调用的错误记录，包括时间戳记和用于确定问题的错误隔离字符串。可禁用这些选项。该文件必须重新命名为 db2uext2.c，并作为 C 程序编译。仅在 UNIX 平台上可用。

第2章 设置

设置 OS/2 环境	32	创建、编目和联编样本数据库	37
设置 UNIX 环境	33	创建	38
设置 Windows 32 位操作系统环境	34	编目	39
在服务器上启用通信	36	联编	40
Windows NT 和 Windows 2000	36	下一步	41

创建一个适合构建和运行 DB2 应用程序的环境，要求正确设置如下项目：

1. 编译程序或解释程序
2. DB2（数据库管理程序、DB2 AD 客户机和客户机连接）
3. 操作系统环境
4. DB2 样本数据库（可选）

检查编译程序或解释程序环境

要开发 DB2 程序，必须使用操作系统所支持的一种程序设计语言的编译程序或解释程序，如第6页的『按平台分类的受支持软件』中所示。建议您首先构建一个非 DB2 应用程序，来确保现存的编译程序或解释程序的环境设置正确。然后，如果遇到问题，请查看与该编译程序或解释程序一起提供的文档。

设置 DB2 环境

要设置 DB2 环境，下列各项必须已安装并且起作用：

- 数据库管理程序，它安装在含有用于您环境的数据库实例的服务器上。如果需要有关数据库实例的信息，参考第343页的『附录A. 关于数据库管理程序实例』。
- DB2 AD 客户机，它安装在将用于开发应用程序的客户机或服务器工作站上。
- 与远程服务器的连接，如果您要在客户机工作站上进行开发。

更新数据库管理程序配置文件

本文件包含应用程序开发的重要设置。可通过输入以下命令更改这些设置：

```
db2 update dbm cfg using <keyword> <value>
```

且可通过输入以下命令查看该设置：

```
db2 get dbm cfg
```

有关这些命令用法的详情，参阅 *Command Reference*。

- 对于存储过程，关键字 `KEEPDARI` 的缺省值为 `yes`。它使存储过程进程保持活动。如果您正在开发存储过程，您可能想测试多次装入同一存储过程库。此缺省设置可能会干扰重新装入库。最好是在开发存储过程时将此关键字的值更改为 `no`，然后在要装入最后版本的存储过程时再将它更改回 `yes`。
- 对于 Java，更新 `JDK11_PATH` 关键字。有关详情，参阅第58页的『设置环境』。

有关安装和设置的详细信息，参考适合您操作系统的快速入门一书。

当以上各项已安装且在工作时，可执行下面其中一节的步骤设置操作系统环境：

- 『设置 OS/2 环境』
- 第33页的『设置 UNIX 环境』
- 第34页的『设置 Windows 32 位操作系统环境』

设置好操作系统环境之后，您可能想创建本书样本程序所使用的样本数据库。要创建数据库，参阅第37页的『创建、编目和联编样本数据库』。

设置 OS/2 环境

大多数 OS/2 编译程序使用环境变量来控制各种选项。可在 `CONFIG.SYS` 文件中设置这些变量，或可创建命令文件来设置它们。

CONFIG.SYS

在 `CONFIG.SYS` 文件中设置环境变量的优点在于：一旦输入，每次启动（引导）计算机时都设置它们。

命令文件

在命令文件中设置环境变量的优点是，路径较短并且可灵活地使用几个编译程序。缺点在于必须在启动每个程序设计对话时运行该命令文件。

如果通过运行命令文件来设置环境变量，则必须在设置环境变量的同一窗口中构建应用程序。如果在另一个窗口中构建应用程序，将不使用在第一个窗口中设置的相同选项。

当安装 DB2 AD 客户机时，将以下语句添加到 `CONFIG.SYS` 文件中：

```
set LIB=%DB2PATH%\lib;%LIB%
```

本书中的命令文件假设上述语句已存在。如果在安装 DB2 AD 客户机之后编辑了 `CONFIG.SYS` 文件，要确保没有删除此语句。

DB2 还会自动更新下列环境变量：

- `PATH`，将包括 `%DB2PATH%\bin` 目录
- `LIBATH`，将包括 `%DB2PATH%\dll` 目录

有关 DB2 更新的 Java 环境变量，参阅第61页的『OS/2』。

另外，如果使用的是以下显示的一种程序设计语言，则 CONFIG.SYS 文件必须有适当的语句：

C/C++ set INCLUDE=%DB2PATH%\include;%INCLUDE%

FORTAN

set FINCLUDE=%DB2PATH%\include;%FINCLUDE%

IBM COBOL

set SYSLIB=%SYSLIB%;%DB2PATH%\include\cobol_a

Micro Focus COBOL

set COBCPY=%DB2PATH%\include\cobol_mf;%COBCPY%

在 OS/2 上，除 DB2PATH 和 DB2INSTPROF 之外，不应再在 CONFIG.SYS 中定义其他的 DB2 环境变量。应在“DB2 实例简要表注册表”中定义全局级、实例级或实例节点级（并行版）的所有 DB2 变量。使用 db2set.exe 命令设置、修改和列出这些变量。

注：如果设置了 DB2INSTDEF 注册表变量，则不需要设置 DB2INSTANCE。如果没有设置 DB2INSTANCE，则它定义使用的缺省实例名。

设置 UNIX 环境

需要设置环境变量，以便可存取安装数据库管理程序时创建的数据库实例。*DB2 (UNIX 版) 快速入门*一书提供了有关设置环境变量的一般信息。本节提供有关设置环境变量以存取数据库实例的具体说明。

每个数据库管理程序实例都有两个文件：db2profile 和 db2cshrc，它们包含用来设置该实例的环境变量的脚本。根据您所使用的 shell，输入下列命令来运行该脚本：

对于 **bash** 或 **Korn shell**：

. \$HOME/sql1lib/db2profile

对于 **C shell**：

source \$HOME/sql1lib/db2cshrc

其中，\$HOME 是实例拥有者的主目录。

为方便起见，可将此命令加入 .profile 或 .login 文件，以便在您注册时它会自动运行。

取决于所在的 UNIX 平台，在创建 DB2 实例期间自动更新下列环境变量：

AIX:

- PATH, 将包括 sqllib/bin 在内的几个 DB2 目录
- LIBPATH, 将包括 sqllib/lib 目录

HP-UX:

- PATH, 将包括 sqllib/bin 在内的几个 DB2 目录
- SHLIB_PATH, 将包括 sqllib/lib 目录

Linux、PTX 和 Solaris:

- PATH, 包括 sqllib/bin 在内的几个 DB2 目录
- LD_LIBRARY_PATH, 包括 sqllib/lib 目录

Silicon Graphics IRIX:

- PATH, 将包括 sqllib/bin 在内的几个 DB2 目录
- LD_LIBRARY_PATH, 将包括目录 sqllib/lib (用于 o32 对象类型应用程序)
- LD_LIBRARYN32_PATH, 将包括目录 sqllib/lib32 (用于 n32 对象类型应用程序)

有关 DB2 更新的 Java 环境变量，参阅第58页的『设置环境』。

设置 Windows 32 位操作系统环境

在 Windows NT 或 Windows 2000 上安装 DB2 AD 客户机时，安装程序使用环境变量 INCLUDE、LIB、PATH、DB2PATH 和 DB2INSTANCE 更新配置注册表。缺省实例为 DB2。

有关 DB2 更新的 Java 环境变量，参阅第66页的『Windows 32 位操作系统』。

在 Windows 98 或 Windows 95 上安装 DB2 AD 客户机时，安装程序将更新 autoexec.bat 文件。

可以重设这些环境变量来设置机器或当前注册用户的值。要重设这些值，使用下列任何一项：

- Windows NT 控制面板
- Windows 2000 控制面板
- Windows 95 或 Windows 98 命令窗口
- Windows 95 或 Windows 98 autoexec.bat 文件

注:

1. 更改这些环境变量时要慎重。不要更改 DB2PATH 环境变量。
2. 在命令中使用 %DB2PATH% 变量时，用引号将完整路径括起，如 set LIB="%DB2PATH%\lib";%LIB% 中所示。在 DB2 版本 7.1 中，此变量的缺省安装值为 \Program Files\sql1lib，它包含空格，因此不使用引号会发生错误。

为了在 Windows 32 位操作系统上运行大多数程序，可更新这些环境变量。另外，为运行 DB2 应用程序必须执行下列特定的步骤：

- 在构建 C 或 C++ 程序时，必须确保 INCLUDE 环境变量包含作为第一个目录的 %DB2PATH%\INCLUDE。

例如，Microsoft Visual C++ 编译程序环境设置文件 Vc\bin\vcvars32.bat 包含以下命令：

```
set INCLUDE=%MSVCDir%\INCLUDE;%MSVCDir%\MFC\INCLUDE;%MSVCDir%\ATL\INCLUDE;%INCLUDE%
```

要让 DB2 使用此文件，首先将 %INCLUDE%（它设置 %DB2PATH%\INCLUDE 路径）从该列表的尾部移至列表的开头，如下所示：

```
set INCLUDE=%INCLUDE%;%MSVCDir%\INCLUDE;%MSVCDir%\MFC\INCLUDE;%MSVCDir%\ATL\INCLUDE;
```

- 当构建 Micro Focus COBOL 程序时，设置 COBCPY 环境变量指向 %DB2PATH%\INCLUDE\cobol_mf。
- 当构建 IBM COBOL 程序时，设置 SYSLIB 环境变量指向 %DB2PATH%\INCLUDE\cobol_a。
- 使用以下语句，确保 LIB 环境变量指向 %DB2PATH%\lib:

```
set LIB="%DB2PATH%\lib";%LIB%
```

- 确保在远程数据库的服务器上已设置 DB2COMM 环境变量。
- 确保在服务器端已启动安全服务，以便进行 SERVER 认证；而在客户机端的安全服务取决于 CLIENT 认证的级别。要启动安全服务，使用 NET START DB2NTSECSERVER 命令。

注:

1. 所有 DB2 环境变量都可在用户的环境中定义或作为注册表变量来设置。有关注册表变量的信息，请参阅管理指南。有关 db2set 命令的信息，参阅 *Command Reference*。
2. DB2INSTANCE 只应定义在用户环境级上。如果使用 DB2INSTDEF 注册表变量（它定义未设置 DB2INSTANCE 时使用的缺省实例名），则不需要它。
3. Windows NT 或 Windows 2000 环境中的数据库管理程序是作为一个 Windows NT 服务或 Windows 2000 服务实现的，因此如果该服务成功启动，尽管可能发生其他问题，也不会返回错误或警告。这就意味着当您运行 db2start 或 NET START 命令时，如果任何通信子系统启动失败，则不会返回警告信息。因此，用

户应经常检查 Windows NT 或 Windows 2000 事件记录或 DB2DIAG.LOG 文件，以查看在这些命令运行期间可能发生的错误。

在服务器上启用通信

本节说明如何连接“DB2 通用数据库”服务器。

在开始安装、编目和联编 sample 数据库之前，应确保服务器可以使用且配置为支持要编目的协议。在服务器上执行下列操作：

1. 确保已设置 db2comm 环境变量。例如，如果使用 TCP/IP，请输入：

```
db2set DB2COMM=tcpip
```

并确保配置了 TCP/IP 支持协议。

参考您平台的快速入门一书，以获得向 Services 文件添加 TCP/IP 设置的说明。

2. 输入以下命令启动数据库实例：

```
db2start
```

必须从客户机完成将实用程序与样本数据库的联编。有关详情，参阅第40页的『联编』。

Windows NT 和 Windows 2000

在 DB2 Windows NT 版或 DB2 Windows 2000 版生产系统中，必须将数据库实例作为服务启动。步骤如下：

- 如果使用通信协议，则确保已在 Windows NT 或 Windows 2000 控制面板的“系统环境变量”部分中设置了 db2comm 环境变量。
- 启动安全服务。可自动完成此操作（查看下面的注意事项），或使用以下命令人工启动此服务：

```
NET START DB2NTSECSEVER
```

- 输入以下命令启动实例：

```
db2start
```

自动启动“安全服务” 通常只有在工作站充当 DB2 客户机，并且与为“客户机认证”配置好的服务器连接时，才需要将安全服务设置为自动启动。要自动启动安全服务，执行下列操作：

Windows NT

1. 单击“开始”按钮。
2. 单击“设置”。

3. 单击“控制面板”。
4. 在“控制面板”中，单击“服务”。
5. 在“服务”窗口中，突出显示“DB2 安全服务器”。
6. 如果它没有列出“已启动”和“自动”设置，则单击“启动”。
7. 单击“自动”。
8. 单击“确认”。
9. 重新引导机器使设置生效。

Windows 2000

1. 单击“开始”按钮。
2. 单击“设置”。
3. 单击“控制面板”。
4. 单击“管理工具”。
5. 单击“服务”。
6. 在“服务”窗口中，突出显示“DB2 安全服务器”。
7. 如果它没有列出“已启动”和“自动”设置，则从顶部菜单单击“操作”。
8. 单击“属性”。
9. 确保您在“常规”标签中。
10. 从“启动类型”下拉菜单中选择“自动”。
11. 单击“确认”。
12. 重新引导机器使设置生效。

创建、编目和联编样本数据库

要使用与 DB2 一起交付的样本程序，需要在服务器工作站上创建 sample 数据库。有关 sample 数据库内容的列表，参考 *SQL Reference*。

如果要使用远程客户机存取服务器上的 sample 数据库，则需要对客户机工作站上对 sample 数据库编目。

另外，如果想使用正在运行另一个版本的 DB2 或正在另一个操作系统上运行的远程客户机存取服务器上的 sample 数据库，则需要将包括 DB2 CLI 在内的数据库实用程序与 sample 数据库联编。

创建

要创建 `sample` 数据库，必须具有 `SYSADM` 权限。有关 `SYSADM` 权限的详情，参考适合您操作系统的快速入门一书。

要创建该数据库，在服务器上执行下列操作：

1. 确保路径中包含 `db2samp1`（创建 `sample` 数据库的程序）的位置。`db2profile` 或 `db2cshrc` 文件将把 `db2samp1` 置于您的路径中，因此，如果不加以更改，该程序将保留在您的路径中。

- 在 UNIX 服务器上，`db2samp1` 位于：

```
$HOME/sqllib/bin
```

其中，`$HOME` 是 DB2 实例拥有者的主目录。

- 在 OS/2 和 Windows 上，`db2samp1` 位于以下目录中：

```
%DB2PATH%\bin
```

其中，`%DB2PATH%` 是安装有 DB2 的路径。

2. 确保将 `DB2INSTANCE` 环境变量设置为要创建 `sample` 数据库的实例的名称。如果未设置它，可使用以下命令进行设置：

- 在 UNIX 平台上：

如果使用 `bash shell` 或 `Korn shell`，则可输入以下命令进行设置：

```
DB2INSTANCE=instance_name  
export DB2INSTANCE
```

如果使用 `C shell`，则可输入以下命令进行设置：

```
setenv DB2INSTANCE instance_name
```

- 在 OS/2 和 Windows 上，输入：

```
set DB2INSTANCE=instance_name
```

其中 `instance_name` 是数据库实例的名称。

3. 输入 `db2samp1`，后跟您想创建的样本数据库的位置，以创建 `sample` 数据库。在 UNIX 平台上，该位置是 `path`（路径），应输入为：

```
db2samp1 path
```

在 OS/2 和 Windows 上，该位置是 `drive`（驱动器），应输入为：

```
db2samp1 drive
```

如果不指定路径或驱动器，安装程序将把样本表安装在数据库管理程序配置文件的 `DFTDBPATH` 参数所指定的缺省路径或驱动器中。如果需要有关该配置文件的信息，参考管理指南。

数据库的认证类型与在其中创建该数据库的实例的认证类型相同。如果需要有关在创建数据库实例时指定认证的更多信息，参考快速入门一书。

在主机或 AS/400 服务器上创建

如果要对主机服务器（如 DB2 OS/390 版或 AS/400 服务器）运行样本程序，则需创建一个包含在 *SQL Reference* 中所述的样本表的数据库。可能要参考样本程序 `expsamp`，它使用 `STAFF` 和 `ORG` 表来演示如何使用 API 将表和表数据调入和调出 DB2 Connect 数据库。

要创建数据库：

1. 使用 `db2samp1` 在 DB2 服务器实例中创建 `sample` 数据库。
2. 连接 `sample` 数据库。
3. 将样本表调出到一个文件。
4. 连接 DB2 Connect 数据库。
5. 创建样本表。
6. 调入样本表。

如果需要有关调出和调入文件的信息，参考 *Data Movement Utilities Guide and Reference*。如果需要有关连接数据库以及创建表的信息，参考 *SQL Reference*。

编目

要从远程客户机存取服务器上的 `sample` 数据库，需要在客户机工作站上对 `sample` 数据库编目。

不需要在服务器工作站上编目 `sample` 数据库，因为在创建该数据库时已将它编目。

编目将使用客户机应用程序要存取的数据库的名称更新客户机工作站上的数据库目录。在处理客户机请求时，数据库管理程序使用已编目的名称来查找和连接数据库。

快速入门一书提供有关编目数据库的一般信息。本节提供有关编目 `sample` 数据库的具体说明。

要在远程客户机工作站上编目样本数据库，输入：

```
db2 catalog database sample as sample at node nodename
```

其中 `nodename` 是服务器节点的名称。

快速入门一书说明如何编目节点，这是设置通信协议的一部分。在可以连接数据库之前，还必须编目远程节点。

联编

如果要从正在运行另一个版本的 DB2 或正在另一个操作系统上运行的远程客户机存取服务器上的 `sample` 数据库，则需要将包括 DB2 CLI 在内的数据库实用程序与 `sample` 数据库联编。

联编会创建一个程序包，当执行应用程序时数据库管理程序需要用它来存取数据库。通过对在预编译期间创建的联编文件指定 `BIND` 命令，可以显式执行联编。

Command Reference 提供有关联编数据库实用程序的一般信息。本节提供有关将数据库实用程序与 `sample` 数据库联编的具体说明。

根据使用的客户机工作站的平台，联编数据库实用程序的方式不同。

在 OS/2 客户机工作站上:

1. 输入如下命令，连接 `sample` 数据库:

```
db2 connect to sample user userid using password
```

其中 *userid* 和 *password* 是 `sample` 数据库所在的实例的用户 ID 和口令。

使用此命令，DB2 自动将实用程序与数据库联编，因此用户不必显式地联编它们。

2. 退出“命令行处理器”，并检查联编信息文件 `bind.msg` 来验证联编成功。

在 UNIX 客户机工作站上:

1. 输入如下命令，连接 `sample` 数据库:

```
db2 connect to sample user userid using password
```

其中 *userid* 和 *password* 是 `sample` 数据库所在的实例的用户 ID 和口令。

2. 输入如下命令，将实用程序与数据库联编:

```
db2 bind BNDPATH/@db2ubind.lst blocking all sqlerror continue \  
messages bind.msg
```

```
db2 bind BNDPATH/@db2cli.lst blocking all sqlerror continue \  
messages cli.msg
```

其中 *BNDPATH* 是联编文件所在的路径，如 `$HOME/sqlllib/bnd`，而 `$HOME` 是 DB2 实例拥有者的主目录。

3. 检查联编信息文件 `bind.msg` 和 `cli.msg`，以验证联编成功。

在运行 **Windows 32** 位操作系统的客户机工作站上:

1. 从“开始”菜单选择“程序”。
2. 从“程序”菜单选择 IBM DB2。
3. 从 IBM DB2 菜单选择 DB2 命令窗口。
显示该命令窗口。

4. 输入如下命令, 连接 `sample` 数据库:

```
db2 connect to sample user userid using password
```

其中 `userid` 和 `password` 是 `sample` 数据库所在的实例的用户 ID 和口令。

5. 输入如下命令, 将实用程序与数据库联编:

```
db2 bind "%DB2PATH%\bnd\@db2ubind.lst" blocking all  
sqlerror continue messages bind.msg
```

其中 `%DB2PATH%` 是安装有 DB2 的路径。

6. 退出命令窗口, 检查联编信息文件 `bind.msg` 以验证联编成功。

对于所有平台

如果在 DRDA 兼容的应用程序服务器上创建了 `sample` 数据库, 则指定下列 `.lst` 文件之一以代替 `db2ubind.lst`:

ddcsmvs.lst

用于 DB2 OS/390 版

ddcsvm.lst

用于 DB2 VM 版

ddcsvse.lst

用于 DB2 VSE 版

ddcs400.lst

用于 DB2 AS/400 版

适合您操作系统的快速入门一书提供了有关联编数据库实用程序的一般信息。

下一步

一旦设置好环境, 就可以构建 DB2 应用程序了。以下各章讨论样本程序, 并演示如何进行编译、连接和运行。首先阅读第43页的『第3章 构建 DB2 应用程序的一般信息』, 然后针对您要构建的应用程序参阅随后的特定章节。

要使用 Java 进行程序设计, 参阅第57页的『第4章 构建 Java Applet 和应用程序』。

要使用 SQL 过程进行程序设计，参阅第83页的『第5章 构建 SQL 过程』。

要使用 DB2 API、DB2 CLI 和嵌入式 SQL 进行程序设计，参阅针对您平台的“构建应用程序”章节。

有关详情，参考以下列书籍：

- *Application Development Guide* 一书描述使用嵌入式 SQL、JDBC 和 SQLJ 以及用户定义函数 (UDF) 构建应用程序。
- *CLI Guide and Reference* 一书描述使用 DB2 CLI 或 ODBC 构建应用程序。
- *Administrative API Reference* 一书描述构建 DB2 API 应用程序。

第3章 构建 DB2 应用程序的一般信息

构建文件、Makefile 和错误检查实用程序	44	DB2 调用层接口 (CLI) 应用程序	51
构建文件	44	嵌入式 SQL 应用程序	51
Makefile	46	存储过程	53
错误检查实用程序	48	用户定义函数 (UDF)	54
Java Applet 和应用程序	49	多线程应用程序	54
DB2 API 应用程序	50	用 C++ 编写 UDF 和存储过程的考虑事项	54

本章信息适用于多种操作系统。大部分主题适用于大多数 DB2 支持的平台。

有关最新的 DB2 应用程序开发情况的更新，请访问以下 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

构建和运行 DB2 程序的一般要点

1. 应用程序环境：

- OS/2: 如果想通过命令文件而非 CONFIG.SYS 文件来设置环境变量，必须在设置环境变量的同一窗口中构建应用程序。
- UNIX: 必须在设置环境变量的外壳中构建和运行 DB2 应用程序。取决于所用的外壳，可通过运行 db2profile 或 db2cshrc 来完成此操作。
- Windows 32 位操作系统: 必须在 DB2 命令窗口中构建应用程序。有关详情，参考第31页的『第2章 设置』。

2. 要构建含有嵌入式 SQL 的 DB2 程序，或要运行任何 DB2 程序，必须启动服务器上的数据库管理程序。要启动数据库管理程序，您需要 SYSADM（系统管理）权限。有关 SYSADM 权限的信息，参考快速入门。

如果数据库管理程序尚未运行，在服务器上输入如下命令启动它：

```
db2start
```

- #### 3. 当构建用于生产的应用程序时，构建到可执行文件中的 DB2 运行期路径应为安装路径，而不是开发应用程序的本地 DB2 实例的路径。本书旨在介绍如何在开发环境中构建应用程序，因此描述了 UNIX 上 sqllib/include 和 sqllib/lib 的实例副本，以及 OS/2 和 Windows 32 位操作系统上 %DB2PATH%\include 和 %DB2PATH%\lib 的实例副本。
- #### 4. 建议在改变或构建样本程序之前，将要使用的语言样本从 UNIX 平台上的 sqllib/samples 目录中，或从 OS/2 平台或 Windows 32 位操作系统上的 %DB2PATH%\samples 目录中复制到自己的工作目录下。这样，可以保留原始样本以备将来参考之用。

构建文件、Makefile 和错误检查实用程序

DB2 提供了一组用于程序开发的构建工具。这些工具便于您构建提供的样本程序，它们演示了大量 DB2 功能。这些工具还可用来构建自己的数据库程序。对于每种受支持的编译程序，DB2 提供了构建文件、makefile 和错误检查实用程序，可在 samples 目录中找到这些文件和样本程序。本节说明如何使用这些工具。

构建文件

下面每一章都使用包含编译和连接命令的文件，以使用受支持的平台编译程序构建程序。这些文件在 OS/2 中称为命令文件，在 UNIX 中称为脚本文件，而在 Windows 32 位操作系统中称为批处理文件。我们将它们统称为构建文件。

DB2 为受支持的平台上的每一种语言提供了构建文件，这些文件位于每种语言的样本程序目录下，用它们构建的各种类型程序可在这些受支持的平台运行。表16列示所有受支持的平台上的所有语言的构建文件，Windows 32 位操作系统上的 C++ 除外。省略了文件扩展名。对于 OS/2，扩展名为 .cmd；对于 Windows 32 位操作系统，扩展名为 .bat。对于 UNIX 平台，没有扩展名。

对于 Windows 32 位操作系统，支持两种 C++ 编译程序：Microsoft Visual C++ 和 IBM VisualAge C++。对于这两种编译程序，已分别在每个构建文件名的 "bld" 后插入 "m" 或 "v" (bldclisp 除外)，变为 bldmclis 或 bldvclis。在第45页的表17中列示了这些构建文件。省略了扩展名 .bat。

表 16. DB2 构建文件

构建文件	构建的程序类型
bldsqlj	Java 嵌入式 SQLJ 应用程序。
bldsqljs	Java 嵌入式 SQLJ 存储过程。
bldcli	使用或不使用嵌入式 SQL 的 DB2 CLI 应用程序。
bldapi	使用或不使用嵌入式 SQL 的 DB2 CLI 应用程序，要求在 utilapi 实用程序文件（包含创建和卸下数据库所需的 DB2 API）中连接。
bldclisp	DB2 CLI 存储过程（非嵌入式 SQL）。
bldapp	使用或不使用嵌入式 SQL 的应用程序。
bldsrv	嵌入式 SQL 存储过程。
bldudf	用户定义函数 (UDF)。
bldmt	多线程嵌入式 SQL 应用程序（只适用于受支持的 UNIX 平台上的 C/C++）。
blddevm	事件监控程序的样本程序 evm（只能在 AIX 和 OS/2 上运行）。

表 17. 用于 Windows 32 位操作系统的 C/C++ 构建文件

构建文件	构建的程序类型
bldmcli	使用或不使用嵌入式 SQL 的 Microsoft Visual C++ DB2 CLI 应用程序。
bldvcli	使用或不使用嵌入式 SQL 的 VisualAge C++ DB2 CLI 应用程序。
bldmapi	使用或不使用嵌入式 SQL 的 Microsoft Visual C++ DB2 CLI 应用程序，要求在 utilapi 实用程序文件（包含创建和卸下数据库所需的 DB2 API）中连接。
bldvapi	使用或不使用嵌入式 SQL 的 VisualAge C++ DB2 CLI 应用程序，要求在 utilapi 实用程序文件（包含创建和卸下数据库所需的 DB2 API）中连接。
bldmclis	Microsoft Visual C++ DB2 CLI 存储过程（非嵌入式 SQL）。
bldvclis	VisualAge C++ DB2 CLI 存储过程（非嵌入式 SQL）。
bldmapp	使用或不使用嵌入式 SQL 的 Microsoft Visual C++ 应用程序。
bldvapp	使用或不使用嵌入式 SQL 的 VisualAge C++ 应用程序。
bldmsrv	Microsoft Visual C++ 嵌入式 SQL 存储过程。
bldvsrv	VisualAge C++ 嵌入式 SQL 存储过程。
bldmudf	Microsoft Visual C++ 用户定义函数 (UDF)。
bldvudf	VisualAge C++ 用户定义函数 (UDF)。

本书中描述了这些构建文件，因为它们非常清晰地演示了用受支持的编译程序构建各种程序所用到的（DB2 建议使用的）编译和连接选项。除此之外，还有很多其他可用的编译和连接选项，用户可以随意试用它们。参阅您的编译程序文档，以了解提供的全部编译和连接选项。除构建样本程序外，开发人员还可以用构建文件构建自己的程序。这些样本程序可用作模板，用户可修改这些模板来辅助程序设计开发。

为方便起见，构建文件被设计为可以构建具有编译程序允许的任何文件名的源文件。这不同于 makefile，后者的程序名被以硬编码的方式写入文件中。在 UNIX 上构建文件使用 \$1 变量内部替换程序名，而在 OS/2 和 Windows 32 位操作系统上使用 %1 变量内部替换程序名。其他类似的命名变量替换可能需要的其他自变量。这些构建文件可用来进行快速简单的实验，因为每一种构建文件都适合特定类型的程序构建，如 DB2 API、DB2 CLI、嵌入式 SQL、存储过程和用户定义函数。凡是受编译程序支持的特定类型的程序，都提供了对应类型的构建文件。

每次构建一个程序后，即使没有修改源文件，也会自动覆盖构建文件所生成的目标文件和可执行文件。这不同于 makefile。这意味着开发人员可重新构建现存的程序，而不必删除之前的目标文件和可执行文件或修改源文件。

构建文件包含样本数据库的缺省设置。如果用户要存取另一个数据库，只需提供另一个参数来覆盖缺省值。如果他们始终使用该数据库，应将数据库名以硬编码的方式写入构建文件来替换 `sample`。

用于构建嵌入式 SQL 程序的构建文件调用另一个文件 `embprep`，该文件包含构建嵌入式 SQL 程序所需的预编译和联编步骤。这些步骤可能需要用户 ID 和口令可选参数，这取决于构建嵌入式 SQL 程序的环境。

如果开发人员在数据库所在的服务器实例上构建程序，则用户 ID 和口令是公共参数，因此不必提供这两个参数。另一方面，如果开发人员在另一个实例中，比如在远程存取服务器数据库的客户机上，则必须提供这两个参数。`makefile` 也使用 `embprep` 文件。有关此文件的详情，参阅下面的『`Makefile`』。

最后，开发人员为了自身的方便，可修改构建文件。除了可更改构建文件中的数据库名之外（以上说明过），开发人员还可轻易地将其他参数以硬编码的方式写入到该文件中、更改编译和连接选项，或更改缺省的 DB2 实例路径。构建文件简单明了、独特的特性使您可以轻松地修改它们，以满足自己的需要。

Makefile

受支持的编译程序的每个 `samples` 目录都包含构建提供的样本程序所需的 `makefile`。`makefile` 构建大多数随编译程序交付的 DB2 样本程序。它使用变量来表示每个样本程序编译时会用到的许多公共元素。`makefile` 的语法及其命令的输出在某些重要的方面与提供的构建文件不同。`make` 命令简单易用且功能强大：

make <program_name>

编译和连接指定的程序。

make all

编译和连接 `makefile` 中列出的所有程序。

make clean

删除 `makefile` 中列出的所有程序的中间文件，如目标文件。

make cleanall

删除 `makefile` 中列出的所有程序的所有中间文件和可执行文件。

与构建文件不同，`makefile` 不会覆盖它里面所列出的程序的现存中间文件和可执行文件。如果部分文件已有可执行文件，使用 `make all` 命令可以更快地为其他文件创建可执行文件，因为 `make all` 将忽略已有可执行文件的那些文件。但这也说明了 `make clean` 和 `make cleanall` 命令的用处，可使用这两个命令删除那些不需要的现存的目标文件和可执行文件。

makefile 可用于程序开发。与构建文件相比，它们使用起来不太方便，但如果您想利用 make 命令强大的功能和简便性，可考虑使用它。要在现存的 makefile 中包括新程序，将一个类似的现存样本程序作为模板，并在语法中定义该程序项。注意，DB2 API、DB2 CLI 和嵌入式 SQL 程序、存储过程和 UDF 的语法是不同的。

下面是说明如何使用提供的 makefile 的一个示例。这个 makefile 位于 AIX 上的 samples/cli 目录下，它用 IBM C 编译程序构建 AIX 上所有提供的 DB2 CLI 样本。此示例演示如何将存储过程程序 spserver 构建为可由客户机应用程序 spclient 调用的共享库。

在使用 makefile 之前，可能需要编辑该文件中包含的下列变量：

- DB** 使用的数据库。缺省情况下，设置为 sample。
- UID** 使用的用户 ID。缺省情况下，不设置任何值。
- PWD** UID 用户 ID 的口令。缺省情况下，不设置任何值。

在 AIX 上，DB2 CLI makefile 定义 DB2PATH、CC、COPY、ERASE、CFLAGS 和 LIBS 变量，如下所示：

```
# Set DB2PATH to where DB2 will be accessed.
# The default is the instance path.
DB2PATH=$(HOME)/sqlib

CC = cc
COPY = cp
ERASE = rm -f

# The required compiler flags
CFLAGS= -I$(DB2PATH)/include

# The required libraries
LIBS= -L$(DB2PATH)/lib -Wl,-rpath,$(DB2PATH)/lib -ldb2
```

makefile 在编译存储过程 spserver 并将共享库复制到 function 子目录中时，会使用这些变量：

```
spserver : utilcli.o
    $(CC) -o spserver spserver.c utilcli.o $(CFLAGS) $(LIBS) \
    -H512 -T512 -bE:spserver.exp -e outlanguage
    $(ERASE) $(DB2PATH)/function/spserver
    $(COPY) spserver $(DB2PATH)/function/spserver
```

对于嵌入式 SQL 程序，makefile 调用 embprep 文件，该文件包含构建这些嵌入式 SQL 程序所需的预编译和联编步骤。使它成为一个独立的文件，这样每个嵌入式

SQL 程序都可以调用它，而不必在 `makefile` 主体内重复这些步骤。此文件使用用户 ID、口令和数据库名的参数与服务器上的数据库连接。当 `makefile` 调用 `embprep` 时，就把这些值传送给它。

缺省情况下，将数据库变量 `DB` 硬编码到 `sample` 数据库中；如果要使用另一个数据库，用户可更改此变量。缺省情况下，不将用户 ID 和口令变量（即 `UID` 和 `PWD`）设置成任何值。如果用户已在服务器数据库所在的同一实例中工作，则不必使用这些可选参数。然而，如果不是这样，例如，如果用户从客户机远程连接服务器，那么用户可以给 `UID` 和 `PWD` 变量赋予恰当的值来修改 `makefile`，然后这两个变量值将自动传送给 `embprep` 预编译和联编文件。以下是一个示例，说明在 Windows NT 上由 Micro Focus COBOL `makefile` 调用 `embprep` 文件来构建嵌入式 SQL 应用程序 `updat`:

```
updat.cb1 : updat.sqb
            embprep updat $(DB) $(UID) $(PWD)
updat.obj : updat.cb1
            $(CC) updat.cb1 ;
updat : updat.obj checkerr.obj
        $(LINK) updat.obj checkerr.obj $(LIBS)
```

错误检查实用程序

DB2 AD 客户机提供几种实用程序文件。这些文件具有错误检查和打印出错信息的功能。一个例外是 CLI 实用程序文件 `utilapi.c`，它调用 DB2 API 来创建和卸下数据库。在 `samples` 目录中为每一种语言都提供了实用程序文件。当用于某个应用程序时，错误检查实用程序文件会提供有用的出错信息，从而使调试 DB2 程序更容易。大多数错误检查实用程序使用 DB2 API `GET SQLSTATE MESSAGE` 和 `GETERROR MESSAGE` 获取与在程序执行过程中所遇到的问题相关的 `SQLSTATE` 和 `SQLCA` 信息。DB2 CLI 实用程序文件 `utilcli.c` 不使用这些 DB2 API，而使用等效的 DB2 CLI 语句。使用所有错误检查实用程序打印描述性出错信息，以便开发人员快速了解问题。

一些 DB2 程序，如存储过程和用户定义函数，不需要使用这些实用程序。Java 也不需要这些实用程序，因为如果发生异常情况，将抛出 `SQLException` 对象。

下面是 DB2 支持的、不同程序设计语言的编译程序所用的错误检查实用程序文件：

```
checkerr.cb1
    用于 COBOL 程序

utilcli.c
    用于 CLI 程序
```


utilapi.c

用于 C 非嵌入式 SQL 程序

utilemb.sqc

用于 C 嵌入式 SQL 程序

utilapi.C

用于 C++ 非嵌入式 SQL 程序

utilemb.sqC

用于 C++ 嵌入式 SQL 程序

为了使用这些实用程序函数，首先必须编译该实用程序文件，然后在创建目标程序的可执行文件期间连接它的目标文件。由 `samples` 目录中的 `makefile` 和构建文件为那些需要错误检查实用程序的程序完成此操作。

以下示例演示在 DB2 程序中如何使用错误检查实用程序。 `utilemb.h` 头文件定义代替 `SqlInfoPrint()` 函数和 `TransRollback()` 函数的 `EMB_SQL_CHECK` 宏：

```
#define EMB_SQL_CHECK( MSG_STR ) \
    if( SqlInfoPrint( MSG_STR, &sqlca, __LINE__, __FILE__ ) != 0 ) TransRollback( );
```

`SqlInfoPrint()` 检查 `SQLCODE` 标记。并打印出与此标记指示的特定错误相关的任何可用信息。还指出该错误发生在源代码中何处。`TransRollback()` 允许实用程序文件安全回滚发生错误的事务。此函数需要调用嵌入式 SQL 语句来连接数据库并执行回滚。下面是一个示例，说明 C++ 程序 `cursor` 如何使用此宏调用实用程序函数，它为 `SqlInfoPrint()` 函数的 `MSG_STR` 参数提供值 "DECLARE CURSOR"。

```
Cursor::Fetch () {
    EXEC SQL DECLARE c1 CURSOR FOR
        SELECT name, dept FROM staff WHERE job='Mgr'
        FOR UPDATE OF job;
    EMB_SQL_CHECK("DECLARE CURSOR") ;
```

`EMB_SQL_CHECK` 宏确保如果 `DECLARE` 语句失败，将安全回滚事务，并打印出一条恰当的出错信息。

鼓励开发人员在创建自己的 DB2 程序时利用这些错误检查实用程序。

Java Applet 和应用程序

要构建 Java Applet 和应用程序，应在所有平台上执行相同的步骤，因此有一章专门介绍这方面的信息，即第57页的『第4章 构建 Java Applet 和应用程序』。每种平台需要特定的设置信息，该章不同节段分别给出了这些信息。在第31页的『第2章 设置』中，除了介绍 DB2 设置信息外，也介绍了这方面的设置信息。

Java 这一章说明如何构建使用 JDBC 驱动程序的 JDBC 程序，并说明如何构建除了使用 JDBC 驱动程序外还使用 Java 支持的嵌入式 SQL 的 SQLJ 程序。该章说明如何构建 JDBC 和 SQLJ 的 Applet、应用程序和存储过程。还说明了如何构建 Java 用户定义函数，这种函数不能包含 JDBC 和 SQLJ 语句。

samples 目录包含适用于 SQLJ 程序的 Java makefile 和构建文件。在命令行上构建 JDBC 程序比较容易，因此没有为这些程序提供构建文件。

DB2 API 应用程序

DB2 AD 客户机包括调用 DB2 API 的样本程序。本书后面的“构建应用程序”几章说明如何使用与该平台的 DB2 AD 客户机一起提供的 DB2 应用程序构建文件，为受支持的编译程序构建样本程序。还可使用提供的 makefile。makefile 和构建文件显示了可使用的编译程序选项。在适当的章节中，为每个平台上受支持的编译程序都定义了这些选项。您可能需要针对自己的环境修改这些选项。

本书中使用下面的样本程序来演示使用受支持的程序设计语言构建和运行 DB2 API 应用程序的步骤。要执行的步骤可能随环境的不同而改变：

client 演示下列 API 的用法：SET CLIENT 和 QUERY CLIENT

有关所有 DB2 API 样本程序的详细说明，参阅第12页的『Sample Program Tables』。

受支持的 DB2 API 样本程序的源文件，在 UNIX 上位于 `sqlllib/samples` 下适当的程序设计语言子目录中；而在 OS/2 和 Windows 32 位操作系统上，位于 `%DB2PATH%\samples` 下适当的程序设计语言子目录中。

当构建样本程序后，可用它们作为模板来创建自己的应用程序。可使用提供的 makefile 或构建文件来构建 DB2 API 程序。

注：当编写 API 应用程序时，所有 API 输入结构应 `memset` 为 0，这样非显式设置的结构元素将被初始化为 0。这对重新编译使用低级别版本的 API 编写的应用程序很重要。当重新编译时，使用新定义的结构，但所有元素可能没有初始化。调用 `memset` 将确保它们初始化。以下是一个示例，显示已 `memset` 为 0 的输入数据结构 `pLoadInStruct`：

```
memset( pLoadInStruct, 0, sizeof(pLoadInStruct) );
```

DB2 调用层接口 (CLI) 应用程序

DB2 AD 客户机带有使用“DB2 调用层接口”(DB2 CLI) 函数调用的样本程序。您可以研究这些样本，以了解如何在应用程序中使用这些函数调用来存取 DB2 数据库。

符合 ODBC 的 DB2 CLI 应用程序可以移植到 ODBC 下工作，只要使用 ODBC SDK (DB2 不带此工具箱) 重新编译该应用程序，而且在该应用程序平台上有一个 ODBC 驱动程序管理器可用。

样本程序、构建文件和 makefile 包含在 UNIX 上的 `sqllib/samples/cli` 目录下或 OS/2 和 Windows 32 位操作系统上的 `%DB2PATH%\samples\cli` 目录下。可能需要针对您的环境修改构建文件和 makefile 中的编译程序选项。

本书中使用下面的样本程序来演示构建和运行 DB2 CLI 应用程序的步骤（要执行的步骤取决于您的环境）：

tbinfo 演示如何获取和设置表级别的信息。

dbusemx

演示如何对单个数据库使用嵌入式 SQL。

dbmconn

演示如何连接和断开多个数据库。

spclient

是客户机 / 服务器示例中的客户机程序；服务器程序是 `spserver`。

spserver

是客户机 / 服务器示例中的服务器程序；客户机程序是 `spclient`。

udfcli 使用由用户定义函数程序 `udfsrv` 创建的函数。

有关所有 DB2 CLI 样本程序的详细说明，参阅第22页的表7。 *CLI Guide and Reference* 说明使用 DB2 CLI 的样本是如何工作的。

嵌入式 SQL 应用程序

注：Java 嵌入式 SQL (SQLJ) 在本章中不作讨论。有关它的完整讨论，参阅第57页的『第4章 构建 Java Applet 和应用程序』。

DB2 AD 客户机包括嵌入 SQL 语句的样本程序。本书后面的“构建应用程序”几章说明如何使用与该平台的 DB2 AD 客户机一起提供的构建文件，为受支持的编译程序构建样本程序。还可使用提供的 makefile。makefile 和构建文件显示了可使

用的编译程序选项。在适当的章节中，为每个平台上受支持的编译程序都定义了这些选项。您可能需要针对自己的环境修改这些选项。

当运行构建文件来构建包含嵌入式 SQL 的样本程序时，该构建文件执行下列步骤：

- 连接数据库。
- 预编译源文件。
- 将联编文件与数据库联编。
- 与数据库断开。
- 编译并连接源文件。

本书中使用下面的样本程序来演示使用受支持的程序设计语言构建和运行嵌入式 SQL 应用程序的步骤。要执行的步骤可能随环境的不同而改变：

updat 使用静态 SQL 更新数据库。

下列样本用来演示嵌入式 SQL 客户机应用程序如何调用使用 C 和 C++ 编写的存储过程和用户定义函数 (UDF)：

spclient

是演示调用存储过程的客户机程序；服务器程序是 spserver。

spserver

是演示调用存储过程的服务器程序；客户机程序是 spclient。

udfcli 使用用户定义函数库 udfsrv 中的 ScalarUDF 函数。

下列样本用来演示使用 COBOL 编写的存储过程和用户定义函数 (UDF)：

outcli 是演示调用存储过程的客户机程序；服务器程序是 outsrv。

outsrv 是演示调用存储过程的服务器程序；客户机程序是 outcli。

calludf

调用用户定义函数库 udf 中的函数。

有关这些样本程序的详细说明，参阅第12页的『Sample Program Tables』。

受支持的这些样本程序的源文件，在 UNIX 上位于 sqllib/samples 下适当的程序设计语言子目录中；而在 OS/2 和 Windows 32 位操作系统上，位于 %DB2PATH%\samples 下适当的程序设计语言子目录中。

当构建样本程序后，可用它们作为模板来创建自己的应用程序。方法是用您自己的 SQL 语句来修改样本程序。可使用提供的 makefile 或构建文件来构建程序。

第11页的『样本程序』列出了所有的样本程序。 *Application Development Guide* 说明包含嵌入式 SQL 的样本是如何工作的。

存储过程

存储过程在服务器上构建，并存储在服务器上。客户机应用程序可远程存取它们。然后存储过程对服务器数据库执行本地处理，并将结果返回给客户机。这就减轻了网络通信量，同时提高了网络整体性能。

存储过程可在受保护的方式下运行，也可在不受保护的方式下运行。不受保护的存储过程在数据库管理程序所在的相同地址空间内运行，因而与受保护存储过程（在与数据库管理程序隔离的地址空间内运行）相比，性能有所提高。但是，使用不受保护的存储过程会带来危险，即用户代码可能破坏数据库控制结构。因此，仅当需要使性能最优时，才运行不受保护的存储过程。确保在将这些程序作为不受保护的程序运行前对它们进行全面的测试。有关详情，参考 *Application Development Guide*。

本书所演示的存储过程存储在服务器上的 `sqllib/function` 路径中。对于 `DB2DARI` 参数格式的存储过程（其中被调用过程与共享库名匹配），此位置指示存储过程是受保护的。如果希望此类型的存储过程不受保护，必须将它移到 `sqllib/function/unfenced` 目录中。对于所有其他类型的 `DB2` 存储过程，可在调用程序中用 `CREATE FUNCTION` 语句指示它是受保护的还是不受保护的。有关创建和使用不同类型的 `DB2` 存储过程的完整讨论，请参阅 *Application Development Guide* 中的“Stored Procedures”一章。

下列样本程序用来演示在服务器上使用 SQL 过程构建并存储一个存储过程库的步骤：

spserver.db2

是一个 CLP 脚本文件，它包含在服务器上创建共享库所用的 SQL 过程。此脚本文件可由 `CLP call` 命令或 `C`、`C++` 和 `CLI` 目录中的 `spclient` 应用程序调用。

下列样本程序用来演示在服务器上使用 `C` 和 `C++` 构建并存储一个存储过程库的步骤：

spserver

是客户机 / 服务器示例中的服务器程序；客户机程序是 `spclient`。

下列样本程序用来演示在服务器上使用 `Java` 构建并存储一个存储过程库的步骤：

Spserver

是客户机 / 服务器示例中的服务器程序；客户机程序是 `Spclient`。

下列样本程序用来演示在服务器上使用 COBOL 构建并存储一个存储过程库的步骤:

outsrv 是客户机 / 服务器示例中的服务器程序; 客户机程序是 **outcli**。

用户定义函数 (UDF)

用户定义函数允许您编写满足您自身需要的 SQL 扩充程序。与存储过程相似, 用户定义函数存储在服务器上, 可由客户机应用程序存取。UDF 不包含嵌入式 SQL 语句。

本书中使用下面的样本程序来演示在服务器上构建和存储 UDF 库的步骤:

udfsrv 创建用户定义函数 (UDF) 库。仅适用于 C 和 C++。客户机应用程序 **udfcli**调用这些函数。

udf 创建用户定义函数 (UDF) 库。仅适用于 COBOL。客户机应用程序 **calludf** 调用这些函数。

UDFsrv 创建用户定义函数 (UDF) 库。仅适用于 Java。**UDFcli** 和 **UDFclie** 分别是调用这些函数的 JDBC 和 SQLJ 客户机应用程序。

多线程应用程序

DB2 在受支持的 UNIX 平台上支持 C 和 C++ 多线程应用程序。这些应用程序允许用户有几个同步进程同时执行几个线程。这样不必使用轮询方案, 就可处理异步事件并创建事件驱动的应用程序。下列样本程序用来演示构建 DB2 多线程应用程序:

thdsrver

演示线程创建和管理。

用 C++ 编写 UDF 和存储过程的考虑事项

在 C++ 中函数名可“过载”。具有相同名称的两个函数如果有不同的自变量, 则可以同时存在, 如下所示:

```
int func( int i )
```

和

```
int func( char c )
```

缺省情况下，C++ 编译程序“用类型修饰”函数名或“提取”函数名。这表示会将自变量类型名附加到它们的函数名后以分辨之，例如表示上述两个示例的 `func_Fi` 和 `func_Fc`。提取的名称在每个平台上都不同，因此显式使用提取的名称的代码是不可移植的。

在 OS/2 和 Windows 32 位操作系统上，可根据 `.obj`（目标）文件来确定类型修饰的函数名。

在 OS/2 和 Windows 上，如果使用 VisualAge C++ 编译程序，可使用 `cppfilt` 命令，根据 `.obj`（目标）文件来确定类型修饰的函数名，如下所示：

```
cppfilt -b /p myprog.obj
```

其中，`myprog.obj` 是您的程序目标文件。

在 Windows 上，如果使用 Microsoft Visual C++ 编译程序，可使用 `dumpbin` 命令，根据 `.obj`（目标）文件来确定类型修饰的函数名，如下所示：

```
dumpbin /symbols myprog.obj
```

其中，`myprog.obj` 是您的程序目标文件。

在 UNIX 平台上，可使用 `nm` 命令，根据 `.o`（目标）文件或共享库来确定类型修饰的函数名。此命令可产生许多输出，因此建议通过 `grep` 用管道输送输出以便查找正确的行，如下所示：

```
nm myprog.o | grep myfunc
```

其中，`myprog.o` 是您的程序目标文件，而 `myfunc` 是程序源文件中的函数。

所有这些命令产生的输出都包括带提取的函数名的一行。例如，在 UNIX 上，此行类似于如下所示：

```
myfunc__FP1T1PsT3PcN35|      3792|unamex|      | ...
```

一旦从以上一个命令中获得提取的函数名，就可在适当的命令中使用它。下面演示如何使用从上述 UNIX 示例获得的提取函数名。在 OS/2 或 Windows 上获得的提取函数名的用法相同。

当用 `CREATE FUNCTION` 注册一个 UDF 时，`EXTERNAL NAME` 子句必须指定提取的函数名：

```
CREATE FUNCTION myfunco(...) RETURNS...
...
EXTERNAL NAME '/whatever/path/myprog!myfunc__FP1T1PsT3PcN35'
...
```

类似地，当调用存储过程时，`CALL` 函数也必须指定提取的函数名：

```
CALL 'myprog!myfunc__FP1T1PsT3PcN35' ( ... )
```

如果存储过程或 UDF 库不包含过载使用的 C++ 函数名，则可选择使用 `extern "C"` 来强制编译程序不用类型修饰函数名。（注意：您始终可在 UDF 中过载使用 SQL 函数名，因为 DB2 会根据库函数的名称和它所带的参数来分辨要调用的库函数）。

```
#include <string.h>
#include <stdlib.h>
#include "sqludf.h"

/*-----*/
/* function fold: output = input string is folded at point indicated */
/*                               by the second argument.                */
/*      inputs: CLOB,             input string                          */
/*              LONG              position to fold on                    */
/*      output: CLOB              folded string                          */
/*-----*/
extern "C" void fold(
    SQLUDF_CLOB      *in1,                /* input CLOB to fold      */
    ...
    ...
)
/* end of UDF: fold */

/*-----*/
/* function find_vowel:          */
/*      returns the position of the first vowel.                        */
/*      returns error if no vowel.                                      */
/*      defined as NOT NULL CALL                                        */
/*      inputs: VARCHAR(500)                                           */
/*      output: INTEGER                                                 */
/*-----*/
extern "C" void findvwl(
    SQLUDF_VARCHAR   *in,                /* input smallint         */
    ...
    ...
)
/* end of UDF: findvwl */
```

在此示例中，用户定义函数 (UDF) `fold` 和 `findvwl` 不是由编译程序用类型修饰的函数，因此应使用它们的普通名称将它们注册在 `CREATE FUNCTION` 语句中。类似地，如果编写 C++ 存储过程时使用了 `extern "C"`，则将在 `CALL` 语句中使用该存储过程的未加修饰的函数名。

第4章 构建 Java Applet 和应用程序

设置环境	58	应用程序	69
AIX	58	存储过程的客户机应用程序	69
HP-UX	59	用户定义函数的客户机应用程序	70
Linux	60	存储过程	70
OS/2	61	SQLJ 程序	71
PTX	62	Applet	74
Silicon Graphics IRIX	63	应用程序	75
Solaris	65	存储过程的客户机程序	75
Windows 32 位操作系统	66	用户定义函数的客户机程序	76
Java 样本程序	68	存储过程	76
JDBC 程序	68	用户定义函数 (UDF)	79
Applet	68	DB2 Java Applet 的一般要点	80

可以用适用于您平台的 Java Development Kit (JDK) 来开发存取 DB2 数据库的 Java 程序。该 JDK 包括“Java 数据库链接”(JDBC), Java 的一个动态 SQL API。

DB2 JDBC 支持作为 DB2 客户机和服务器上的 Java Enablement 选项的一部分提供。利用此支持,可构建和运行 JDBC 应用程序和 Applet。它们只包括动态 SQL,并使用 Java 调用接口将 SQL 语句传送至 DB2。

DB2 Java 嵌入式 SQL (SQLJ) 支持作为 DB2 AD 客户机的一部分提供。利用 DB2 SQLJ 支持和 DB2 JDBC 支持,可以构建和运行 SQLJ Applet 和应用程序。它们包含静态 SQL,且使用与 DB2 数据库联编的嵌入式 SQL 语句。

DB2 AD 客户机提供的 SQLJ 支持包括:

- DB2 SQLJ 译码器 `sqlj`, 它使用 Java 源语句置换 SQLJ 程序中的嵌入式 SQL 语句,并生成一个串行化简要表,该简要表包含有关在 SQLJ 程序中找到的 SQL 操作的信息。
- DB2 SQLJ 简要表定制器 `db2profc`, 它预编译存储在串行化简要表中的 SQL 语句,将它们定制成运行期函数调用,并在该 DB2 数据库中生成一个程序包。有关 `db2profc` 命令的更多信息,参阅 *Command Reference*。
- DB2 SQLJ 简要表打印机 `db2profp`, 它以纯文本格式打印简要表的 DB2 定制版本的内容。

要运行 DB2 Java 应用程序,必须安装并调用提供本机线程支持的 Java 虚拟机 (JVM)。要使用本机线程执行 Java 应用程序,可在命令中使用 `-native` 选项。例如,要运行 Java 样本应用程序 `App.class`,可使用以下命令:

```
java -native App
```

您可能可以将本机线程指定为某些 Java 虚拟机的缺省线程支持。本章信息假设本机线程支持为缺省值。有关使本机线程成为系统缺省线程的说明，请参考 JVM 文档。

要运行 DB2 Java Applet，可调用提供本机线程或绿线程支持的 Java 虚拟机。

有关使用 Java 进行 DB2 编程的详情，参考 *Application Development Guide* 中“Programming in Java”一章。

有关最新的、更新过的 DB2 Java 信息，访问下列位置的 Web 页面：

<http://www.ibm.com/software/data/db2/java>

设置环境

AIX

要在 AIX 上使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. 下列情况之一：
 - IBM 公司开发的 Java Development Kit (JDK) 版本 1.1.8 AIX 版和 Java Runtime Environment (JRE) 版本 1.1.8 AIX 版。它们与 DB2 一起安装。（参考 <http://www.ibm.com/software/data/db2/java>。）
 - IBM 公司开发的 Java Development Kit (JDK) 版本 1.2.2 AIX 版和 Java Runtime Environment (JRE) 版本 1.2.2 AIX 版。（参考 <http://www.ibm.com/software/data/db2/java>。）
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 AIX 版”的客户机和服务器上提供。

JDBC 1.22 驱动程序仍是所有操作系统（包括 AIX 在内）上的缺省驱动程序。要利用 JDBC 2.0 的新功能部件，必须安装适用于您的平台的 JDBC 2.0 驱动程序和 JDK 1.2 支持。要安装 JDBC 2.0 驱动程序 AIX 版，在 `sqllib/java12` 目录下输入 `usejdbc2` 命令。此命令执行下列任务：

- 为 1.22 驱动程序文件创建一个 `sqllib/java11` 目录
- 将 JDBC 1.22 驱动程序文件备份到 `sqllib/java11` 目录中
- 将 JDBC 2.0 驱动程序文件从 `sqllib/java12` 目录中复制到恰当的目录中

要切换回 JDBC 1.22 驱动程序，在 `sqllib/java12` 目录下执行 `usejdbc1` 命令。

要运行 DB2 Java 存储过程或 UDF，还需要更新服务器上的 DB2 数据库管理程序配置，以包括在该机器上安装 JDK 的路径。为此，可在服务器命令行上输入如下命令：

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk11
```

其中 /home/db2inst/jdk11 是 JDK 的安装路径。

可在服务器上输入如下命令，检查 DB2 数据库管理程序配置来验证 JDK11_PATH 字段的值是否正确：

```
db2 get dbm cfg
```

为便于查看，您可能希望将输出重定向至文件。JDK11_PATH 字段在靠近输出开始处出现。有关这些命令的更多信息，参阅 *Command Reference*。

要在 AIX 上使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序，应将更新 AIX Java 环境的命令加入数据库管理程序文件 db2profile 和 db2cshrc 中。当创建 DB2 实例时，修改 .profile 和 / 或 .cshrc 以使 CLASSPATH 包括：

- "."（当前目录）
- 文件 sqllib/java/db2java.zip

要构建 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
sqllib/java/sqlj.zip
```

要运行 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
sqllib/java/runtime.zip
```

HP-UX

要在 HP-UX 上使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. Hewlett-Packard 公司开发的 HP-UX Developer's Kit for Java 发行版 1.1.8 或更高版本（参考 <http://www.ibm.com/software/data/db2/java>）。
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 HP-UX 版”的客户机和服务器上提供。

要运行 DB2 Java 存储过程或 UDF，还需要更新服务器上的 DB2 数据库管理程序配置，以包括在该机器上安装 JDK 的路径。为此，可在服务器命令行上输入如下命令：

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk11
```

其中 `/home/db2inst/jdk11` 是安装 JDK 的路径。

可在服务器上输入如下命令，检查 DB2 数据库管理程序配置来验证 `JDK11_PATH` 字段的值是否正确：

```
db2 get dbm cfg
```

为便于查看，您可能希望将输出重定向至文件。`JDK11_PATH` 字段在靠近输出开始处出现。有关这些命令的更多信息，参阅 *Command Reference*。

要在 HP-UX 上使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序，应将更新 HP-UX Java 环境的命令加入数据库管理程序文件 `db2profile` 和 `db2cshrc` 中。当创建 DB2 实例时，修改 `.profile` 和 / 或 `.cshrc` 以便：

1. 将 `THREADS_FLAG` 设置为 "native"。
2. `CLASSPATH` 包括：
 - "."（当前目录）
 - 文件 `sqllib/java/db2java.zip`

要构建 SQLJ 程序，还须更新 `CLASSPATH` 以包括文件：

```
sqllib/java/sqlj.zip
```

要运行 SQLJ 程序，还须更新 `CLASSPATH` 以包括文件：

```
sqllib/java/runtime.zip
```

Linux

要在 Linux 上使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. IBM Developer Kit Linux 版和 Runtime Environment Linux 版的版本 1.1.8。
(参考 <http://www.ibm.com/software/data/db2/java>)。
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 Linux 版”的客户机和服务器上提供。

要运行 DB2 Java 存储过程或 UDF，还需要更新服务器上的 DB2 数据库管理程序配置，以包括在该机器上安装 JDK 的路径。为此，可在服务器命令行上输入如下命令：

```
db2 update dbm cfg using JDK11_PATH /usr/local/jdk118
```

其中 `/usr/local/jdk118` 是 JDK 的安装路径。

可在服务器上输入如下命令，检查 DB2 数据库管理程序配置来验证 JDK11_PATH 字段的值是否正确：

```
db2 get dbm cfg
```

为便于查看，您可能希望将输出重定向至文件。JDK11_PATH 字段在靠近输出开始处出现。有关这些命令的更多信息，参阅 *Command Reference*。

注：在 Linux 上，在 "setuid" 环境中运行的程序内，“Java 虚拟机”实施得并不好。包含 Java 解释程序的共享库 libjava.so 可能会无法装入。一个解决方法是，使用类似如下所示的命令（根据您的机器上 Java 的安装位置），为 /usr/lib 中的 JVM 共享库创建符号链路：

```
ln -s /usr/local/jdk118/lib/linux/native_threads/libjava.so /usr/lib
```

有关此解决方法以及其他可用的解决方法的更多信息，请访问：

<http://www.ibm.com/software/data/db2/java/faq.html>

要在 Linux 上使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序，应将更新 Linux Java 环境所需的命令加入数据库管理程序文件 db2profile 和 db2cshrc 中。当创建 DB2 实例时，修改 .bashrc、.profile 和 / 或 .cshrc 以便：

1. 将 THREADS_FLAG 设置为 "native"。
2. CLASSPATH 包括：
 - "."（当前目录）
 - 文件 sqllib/java/db2java.zip

要构建 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
sqllib/java/sqlj.zip
```

要运行 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
sqllib/java/runtime.zip
```

OS/2

要在 OS/2 上使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. IBM 公司开发的 Java Development Kit (JDK) 版本 1.1.8 OS/2 版和 Java Runtime Environment (JRE) 版本 1.1.8 OS/2 版。它们与 DB2 一起安装。（参考 <http://www.ibm.com/software/data/db2/java/>。）
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 OS/2 版”的客户机和服务器上提供。

必须在 HPFS 驱动器中安装 JDK，以允许扩展名超过三个字符的长文件名，如 .java。Java 工作目录也必须在 HPFS 驱动器上。如果要在 OS/2 服务器上运行 Java 存储过程或 UDF，必须在该服务器上的 HPFS 驱动器中安装 DB2，以便可将存储过程或 UDF 的 .class 文件放入 %DB2PATH%\function 目录中。

要运行 DB2 Java 存储过程或 UDF，还需要更新服务器上的 DB2 数据库管理程序配置，以包括在该机器上安装 JDK 的路径。为此，可在服务器命令行上输入如下命令：

```
db2 update dbm cfg using JDK11_PATH c:\jdk11
```

其中 c:\jdk11 是安装 JDK 的路径。

可在服务器上输入如下命令，检查 DB2 数据库管理程序配置来验证 JDK11_PATH 字段的值是否正确：

```
db2 get dbm cfg
```

为便于查看，您可能希望将输出重定向至文件。JDK11_PATH 字段在靠近输出开始处出现。有关这些命令的更多信息，参阅 *Command Reference*。

要在 OS/2 上使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序，当安装 DB2 时应自动更新 CLASSPATH 环境变量，以包括：

- "."（当前目录）
- 文件 %DB2PATH%\java\db2java.zip

要构建 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
%DB2PATH%\java\sqlj.zip
```

要运行 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
%DB2PATH%\java\runtime.zip
```

PTX

要在 PTX 上使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. ptx/JSE 版本 3.0，相当于 Sun Microsystem 公司开发的 JDK 1.2（参考 <http://www.ibm.com/software/data/db2/java>）。
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 NUMA-Q 版”的客户端和服务器的提供。

要运行 DB2 Java 存储过程或 UDF，还需要更新服务器上的 DB2 数据库管理程序配置，以包括在该机器上安装 ptx/JSE 的路径。为此，可在服务器命令行上输入如下命令：

```
db2 update dbm cfg using JDK11_PATH /opt/jse3.0
```

其中 /opt/jse3.0 是 ptx/JSE 安装路径。

可在服务器上输入如下命令，检查 DB2 数据库管理程序配置来验证 JDK11_PATH 字段的值是否正确：

```
db2 get dbm cfg
```

为便于查看，您可能希望将输出重定向至文件。JDK11_PATH 字段在靠近输出开始处出现。有关这些命令的更多信息，参阅 *Command Reference*。

要在 PTX 上使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序，应将更新 PTX Java 环境所需的命令加入数据库管理程序文件 db2profile 和 db2cshrc 中。当创建 DB2 实例时，修改 .profile 和 / 或 .cshrc，以便 CLASSPATH 包括：

- "."（当前目录）
- 文件 sqllib/java/db2java.zip

要构建 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
sqllib/java/sqlj.zip
```

要运行 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

```
sqllib/java/runtime.zip
```

Silicon Graphics IRIX

要在 Silicon Graphics IRIX 上使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. Silicon Graphics 公司开发的 Java2 Software Development Kit 版本 1.2.1 (JDK 1.2.1)（参考 <http://www.ibm.com/software/data/db2/java>）。
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 Silicon Graphics IRIX 版”的客户机上提供。

注：在 Silicon Graphics IRIX 上的 SQLJ 应用程序只能使用 o32 对象类型构建。要将 Java 缺省对象类型更改为 o32，如果使用 Korn shell，使用以下命令设置 SGI_ABI 环境变量：

```
export SGI_ABI=-o32
```

如果使用 C shell，使用以下命令：

```
setenv SGI_ABI -o32
```

DB2 Silicon Graphics IRIX 版只用于客户机。要运行 DB2 应用程序和 Applet 并构建嵌入 SQL 的 DB2 应用程序和 Applet, 则需要从客户机存取服务器上的 DB2 数据库。该服务器可能正运行着不同的操作系统。有关配置客户机至服务器通信的信息, 参阅 *DB2 (UNIX 版) 快速入门*。

另外, 因为您将从在不同操作系统上运行的远程客户机存取服务器上的数据库, 因此您需要将数据库实用程序 (包括 DB2 CLI) 与该数据库联编。有关详情, 参阅第40页的『联编』。

要运行 DB2 Java 存储过程或 UDF, 还需要更新服务器上的 DB2 数据库管理程序配置, 以包括在该机器上安装 JDK 的路径。为此, 可在服务器命令行上输入如下命令:

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk11
```

其中 /home/db2inst/jdk11 是安装 JDK 的路径。

可在服务器上输入如下命令, 检查 DB2 数据库管理程序配置来验证 JDK11_PATH 字段的值是否正确:

```
db2 get dbm cfg
```

为便于查看, 您可能希望将输出重定向至文件。JDK11_PATH 字段在靠近输出开始处出现。有关这些命令的更多信息, 参阅 *Command Reference*。

要在 Silicon Graphics IRIX 上使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序, 应将更新 Silicon Graphics IRIX Java 环境所需的命令加入数据库管理程序文件 db2profile 和 db2cshrc 中。当创建 DB2 实例时, 修改 .profile 和 / 或 .cshrc, 以便 CLASSPATH 包括:

- "." (当前目录)
- 文件 sqlllib/java/db2java.zip

要构建 SQLJ 程序, 还须更新 CLASSPATH 以包括文件:

```
sqlllib/java/sqlj.zip
```

要运行 SQLJ 程序, 还须更新 CLASSPATH 以包括文件:

```
sqlllib/java/runtime.zip
```

注: 在 Silicon Graphics IRIX 上, 连接上下文 close() 方法可能导致一个陷阱。解决方法是, 在无用存储单元收集期间使连接上下文自动关闭。

Solaris

要在 Solaris 操作环境中使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. Sun Microsystems 公司开发的 Java Development Kit (JDK) 版本 1.1.8 或 1.2 Solaris 版（参考 <http://www.ibm.com/software/data/db2/java>）。
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 Solaris 版”的客户机和服务器上提供。

JDBC 1.22 驱动程序仍是所有操作系统（包括 Solaris 在内）上的缺省驱动程序。要利用 JDBC 2.0 的新功能部件，必须安装适用于您的平台的 JDBC 2.0 驱动程序和 JDK 1.2 支持。要安装 JDBC 2.0 驱动程序 Solaris 版，从 `sql1lib/java12` 目录下输入 `usejdbc2` 命令。此命令执行下列任务：

- 为 1.22 驱动程序文件创建一个 `sql1lib/java11` 目录
- 将 JDBC 1.22 驱动程序文件备份到 `sql1lib/java11` 目录中
- 将 JDBC 2.0 驱动程序文件从 `sql1lib/java12` 目录中复制到恰当的目录中

要切换回 JDBC 1.22 驱动程序，在 `sql1lib/java12` 目录下执行 `usejdbc1` 命令。

要运行 DB2 Java 存储过程或 UDF，还需要更新服务器上的 DB2 数据库管理程序配置，以包括在该机器上安装 JDK 的路径。为此，可在服务器命令行上输入如下命令：

```
db2 update dbm cfg using JDK11_PATH /usr/java
```

其中 `/usr/java` 是 JDK 的安装路径。

可在服务器上输入如下命令，检查 DB2 数据库管理程序配置来验证 `JDK11_PATH` 字段的值是否正确：

```
db2 get dbm cfg
```

为便于查看，您可能希望将输出重定向至文件。`JDK11_PATH` 字段在靠近输出开始处出现。有关这些命令的更多信息，参阅 *Command Reference*。

注：在 Solaris 操作环境中，某些 Java 虚拟机实现在“setuid”环境中运行的程序内不能正常工作。包含 Java 解释程序的共享库 `libjava.so` 可能会无法装入。一个解决方法是，使用类似如下所示的命令（并根据您的机器上 Java 所安装的位置），为 `/usr/lib` 中所有需要的 JVM 共享库创建符号链路：

```
ln -s /usr/java/lib/*.so /usr/lib
```

有关此解决方法以及其他可用的解决方法的更多信息，请访问：

<http://www.ibm.com/software/data/db2/java/faq.html>

要在 Solaris 操作环境中使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序，应将更新 Solaris Java 环境所需的命令加入数据库管理程序文件 db2profile 和 db2cshrc 中。当创建 DB2 实例时，修改 .profile 和 / 或 .cshrc 以便：

1. 将 THREADS_FLAG 设置为 "native"。
2. CLASSPATH 包括：
 - "."（当前目录）
 - 文件 sqllib/java/db2java.zip

要构建 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

sqllib/java/sqlj.zip

要运行 SQLJ 程序，还须更新 CLASSPATH 以包括文件：

sqllib/java/runtime.zip

Windows 32 位操作系统

要在 Windows 32 位操作系统上使用 DB2 JDBC 支持构建 Java 应用程序，需要在用于开发的机器上安装并配置下列各项：

1. 下列情况之一：
 - IBM 公司开发的 Java Development Kit (JDK) 1.1.8 Win32 版和 Java Runtime Environment (JRE) 1.1.8 Win32 版。它们与 DB2 一起安装。（参考 <http://www.ibm.com/software/data/db2/java>。）
 - Sun Microsystems 公司开发的 Java Development Kit (JDK) 版本 1.2 Win32 版（参考 <http://www.ibm.com/software/data/db2/java>）。
 - Microsoft Software Developer's Kit for Java 版本 3.1（参考 <http://www.ibm.com/software/data/db2/java>）。
2. DB2 Java Enablement，在“DB2 通用数据库版本 7.1 Windows 32 位操作系统版”的客户机和服务器上提供。

JDBC 1.22 驱动程序仍是所有操作系统（包括 Windows 32 位操作系统在内）上的缺省驱动程序。要利用 JDBC 2.0 的新功能部件，必须安装适用于您的平台的 JDBC 2.0 驱动程序和 JDK 1.2 支持。要安装 JDBC 2.0 驱动程序 Windows 32 位操作系统版，从 sqllib/java12 目录下输入 usejdbc2 命令。此命令执行下列任务：

- 为 1.22 驱动程序文件创建一个 sqllib/java11 目录
- 将 JDBC 1.22 驱动程序文件备份到 sqllib/java11 目录中
- 将 JDBC 2.0 驱动程序文件从 sqllib/java12 目录中复制到恰当的目录中

要切换回 JDBC 1.22 驱动程序，在 `sqllib/java12` 目录下执行 `usejdbc1` 批处理文件。

要运行 DB2 Java 存储过程或 UDF，还需要更新服务器上的 DB2 数据库管理程序配置，以包括在该机器上安装 JDK 的路径。为此，可在服务器命令行上输入如下命令：

```
db2 update dbm cfg using JDK11_PATH c:\jdk11
```

其中 `c:\jdk11` 是安装 JDK 的路径。

注：如果 JDK 的安装路径包含带一个或多个空格的目录名，应用单引号将路径引起来。例如：

```
db2 update dbm cfg using JDK11_PATH 'c:\Program Files\jdk11'
```

可在服务器上输入如下命令，检查 DB2 数据库管理程序配置来验证 `JDK11_PATH` 字段的值是否正确：

```
db2 get dbm cfg
```

为便于查看，您可能希望将输出重定向至文件。`JDK11_PATH` 字段在靠近输出开始处出现。有关这些命令的更多信息，参阅 *Command Reference*。

要在受支持的 Windows 平台上使用 DB2 JDBC 支持运行 JDBC 和 SQLJ 程序，当安装 DB2 时应自动更新 `CLASSPATH`，以包括：

- `."`（当前目录）
- 文件 `%DB2PATH%\java\db2java.zip`

要构建 SQLJ 程序，还须更新 `CLASSPATH` 以包括文件：

```
%DB2PATH%\java\sqlj.zip
```

要运行 SQLJ 程序，还须更新 `CLASSPATH` 以包括文件：

```
%DB2PATH%\java\runtime.zip
```

为了指定用于 DB2 SQLJ 的 Java Development Kit，DB2 会在 Windows 32 位操作系统上安装环境变量 `DB2JVVIEW`。该环境变量适用于所有 DB2 SQLJ 命令（`db2prof`、`db2profp`、`profdb`、`profp` 和 `sqlj`）。

如果使用缺省设置 `"DB2JVVIEW=0"`，或未设置 `DB2JVVIEW`，将使用 Sun JDK；即，如果调用 `"profp"`，它将作为 `"java sqlj.runtime.profile.util.ProfilePrinter"` 运行。如果 `"DB2JVVIEW=1"`，将使用 Microsoft SDK for Java；即，如果调用 `"profp"`，它将作为 `"jview sqlj.runtime.profile.util.ProfilePrinter"` 运行。

Java 样本程序

DB2 提供了样本程序，在下面几节中会使用它们，这些样本程序演示如何构建并运行专门使用动态 SQL 的 JDBC 程序和使用静态 SQL 的 SQLJ 程序。在 UNIX 平台上，这些 Java 样本位于 `sqllib/samples/java` 中。在 OS/2 和 Windows 32 位操作系统上，这些样本位于 `%DB2PATH%\samples\java` 中。`samples` 目录还包含 `README`、`makefile` 和构建文件。

Java `makefile` 构建所有提供的样本程序。它需要一个兼容的 `make` 可执行程序，而通常 Java Development Kit 不提供这个程序。参阅 `makefile` 文本开始处的注解以获得更多信息。某些 Java `makefile` 命令与其他语言不同：`make clean` 命令除去由 Java 编译程序生成的所有文件，如 `.class` 文件。`make cleanall` 命令除去这些文件以及 SQLJ 译码器生成的任何文件。

在命令行上构建 JDBC 程序相对简单，因此没有包括它们的构建文件。提供了两个 SQLJ 构建文件：`bldsqlj` 构建 SQLJ Applet 和应用程序；`bldsqljs` 构建 SQLJ 存储过程。在第71页的『SQLJ 程序』一节中演示了这些构建文件。

OS/2

在 OS/2 上，工作目录必须在一个 HPFS 驱动器上。因为在 OS/2 上提供的 DB2 样本程序要与 FAT 驱动器兼容，因此文件名最多有三个字符的扩展名。为了符合此限制，Java 样本文件扩展名已被截断。将 Java 文件复制到您的工作目录之后，可使用下列命令重命名这些文件：

```
move *.jav *.java
move *.htm *.html
move *.sql *.sqlj
```

JDBC 程序

Applet

`DB2App1t` 演示一个动态 SQL Java Applet，它使用 JDBC Applet（或“net”）驱动程序存取 DB2 数据库。

要通过在命令行输入命令来构建并运行此 Applet：

1. 确保在您的 DB2 机器（服务器或客户机）上安装了 web 服务器，且它正在运行。
2. 根据文件中的说明修改 `DB2App1t.html` 文件。

3. 在 DB2Appl1.html 中指定的 TCP/IP 端口上启动 JDBC Applet 服务器，例如，如果在 DB2Appl1.html 中指定了 param name=port value='6789'，应输入：

```
db2jstrt 6789
```
4. 使用如下命令编译 DB2Appl1.java 以生成文件 DB2Appl1.class：

```
javac DB2Appl1.java
```
5. 确保您的 web 浏览器可存取工作目录。否则，将 DB2Appl1.class 和 DB2Appl1.html 复制到可存取的目录。
6. 将 OS/2 或 Windows 32 位操作系统上的文件 %DB2PATH%\java\db2java.zip，或 UNIX 上的文件 sqllib/java/db2java.zip，复制到 DB2Appl1.class 和 DB2Appl1.html 所在的目录中。
7. 在您的客户机上，启动 web 浏览器（它必须支持 JDK 1.1）并装入 DB2Appl1.html。

步骤 (1)、(5) 和 (7) 的替代方法是，可在客户机的工作目录中输入如下命令，以使用 Java Development Kit 所带的 Applet 浏览器：

```
appletviewer DB2Appl1.html
```

也可使用 Java makefile 构建此程序。

应用程序

DB2Appl 演示一个动态 SQL Java 应用程序，它使用 JDBC 应用程序（或 "app"）驱动程序存取 DB2 数据库。

要通过在命令行输入命令来构建并运行此应用程序：

1. 使用如下命令编译 DB2Appl.java 来生成文件 DB2Appl.class：

```
javac DB2Appl.java
```

2. 使用如下命令，对该应用程序运行 java 解释程序：

```
java DB2Appl
```

也可使用 Java makefile 构建此程序。

存储过程的客户机应用程序

Spclient 是一个客户机应用程序，它使用 JDBC 应用程序的驱动程序调用 Java 存储过程类 Spserver。在构建并运行此客户机应用程序之前，在服务器上构建该存储过程类。参阅第70页的『存储过程』。

要通过在命令行输入命令来构建并运行此客户机程序：

1. 使用以下命令编译 Spclient.java 来生成文件 Spclient.class：

```
javac Spclient.java
```

2. 使用以下命令对客户机程序运行 Java 解释程序:

```
java Spclient
```

也可使用 Java makefile 构建此程序。

用户定义函数的客户机应用程序

UDFcli 是一个客户机程序，它使用 JDBC 应用程序的驱动程序来调用在用户定义函数服务器程序 UDFsrv 中实现的用户定义函数。在构建并运行此客户机应用程序之前，在服务器上构建用户定义函数程序 UDFsrv。参阅第79页的『用户定义函数 (UDF)』。

要通过在命令行输入命令来构建并运行此客户机程序:

1. 使用如下命令编译 UDFcli.java 来生成文件 UDFcli.class:

```
javac UDFcli.java
```

2. 使用以下命令对客户机程序运行 Java 解释程序:

```
java UDFcli
```

也可使用 Java makefile 构建此程序。

存储过程

Spserver 演示使用 JDBC 应用程序驱动程序的动态 SQL PARAMETER STYLE JAVA 存储过程。存储过程在服务器上编译并存储。当客户机应用程序调用存储过程时，它们要存取服务器数据库并将信息返回至客户机应用程序。

要通过在命令行输入命令来构建并运行此服务器程序:

1. 使用以下命令编译 Spserver.java 来生成文件 Spserver.class:

```
javac Spserver.java
```

2. 在 OS/2 和 Windows 32 位操作系统上，将 Spserver.class 复制到 %DB2PATH%\function 目录中；在 UNIX 上，复制到 sqllib/function 目录中。
3. 接着在服务器上运行 Spcreate.db2 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已编目，可使用以下命令卸下它们:

```
db2 -td@ -vf Spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf Spcreate.db2
```

4. 最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。
5. 编译并运行 Spclient 客户机应用程序, 以存取存储过程类。参阅第69页的『存储过程的客户机应用程序』。

也可使用 Java makefile 构建此程序。

SQLJ 程序

注: 要使用 IBM Java Development Kit UNIX 版、OS/2 版以及 Windows 32 位操作系统版构建并运行 SQLJ 程序, 必须根据操作系统使用以下命令关闭 JDK 的即时编译程序:

在 **OS/2** 和 **Windows** 上:

```
SET JAVA_COMPILER=NONE
```

在 **UNIX:**

```
export JAVA_COMPILER=NONE
```

构建文件 bldsqlj 包含构建 SQLJ Applet 或应用程序所需的命令。在 UNIX 上它是一个脚本文件。在 OS/2 上, 它是命令文件 bldsqlj.cmd, 而在 Windows 上它是批处理文件 bldsqlj.bat。命令和批处理文件的内容是相同的, 此版本先提供, 然后才提供 UNIX 脚本文件。后面的 Applet 和应用程序构建部分将说明这些构建文件。

注: 与 DB2 一起交付的 SQLJ 译码器将已转换的 .java 文件编译为 .class 文件。因此, 本节中的构建文件不使用 Java 编译程序。

在以下用于 OS/2 和 Windows 32 位操作系统的构建文件中, 第一个参数 %1 指定源文件的名称。第二个参数 %2 指定要连接的数据库的名称。第三个参数 %3 指定数据库的用户 ID, 而第四个参数 %4 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名, 则该程序使用缺省的 sample 数据库。

```
@echo off
rem bldsqlj -- OS/2 and Windows 32-bit operating systems
rem Builds a Java embedded SQL (SQLJ) program.
rem Usage: bldsqlj prog_name [ db_name [ userid password ] ]

if "%1" == "" goto error

rem Translate and compile the SQLJ source file
```

```

rem and bind the package to the database.
if "%2" == "" goto case1
if "%3" == "" goto case2
if "%4" == "" goto error
goto case3
:case1
sqlj %1.sqlj
  db2profrc -url=jdbc:db2:sample -preoptions="package using %1" %1_SJProfile0
  goto continue
:case2
  sqlj      -url=jdbc:db2:%2 %1.sqlj
  db2profrc -url=jdbc:db2:%2 -preoptions="package using %1" %1_SJProfile0
  goto continue
:case3
  sqlj      -url=jdbc:db2:%2 -user=%3 -password=%4 %1.sqlj
  db2profrc -url=jdbc:db2:%2 -user=%3 -password=%4 -preoptions="package using %1"
    %1_SJProfile0
  goto continue
:continue

goto exit

:error
echo Usage: bldsqlj prog_name [ db_name [ userid password ]]

:exit

@echo on

```


bldsqlj 的译码器和预编译选项

sqlj	SQLJ 译码器 (也编译该程序)。
%1.sqlj	SQLJ 源文件。
%1.java	从 SQLJ 源文件转换的 Java 文件。
db2profrc	DB2 Java 版简要表定制器。
-url	指定用于建立数据库连接的 JDBC URL, 如 jdbc:db2:sample。
-user	指定用户 ID (可选参数)。
-password	指定口令 (可选参数)。
-preoptions	用字符串 "package using %1" 指定数据库的程序包名, 其中 %1 是 SQLJ 源文件名。
%1_SJProfile0	指定该程序的串行化简要表。

在以下的 UNIX 脚本文件中, 第一个参数 \$1 指定源文件的名称。第二个参数 \$2 指定要连接的数据库名称。第三个参数 \$3 指定数据库的用户 ID, 而第四个参数 \$4 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名, 则该程序使用缺省的 sample 数据库。

```
#!/bin/ksh
# bldsqlj script file -- UNIX platforms
# Builds a Java embedded SQL (SQLJ) sample
# Usage: bldsqlj <prog_name> [ <db_name> [ <userid> <password> ]]

# Translate and compile the SQLJ source file
# and bind the package to the database.
if (($# < 2))
then
    sqlj $1.sqlj
    db2profrc -url=jdbc:db2:sample -preoptions="package using $1" $1_SJProfile0
elif (($# < 3))
then
    sqlj -url=jdbc:db2:$2 $1.sqlj
    db2profrc -url=jdbc:db2:$2 -preoptions="package using $1" $1_SJProfile0
else
    sqlj -url=jdbc:db2:$2 -user=$3 -password=$4 $1.sqlj
    db2profrc -url=jdbc:db2:$2 -user=$3 -password=$4 -preoptions="package using $1"
        $1_SJProfile0
fi
```

bldsqlj 的译码器和预编译选项

sqlj	SQLJ 译码器（也编译该程序）。
\$1.sqlj	SQLJ 源文件。
\$1.java	从 SQLJ 源文件转换的 Java 文件。
db2profrc	DB2 Java 版简要表定制器。
-url	指定用于建立数据库连接的 JDBC URL，如 jdbc:db2:sample。
-user	指定用户 ID（可选参数）。
-password	指定口令（可选参数）。
-preoptions	用字符串 "package using %1" 指定数据库的程序包名，其中 %1 是 SQLJ 源文件名。
\$1_SJProfile0	指定该程序的串行化简要表。

Applet

Applet 演示存取 DB2 数据库的 SQLJ Applet。

要使用构建文件 bldsqlj 构建此 Applet，然后运行它：

1. 确保在您的 DB2 机器（服务器或客户机）上安装了 web 服务器，且它正在运行。
2. 根据文件中的指示修改 Applet.html 文件。
3. 在 Applet.html 中指定的 TCP/IP 端口上启动 JDBC Applet 服务器。例如，如果在 Applet.html 中指定了 param name=port value='6789'，应输入：

```
db2jstrt 6789
```

4. 使用如下命令构建该 Applet:

```
bldsqlj Applet [ <db_name> [ <userid> <password> ] ]
```

其中，可选参数 <db_name> 允许您存取另一个数据库而不是缺省数据库 sample。如果要存取的数据库在另一个实例上，比如从远程客户机存取服务器，则需要可选参数 <userid> 和 <password>。

5. 确保您的 web 浏览器可存取工作目录。否则，将下列文件复制到可存取的目录：

```
Applt.html,                               Applt.class,
Applt_Cursor1.class,                       Applt_Cursor2.class,
Applt_SJProfileKeys.class,                 Applt_SJProfile0.ser
```

6. 将 OS/2 和 Windows 32 位操作系统上的文件 %DB2PATH%\java\db2java.zip 和 %DB2PATH%\java\runtime.zip，或在 UNIX 上的文件 sqllib/java/db2java.zip 和 sqllib/java/runtime.zip，复制到其他 Applt 文件所在的目录中。
7. 在您的客户机上，启动 web 浏览器（它必须支持 JDK 1.1）并装入 Applt.html。

步骤 (1)、(5) 和 (7) 的替代方法是，可在客户机的工作目录中输入如下命令，以使用 Java Development Kit 所带的 Applet 浏览器：

```
appletviewer Applt.html
```

也可使用 Java makefile 构建此程序。

应用程序

App 演示存取 DB2 数据库的 SQLJ 应用程序。

要使用构建文件 bldsqlj 来构建此应用程序，输入如下命令：

```
bldsqlj App [ <db_name> [ <userid> <password> ] ]
```

其中，可选参数 <db_name> 允许您存取另一个数据库而不是缺省数据库 sample。如果要存取的数据库在另一个实例上，比如从远程客户机存取服务器，则需要可选参数 <userid> 和 <password>。

使用如下命令，对该应用程序运行 Java 解释程序：

```
java App
```

也可使用 Java makefile 构建此程序。

存储过程的客户机程序

Stclient 是客户机程序，它使用 JDBC 应用程序驱动程序调用 SQLJ 存储过程类 Stserver。在构建并运行此客户机应用程序之前，在服务器上构建该存储过程类。参阅第76页的『存储过程』。

要使用构建文件 bldsqlj 来构建此客户机程序，输入如下命令：

```
bldsqlj Stclient [ <db_name> [ <userid> <password> ] ]
```

其中，可选参数 <db_name> 允许您存取另一个数据库而不是缺省数据库 sample。如果要存取的数据库在另一个实例上，比如从远程客户机存取服务器，则需要可选参数 <userid> 和 <password>。

使用如下命令，对客户机应用程序运行 Java 解释程序：

```
java Stclient
```

也可使用 Java makefile 构建此程序。

用户定义函数的客户机程序

UDFcli 是一个客户机程序，它使用 JDBC 应用程序的驱动程序来调用在服务器程序 UDFsrv 中实现的用户定义函数。在构建并运行此客户机应用程序之前，在服务器上构建程序 UDFsrv。参阅第79页的『用户定义函数 (UDF)』。

要使用构建文件 bldsqlj 来构建此 SQLJ 客户机程序，输入如下命令：

```
bldsqlj UDFclie [ <db_name> [ <userid> <password> ]]
```

其中，可选参数 <db_name> 允许您存取另一个数据库而不是缺省数据库 sample。如果要存取的数据库在另一个实例上，比如从远程客户机存取服务器，则需要可选参数 <userid> 和 <password>。

使用如下命令，对客户机应用程序运行 Java 解释程序：

```
java UDFclie
```

也可使用 Java makefile 构建此程序。

存储过程

构建文件 bldsqljs 包含构建 SQLJ 存储过程所需的命令。在 UNIX 上它是一个脚本文件。在 OS/2 上，它是命令文件 bldsqljs.cmd，而在 Windows 上它是批处理文件 bldsqljs.bat。命令和批处理文件的内容是相同的，此版本先提供，然后才提供 UNIX 脚本文件。

在以下用于 OS/2 和 Windows 32 位操作系统的构建文件中，第一个参数 %1 指定源文件的名称。第二个参数 %2 指定要连接的数据库的名称。因为必须在数据库所在的同一实例中构建存储过程，所以没有用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。如果未提供数据库名，则该程序使用缺省的 sample 数据库。

```
@echo off
rem bldsqljs -- OS/2 and Windows 32-bit operating systems
rem Builds a Java embedded SQL (SQLJ) stored procedure
```

```

rem Usage: bldsqljs prog_name [ db_name ]

if "%1" == "" goto error

rem Translate and compile the SQLJ source file
rem and bind the package to the database.
if "%2" == "" goto case1
goto case2
:case1
sqlj %1.sqlj
    db2profrc -url=jdbc:db2:sample -preoptions="package using %1" %1_SJProfile0
    goto continue
:case2
    sqlj      -url=jdbc:db2:%2 %1.sqlj
    db2profrc -url=jdbc:db2:%2 -preoptions="package using %1" %1_SJProfile0
:continue

rem Copy the *.class and *.ser files to the 'function' directory.
copy %1*.class "%DB2PATH%\function"
copy %1*.ser  "%DB2PATH%\function"

goto exit
:error
echo Usage: bldsqljs prog_name [ db_name ]
:exit
@echo on

```

bldsqljs 的译码器和预编译选项

<p>sqlj SQLJ 译码器（也编译该程序）。</p> <p>%1.sqlj SQLJ 源文件。</p> <p>%1.java 从 SQLJ 源文件转换的 Java 文件。</p> <p>db2profrc DB2 Java 版简要表定制器。</p> <p>-url 指定用于建立数据库连接的 JDBC URL，如 jdbc:db2:sample。</p> <p>-preoptions 用字符串 "package using %1" 指定数据库的程序包名，其中 %1 是 SQLJ 源文件名。</p> <p>%1_SJProfile0 指定该程序的串行化简要表。</p>

在以下的 UNIX 脚本文件中，第一个参数 \$1 指定源文件的名称。第二个参数 \$2 指定要连接的数据库名称。因为必须在数据库所在的同一实例中构建存储过程，所以没有用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldsqljs script file -- UNIX platforms
# Builds a Java embedded SQL (SQLJ) stored procedure
# Usage: bldsqljs <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Translate and compile the SQLJ source file
# and bind the package to the database.
if (($# < 2))
then
    sqlj $1.sqlj
    db2profc -url=jdbc:db2:sample -prepoptions="package using $1" $1_SJProfile0
else
    sqlj      -url=jdbc:db2:$2 $1.sqlj
    db2profc -url=jdbc:db2:$2 -prepoptions="package using $1" $1_SJProfile0
fi

# Copy the *.class and *.ser files to the 'function' directory.
rm -f $DB2PATH/function/$1*.class
rm -f $DB2PATH/function/$1*.ser
cp $1*.class $DB2PATH/function
cp $1*.ser $DB2PATH/function
```

bldsqljs 的译码器和预编译选项

sqlj SQLJ 译码器（也编译该程序）。

\$1.sqlj
SQLJ 源文件。

\$1.java
从 SQLJ 源文件转换的 Java 文件。

db2profc
DB2 Java 版简要表定制器。

-url 指定用于建立数据库连接的 JDBC URL，如 `jdbc:db2:sample`。

-prepoptions
用字符串 `"package using %1"` 指定数据库的程序包名，其中 `%1` 是 SQLJ 源文件名。

\$1_SJProfile0
指定该程序的串行化简要表。

Stserver 演示 PARAMETER STYLE JAVA 存储过程，它使用 JDBC 应用程序驱动程序来存取 DB2 数据库。存储过程在服务器上编译并存储。当客户机应用程序调用存储过程时，它们存取服务器数据库并将信息返回至客户机应用程序。

要使用构建文件 bldsqljs 构建此存储过程类：

1. 输入以下命令：

```
bldsqljs Stserver [ <db_name> ]
```

其中，可选参数 <db_name> 允许您存取另一个数据库而不是缺省数据库 sample。

2. 接着在服务器上运行 Stcreate.db2 脚本文件以编目存储过程。首先连接数据库：

```
db2 connect to sample
```

如果存储过程先前已编目，可使用以下命令卸下它们：

```
db2 -td@ -vf Stdrops.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf Stcreate.db2
```

3. 最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。
4. 编译并运行 Stclient 客户机应用程序，以调用存储过程。参阅第75页的『存储过程的客户机程序』。

也可使用 Java makefile 构建此程序。

用户定义函数 (UDF)

UDFsrv 演示 Java 用户定义函数。因为 UDF 程序不包含 SQL 语句，因此 Java UDF 程序也不能包含 SQLJ 语句。一旦在服务器上创建了 UDFsrv 库，则可由客户机应用程序存取它。DB2 提供 JDBC 客户机应用程序 UDFcli 和 SQLJ 客户机应用程序 UDFclie。可使用任何一个来存取 UDFsrv 库。

要通过在命令行输入命令来构建并运行 UDF 程序：

1. 使用如下命令编译 UDFsrv.java 来生成文件 UDFsrv.class：

```
javac UDFsrv.java
```

2. 在 OS/2 和 Windows 32 位操作系统上将 UDFsrv.class 复制到 %DB2PATH%\function 目录，或在 UNIX 上复制到 sqllib/function 目录。

3. 要存取 UDFsrv 库，可使用 JDBC 或 SQLJ 客户机应用程序。要编译并运行 JDBC 客户机应用程序 UDFcli，参阅第70页的『用户定义函数的客户机应用程序』。要编译并运行 SQLJ 客户机应用程序 UDFclie，参阅第76页的『用户定义函数的客户机程序』。

UDFcli 和 UDFclie 包含 CREATE FUNCTION SQL 语句，可使用它向数据库注册 UDFsrv 中包含的 UDF。UDFcli 和 UDFclie 还包含使用 UDF 的 SQL 语句（一旦注册了这些 UDF）。

DB2 Java Applet 的一般要点

1. 对于由几个 Java 类组成的一个较大的 JDBC 或 SQLJ Applet，可以选择将它的所有类封装到一个 JAR 文件中。对于 SQLJ Applet，还必须将它的串行化简要表与它的类一起封装。如果您选择这样做，则要将 JAR 文件添加到 "applet" 标记中的 archive 参数中。有关详情，参阅 JDK 版本 1.1 的文档。

对于 SQLJ Applet: 某些浏览器尚不支持从与 Applet 相关的资源文件装入串行化对象。例如，当尝试在那些浏览器中装入 Applet Applt 时，您将收到以下出错信息：

```
java.lang.ClassNotFoundException: Applt_SJProfile0
```

一个解决方法是，用一个实用程序将串行化简要表转换为以 Java 类格式存储的简要表。该实用程序是一个 Java 类，称为 `sqlj.runtime.profile.util.SerProfileToClass`。它将串行化简要表资源文件作为输入，并生成一个包含该简要表的 Java 类作为输出。可使用下列命令之一转换简要表：

```
profconv Applt_SJProfile0.ser
```

或

```
java sqlj.runtime.profile.util.SerProfileToClass Applt_SJProfile0.ser
```

最后，创建类 `Applt_SJProfile0.class`。用 `.class` 格式的简要表替换 Applet 使用的 `.ser` 格式的所有简要表，这样问题应当可解决。

2. 您可能希望将文件 `db2java.zip`（对于 SQLJ Applet，还有文件 `runtime.zip`）放入可从 Web 站点装入的几个 Applet 共享的目录中。在 OS/2 和 Windows 32 位操作系统上，这些文件位于 `%DB2PATH%\java` 目录中；而在 UNIX 上，则位于 `sqllib/java` 目录中。您可能需要将 `codebase` 参数添加到 HTML 文件的 "applet" 标记中，以标识该目录。有关详情，参阅 JDK 版本 1.1 的文档。

3. 自 DB2 版本 5.2 起, 已将信号处理添加到 JDBC Applet 服务器 (监听程序) db2jd, 使其更强健。这样, 就不能使用 CTRL-C 命令来删掉 db2jd。因此, 终止监听程序的唯一方法是停止该进程。
4. 有关在 web 服务器, 尤其是 Domino Go Webserver 上运行 DB2 Java Applet 的信息, 参阅:

<http://www.ibm.com/software/data/db2/db2lotus/gojava.htm>

第5章 构建 SQL 过程

设置 SQL 过程环境	83	OS/2 嵌入式 SQL 客户机应用程序	88
创建 SQL 过程	87	UNIX DB2 CLI 客户机应用程序	89
调用 SQL 过程	87	UNIX 嵌入式 SQL 客户机应用程序	89
使用 CALL 命令	87	Windows DB2 CLI 客户机应用程序	90
OS/2 DB2 CLI 客户机应用程序	88	Windows 嵌入式 SQL 客户机应用程序	90

本章提供构建 DB2 SQL 过程的详细信息。

在 UNIX 平台上，DB2 SQL 过程样本程序位于 `sqllib/samples/sqlproc` 目录中，而在 OS/2 和 Windows 32 位操作系统上，这些文件位于 `%DB2PATH%\samples\sqlproc` 目录中。有关样本程序的说明，参阅第25页的表10。

设置 SQL 过程环境

以下说明是对第31页的『第2章 设置』中设置 DB2 环境说明的补充。

要得到 SQL 过程支持，必须在服务器上安装应用程序开发客户机和 DB2 支持的 C 或 C++ 编译程序。有关安装应用程序开发客户机的信息，参考您的平台的快速入门一书。有关您的平台上 DB2 支持的 C 和 C++ 编译程序，参阅第6页的『按平台分类的受支持软件』。

注：在 OS/2 FAT 文件系统上，仅限于使用八个或更少字符的 SQL 过程模式名。对于超过八个字符的模式名，必须使用 HPFS 文件系统。

配置编译程序环境

要创建 SQL 过程，使用下列步骤在服务器上配置 DB2，以便使用受支持的 C 或 C++ 编译程序：

- 创建一个设置编译程序环境的可执行文件。在 OS/2 上，它是个命令文件，在 UNIX 上，它是个脚本文件，而在 Windows 上，它是个批处理文件。编译程序可能需要 `path`、`include` 和 `library` 环境变量。
- 用以下命令将 `DB2_SQLROUTINE_COMPILER_PATH` DB2 注册表变量设置为可执行文件：

```
db2set DB2_SQLROUTINE_COMPILER_PATH=executable_file
```

其中 *executable_file* 是该 C 编译程序环境文件的完整路径名。

如果没有设置 `DB2_SQLROUTINE_COMPILER_PATH` DB2 注册表变量, DB2 将其设置为缺省文件。取决于您的操作系统, 此文件将具有以下其中一个路径和文件名:

OS/2:

```
%DB2PATH%\function\routine\sr_cpath.cmd
```

UNIX:

```
$HOME/sqlllib/function/routine/sr_cpath
```

Windows:

```
%DB2PATH%\function\routine\sr_cpath.bat
```

只要您修改此缺省文件, 以反映您正使用的服务器操作系统和 C 或 C++ 编译程序所需的设置, 您就可以使用该文件。

注: 在 Windows NT 和 Windows 2000 上, 如果您将编译程序的环境变量作为 `SYSTEM` 变量存储, 则不必设置 `DB2_SQLROUTINE_COMPILER_PATH` DB2 注册表变量。

定制编译程序选项

DB2 对每个平台上它支持的其中一个编译程序提供缺省值。要使用其他编译程序, 使用 `DB2_SQLROUTINE_COMPILE_COMMAND` DB2 注册表变量设置 SQL 过程编译程序选项。要为 SQL 过程指定定制的 C 或 C++ 编译程序选项, 用以下命令在 DB2 注册表中存储包括所有选项的整个命令行:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=compiler_command
```

其中 *compiler_command* 是 C 或 C++ 编译命令, 包括创建存储过程所需的选项和参数。

在编译程序命令中, 使用关键字 `SQLROUTINE_FILENAME` 替换生成的 `SQC`、`C`、`PDB`、`DEF`、`EXP`、信息记录和共享库文件的文件名。仅对 AIX, 使用关键字 `SQLROUTINE_ENTRY` 替换入口点名称。

作为受支持的 C 或 C++ 编译程序的 `DB2_SQLROUTINE_COMPILE_COMMAND` 缺省值的示例, 此处提供 AIX、Solaris 和 Windows 32 位操作系统上的缺省编译程序值。同时提供为返回调试信息而进行的建议更改。可进行类似的更改, 以便在其他平台上返回调试信息。

AIX 这是 IBM C Set++ AIX 版的版本 3.6.6 缺省编译程序命令值:

```
x1C_r -+ -H512 -T512 -I$HOME/sqlllib/include SQLROUTINE_FILENAME.c \  
-bE:SQLROUTINE_FILENAME.exp -e SQLROUTINE_ENTRY \  
-o SQLROUTINE_FILENAME -L$HOME/sqlllib/lib -lc -ldb2
```

要返回调试信息，更改该缺省值，将 `-g` 选项添加至 `DB2_SQLROUTINE_COMPILE_COMMAND` 中，如下所示：

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="x1C_r -+ -H512 -T512 -g \  
-I$HOME/sql1ib/include SQLROUTINE_FILENAME.c \  
-bE:SQLROUTINE_FILENAME.exp -e SQLROUTINE_ENTRY \  
-o SQLROUTINE_FILENAME -L$HOME/sql1ib/lib -lc -ldb2"
```

其中 `"\"` 仅用于指示回车。

注：要在 AIX 上编译 64 位 SQL 过程，将 `-q64` 选项添加到以上命令中。

Solaris

这是 SPARCCompiler C++ 版本 4.2 和 5.0 的缺省编译程序命令值：

```
cc -# -Kpic -I$HOME/sql1ib/include SQLROUTINE_FILENAME.c -G \  
-o SQLROUTINE_FILENAME -L$HOME/sql1ib/lib -R$HOME/sql1ib/lib -ldb2
```

要返回调试信息，更改该缺省值，将 `-g` 选项添加至 `DB2_SQLROUTINE_COMPILE_COMMAND` 中，如下所示：

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="cc -# -Kpic -g \  
-I$HOME/sql1ib/include SQLROUTINE_FILENAME.c -G \  
-o SQLROUTINE_FILENAME -L$HOME/sql1ib/lib \  
-R$HOME/sql1ib/lib -ldb2"
```

其中 `"\"` 仅用于指示回车。

注：要在 Solaris 上编译 64 位 SQL 过程，将 `-xarch=v9` 选项添加到以上命令中。

Windows 32 位操作系统

这是 Microsoft Visual C++ 版本 5.0 和 6.0 的缺省编译程序命令值：

```
cl -Od -W2 /TC -D_X86_=1 -I%DB2PATH%\include SQLROUTINE_FILENAME.c  
/link -dll -def:SQLROUTINE_FILENAME.def /out:SQLROUTINE_FILENAME.dll  
%DB2PATH%\lib\db2api.lib
```

要返回调试信息，则更改缺省值，如下所示：

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="cl -Od -W2 /TC -D_X86_=1  
-Z7 -I%DB2PATH%\include SQLROUTINE_FILENAME.c /link -dll  
-def:SQLROUTINE_FILENAME.def /out:SQLROUTINE_FILENAME.dll  
-debug:full -pdb:none -debugtype:cv %DB2PATH%\lib\db2api.lib"
```

注：对于 Windows 32 位操作系统，您必须在一行上输入编译程序命令值。

要还原为缺省的编译程序选项，用以下命令将 `DB2_SQLROUTINE_COMPILE_COMMAND` 的 DB2 注册表值设置为空值：

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=
```

保留中间文件

在发出 CREATE PROCEDURE 语句时，DB2 将创建许多中间文件，正常情况下，如果 DB2 成功完成该语句，这些中间文件将被删除。如果 SQL 过程没有按照期望的那样执行，您会发现检查 DB2 创建的 SQC、C、PDB 和信息记录文件是很有用的。要保存 DB2 在成功执行 CREATE PROCEDURE 语句期间创建的文件，必须在将 DB2_SQLROUTINE_KEEP_FILES DB2 注册表变量的值设置为 "1"、"y" 或 "yes"，如下列命令所示：

```
db2set DB2_SQLROUTINE_KEEP_FILES=1
```

取决于您的操作系统，这些中间文件将保留在以下其中一个目录中：

UNIX

```
$HOME/sqllib/function/routine/sqlproc/database_name/schema_name
```

其中 *database_name* 和 *schema_name* 分别为用来创建 SQL 过程的数据库和模式。

OS/2 和 Windows

```
%DB2PATH%\function\routine\sqlproc\database_name\schema_name
```

其中 *database_name* 和 *schema_name* 分别为用来创建 SQL 过程的数据库和模式。

定制预编译和联编选项

可通过设置 DB2_SQLROUTINE_PREPOPTS DB2 注册表变量来定制预编译和联编选项。在过程级别，则不能定制这些选项。要为 SQL 过程指定定制的预编译选项，使用以下命令将 DB2 预编译程序将要使用的预编译选项列表放在 DB2 注册表中：

```
db2set DB2_SQLROUTINE_PREPOPTS=options
```

其中 *options* 指定 DB2 预编译程序将要使用的预编译选项列表。仅允许指定下列选项：

```
BLOCKING {UNAMBIG | ALL | NO}
DATETIME {DEF | USA | EUR | ISO | JIS | LOC}
DEGREE {1 | degree-of-parallelism | ANY}
DYNAMICRULES {BIND | RUN}
EXPLAIN {NO | YES | ALL}
EXPLAINSAP {NO | YES | ALL}
INSERT {DEF | BUF}
ISOLATION {CS |RR |UR |RS |NC}
QUERYOPT optimization-level
SYNCPOINT {ONEPHASE | TWOPHASE | NONE}
```

创建 SQL 过程

“DB2 命令行处理器”脚本（在 UNIX 上，位于 `sqllib/samples/sqlproc` 目录中；在 OS/2 和 Windows 上，位于 `%DB2PATH%\samples\sqlproc` 目录中；其扩展名为 `.db2`）在服务器上执行创建存储过程的 `CREATE PROCEDURE` 语句。执行此语句前必须连接数据库。每个 CLP 脚本具有一个对应的同名客户机应用程序，其扩展名为 `.sqc` 或 `.c`。

运行 `CREATE PROCEDURE CLP` 脚本之前，使用以下命令连接样本数据库：

```
db2 connect to sample user userid using password
```

其中 `userid` 和 `password` 是 `sample` 数据库所在的实例的用户 ID 和口令。

要执行 `rsultset.db2` 脚本文件中包含的 `CREATE PROCEDURE` 语句，输入以下命令：

```
db2 -td@ -vf rsultset.db2
```

接着，停止并重新启动 DB2。现在，您可以按照下面的『调用 SQL 过程』中的说明调用 SQL 过程。

调用 SQL 过程

可使用命令行处理器 (CLP) `call` 命令或构建客户机应用程序来调用 SQL 过程。

使用 CALL 命令

要使用 `call` 命令，必须输入存储过程名以及该存储过程必需的任何 `IN` 或 `INOUT` 自变量。不必输入 `OUT` 参数。

首先，按照『创建 SQL 过程』中的步骤创建 SQL 过程。

要调用该 SQL 过程，必须首先连接数据库：

```
db2 connect to sample user userid using password
```

其中 `userid` 和 `password` 是 `sample` 数据库所在的实例的用户 ID 和口令。

存储过程的参数在程序源文件中的存储过程的 `CREATE PROCEDURE` 语句中给定的。例如，在源文件 `whiles.db2` 中的 `DEPT_MEDIAN` 过程的 `CREATE PROCEDURE` 语句中：

```
CREATE PROCEDURE DEPT_MEDIAN  
(IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
```

要调用此过程，您需要为 IN 参数 deptNumber 指定一个有效的 SMALLINT 值。可从样本数据库中相应的表获得有效值，或者检查客户机调用程序源文件找到它使用的那个值。在 whiles.sqc 中，您将会发现使用了值 "51"：

```
printf("Use CALL with Host Variables to invoke the Server Procedure "  
      "named %s\n", procname);  
dept = 51;                               /* get median for dept. 51 */
```

输入 call 命令以及过程名和 IN 参数的值。过程的参数必须括在圆括号中，且必须使用引号，如下所示：

```
db2 "call DEPT_MEDIAN (51)"
```

您应得到这个结果：

```
MEDIANSALARY: 1.76545000000000e+04
```

在使用 call 命令时切记以下几点：

- 结果列最多可有 1023 个字符。
- 必须在目录中定义调用的存储过程。
- 不支持 LOB 和二进制数据 (FOR BIT DATA, VARBINARY, LONGVARBINARY, GRAPHIC, VARGRAPHIC, LONGVARGRAPHIC)。

OS/2 DB2 CLI 客户机应用程序

%DB2PATH%\samples\sqlproc 中的命令文件 bldcli.cmd 包含构建 SQL 过程的 DB2 CLI 客户机应用程序所需的命令。有关 bldcli.cmd 的详情，参阅第202页的『DB2 CLI 应用程序』。

要根据源文件 resultset.c 构建 DB2 CLI 客户机应用程序 resultset，输入：

```
bldcli resultset
```

此命令创建可执行文件 resultset。

要调用该存储过程，输入可执行文件名、要连接的数据库的名称和该数据库实例的用户 ID 和口令运行样本客户机应用程序：

```
resultset database userid password
```

OS/2 嵌入式 SQL 客户机应用程序

%DB2PATH%\samples\sqlproc 中的命令文件 bldapp.cmd 包含构建 SQL 过程的嵌入式 SQL 客户机应用程序所需的命令。有关 bldapp.cmd 的详情，参阅第206页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `basecase.sqc` 构建嵌入式 SQL 客户机应用程序 `basecase`，输入命令文件名、可执行文件名、要连接的数据库和该数据库实例的用户 ID 和口令：

```
bldapp basecase database userid password
```

产生可执行文件 `basecase`。

要调用存储过程，可输入如下命令运行客户机应用程序：

```
basecase database userid password
```

UNIX DB2 CLI 客户机应用程序

`sqllib/samples/sqlproc` 中的脚本文件 `bldcli` 包含构建 SQL 过程的 DB2 CLI 客户机应用程序所需的命令。有关 `bldcli` 脚本文件的详情，参阅您的 UNIX 平台的“构建应用程序”一章中的“DB2 CLI应用程序”一节。

要根据源文件 `resultset.c` 构建 DB2 CLI 客户机应用程序 `resultset`，输入：

```
bldcli resultset
```

此命令创建可执行文件 `resultset`。

要调用该存储过程，输入可执行文件名、要连接的数据库的名称和该数据库实例的用户 ID 和口令运行样本客户机应用程序：

```
resultset database userid password
```

UNIX 嵌入式 SQL 客户机应用程序

`sqllib/samples/sqlproc` 中的脚本文件 `bldapp` 包含构建 SQL 过程的嵌入式 SQL 客户机应用程序所需的命令。有关 `bldapp` 脚本文件的详情，参阅您的 UNIX 平台的“构建应用程序”一章中的“DB2 API和嵌入式 SQL 应用程序”一节。

要根据源文件 `basecase.sqc` 构建嵌入式 SQL 客户机应用程序 `basecase`，输入脚本文件名、可执行文件名、要连接的数据库名和该数据库实例的用户 ID 和口令：

```
bldapp basecase database userid password
```

产生可执行文件 `basecase`。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
basecase database userid password
```

Windows DB2 CLI 客户机应用程序

%DB2PATH%\samples\sqlproc 目录包含两个用于构建 DB2 CLI 客户机应用程序的构建文件: `bldmcli` 用于 Microsoft Visual C++ 编译程序, `bldvcli` 用于 IBM VisualAge C++ 编译程序。有关 `bldmcli` 的详情, 参阅第304页的『DB2 CLI 应用程序』。有关 `bldvcli` 的详情, 参阅第318页的『DB2 CLI 应用程序』。

要根据源文件 `resultset.c` 构建 DB2 CLI 客户机应用程序 `resultset`, 取决于您使用的编译程序, 输入:

```
bldmcli resultset
```

或

```
bldvcli resultset
```

这些命令创建可执行文件 `resultset`。

要调用该存储过程, 输入可执行文件名、要连接的数据库的名称和该数据库实例的用户 ID 和口令来运行样本客户机应用程序:

```
resultset database userid password
```

Windows 嵌入式 SQL 客户机应用程序

%DB2PATH%\samples\sqlproc 目录包含两个用于构建嵌入式 SQL 客户机应用程序的构建文件: `bldmapp` 用于 Microsoft Visual C++编译程序, `bldvapp` 用于 IBM VisualAge C++ 编译程序。有关 `bldmapp` 的详情, 参阅第310页的『DB2 API 和嵌入式 SQL 应用程序』。有关 `bldvapp` 的详情, 参阅第323页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `basecase.sqc` 构建嵌入式 SQL 客户机应用程序 `basecase`, 输入脚本文件名、可执行文件名、要连接的数据库名和该数据库实例的用户 ID 和口令。取决于您使用的编译程序, 此命令将会是:

```
bldmapp basecase database userid password
```

或

```
bldvapp basecase database userid password
```

产生可执行文件 `basecase`。

要调用存储过程, 可输入如下命令来运行样本客户机应用程序:

```
basecase database userid password
```

第6章 构建 AIX 应用程序

重要考虑事项	91	VisualAge C++ 版本 4.0.	120
安装和运行 IBM 与 Micro Focus COBOL	92	DB2 CLI 应用程序	121
存储过程和 UDF 的入口点	92	构建和运行嵌入式 SQL 应用程序	122
存储过程和 CALL 语句	93	使用 DB2 API 的 DB2 CLI 应用程序	123
UDF 和 CREATE FUNCTION 语句	94	DB2 CLI 存储过程	124
IBM C	95	DB2 API 应用程序	127
DB2 CLI 应用程序	95	嵌入式 SQL 应用程序	128
构建和运行嵌入式 SQL 应用程序	97	嵌入式 SQL 存储过程	130
使用 DB2 API 的 DB2 CLI 应用程序	98	用户定义函数 (UDF)	133
DB2 CLI 存储过程	98	IBM COBOL Set AIX 版	135
DB2 API 和嵌入式 SQL 应用程序	101	使用编译程序	135
构建和运行嵌入式 SQL 应用程序	103	DB2 API 和嵌入式 SQL 应用程序	136
嵌入式 SQL 存储过程	104	构建和运行嵌入式 SQL 应用程序	137
用户定义函数 (UDF)	107	嵌入式 SQL 存储过程	138
多线程应用程序	109	Micro Focus COBOL	141
IBM C Set++	110	使用编译程序	141
DB2 API 和嵌入式 SQL 应用程序	110	DB2 API 和嵌入式 SQL 应用程序	142
构建和运行嵌入式 SQL 应用程序	113	构建和运行嵌入式 SQL 应用程序	143
嵌入式 SQL 存储过程	113	嵌入式 SQL 存储过程	144
用户定义函数 (UDF)	116	退出存储过程	148
多线程应用程序	119	REXX	148

本章提供在 AIX 上构建应用程序的详细信息。在脚本文件中，以 db2 开始的命令是“命令行处理器”(CLP)命令。有关 CLP 命令的详情，可参考 *Command Reference*。

有关 AIX 的最新 DB2 应用程序开发情况的更新，请访问以下 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

注：要使用本章中的构建文件构建 64 位应用程序，您可取消每个构建文件中指示的命令的注解，或使用以下命令设置 64 位对象方式环境：

```
export OBJECT_MODE=64
```

重要考虑事项

本节提供针对 AIX 的信息，介绍如何用各种受支持的编译程序构建 DB2 应用程序。包括：

- 安装和运行 IBM 与 Micro Focus COBOL
- 存储过程和 UDF 的入口点

- 存储过程和 CALL 语句
- UDF 和 CREATE FUNCTION 语句

安装和运行 IBM 与 Micro Focus COBOL

AIX 装入存储过程的方式以及分辨存储过程所引用库的方式有其自身的特点，因此对如何安装 COBOL 有一定的要求。这些要求是 COBOL 程序在运行期装入共享库（存储过程）时要考虑的一个因素。

当装入一个存储过程时，也必须装入它引用的一系列库。当 AIX 搜索只能由您的程序间接引用的库时，它必须使用已编译到该库中的路径，该库在语言提供者（IBM COBOL 或 Micro Focus COBOL）构建该路径时会引用该路径。此路径可能与编译程序所在的路径完全不同。如果在一系列库中找不到该库，则存储过程的装入将失败，您会收到 SQLCODE -10013。

要确保不发生这种情况，在您期望之处安装编译程序，然后为所有语言库创建从安装目录到目录 /usr/lib（当需要装入库时，几乎总是要搜索的目录）的符号链路。可将这些库与 sqllib/function（存储过程目录）连接，但这只能用于一个数据库实例；而 /usr/lib 用于机器上的每个数据库实例。强烈建议您不要复制其中的库；对于 Micro Focus COBOL，当有这些库的多个副本存在时更应如此。

Micro Focus COBOL 的一个样本符号链路如下所示（假定它安装在 /usr/lpp/cobdir 中）：

```
[1]> su root
[2]> cd /usr/lib
[1]> ln -sf /usr/lpp/cobdir/coblib/*.a
```

存储过程和 UDF 的入口点

存储过程是存取数据库并将信息返回给客户机应用程序的一组程序。“用户定义函数”（UDF）是您自己的标量函数或表函数。存储过程和 UDF 在服务器上编译，并在服务器上的共享库中存储和执行。这些共享库是在编译存储过程和 UDF 时创建的。

每个共享库有一个入口点，可从服务器调用该入口点来存取共享库中的过程。AIX 上的 IBM C 编译程序允许您将库中任何调出的函数名指定为缺省入口点。此函数是在存储过程调用或在 CREATE FUNCTION 语句中仅指定了库名时被调用的函数。该入口点的指定可在连接步骤中用 -e 选项来完成。例如：

```
-e funcname
```

使 funcname 成为缺省入口点。有关该函数如何与 CREATE FUNCTION 语句相关的信息，参阅第94页的『UDF 和 CREATE FUNCTION 语句』。

在其他 UNIX 平台上，不存在这样的机制，因此 DB2 假定：缺省入口点与共享库本身同名。

AIX 要求您提供调出文件，该文件指定共享库中的哪些全局函数是可从外部调用的。此文件必须包括共享库中所有存储过程和 / 或用户定义函数的名称。其他 UNIX 平台则是调出共享库中所有全局函数。以下是 AIX 调出文件的示例：

```
#!/outsrv export file
outsrv
```

调出文件 `outsrv.exp` 列出存储过程 `outsrv`。连接程序使用 `outsrv.exp` 来创建共享库 `outsrv`，该库包含与它同名的存储过程。

注：当构建了共享库后，一般是将它复制到 DB2 将从其中存取该库的目录中。当试图置换存储过程或用户定义函数的共享库时，应运行 `/usr/sbin/slibclean` 以清除 AIX 共享库高速缓存，或从目标目录中除去该库，然后从源目录中将该库复制到目标目录。否则，因为 AIX 会保留所引用库的高速缓存且不允许覆盖该库，从而可能导致复制操作失败。

AIX 编译程序文档包含有关调出文件的其他信息。

存储过程和 CALL 语句

Application Development Guide 描述如何编写存储过程。*SQL Reference* 描述如何在数据库的所在位置使用 CALL 语句调用存储过程。本节描述如何按 CALL 语句中提供的信息编译和连接存储过程。

当编译并连接程序时，可用以下两种方式标识函数：

- 使用 `-e` 选项。

例如，可在连接步骤中指定：

```
-e modify
```

它指示连接的库的缺省入口点是函数 `modify`。

如果正在连接目录 `/u/mydir/procs` 中的库 `mystored`，且要使用上面所指定的缺省入口点 `modify`，则按如下所示编写 CALL 语句：

```
CALL '/u/mydir/procs/mystored'
```

这会将库 `mystored` 装入内存，且 DB2 会挑出函数 `modify` 作为缺省入口点，并执行它。

- 使用由 `-bE`：选项指定的调出文件。

一般而言，当库中有多个存储过程且还要存取作为存储过程的其他函数时，将使用此连接选项。

继续上述示例，假定库 `mystored` 包含三个存储过程：如上所示的 `modify`，另外还有 `remove` 和 `add`。如上所示将 `modify` 标识为缺省入口点，并将 `remove` 和 `add` 包括在调出文件中，以便在连接步骤中指示它们是附加入口点。

在连接步骤中，指定：

```
-bE:mystored.exp
```

它标识调出文件 `mystored.exp`。

调出文件将是一个存储过程函数列表，缺省入口点列示在最前面：

```
modify  
remove  
add
```

最后，存储过程中调用 `remove` 和 `add` 函数的两个 `CALL` 语句按如下方式编写：

```
CALL '/u/mydir/procs/mystored!remove'
```

和

```
CALL '/u/mydir/procs/mystored!add'
```

UDF 和 CREATE FUNCTION 语句

Application Development Guide 描述如何编写 UDF。 *SQL Reference* 描述如何使用 `CREATE FUNCTION` 语句将 UDF 注册到 DB2 中。本节说明编译和连接 UDF 与您在 `CREATE FUNCTION` 语句的 `EXTERNAL NAME` 子句中提供的信息之间的关系。

当编译并连接程序时，可用以下两种方式标识函数：

- 使用 `-e` 选项。

例如，可在连接步骤中指定：

```
-e modify
```

它指示连接的库的缺省入口点是函数 `modify`。

如果正在连接目录 `/u/mydir/procs` 中的库 `myudfs`，且要使用上面所指定的缺省入口点 `modify`，则在 `CREATE FUNCTION` 语句中包括如下内容：

```
EXTERNAL NAME '/u/mydir/procs/myudfs'
```

DB2 会挑出函数 `modify` 作为库 `myudfs` 的缺省入口点。

- 使用由 `-bE:` 选项指定的调出文件。

一般而言，当库中有多个 UDF 且还要存取作为 UDF 的其他函数时，将使用此连接选项。

继续上述示例，假定库 `myudfs` 包含三个 UDF：如上所示的 `modify`，另外还有 `remove` 和 `add`。如上所示将 `modify` 标识为缺省入口点，并将 `remove` 和 `add` 包括在调出文件中，以便在连接步骤中指示它们是附加入口点。

在连接步骤中，指定：

```
-bE:myudfs.exp
```

它标识调出文件 `myudfs.exp`。

该调出文件如下所示：

```
* additional entry points for myudfs
#!
remove
add
```

最后，由 `remove` 和 `add` 函数实现的 UDF 的两个 `CREATE FUNCTION` 语句将包含如下的 `EXTERNAL NAME` 子句：

```
EXTERNAL NAME '/u/mydir/procs/myudfs!remove'
```

和

```
EXTERNAL NAME '/u/mydir/procs/myudfs!add'
```

IBM C

本节说明如何使用 IBM C 来构建使用下列种类的 DB2 接口的程序：

- DB2 CLI
- DB2 API
- 嵌入式 SQL

DB2 CLI 应用程序

`sqllib/samples/cli` 中的脚本文件 `blcli` 包含构建 DB2 CLI 程序所需的命令。参数 `$1` 指定源文件名。

这是唯一的必需参数，且是不包含嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还需提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

如果该程序包含嵌入式 SQL（根据扩展名 .sql 来判断），则调用脚本 embprep 来预编译该程序，生成一个扩展名为 .c 的程序文件。

```
#!/bin/ksh
# bldcli script file -- AIX
# Builds a CLI program with IBM C.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sql" ]]
then
    embprep $1 $2 $3 $4
fi

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# Compile the error-checking utility.
xlc $CFLAGS_64 -I$DB2PATH/include -c utilcli.c

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c

# Link the program.
xlc $CFLAGS_64 -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2
```


bldcli 的编译和连接选项

编译选项:

xlc IBM C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

连接选项:

xlc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-o \$1 指定可执行程序。

\$1.o 指定目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`, 输入:

```
bldcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名, 运行该可执行文件:

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldcli dbusemx database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldcli dbusemx database userid password
```

产生可执行文件 dbusemx。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 sample 数据库，只须输入可执行文件名：

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

sqllib/samples/cli 中的脚本文件 bldapi 包含使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 utilapi 实用程序文件（该文件包含创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 bldcli 脚本文件之间唯一的区别。有关 bldapi 和 bldcli 之间公共的编译和连接选项，请参阅第95页的『DB2 CLI 应用程序』。

要根据源文件 dbmconn.c 构建样本程序 dbmconn，输入：

```
bldapi dbmconn
```

产生可执行文件 dbmconn。可输入可执行文件名，运行该可执行文件：

```
dbmconn
```

DB2 CLI 存储过程

sqllib/samples/cli 中的脚本文件 bldclisp 包含构建 DB2 CLI 存储过程所需的命令。参数 \$1 指定源文件名；参数 \$2 指定作为共享库入口点的存储过程函数。

```
#!/bin/ksh
# bldclisp script file -- AIX
# Builds a CLI stored procedure in IBM C.
```

```

# Usage: bldclisp <prog_name> [ <entry_point> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# Compile the error-checking utility.
xlc $CFLAGS_64 -I$DB2PATH/include -c utilcli.c

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c

# Link the program.
xlc $CFLAGS_64 -o $1 $1.o utilcli.o -L$DB2PATH/lib \
    -ldb2 -lm -H512 -T512 -bE:$1.exp -e $2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldclisp 的编译和连接选项

编译选项:

xlc IBM C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。

-c

只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

bldclisp 的编译和连接选项

连接选项:

xlc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-o \$1 指定可执行程序。

\$1.o 指定目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sql/lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

-ldb2 连接 DB2 库。

-lm 与数学库连接。

-H512 指定输出文件对齐。

-T512 指定输出文件文本段起始地址。

-bE:\$exp

指定调出文件。该调出文件包含存储过程的列表。

-e \$2 指定共享库的缺省入口点。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.c` 构建样本程序 `spserver`, 输入构建文件名、程序名以及作为共享库入口点的存储过程函数的名称:

```
bldclisp spserver outlanguage
```

此脚本文件将存储过程复制到 `sql/lib/function` 目录中。

接着在服务器上运行 `spcreate.db2` 脚本以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`，就可构建调用共享库中的存储过程的 CLI 客户机应用程序 `spclient`。

可使用脚本文件 `bldcli` 构建 `spclient`。有关详情，参考第95页的『DB2 CLI 应用程序』。

要存取该共享库，可输入以下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/c` 中的构建文件 `bldapp` 包含构建 DB2 应用程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是唯一的必需参数，且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定数据库的用户 ID；第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序，`bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名，则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- AIX
# Builds a C application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ] ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
```

```

DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# If embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.c error-checking utility.
    xlc $CFLAGS_64 -I$DB2PATH/include -c utilemb.c
else
    # Compile the utilapi.c error-checking utility.
    xlc $CFLAGS_64 -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
    # Link the program with utilemb.o
    xlc $CFLAGS_64 -o $1 $1.o utilemb.o -ldb2 -L$DB2PATH/lib
else
    # Link the program with utilapi.o
    xlc $CFLAGS_64 -o $1 $1.o utilapi.o -ldb2 -L$DB2PATH/lib
fi

```

bldapp 的编译和连接选项

编译选项:

xlc IBM C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。

-c

只执行编译; 不连接。编译和连接是两个独立的步骤。

bldapp 的编译和连接选项

连接选项:

x1c 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-o \$ 指定可执行程序。

\$1.o 指定该程序的目标文件。

utilemb.o

如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.o

如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

-ldb2 连接数据库管理程序库。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sql/lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.c` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。

要运行可执行文件, 输入可执行文件名:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名称:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

`sqllib/samples/c` 中的脚本文件 `bldsrv` 包含构建存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定作为共享库入口点的存储过程函数。第三个参数 `$3` 指定您想连接的数据库的名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要用户 ID 和口令的参数。

只需要前两个参数, 即源文件名和入口点。数据库名称是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldsrv script file -- AIX
# Builds a C stored procedure
# Usage: bldsrv <prog_name> <entry_point> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $3

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c

# Link the program using the export file $1.exp,
# creating shared library $1 with entry point $2.
```



```
xlc $CFLAGS_64 -o $1 $1.o -ldb2 -L$DB2PATH/lib -H512 -T512 -bE:$1.exp -e $2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv 的编译和连接选项	
编译选项:	
xlc	IBM C 编译程序。
\$CFLAGS_64	如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。
-I\$DB2PATH/include	指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。
-c	只执行编译; 不连接。编译和连接是两个独立的步骤。
连接选项:	
xlc	使用编译程序作为连接程序的前端。
\$CFLAGS_64	如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。
-o \$1	指定输出为共享库文件。
\$1.o	指定存储过程的目标文件。
-ldb2	连接 DB2 库。
-L\$DB2PATH/lib	指定 DB2 运行期共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。
-H512	指定输出文件对齐。
-T512	指定输出文件文本段起始地址。
-bE:\$1.exp	指定调出文件。该调出文件包含存储过程的列表。
-e \$1	指定共享库的缺省入口点。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入构建文件名、程序名以及作为共享库入口点的存储过程函数的名称:

```
bldsrv spserver outlanguage
```

如果连接另一个数据库，还须输入该数据库名：

```
bldsrv spserver outlanguage database
```

该脚本文件将存储过程复制到服务器的 `sqllib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库：

```
db2 connect to sample
```

如果存储过程先前已经编目，可使用以下命令卸下它们：

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`，就可构建存取该共享库的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第101页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

sqllib/samples/c 中的脚本文件 bldudf 包含构建 UDF 所需的命令。UDF 的编译与存储过程相似。它们不能包含 SQL 语句。这意味着要构建 UDF 程序，不必连接数据库来预编译和联编该程序。

第一个参数 \$1 指定源文件的名称。第二个参数 \$2 指定作为共享库入口点的存储过程函数。此脚本文件将源文件名 \$1 用作共享库名。

```
#!/bin/ksh
# bldudf script file -- AIX
# Builds a C UDF library
# Usage: bldudf <prog_name> <entry_point>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c

# Link the program.
xlc $CFLAGS_64 -o $1 $1.o -ldb2 -ldb2apie -L$DB2PATH/lib -H512 -T512 -bE:$1.exp -e $2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bludf 的编译和连接选项

编译选项:

xlc IBM C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqlllib/include。

-c 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

连接选项:

xlc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-o \$1 指定输出为共享库文件。

\$1.o 指定共享库的目标文件。

-ldb2 连接数据库管理程序库。

-ldb2apie

连接 "DB2 API 引擎" 库, 以便可使用 LOB 定位器。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

-H512 指定输出文件对齐。

-T512 指定输出文件文本段起始地址。

-bE:\$1.exp

指定调出文件。该调出文件包含 UDF 的列表。

-e \$2 指定共享库的缺省入口点。

有关其他编译程序选项, 参考编译程序文档。有关创建 UDF 的详情, 参考第94页的『UDF 和 CREATE FUNCTION 语句』。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入构建文件名、程序名以及作为共享库入口点的 UDF 函数:

```
bludf udfsrv ScalarUDF
```

此脚本文件将 UDF 复制到 sqllib/function 目录中。

必要时，将该 UDF 设置为文件方式，以便客户机应用程序可存取它。

一旦构建了 udfsrv，就可构建调用它的客户机应用程序 udfcli。提供了此程序的 DB2 CLI 和嵌入式 SQL 版本。

可根据 sqllib/samples/cli 中的源文件 udfcli.c，使用脚本文件 bldcli 构建 DB2 CLI udfcli 程序。有关详情，参考第95页的『DB2 CLI 应用程序』。

可根据 sqllib/samples/c 中的源文件 udfcli.sqc，使用脚本文件 bldapp 构建嵌入式 SQL udfcli 程序。有关详情，参考第101页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF，可输入可执行文件名运行样本调用应用程序：

```
udfcli
```

调用应用程序从 udfsrv 库中调用 ScalarUDF 函数。

多线程应用程序

AIX 版本 4 上的 C 多线程应用程序需要使用 xlc_r 编译程序而不是 xlc 编译程序来编译和连接，而对于 C++，应使用 x1C_r 编译程序而不是 x1C 编译程序。如果要使用 AIX 4.3 或更高版本来运行 32 位应用程序，应使用 xlc_r7 或 x1C_r7 编译程序。_r 版本（以及其他多线程编译程序前端）为多线程编译设置恰当的预处理程序定义，并向连接程序提供恰当的线程库名称。

如果使用 3.1 版本的编译程序，可从 /etc/x1C.cfg 获取有关使用多线程编译程序前端设置编译程序和连接标记的其他信息；如果使用 3.6 或更高版本的编译程序，可从 /etc/ibmcxx.cfg 获取这些信息。

sqllib/samples/c 中的脚本文件 bldmt 包含构建嵌入式 SQL 多线程程序所需的命令。第一个参数 \$1 指定源文件的名称。第二个参数 \$2 指定要连接的数据库名称。参数 \$3 指定数据库的用户 ID，而 \$4 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 sample 数据库。

```
#!/bin/ksh
# bldmt script file -- AIX
# Builds a C multi-threaded embedded SQL program.
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
```

```

# Precompile and bind the program.
embprep $1 $2 $3 $4

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# Compile the program.
xlc_r $CFLAGS_64 -I$DB2PATH/include -c $1.c

# Link the program.
xlc_r $CFLAGS_64 -o $1 $1.o -L$DB2PATH/lib -ldb2

```

除了以上讨论的 `xlc_r` 编译程序之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第101页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sqc` 构建多线程样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入该可执行文件名：

```
thdsrver
```

IBM C Set++

本节包括下列主题：

- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 API 和嵌入式 SQL 应用程序

`sqlllib/samples/cpp` 中的构建文件 `bldapp` 包含构建 DB2 API 和嵌入式 SQL 应用程序所需的命令。

第一个参数 \$1 指定源文件的名称。这是唯一的必需参数，且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- AIX
# Builds a C++ application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.C error-checking utility.
    xlc $CFLAGS_64 -I$DB2PATH/include -c utilemb.C
else
    # Compile the utilapi.C error-checking utility.
    xlc $CFLAGS_64 -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.C

if [[ -f $1".sqc" ]]
then
    # Link the program with utilemb.o
    xlc $CFLAGS_64 -o $1 $1.o utilemb.o -ldb2 -L$DB2PATH/lib
else
    # Link the program with utilapi.o
    xlc $CFLAGS_64 -o $1 $1.o utilapi.o -ldb2 -L$DB2PATH/lib
fi
```

bldapp 的编译和连接选项

编译选项:

x1C IBM C Set ++ 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sql1lib/include。

-c

只执行编译; 不连接。编译和连接是两个独立的步骤。

连接选项:

x1C 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-o \$1 指定可执行程序。

-o \$1 指定该程序的目标文件。

utilapi.o

包括用于非嵌入式 SQL 程序的 API 实用程序目标文件。

utilemb.o

包括用于嵌入式 SQL 程序的嵌入式 SQL 实用程序目标文件。

-ldb2 连接数据库管理程序库。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.C` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```


构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

注: 请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ 存储过程的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldsrv` 包含构建存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定作为共享库入口点的存储过程函数。第三个参数 `$3` 指定您想连接的数据库的名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要用户 ID 和口令的参数。

只需要前两个参数, 即源文件名和入口点。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldsrv script file -- AIX
# Builds a C++ stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
```

```

DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
    LFLAGS_64=-X64
else
    CFLAGS_64=
    LFLAGS_64=
fi

# Compile the program.
x1c $CFLAGS_64 -I$DB2PATH/include -c $1.C

# Link using export file $1.exp, creating shared library $1
makeC++SharedLib $LFLAGS_64 -p 1024 -o $1 $1.o -L$DB2PATH/lib -ldb2 -E $1.exp

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bidsrv 的编译和连接选项

编译选项:

x1c IBM C Set ++ 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。

-c

只执行编译; 不连接。编译和连接是两个独立的步骤。

blsrv 的编译和连接选项

连接选项:

makeC++SharedLib

含静态构造程序的存储过程所用的连接程序脚本。

\$LFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-X64"; 否则, 它不包含任何值。

-p 1024

将优先级设置为任意值 1024。

-o \$1 指定输出为共享库文件。

\$1.o 指定该程序的目标文件。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

-ldb2 连接数据库管理程序库。

-E \$1.exp

指定调出文件。该调出文件包含存储过程的列表。

-e \$2 指定共享库的入口点。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入构建文件名、程序名以及作为共享库入口点的存储过程函数的名称:

```
blsrv spserver outlanguage
```

如果连接另一个数据库, 还须输入该数据库名:

```
blsrv spserver outlanguage database
```

此脚本文件将共享库复制到服务器的 `sqllib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`，就可构建调用共享库中的存储过程的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第110页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数，然后将输出返回给客户机应用程序。

用户定义函数 (UDF)

注：请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ UDF 的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldudf` 包含构建 UDF 所需的命令。UDF 不能包含嵌入式 SQL 语句。因此，要构建 UDF 程序，不必连接数据库来预编译和联编该程序。

参数 `$1` 指定源文件的名称。参数 `$2` 指定作为共享库入口点的用户定义函数。此脚本文件将源文件名 `$1` 用作共享库名。

```
#!/bin/ksh
# bldudf script file -- AIX
# Builds a C++ UDF library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
```

```

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
    LFLAGS_64=-X64
else
    CFLAGS_64=
    LFLAGS_64=
fi

# Compile the program.
if [[ -f $1".c" ]]
then
    x1C $CFLAGS_64 -I$DB2PATH/include -c $1.c
elif [[ -f $1".C" ]]
then
    x1C $CFLAGS_64 -I$DB2PATH/include -c $1.C
fi

# Link using export file $1.exp, creating shared library $1
makeC++SharedLib $LFLAGS_64 -p 1024 -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie -E $1.exp

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bludf 的编译和连接选项

编译选项:

x1C IBM C Set ++ 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-q64"; 否则, 它不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。

-c 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

bldudf 的编译和连接选项

连接选项:

makeC++SharedLib

含静态构造程序的存储过程所用的连接程序脚本。

\$LFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-x64"; 否则, 它不包含任何值。

-p 1024

将优先级设置为任意值 1024。

-o \$1 指定输出为共享库文件。

\$1.o 指定该程序的目标文件。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sql/lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

-ldb2 连接数据库管理程序库。

-ldb2apie

连接“DB2 API 引擎”库, 以便可使用 LOB 定位器。

-E \$1.exp

指定调出文件。该调出文件包含存储过程的列表。

有关其他编译程序选项, 参考编译程序文档。有关创建 UDF 的详情, 参考第94页的『UDF 和 CREATE FUNCTION 语句』。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入构建文件名、程序名以及作为共享库入口点的 UDF 函数:

```
bldudf udfsrv ScalarUDF
```

此脚本文件将 UDF 复制到服务器的 `sql/lib/function` 路径中。

必要时, 将该 UDF 设置为文件方式, 以便 DB2 实例可运行它。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。可根据 `sql/lib/samples/cpp` 中的 `udfcli.sqc` 源文件, 使用脚本文件 `bldapp` 来构建 `udfcli` 程序。有关详情, 参考第110页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF, 可输入可执行文件名运行样本调用应用程序:

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

AIX 版本 4 上的 C++ 多线程应用程序需要使用 `x1C_r` 编译程序而不是 `x1C` 编译程序来编译和连接，而对于 C，应使用 `x1c_r` 编译程序而不是 `x1c` 编译程序。如果使用 AIX 4.3 或更高版本来运行 32 位应用程序，应使用 `x1C_r7` 或 `x1c_r7` 编译程序。`_r` 版本（以及其他多线程编译程序前端）为多线程编译设置恰当的预处理程序定义，并向连接程序提供恰当的线程库名称。

如果使用 3.1 版本的编译程序，可从 `/etc/x1C.cfg` 获取有关使用多线程编译程序前端设置编译程序和连接标记的其他信息；如果使用 3.6 或更高版本的编译程序，可从 `/etc/ibmcxx.cfg` 获取这些信息。

`sqllib/samples/cpp` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。参数 `$3` 指定数据库的用户 ID，而 `$4` 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldmt script file -- AIX
# Builds a C++ multi-threaded embedded SQL program
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-q64
else
    CFLAGS_64=
fi

# Compile the program.
x1C_r $CFLAGS_64 -I$DB2PATH/include -c $1.C

# Link the program.
x1C_r $CFLAGS_64 -o $1 $1.o -L$DB2PATH/lib -ldb2
```

除了以上讨论的 `x1C_r` 编译程序之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第110页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sql` 构建多线程样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入该可执行文件名：

```
thdsrver
```

VisualAge C++ 版本 4.0

此 VisualAge C++ 编译程序可在 AIX、OS/2 和 Windows 32 位操作系统上运行。本节中的信息适用于所有这些平台。

此 VisualAge C++ 编译程序与本书中描述的其他编译程序不同。要用 VisualAge C++ 版本 4.0 编译程序，必须首先创建一个配置文件。参阅与该编译程序一起提供的文档以获得更多信息。

DB2 为可用 VisualAge C++ 编译程序构建的不同类型的 DB2 程序提供配置文件。要使用 DB2 配置文件，首先将一个环境变量设置为要编译的程序名。接着用 VisualAge C++ 提供的命令编译该程序。以下是由 DB2 提供的配置文件，和描述如何使用它们编译程序的段落：

cli.icc

DB2 CLI 配置文件。有关详情，参阅第121页的『DB2 CLI 应用程序』。

cliapi.icc

使用 DB2 API 的 DB2 CLI 配置文件。有关详情，参阅第123页的『使用 DB2 API 的 DB2 CLI 应用程序』。

clis.icc

DB2 CLI 存储过程配置文件。有关详情，参阅第124页的『DB2 CLI 存储过程』。

api.icc

DB2 API 配置文件。有关详情，参阅第127页的『DB2 API 应用程序』。

emb.icc

嵌入式 SQL 配置文件。有关详情，参阅第128页的『嵌入式 SQL 应用程序』。

stp.icc

嵌入式 SQL 存储过程配置文件。有关详情，参阅第130页的『嵌入式 SQL 存储过程』。

udf.icc

用户定义函数配置文件。有关详情，参阅第133页的『用户定义函数 (UDF)』。

DB2 CLI 应用程序

配置文件 cli.icc (在 AIX 上，位于 sqllib/samples/cli 中；在 OS/2 和 Windows 32 位操作系统上，位于 %DB2PATH%\samples\cli 中) 允许您构建 DB2 CLI 程序。

```
// cli.icc configuration file for DB2 CLI applications
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export CLI=prog_name'
// To use on OS/2 and Windows, enter: 'set CLI=prog_name'
// Then compile the program by entering: 'vacblcl cli.icc'

if defined( $CLI )
{
    prog_name = $CLI
}
else
{
    error "Environment Variable CLI is not defined."
}

infile   = prog_name".c"
utilcli = "utilcli.c"

if defined( $__TOS_AIX__ )
{
    // Set db2path to where DB2 will be accessed.
    // The default is the standard instance path.
    db2path   = $HOME"/sqllib"
    outfile   = prog_name
    group lib  = "libdb2.a"
    option opts = link( libsearchpath, db2path"/lib" ),
                    incl( searchPath, db2path"/include" )
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
    db2path   = $DB2PATH
    outfile   = prog_name".exe"
    group lib  = "db2cli.lib"
    option opts = link( libsearchpath, db2path"\\lib" ),
                    incl( searchPath, db2path"\\include" )
}

option opts
```

```

{
  target type(exe) outfile
  {
    source infile
    source utilcli
    source lib
  }
}

```

VisualAge C++ 版本 4.0 定义下列其中一个环境变量，这取决于它安装在哪种操作系统上：__TOS_AIX__、__TOS_OS2__、__TOS_WIN__。

要根据源文件 `tbinfo.c` 使用配置文件构建 DB2 CLI 样本程序 `tbinfo`，执行以下操作：

1. 输入以下命令，将 CLI 环境变量设置为该程序名：

```
export CLI=tbinfo
```

2. 如果在工作目录中有 `cli.ics` 文件，该文件是用 `cli.icc` 文件构建另一个程序生成的，可使用以下命令删除 `cli.ics` 文件：

```
rm cli.ics
```

不必删除对将要再次构建的同一程序生成的现存 `cli.ics` 文件。

3. 输入以下命令，编译样本程序：

```
vacbld cli.icc
```

注：`vacbld` 命令是由 VisualAge C++ 版本 4.0 提供的。

产生可执行文件 `tbinfo`。可输入可执行文件名来运行该程序：

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

当在 AIX 上使用 `embprep` 文件、在 OS/2 上使用 `embprep.cmd` 文件或在 Windows 32 位操作系统上使用 `embprep.bat` 文件预编译嵌入式 SQL 程序之后，就可使用 `cli.icc` 配置文件来编译该程序。此文件预编译源文件，并将程序与数据库联编。

有三种方法可从源文件 `dbusemx.sqc` 预编译嵌入式 SQL 应用程序 `dbusemx`：

1. 如果与同一实例中的 `sample` 数据库连接，输入：

```
embprep dbusemx
```

2. 如果与同一实例中的另一个数据库连接，还须输入数据库名：

```
embprep dbusemx database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
embprep dbusemx database userid password
```

结果是预编译 C 文件 `dbusemx.c`。

预编译之后，可使用 `cli.icc` 文件编译 C 文件，如下所示：

1. 输入以下命令，将 CLI 环境变量设置为该程序名：

```
export CLI=dbusemx
```

2. 如果在工作目录中有 `cli.ics` 文件，该文件是用 `cli.icc` 文件构建另一个程序生成的，可使用以下命令删除 `cli.ics` 文件：

```
rm cli.ics
```

不必删除对将要再次构建的同一程序生成的现存 `cli.ics` 文件。

3. 输入以下命令，编译样本程序：

```
vacbld cli.icc
```

注：`vacbld` 命令是由 VisualAge C++ 版本 4.0 提供的。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名：

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第 22 页的表 7 中的 CLI 样本程序的说明指示使用 DB2 API 的样本。配置文件 `cliapi.icc`（在 AIX 上，位于 `sqllib/samples/cli` 中；在 OS/2 和 Windows 32 位操作系统上，位于 `%DB2PATH%\samples\cli` 中）允许您构建使用 DB2 API 的 DB2 CLI 程序。

此文件在 `utilapi` 实用程序文件（该文件包含创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 `cli.icc` 配置文件之间唯一的区别。

要根据源文件 `dbmconn.c` 构建 DB2 CLI 样本程序 `dbmconn`，执行以下操作：

1. 输入以下命令，将 CLI API 环境变量设置为程序名：

```
export CLI_API=dbmconn
```

2. 如果在工作目录中有 cliapi.ics 文件（该文件是使用 cliapi.icc 文件构建另一个程序生成的），应使用以下命令删除 cliapi.ics 文件：

```
rm cliapi.ics
```

不必删除对将要再次构建的同一程序生成的现存 cliapi.ics 文件。

3. 输入以下命令，编译样本程序：

```
vacbld cliapi.icc
```

注：vacbld 命令是由 VisualAge C++ 版本 4.0 提供的。

产生可执行文件 dbmconn。可输入可执行文件名来运行该程序：

```
dbmconn
```

DB2 CLI 存储过程

配置文件 clis.icc（在 AIX 上，位于 sqllib/samples/cli 中；在 OS/2 和 Windows 32 位操作系统上，位于 %DB2PATH%\samples\cli 中）允许您构建 DB2 CLI 存储过程。

```
// clis.icc configuration file for DB2 CLI stored procedures
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export CLIS=prog_name'
// To use on OS/2 and Windows, enter: 'set CLIS=prog_name'
// Then compile the program by entering: 'vacbld clis.icc'
```

```
if defined( $CLIS )
{
  prog_name = $CLIS
}
else
{
  error "Environment Variable CLIS is not defined."
}
```

```
infile   = prog_name".c"
utilcli  = "utilcli.c"
expfile  = prog_name".exp"
```

```
if defined( $_TOS_AIX_ )
{
  // Set db2path to where DB2 will be accessed.
  // The default is the standard instance path.
  db2path   = $HOME"/sqllib"
  outfile   = prog_name
  group lib = "libdb2.a"
  option opts = link( exportList, expfile ),
                 link( libsearchpath, db2path"/lib" ),
                 incl( searchPath, db2path"/include" )
  cpcmd     = "cp"
```

```

    funcdir    = db2path"/function"
}
else /* if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ ) */
{
    db2path    = $DB2PATH
    outfile    = prog_name".dll"
    if defined( $__TOS_WIN__ )
    {
        expfile = prog_name"v4.exp"
    }
    group lib  = "db2cli.lib"
    option opts = link( exportList, expfile ),
                  link( libsearchpath, db2path"\\lib" ),
                  incl( searchPath, db2path"\\include" )

    cpcmd      = "copy"
    funcdir    = db2path"\\function"
}

option opts
{
    target type(dll) outfile
    {
        source infile
        source utilcli
        source lib
    }
}

if defined( $__TOS_AIX__ )
{
    rmcmd      = "rm -f"
    run after rmcmd " " funcdir "/" outfile
}

run after cpcmd " " outfile " " funcdir

```

VisualAge C++ 版本 4.0 定义下列其中一个环境变量，这取决于它安装在哪种操作系统上：__TOS_AIX__、__TOS_OS2__、__TOS_WIN__。

要根据源文件 spserver.c 使用配置文件构建 DB2 CLI 存储过程 spserver，执行以下操作：

1. 输入以下命令，将 CLIS 环境变量设置为该程序名：

```
export CLIS=spserver
```

2. 如果在工作目录中有 clis.ics 文件，该文件是用 clis.icc 文件构建另一个程序生成的，可使用以下命令删除 clis.ics 文件：

```
rm clis.ics
```

不必删除对将要再次构建的同一程序生成的现存 clis.ics 文件。

3. 输入以下命令，编译样本程序：

```
vacbld clis.icc
```

注: vacbld 命令是由 VisualAge C++ 版本 4.0 提供的。

该存储过程被复制到服务器上的 `sqllib/function` 路径 (AIX 上) 或 `%DB2PATH%\function` 路径 (OS/2 和 Windows 32 位操作系统上) 中。

接着在服务器上运行 `spscreate.db2` 脚本文件以编目存储过程。首先使用数据库所在的实例的用户 ID 和口令连接数据库:

```
db2 connect to sample userid password
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`, 就可构建调用该存储过程的 CLI 客户机应用程序 `spclient`。可使用配置文件 `cli.icc` 构建 `spclient`。有关详情, 参考第121页的『DB2 CLI 应用程序』。

要调用存储过程, 可输入如下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。该名称可以是 `sample`、其远程别名或其他名称。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

DB2 API 应用程序

配置文件 `api.icc` (在 AIX 上, 位于 `sqllib/samples/c` 和 `sqllib/samples/cpp` 中; 在 OS/2 和 Windows 32 位操作系统上, 位于 `%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中) 允许您用 C 或 C++ 构建 DB2 API 程序。

```
// api.icc configuration file for DB2 API programs
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export API=prog_name'
// To use on OS/2 and Windows, enter: 'set API=prog_name'
// Then compile the program by entering: 'vacbld api.icc'

if defined( $API )
{
    prog_name = $API
}
else
{
    error "Environment Variable API is not defined."
}

infile = prog_name".c"
util = "utilapi.c"

if defined( $__TOS_AIX__ )
{
    // Set db2path to where DB2 will be accessed.
    // The default is the standard instance path.
    db2path = $HOME"/sqllib"
    outfile = prog_name
    group lib = "libdb2.a"
    option opts = link( libsearchpath, db2path"/lib" ),
        incl( searchPath, db2path"/include" )
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
    db2path = $DB2PATH
    outfile = prog_name".exe"
    group lib = "db2api.lib"
    option opts = link( libsearchpath, db2path"\\lib" ),
        incl( searchPath, db2path"\\include" )
}

option opts
{
    target type(exe) outfile
    {
        source infile
        source util
        source lib
    }
}
```

VisualAge C++ 版本 4.0 定义下列其中一个环境变量，这取决于它安装在哪种操作系统上：__TOS_AIX__、__TOS_OS2__、__TOS_WIN__。

要根据源文件 `client.c` 使用配置文件来构建 DB2 API 样本程序 `client`，执行以下操作：

1. 输入以下命令，将 API 环境变量设置为该程序名：

```
export API=client
```

2. 如果在工作目录中有 `api.ics` 文件，该文件是用 `api.icc` 文件构建另一个程序生成的，可使用以下命令删除 `api.ics` 文件：

```
rm api.ics
```

不必删除对将要再次构建的同一程序生成的现存的 `api.ics` 文件。

3. 输入以下命令，编译样本程序：

```
vacbld api.icc
```

注：`vacbld` 命令是由 VisualAge C++ 版本 4.0 提供的。

产生可执行文件 `client`。可输入可执行文件名来运行该程序：

```
client
```

嵌入式 SQL 应用程序

配置文件 `emb.icc`（在 AIX 上，位于 `sqllib/samples/c` 和 `sqllib/samples/cpp` 中；在 OS/2 和 Windows 32 位操作系统上，位于 `%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中）允许您用 C 和 C++ 构建 DB2 嵌入式 SQL 应用程序。

```
// emb.icc configuration file for embedded SQL applications
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export EMB=prog_name'
// To use on OS/2 and Windows, enter: 'set EMB=prog_name'
// Then compile the program by entering: 'vacbld emb.icc'

if defined( $EMB )
{
    prog_name = $EMB
}
else
{
    error "Environment Variable EMB is not defined."
}

// To connect to another database, replace "sample"
// For user ID and password, update 'user' and 'passwd'
// and take out the comment in the line: 'run before "embprep "'
dbname = "sample"
```



```

user = ""
passwd = ""

// Precompiling the source program file
run before "embprep " prog_name " " dbname // " " user " " passwd

infile = prog_name".c"
util = "utilemb.sqc"

if defined( $__TOS_AIX__ )
{
    // Set db2path to where DB2 will be accessed.
    // The default is the standard instance path.
    db2path = $HOME"/sql1lib"
    outfile = prog_name
    group lib = "libdb2.a"
    option opts = link( libsearchpath, db2path"/lib" ),
                  incl( searchPath, db2path"/include" )
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
    db2path = $DB2PATH
    outfile = prog_name".exe"
    group lib = "db2api.lib"
    option opts = link( libsearchpath, db2path"\\lib" ),
                  incl( searchPath, db2path"\\include" )
}

option opts
{
    target type(exe) outfile
    {
        source infile
        source util
        source lib
    }
}

```

VisualAge C++ 版本 4.0 定义下列其中一个环境变量，这取决于它安装在哪种操作系统上： `__TOS_AIX__`、`__TOS_OS2__`、`__TOS_WIN__`。

要根据源文件 `updat.sqc` 使用配置文件来构建嵌入式 SQL 应用程序 `updat`，执行以下操作：

1. 输入以下命令，将 `EMB` 环境变量设置为该程序名：

```
export EMB=updat
```
2. 如果在工作目录中有 `emb.ics` 文件，该文件是用 `emb.icc` 文件构建另一个程序生成的，可使用以下命令删除 `emb.ics` 文件：

```
rm emb.ics
```

不必删除对将要再次构建的同一程序生成的现存 `emb.ics` 文件。

3. 输入以下命令，编译样本程序:

```
vacbld emb.icc
```

注: vacbld 命令是由 VisualAge C++ 版本 4.0 提供的。

产生可执行文件 updat。可输入可执行文件名来运行该程序:

```
updat
```

嵌入式 SQL 存储过程

配置文件 stp.icc (在 AIX 上, 位于 sqllib/samples/c 和 sqllib/samples/cpp 中; 在 OS/2 和 Windows 32 位操作系统上, 位于 %DB2PATH%\samples\c 和 %DB2PATH%\samples\cpp 中) 允许您用 C 和 C++ 构建 DB2 嵌入式 SQL 存储过程。

```
// stp.icc configuration file for embedded SQL stored procedures
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export STP=prog_name'
// To use on OS/2 and Windows, enter: 'set STP=prog_name'
// Then compile the program by entering: 'vacbld emb.icc'

if defined( $STP )
{
    prog_name = $STP
}
else
{
    error "Environment Variable STP is not defined."
}

// To connect to another database, replace "sample"
// For user ID and password, update 'user' and 'passwd'
// and take out the comment in the line: 'run before "embprep "'
dbname = "sample"
user   = ""
passwd = ""

// Precompiling the source program file
run before "embprep " prog_name " " dbname // " " user " " passwd

infile  = prog_name".c"
outfile = prog_name".exp"

if defined( $__TOS_AIX__ )
{
    // Set db2path to where DB2 will be accessed.
    // The default is the standard instance path.
    db2path      = $HOME"/sqllib"
    outfile      = prog_name
    group lib    = "libdb2.a"
    option opts  = link( exportList, outfile ),
```

```

        link( libsearchpath, db2path"/lib" ),
        incl( searchPath, db2path"/include" )
    cpcmd      = "cp"
    funcdir    = db2path"/function"
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
    db2path    = $DB2PATH
    outfile    = prog_name".dll"
    if defined( $__TOS_WIN__ )
    {
        expfile = prog_name"v4.exp"
    }
    group lib  = "db2api.lib"
    option opts = link( exportList, expfile ),
                  link( libsearchpath, db2path"\\lib" ),
                  incl( searchPath, db2path"\\include" )
    cpcmd      = "copy"
    funcdir    = db2path"\\function"
}

option opts
{
    target type(dll) outfile
    {
        source infile
        source lib
    }
}

if defined( $__TOS_AIX__ )
{
    rmcmd      = "rm -f"
    run after rmcmd " " funcdir "/" outfile
}

run after cpcmd " " outfile " " funcdir

```

VisualAge C++ 版本 4.0 定义下列其中一个环境变量，这取决于它安装在哪种操作系统上：__TOS_AIX__、__TOS_OS2__、__TOS_WIN__。

要根据源文件 spserver.sqc 使用配置文件构建嵌入式 SQL 存储过程 spserver，执行以下操作：

1. 输入以下命令，将 STP 环境变量设置为程序名：

```
export STP=spserver
```

2. 如果在工作目录中有 stp.ics 文件，该文件是用 stp.icc 文件构建另一个程序生成的，可使用以下命令删除 stp.ics 文件：

```
rm stp.ics
```

不必删除对将要再次构建的同一程序生成的现存 stp.ics 文件。

3. 输入以下命令，编译样本程序：

```
vacbld stp.icc
```

注：vacbld 命令是由 VisualAge C++ 版本 4.0 提供的。

该存储过程被复制到服务器上的 `sllib/function` 路径（AIX 上）或 `%DB2PATH%\function` 路径（在 OS/2 和 Windows 32 位操作系统上）中。

接着在服务器上运行 `screate.db2` 脚本文件以编目存储过程。首先连接数据库：

```
db2 connect to sample
```

如果存储过程先前已经编目，可使用以下命令卸下它们：

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf screate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`，就可构建调用此存储过程的客户机应用程序 `spclient`。可使用配置文件 `emb.icc` 构建 `spclient`。有关详情，参考第128页的『嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可以是 `sample`、其远程别名或其他名称。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

配置文件 `udf.icc` (在 AIX 上, 位于 `sqllib/samples/c` 和 `sqllib/samples/cpp` 中; 在 OS/2 和 Windows 32 位操作系统上, 位于 `%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中) 允许您用 C 和或 C++ 构建用户定义函数。

```
// udf.icc configuration file for user-defined functions
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export UDF=prog_name'
// To use on OS/2 and Windows, enter: 'set UDF=prog_name'
// Then compile the program by entering: 'vacbld udf.icc'

if defined( $UDF )
{
    prog_name = $UDF
}
else
{
    error "Environment Variable UDF is not defined."
}

infile = prog_name".c"
expfile = prog_name".exp"

if defined( $__TOS_AIX__ )
{
    // Set db2path to where DB2 will be accessed.
    // The default is the standard instance path.
    db2path = $HOME"/sqllib"
    outfile = prog_name
    group lib = "libdb2.a", "libdb2apie.a"
    option opts = link( exportList, expfile ),
                  link( libsearchpath, db2path"/lib" ),
                  incl( searchPath, db2path"/include" )
    cpcmd = "cp"
    funcdir = db2path"/function"
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
    db2path = $DB2PATH
    outfile = prog_name".dll"
    if defined( $__TOS_WIN__ )
    {
        expfile = prog_name"v4.exp"
    }
    group lib = "db2api.lib", "db2apie.lib"
    option opts = link( exportList, expfile ),
                  link( libsearchpath, db2path"\\lib" ),
                  incl( searchPath, db2path"\\include" )
    cpcmd = "copy"
    funcdir = db2path"\\function"
}

option opts
```

```

{
  target type(dll) outfile
  {
    source infile
    source lib
  }
}
if defined( $__TOS_AIX__ )
{
  rmcmd      = "rm -f"
  run after rmcmd " " funcdir "/" outfile
}

run after cpcmd " " outfile " " funcdir

```

VisualAge C++ 版本 4.0 定义下列其中一个环境变量，这取决于它安装在哪种操作系统上：__TOS_AIX__、__TOS_OS2__、__TOS_WIN__。

要根据源文件 `udf.c` 使用配置文件构建用户定义函数程序 `udfsrv`，执行以下操作：

1. 输入以下命令，将 UDF 环境变量设置为程序名：

```
export UDF=udfsrv
```

2. 如果在工作目录中有 `udf.ics` 文件，该文件是用 `udf.icc` 文件构建另一个程序生成的，可使用以下命令删除 `udf.ics` 文件：

```
rm udf.ics
```

不必删除对将要再次构建的同一程序生成的现存 `udf.ics` 文件。

3. 输入以下命令，编译样本程序：

```
vacblid udf.icc
```

注：`vacblid` 命令是由 VisualAge C++ 版本 4.0 提供的。

UDF 库被复制到服务器的 `sqllib/function` 路径中。

必要时，将此用户定义函数设置为文件方式，以便 DB2 实例可运行它。

一旦构建了 `udfsrv`，就可构建调用它的客户机应用程序 `udfcli`。提供了此程序的 DB2 CLI 和嵌入式 SQL 版本。

可根据 `udfcli.c` 源文件（在 AIX 上，位于 `sqllib/samples/cli` 中；在 OS/2 和 Windows 32 位操作系统上，位于 `%DB2PATH%\samples\cli` 中），使用配置文件 `cli.icc` 构建 DB2 CLI `udfcli` 程序。有关详情，参考第121页的『DB2 CLI 应用程序』。

可根据 `udfcli.sqc` 源文件（在 AIX 上，位于 `sqllib/samples/c` 中；在 OS/2 和 Windows 32 位操作系统上，位于 `%DB2PATH%\samples\cli` 中），使用配置文件 `emb.icc` 构建嵌入式 SQL `udfcli` 程序。有关详情，参考第128页的『嵌入式 SQL 应用程序』。

要调用 UDF，可输入可执行文件名来运行样本调用应用程序：

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

IBM COBOL Set AIX 版

本节包括下列主题：

- 使用编译程序
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程

使用编译程序

如果开发包含嵌入式 SQL 和 DB2 API 调用的应用程序，且使用的是 IBM COBOL Set AIX 版编译程序，则切记以下几点：

- 当使用命令行处理器命令 `db2 prep` 预编译应用程序时，要使用 `target ibmcob` 选项。
- 在源文件中不要使用制表符。
- 在源文件的首行可使用 `PROCESS` 和 `CBL` 关键字，用以设置编译选项。
- 如果您的应用程序只包含嵌入式 SQL，而不包含 DB2 API 调用，则不需要使用 `pgmname(mixed)` 编译选项。如果使用 DB2 API 调用，则必须使用 `pgmname(mixed)` 编译选项。
- 如果使用的是 IBM COBOL Set AIX 版编译程序的“System/390 主机数据类型支持”功能部件，则您的应用程序的 DB2 包含文件位于如下目录中：

```
$HOME/sqllib/include/cobol_i
```

如果要使用提供的脚本文件构建 DB2 样本程序，必须将在脚本文件中指定的包含文件路径更改为指向 `cobol_i` 目录而不是 `cobol_a` 目录。

如果未使用 IBM COBOL Set AIX 版编译程序的“System/390 主机数据类型支持”功能部件，或使用的是此编译程序的较早版本，则您的应用程序的 DB2 包含文件位于如下目录中：

```
$HOME/sqllib/include/cobol_a
```

按如下所示，指定包括 .cbl 扩展名的 COPY 文件名：
COPY "sql.cbl"。

DB2 API 和嵌入式 SQL 应用程序

sqlllib/samples/cobol 中的构建文件 bldapp 包含构建 DB2 应用程序所需的命令。

第一个参数 \$1 指定源文件的名称。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- AIX
# Builds an IBM COBOL application program
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
    embprep $1 $2 $3 $4
fi

# Compile the checkerr.cbl error checking utility.
cob2 -qpgmname\(\mixed\) -qlib -I$DB2PATH/include/cobol_a \
    -c checkerr.cbl

# Compile the program.
cob2 -qpgmname\(\mixed\) -qlib -I$DB2PATH/include/cobol_a \
    -c $1.cbl

# Link the program.
cob2 -o $1 $1.o checkerr.o -ldb2 -L$DB2PATH/lib
```


bldapp 的编译和连接选项	
编译选项:	
cob2	IBM COBOL Set 编译程序。
-qpgmname\<i>(mixed\)</i>	指示编译程序可用混合大小写的名称调用 (CALL) 库入口点。
-qlib	指示编译程序处理 COPY 语句。
-I\$DB2PATH/include/cobol_a	指定 DB2 包含文件的位置。例如: \$HOME/sql1lib/include/cobol_a。
-c	只执行编译; 不连接。编译和连接是两个独立的步骤。
连接选项:	
cob2	使用编译程序作为连接程序的前端。
-o \$1	指定可执行程序。
\$1.o	指定该程序的目标文件。
checkerr.o	包括该实用程序的目标文件以便检查错误。
-ldb2	连接数据库管理程序库。
-L\$DB2PATH/lib	指定 DB2 运行期共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `client.cbl` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名：

```
updat
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：

```
updat database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
updat database userid password
```

嵌入式 SQL 存储过程

`sqllib/samples/cobol` 中的脚本文件 `bldsrv` 包含构建存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例上构建存储过程，所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

此脚本文件将源文件名 `$1` 用作共享库名和共享库的入口点。如果要构建入口点函数名与源文件名不同的存储过程，可修改此脚本文件以接受入口点的另一个参数。建议将数据库参数重命名为 `$3`。然后可将入口点连接选项更改为 `-e $2`，并在运行此脚本文件时在命令行上指定附加参数。

```
#!/bin/ksh
# bldsrv script file -- AIX
# Builds an IBM COBOL stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# Compile the checkerr.cbl error checking utility.
cob2 -pqgname\ (mixed\ ) -qlib -I$DB2PATH/include/cobol_a \
```

```

-c checkerr.cbl

# Compile the program.
cob2 -qpgmname\(mixed\) -qlib -c -I$DB2PATH/include/cobol_a $1.cbl

# Link the program using the export file $1.exp
# creating shared library $1 with entry point $1.
cob2 -o $1 $1.o checkerr.o -H512 -T512 -e $1 -bE:$1.exp \
-L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory of the DB2 instance.
# This assumes the user has write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldsrv 的编译和连接选项

编译选项:

cob2 IBM COBOL Set 编译程序。

-qpgmname\(mixed\)

指示编译程序可用混合大小写的名称调用 (CALL) 库入口点。

-qlib 指示编译程序处理 COPY 语句。

-c 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

-I\$DB2PATH/include/cobol_a

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include/cobol_a。

bldsrv 的编译和连接选项

连接选项:

cob2 使用编译程序来连接编辑。

-o \$1 指定输出为共享库文件。

\$1.o 指定存储过程的目标文件。

checkerr.o

包括该实用程序的目标文件以便检查错误。

-H512 指定输出文件对齐。

-T512 指定输出文件文本段起始地址。

-e \$1 指定共享库的缺省入口点。

-bE:\$1.exp

指定调出文件。该调出文件包含存储过程的列表。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sql/lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

-ldb2 连接数据库管理程序库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv outsrv
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv outsrv database
```

该脚本文件将存储过程复制到服务器的 `sql/lib/function` 路径中。

必要时, 将该存储过程设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了存储过程 `outsrv`, 就可构建调用此存储过程的客户机应用程序 `outcli`。可使用 `bldapp` 脚本文件构建 `outcli`。有关详情, 参考第136页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程, 可输入如下命令运行样本客户机应用程序:

```
outcli database userid password
```

其中,

database

是要连接的数据库的名称。该名称可以是 `sample`、其远程别名或其他名称。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `outsrv`，并在服务器数据库上执行同名的存储过程函数，然后将输出返回给客户机应用程序。

Micro Focus COBOL

本节包括下列主题：

- 使用编译程序
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程

使用编译程序

如果开发包含嵌入式 SQL 并调用 DB2 API 的应用程序，且使用的是 Micro Focus COBOL 编译程序，则切记以下几点：

- 当使用命令行处理器命令 `db2 prep` 预编译应用程序时，使用 `target mfcob` 选项（缺省项）。
- 为了使用内部预编译程序前端、运行期解释程序或动画制作程序调试程序，可执行由 Micro Focus 提供的 `mkrts` 命令，将“DB2 类属 API”入口点添加到 Micro Focus 运行期模块 `rts32` 中，如下所示：

1. 作为超级用户注册。
2. 带以下目录中提供的自变量执行 `mkrts` 命令：

```
/usr/lpp/db2_06_01/lib/db2mkrts.args
```

- 在 Micro Focus COBOL 环境变量 `COBCPY` 中必须包括 DB2 COBOL COPY 文件目录。`COBCPY` 环境变量指定 `COPY` 文件的位置。Micro Focus COBOL 的 DB2 COPY 文件驻留在该数据库实例目录下的 `sqllib/include/cobol_mf` 中。

要包括此目录，输入：

```
export COBCPY=$COBCPY:$HOME/sqllib/include/cobol_mf
```

注：应在 `.profile` 文件中设置 `COBCPY`。

DB2 API 和嵌入式 SQL 应用程序

sqlllib/samples/cobol_mf 中的构建文件 bldapp 包含构建 DB2 应用程序所需的命令。

第一个参数 \$1 指定源文件的名称。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- AIX
# Builds a Micro Focus COBOL application program
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
    embprep $1 $2 $3 $4
fi

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY

# Compile the checkerr.cbl error checking utility.
cob -c -x checkerr.cbl

# Compile the program.
cob -c -x $1.cbl

# Link the program.
cob -x -o $1 $1.o checkerr.o -ldb2 -ldb2gmf -L$DB2PATH/lib
```

bldmfapi 的编译和连接选项

编译选项:

- cob** COBOL 编译程序。
- c** 只执行编译；不连接。
- x** 产生可执行程序。

bldmfapi 的编译和连接选项

连接选项:

- cob** 使用编译程序作为连接程序的前端。
- x** 产生可执行程序。
- o \$1** 指定可执行程序。
- \$1.o** 指定该程序的目标文件。
- ldb2** 连接 DB2 库。
- ldb2gmf**
与 Micro Focus COBOL 的 DB2 异常处理程序库连接。
- L\$DB2PATH/lib**
指定 DB2 运行期共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.cbl` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
updat database userid password
```

嵌入式 SQL 存储过程

注：

1. 在 AIX 4.3 上使用 Micro Focus 4.1 编译程序构建存储过程之前，执行下列命令：

```
db2stop
db2set DB2LIBPATH=$LIBPATH
db2set DB2ENVLIST="COBDIR LIBPATH"
db2set
db2start
```

确保使用命令 `db2stop` 停止数据库，且在您的外壳环境中正确地设置了 `LIBPATH`。发出最后的 `db2set` 命令以显示您的设置：确保 `DB2LIBPATH` 和 `DB2ENVLIST` 设置正确。

2. 某些在 AIX 版本 4 平台上使用的最新版本的 Micro Focus COBOL 编译程序，不能用来创建静态连接的存储过程。因此，修改了 `makefile` 和脚本文件 `bldsrv`，以便可创建动态连接的存储过程。

为使远程客户机应用程序可以成功地调用此动态连接的存储过程，在执行该存储过程之前，需要在该存储过程所驻留的服务器上调用 Micro Focus COBOL 例程 `cobinit()`。完成此操作的封装器程序是在 `makefile` 或脚本文件 `bldsrv` 的执行期间创建的。然后将它与存储过程代码连接，以构成存储过程的共享库。因为使用此封装器程序，所以要使客户机应用程序调用名为 `x` 的存储过程，它必须调用 `x_wrap` 而不是 `x`。

本节后面会解释该封装器程序的详情。

`sqllib/samples/cobol_mf` 中的脚本文件 `bldsrv` 包含构建存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例上构建存储过程，所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

此脚本文件将源文件名 `$1` 用作共享库名和共享库的入口点。如果要构建入口点函数名与源文件名不同的存储过程，可修改此脚本文件以接受入口点的另一个参

数。建议将数据库参数重命名为 \$3。然后可将入口点连接选项更改为 -e \$2，并在运行此脚本文件时在命令行上指定附加参数。

```
#!/bin/ksh
# bldsrv script file -- AIX
# Builds a Micro Focus COBOL stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY

# Compile the program.
cob -c -x $1.cbl

# Create the wrapper program for the stored procedure.
wrapsrv $1

# Link the program using export file ${1}_wrap.exp
# creating shared library $1 with entry point ${1}_wrap.
cob -x -o $1 ${1}_wrap.c $1.o -Q -bE:${1}_wrap.exp -Q "-e $1" \
-Q -bI:$DB2PATH/lib/db2g.imp -ldb2gmf -L$DB2PATH/lib

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv 的编译和连接选项

编译选项:

- | | |
|------------|------------------------------|
| cob | COBOL 编译程序。 |
| -c | 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。 |
| -x | 产生可执行程序。 |

bldsrv 的编译和连接选项

连接选项:

- cob** 使用编译程序来连接编辑。
- x** 产生可执行程序。
- o \$1** 指定可执行程序。
- o \${1}_wrap.c**
指定封装器程序。
- \$1.o** 指定该程序的目标文件。
- Q -bE:\${1}_wrap.exp**
指定调出文件。该调出文件包含存储过程入口点的列表。如果一个存储过程称为 x，则它的入口点将是 x_wrap。
- Q "-e \$1"**
指定共享库的缺省入口点。
- Q -bI:\$DB2PATH/lib/db2g.imp**
提供至 DB2 应用程序库的入口点的列表。
- ldb2gmf**
与 Micro Focus COBOL 的 DB2 异常处理程序库连接。
- L\$DB2PATH/lib**
指定 DB2 运行期共享库的位置。例如: \$HOME/sql/lib/lib。如果不指定 -L 选项，则编译程序假定如下路径: /usr/lib:/lib。

有关其他编译程序选项，参考编译程序文档。

封装器程序 `wrapsrv` 可使 Micro Focus COBOL 例程 `cobinit()` 刚好在执行该存储过程之前被调用。以下显示的是它的内容。

```

#!/bin/ksh
# wrapsrv script file
# Creates the wrapper program for Micro Focus COBOL stored procedures
# Usage: wrapsrv <stored_proc>

# Note: The client program calls "<stored_proc>_wrap" not "<stored_proc>"

# Create the wrapper program for the stored procedure.
cat << WRAPPER_CODE > ${1}_wrap.c
#include <stdio.h>
void cobinit(void);
int $1(void *p0, void *p1, void *p2, void *p3);

int main(void)
{
    return 0;
}

int ${1}_wrap(void *p0, void *p1, void *p2, void *p3)
{
    cobinit();
    return $1(p0, p1, p2, p3);
}
WRAPPER_CODE
# Create the export file for the wrapper program
echo $1_wrap > ${1}_wrap.exp

```

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`，如果连接 `sample` 数据库，则输入：

```
bldsrv outsrv
```

如果连接另一个数据库，还须输入该数据库名：

```
bldsrv outsrv database
```

此脚本文件将共享库复制到服务器的 `sqllib/function` 路径中。

必要时，将该共享库设置为文件方式，以便客户机应用程序可存取它。

一旦构建了存储过程 `outsrv`，就可构建调用它的客户机应用程序 `outcli`。可使用 `bldapp` 脚本文件构建 `outcli`。有关详情，参考第142页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
outcli database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `outsrv`，并在服务器数据库上执行同名的存储过程函数。然后将输出返回给该客户机应用程序。

退出存储过程

当您开发存储过程时，可使用如下语句退出存储过程：

```
move SQLZ-HOLD-PROC to return-code
```

借助于此语句，存储过程将正确地返回到客户机应用程序。当本地 COBOL 客户机应用程序调用该存储过程时，这一点尤其重要。

REXX

不要预编译或联编 REXX 程序。

要在 AIX 上运行 DB2 REXX/SQL 程序，必须设置 LIBPATH 环境变量，以包括 DB2 安装目录下的 `sqllib/lib`。

输入：

```
export LIBPATH=$LIBPATH:/lib:/usr/lib:/usr/lpp/db2_07_01/sqllib/lib
```

在 AIX 上，您的应用程序文件可有任何文件扩展名。可使用下列两个方法中的一种运行您的应用程序：

1. 在 shell 命令提示符处，输入 `rexx name`，其中 `name` 是 REXX 程序的名称。
2. 如果 REXX 程序的首行包含“特殊符号”（#!），且标识出 REXX/6000 解释程序所在的目录，则可在 shell 命令提示符处输入 REXX 程序名，以运行它。例如，如果 REXX/6000 解释程序文件在 `/usr/bin` 目录中，则将如下行作为 REXX 程序的首行：

```
#!/usr/bin/rexx
```

然后，在 shell 命令提示符处输入如下命令，使该程序成为可执行程序：

```
chmod +x name
```

在 shell 命令提示符处输入 REXX 程序的文件名，以运行它。

REXX 样本程序在目录 `sqllib/samples/rexx` 中。要运行样本 REXX 程序 `updat.cmd`，执行下列操作之一：

- 将 `#!/usr/bin/rexx` 一行添加至程序源文件的顶部（如果该行不存在的话）。然后，输入以下命令直接运行此程序：

```
updat.cmd
```

- 输入以下命令，指定 REXX 解释程序和程序：

```
rexx updat.cmd
```

有关 REXX 和 DB2 的详情，参考 *Application Development Guide* 中的“Programming in REXX”一章。

第7章 构建 HP-UX 应用程序

HP-UX C	152	DB2 API 和嵌入式 SQL 应用程序	165
DB2 CLI 应用程序	152	构建和运行嵌入式 SQL 应用程序	167
构建和运行嵌入式 SQL 应用程序	153	嵌入式 SQL 存储过程	167
使用 DB2 API 的 DB2 CLI 应用程序	154	用户定义函数 (UDF)	170
DB2 CLI 存储过程	154	多线程应用程序	172
DB2 API 和嵌入式 SQL 应用程序	157	Micro Focus COBOL	173
构建和运行嵌入式 SQL 应用程序	159	使用编译程序	173
嵌入式 SQL 存储过程	159	DB2 API 和嵌入式 SQL 应用程序	174
用户定义函数 (UDF)	161	构建和运行嵌入式 SQL 应用程序	175
多线程应用程序	163	嵌入式 SQL 存储过程	176
HP-UX C++	165	退出存储过程	178

本章提供在 HP-UX 上构建 DB2 应用程序的详细信息。在脚本文件中，以 db2 开始的命令是“命令行处理器”（CLP）命令。有关 CLP 命令的详情，可参考 *Command Reference*。

有关 HP-UX 的最新 DB2 应用程序开发情况的更新，请访问 DB2 应用程序开发 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

注：

1. 在 DB2 构建文件和 makefile 的编译和连接步骤中使用 +DAportable 选项。此选项生成在 PA_RISC 1.1 和 2.0 工作站上兼容的代码。使用此选项可能稍微降低性能。要提高性能，可从位于 sqllib/samples 目录中的构建文件和 makefile 中除去 +DAportable 选项。如果不使用此选项，当构建 HP-UX 程序时可能收到类似如下的警告：

（警告）检测到至少一个 PA 2.0 目标文件 (<filename>.o)。
连接的目标不能在 PA 1.x 系统上运行。

其中 <filename> 是您正在编译的程序文件。

除非您使用的是 PA_RISC 1.1 或 2.0 系统，否则此警告不适用。

2. 如果您要将 DB2 从 HP-UX 版本 10 或更早版本迁移至 HP-UX 版本 11，则您的 DB2 程序必须用 HP-UX 版本 11 上的 DB2 重新预编译（如果这些程序包括嵌入式 SQL），且必须重新编译。这包括所有 DB2 应用程序、存储过程、用户定义函数和用户出口程序。另外，在 HP-UX 版本 11 上编译的 DB2 程序可能无法在 HP-UX 版本 10 或更早版本上运行。在 HP-UX 版本 10 上编译和运行的 DB2 程序可以与 HP-UX 版本 11 服务器远程连接。

本节包括下列主题:

- DB2 CLI 应用程序
- DB2 CLI 存储过程
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 CLI 应用程序

sqllib/samples/cli 中的脚本文件 bldcli 包含构建 DB2 CLI 程序所需的命令。参数 \$1 指定源文件的名称。

这是唯一的必需参数，且是不包含嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

如果该程序包含嵌入式 SQL（根据扩展名 .sql 来判断），则调用脚本 embprep 来预编译该程序，生成一个扩展名为 .c 的程序文件。

```
#!/bin/ksh
# bldcli script file -- HP-UX
# Builds a CLI program with HP-UX C.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sql" ]]
then
    embprep $1 $2 $3 $4
fi

# Compile the error-checking utility.
cc +DAportable -Aa +e -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc +DAportable -Aa +e -I$DB2PATH/include -c $1.c

# Link the program.
cc +DAportable -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2
```


blcli 的编译和连接选项

编译选项:

cc 使用 C 编译程序。

+DAportable

生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。

-Aa 使用 ANSI 标准方式。

+e 当用 ANSI C 方式编译时启用 HP 增值功能部件。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sql1lib/include

-c 只执行编译; 不连接。编译和连接是两个独立的步骤。

连接选项:

cc 使用编译程序作为连接程序的前端。

+DAportable

使用在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。

-o \$1 指定可执行程序。

-o \$1.o

指定目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。 例如: \$HOME/sql1lib/lib

-ldb2 连接数据库管理程序库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`, 输入:

```
blcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名, 运行该可执行文件:

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
blcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接，还须输入该数据库名：

```
bldcli dbusemx database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldcli dbusemx database userid password
```

产生可执行文件 `dbusemx`。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名：

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

`sqllib/samples/cli` 中的脚本文件 `bldapi` 包含使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 `utilapi` 实用程序文件（该文件包含创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 `bldcli` 脚本文件之间唯一的区别。有关 `bldapi` 和 `bldcli` 之间公共的编译和连接选项，请参阅第152页的『DB2 CLI 应用程序』。

要根据源文件 `dbmconn.c` 构建样本程序 `dbmconn`，输入：

```
bldapi dbmconn
```

产生可执行文件 `dbmconn`。可输入可执行文件名，运行该可执行文件：

```
dbmconn
```

DB2 CLI 存储过程

`sqllib/samples/cli` 中的脚本文件 `bldclisp` 包含构建 DB2 CLI 存储过程所需的命令。参数 `$1` 指定源文件的名称。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldclisp script file -- HP-UX
# Builds a CLI stored procedure in HP-UX C.
# Usage: bldclisp <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the error-checking utility.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c $1.c

# Link the program.
ld -b -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp 的编译和连接选项

编译选项:

- cc** C 编译程序。
- +DAportable**
生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
- +u1** 允许存取未对齐的数据。仅在应用程序使用未对齐的数据时使用。
- +z** 生成与位置无关的代码。
- Aa** 使用 ANSI 标准方式（只用于 C 编译程序）。
- +e** 当用 ANSI C 方式编译时启用 HP 增值功能部件。
- I\$DB2PATH/include**
指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include
- c** 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。

bldclisp 的编译和连接选项

连接选项:

- ld** 使用连接程序连接编辑。
- b** 创建一个共享库而不是通常的可执行文件。
- o \$1** 指定可执行文件。
- \$1.o** 指定目标文件。
- L\$DB2PATH/lib**
指定 DB2 运行期共享库的位置。例如: `-L$HOME/sqlllib/lib`。如果不指定 `-L` 选项, 则假定 `/usr/lib:/lib`。
- ldb2** 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.c` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入:

```
bldclisp spserver
```

此脚本文件将共享库复制到服务器的 `sqlllib/function` 路径中。

接着在服务器上运行 `spscreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`, 就可构建存取该共享库的 CLI 客户机应用程序 `spclient`。

可使用脚本文件 `bldcli` 构建 `spclient`。有关详情, 参考第152页的『DB2 CLI 应用程序』。

要存取该共享库, 可输入以下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数，然后将输出返回给客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/c` 中的脚本文件 `bldapp` 包含构建 DB2 应用程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是唯一的必需参数，且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定数据库的用户 ID；第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序，`bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名，则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- HP-UX
# Builds a C application program
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sql" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.c error-checking utility.
    cc +DAportable -Aa +e -I$DB2PATH/include -c utilemb.c
else
    # Compile the utilapi.c error-checking utility.
    cc +DAportable -Aa +e -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
cc +DAportable -Aa +e -I$DB2PATH/include -c $1.c

if [[ -f $1".sql" ]]
then
```

```

# Link the program with utilemb.o
cc +DAportable -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2
else
# Link the program with utilapi.o
cc +DAportable -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2
fi

```

bldapp 的编译和连接选项	
编译选项:	
cc	C 编译程序。
+DAportable	生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
-Aa	使用 ANSI 标准方式 (只用于 C 编译程序)。
+e	当用 ANSI C 方式编译时启用 HP 增值功能部件。
-I\$DB2PATH/include	指定 DB2 包含文件的位置。例如: <code>-I\$DB2PATH/include</code>
-c	只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
连接选项:	
cc	使用编译程序来连接编辑。
+DAportable	使用在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
-o \$1	指定可执行文件。
\$1.o	指定该程序的目标文件。
utilemb.o	如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。
utilapi.o	如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。
-L\$DB2PATH/lib	指定 DB2 运行期共享库的位置。例如: <code>-L\$DB2PATH/lib</code> 。如果不指定 <code>-L</code> 选项, 则假定 <code>/usr/lib:/lib</code> 。
-ldb2	连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `client.c` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。

要运行可执行文件, 输入可执行文件名:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

`sqllib/samples/c` 中的脚本文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要指定用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldsrv script file -- HP-UX
# Builds a C stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
```

```

embprep $1 $2

# Compile the program.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c $1.c

# Link the program to create a shared library
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldsrv 的编译和连接选项

编译选项:

cc C 编译程序。

+DAportable 生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。

+u1 允许存取未对齐的数据。仅在应用程序使用未对齐的数据时使用。

-Aa 使用 ANSI 标准方式（只用于 C 编译程序）。

+z 生成与位置无关的代码。

+e 当用 ANSI C 方式编译时启用 HP 增值功能部件。

-I\$DB2PATH/include 指定 DB2 包含文件的位置。例如: `-I$DB2PATH/include`。

-c 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。

连接选项:

ld 使用连接程序连接编辑。

-b 创建一个共享库而不是通常的可执行文件。

-o \$1 指定输出为共享库文件。

\$1.o 指定该程序的目标文件。

-L\$DB2PATH/lib 指定 DB2 运行期共享库的位置。例如: `$HOME/sqllib/lib`。如果不指定 `-L` 选项，则假定 `/usr/lib:/lib`。

-ldb2 连接 DB2 库。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`，如果连接 `sample` 数据库，则输入：

```
bldsrv spserver
```

如果连接另一个数据库，还须输入该数据库名：

```
bldsrv spserver database
```


此脚本文件将共享库复制到服务器的 `sqllib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`, 就可构建存取它的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情, 参考第157页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用共享库中的存储过程, 可输入以下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

`sqllib/samples/c` 中的脚本文件 `bldudf` 包含构建 UDF 所需的命令。UDF 的编译与存储过程相似。UDF 不能包含嵌入式 SQL 语句。这意味着要构建 UDF 程序, 不必连接数据库来预编译和联编该程序。

参数 \$1 指定源文件的名称。此脚本文件将源文件名用作共享库名。

```
#!/bin/ksh
# bldudf script file -- HP-UX
# Builds a C UDF library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlllib

# Compile the program.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqlllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf 的编译和连接选项

编译选项:

- cc** C 编译程序。
- +DAportable** 生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
- +u1** 允许存取未对齐的数据。仅在应用程序使用未对齐的数据时使用。
- Aa** 使用 ANSI 标准方式（只用于 C 编译程序）。
- +z** 生成与位置无关的代码。
- +e** 当用 ANSI C 方式编译时启用 HP 增值功能部件。
- I\$DB2PATH/include** 指定 DB2 包含文件的位置。例如: \$HOME/sqlllib/include。
- c** 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。

bldudf 的编译和连接选项

连接选项:

- ld** 使用连接程序连接编辑。
- b** 创建一个共享库而不是通常的可执行文件。
- o \$1** 指定输出为共享库文件。
- \$1.o** 指定该程序的目标文件。
- L\$DB2PATH/lib**
指定 DB2 运行期共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。
- ldb2** 连接 DB2 库。
- ldb2apie**
连接 “DB2 API 引擎” 库, 以便可使用 LOB 定位器。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到 `sqllib/function` 目录中。

必要时, 将该 UDF 设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。提供了此程序的 DB2 CLI 和嵌入式 SQL 版本。

可根据 `sqllib/samples/cli` 中的源文件 `udfcli.c`, 使用脚本文件 `bldcli` 构建 DB2 CLI `udfcli` 程序。有关详情, 参考第152页的『DB2 CLI 应用程序』。

可根据 `sqllib/samples/c` 中的源文件 `udfcli.sqc`, 使用脚本文件 `bldapp` 构建嵌入式 SQL `udfcli` 程序。有关详情, 参考第157页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF, 可输入可执行文件名运行样本调用应用程序:

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

注: HP-UX 提供 POSIX 线程库和 DCE 线程库。DB2 支持使用 POSIX 线程库的多线程应用程序。

在 HP-UX 上构建多线程应用程序需要定义 `_REENTRANT`，以便编译它们。HP-UX 文档建议使用 `-D_POSIX_C_SOURCE=199506L` 来编译。这也将确保定义 `_REENTRANT`。应用程序还需要使用 `-lpthread` 来连接。

`sqllib/samples/c` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID，第四个参数 `$4` 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldmt script file -- HP-UX
# Builds a C multi-threaded embedded SQL program
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# Compile the program.
cc +DAportable -Aa +e -I$DB2PATH/include -D_POSIX_C_SOURCE=199506L -c $1.c

# Link the program
cc +DAportable -o $1 $1.o -L$DB2PATH/lib -ldb2 -lpthread
```

除了以上讨论的 `-D_POSIX_C_SOURCE=199506L` 编译选项和 `-lpthread` 连接选项外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与 `bldapp` 文件所用的相同。有关这些选项的信息，参阅第157页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据 `thdsrver.sqc` 源文件构建样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入该可执行文件名称：

```
thdsrver
```

本节包括下列主题:

- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 API 和嵌入式 SQL 应用程序

sqllib/samples/cpp 中的脚本文件 bldapp 包含构建 DB2 应用程序所需的命令。

第一个参数 \$1 指定源文件的名称。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库, 因此还须提供三个可选参数: 第二个参数 \$2 指定您想连接的数据库的名称; 第三个参数 \$3 指定数据库的用户 ID; 第四个参数 \$4 指定口令。

对于嵌入式 SQL 程序, bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名, 则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时, 才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- HP-UX
# Builds a C++ application program
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.C error-checking utility.
    CC +DAportable +a1 -ext -I$DB2PATH/include -c utilemb.C
else
    # Compile the utilapi.C error-checking utility.
    CC +DAportable +a1 -ext -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
CC +DAportable +a1 -ext -I$DB2PATH/include -c $1.C

if [[ -f $1".sqC" ]]
then
    # Link the program with utilemb.o
    CC +DAportable -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2
```

```

else
  # Link the program with utilapi.o
  CC +DAportable -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2
fi

```

bldapp 的编译和连接选项

编译选项:

- CC** C 编译程序。
- +DAportable**
生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
- +a1** 指示该编译程序使用 ANSI C/C++。
- ext** 允许各种 C++ 扩展名, 包括“超长”支持。
- I\$DB2PATH/include**
指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include
- c** 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

连接选项:

- CC** 使用编译程序作为连接程序的前端。
- +DAportable**
使用在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
- o \$1** 指定可执行文件。
- \$1.o** 指定该程序的目标文件。
- utilemb.o**
如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。
- utilapi.o**
如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。
- L\$DB2PATH/lib**
指定 DB2 运行期共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。
- ldb2** 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.C` 构建非嵌入式 SQL DB2 API 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

注: 请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ 存储过程的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

此脚本文件将源文件名 `$1` 用作共享库名。

```
#!/bin/ksh
# bldsrv script file -- HP-UX
# Builds a C++ stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]
```

```

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlllib

# Precompile and bind the program.
embprep $1 $2

# Compile the program. First ensure it is coded with extern "C".
CC +DAportable +a1 +z -ext -I$DB2PATH/include -c $1.C

# Link the program to create a shared library.
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqlllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldsrv 的编译和连接选项

编译选项:

CC C++ 编译程序。

+DAportable 生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。

+a1 指示该编译程序使用 ANSI C/C++。

+z 生成与位置无关的代码。

-ext 允许各种 C++ 扩展名, 包括“超长”支持。

-I\$DB2PATH/include 指定 DB2 包含文件的位置。例如: \$DB2PATH/include

-c 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

连接选项:

ld 使用连接程序连接编辑。

-b 创建一个共享库而不是通常的可执行文件。

-o \$1 指定可执行文件。

\$1.o 指定该程序的目标文件。

-L\$DB2PATH/lib 指定 DB2 运行期共享库的位置。例如: -L\$DB2PATH/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 spserver.sqc 构建样本程序 spserver, 如果连接 sample 数据库, 则输入:

```
bldsrv spserver
```


如果连接另一个数据库，还须输入该数据库名：

```
bldsrv spserver database
```

此脚本文件将共享库复制到服务器的 `sqllib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库：

```
db2 connect to sample
```

如果存储过程先前已经编目，可使用以下命令卸下它们：

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`，就可构建调用共享库中的存储过程的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第165页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用共享库中的存储过程，可输入以下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数。存储过程将输出返回给客户机应用程序。

用户定义函数 (UDF)

注: 请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ UDF 的信息。

sqllib/samples/c 中的脚本文件 bldudf 包含构建 UDF 所需的命令。用户定义程序不能包含嵌入式 SQL 语句。这意味着要构建 UDF 程序, 不必连接数据库来预编译和联编该程序。

参数 \$1 指定源文件名。此脚本文件将源文件名用作共享库名。

```
#!/bin/ksh
# bldudf script file -- HP-UX
# Builds a C or C++ UDF library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the program.
if [[ -f $1".c" ]]
then
    CC +DAportable -ext +a1 +z -I$DB2PATH/include -c $1.c
elif [[ -f $1".C" ]]
then
    CC +DAportable -ext +a1 +z -I$DB2PATH/include -c $1.C
fi

# Link the program.
CC +DAportable -b -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf 的编译和连接选项

编译选项:

- CC** C++ 编译程序。
- +DAportable**
生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
- ext** 允许各种 C++ 扩展名, 包括“超长”支持。
- +a1** 指示该编译程序使用 ANSI C/C++。
- +z** 生成与位置无关的代码。
- I\$DB2PATH/include**
指定 DB2 包含文件的位置。例如: \$DB2PATH/include
- c** 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

连接选项:

- CC** 使用编译程序作为连接程序的前端。
- b** 创建一个共享库而不是通常的可执行文件。
- o \$1** 指定可执行文件。
- \$1.o** 指定该程序的目标文件。
- L\$DB2PATH/lib**
指定 DB2 运行期共享库的位置。例如: -L\$DB2PATH/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。
- ldb2** 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到 `sqllib/function` 目录中。

必要时, 将该 UDF 设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。可根据 `sqllib/samples/cpp` 中的 `udfcli.sqc` 源文件, 使用脚本文件 `bldapp` 来构建 `udfcli` 程序。有关详情, 参考第165页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF, 可输入可执行文件名运行样本调用应用程序:

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

注: HP-UX 提供 POSIX 线程库和 DCE 线程库。HP-UX 上的 DB2 支持使用 POSIX 线程库的多线程应用程序。

在 HP-UX 上构建多线程应用程序需要定义 `_REENTRANT`，以便编译它们。HP-UX 文档建议使用 `-D_POSIX_C_SOURCE=199506L` 来编译。这也将确保定义了 `_REENTRANT`。对于 HP-UX C++ 编译程序，还必须使用 `-D_HPUX_SOURCE` 来定义 `rand_r`。且需使用 `-lpthread` 连接应用程序。

`sqllib/samples/cpp` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID，而第四个参数 `$4` 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldmt script file -- HP-UX
# Builds a C++ multi-threaded embedded SQL program
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# Compile the program.
CC +DAportable +a1 -ext -I$DB2PATH/include \
    -D_HPUX_SOURCE -D_POSIX_C_SOURCE=199506L -c $1.C

# Link the program
CC -o $1 $1.o -L$DB2PATH/lib -ldb2 -lpthread
```

除了以上讨论的 `-D_HPUX_SOURCE` 和 `-D_POSIX_C_SOURCE=199506L` 编译选项以及 `-lpthread` 连接选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第165页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sqC` 构建样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入该可执行文件名称：

Micro Focus COBOL

本节包括下列主题:

- 使用编译程序
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程

使用编译程序

如果开发包含嵌入式 SQL 和 DB2 API 调用的应用程序，且使用的是 Micro Focus COBOL 编译程序，则切记以下几点:

- 当使用命令行处理器命令 `db2 prep` 预编译应用程序时，使用 `target mfcob` 选项（缺省项）。
- 为了使用内部预编译程序前端、运行期解释程序或动画制作程序调试程序，可执行由 Micro Focus 提供的 `mrts` 命令，将“DB2 类属 API”入口点添加到 Micro Focus 运行期模块 `rts32` 中。您还需运行 `mkcheck` 更新 `check` 文件。如果没有运行此文件，您将在 `SQLGSTRT` 中收到代码为 173 的错误。

在运行 `mrts` 和 `mkcheck` 之前，必须在下列步骤中设置 `COBOPT`:

1. 作为超级用户注册。
2. 在目录 `$COBDIR/src/rts` 下输入:

```
COBOPT=/opt/IBMDB2/V7.1/lib/db2mrts.args; export COBOPT
ksh mrts
mv $COBDIR/rts32 $COBDIR/rts32.orig
cp rts32 $COBDIR/rts32
```

3. 还必须重建 Hewlett-Packard 在交付本产品时提供的 `check` 可执行文件。如果未重建位于 `$COBDIR` 目录中的 `check` 可执行文件，则使用 `cob -C SQL` 所进行的编译尝试将失败，并收到一个运行期系统 173 错误，这是因为 DB2 预处理程序要调用 DB2 库。要重建 `check` 文件，应以超级用户的身份切换到 `$COBDIR` 目录下的 `src/sql` 目录中，然后运行 `mkcheck` 脚本。该脚本运行完毕后，需将生成的 `check` 可执行文件移动到 `$COBDIR` 目录下。在 `$COBDIR/src/sql` 目录下输入:

```
COBOPT=/opt/IBMDB2/V7.1/lib/db2mrts.args; export COBOPT
ksh mkcheck
mv $COBDIR/check $COBDIR/check.orig
cp check $COBDIR/check
```

现在，可带以下目录中提供的自变量执行 `mrts` 命令:

```
/opt/IBMDB2/V7.1/lib/db2mrts.args
```

- 在 Micro Focus COBOL 环境变量 COBCPY 中必须包括 DB2 COBOL COPY 文件目录。COBCPY 环境变量指定 COPY 文件的位置。Micro Focus COBOL 的 DB2 COPY 文件驻留在该数据库实例目录下的 sqllib/include/cobol_mf 中。

要包含此目录，输入：

```
export COBCPY=$COBCPY:/opt/IBMDB2/V7.1/include/cobol_mf
```

注：应在 .profile 文件中设置 COBCPY。

DB2 API 和嵌入式 SQL 应用程序

sqllib/samples/cobol_mf 中的脚本文件 bldapp 包含构建 DB2 API 和嵌入式 SQL 应用程序所需的命令。

第一个参数 \$1 指定源文件的名称。这是唯一的必需参数，且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- HP-UX
# Builds a Micro Focus COBOL application program
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
    embprep $1 $2 $3 $4
fi

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# Compile the checkerr.cbl error checking utility.
cob +DAportable -cx checkerr.cbl

# Compile the program.
cob +DAportable -cx $1.cbl
```

```
# Link the program.
cob +DAportable -x $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

bldapp 的编译和连接选项	
编译选项:	
cob	Micro Focus COBOL 编译程序。
+DAportable	生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
-cx	编译为目标模块。
连接选项:	
cob	使用编译程序作为连接程序的前端。
+DAportable	使用在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
-x	指定可执行程序。
\$1.o	包括该程序的目标文件。
checkerr.o	包括该实用程序的目标文件以便检查错误。
-L\$DB2PATH/lib	指定 DB2 运行期共享库的位置。例如: \$HOME/sql/lib/lib。
-ldb2	连接 DB2 库。
-ldb2gmf	与 Micro Focus COBOL 的 DB2 异常处理程序库连接。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `client.cb1` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

`sqllib/samples/cobol_mf` 中的脚本文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为服务器上可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例上构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。此脚本文件将源文件名 `$1` 用作共享库名。

```
#!/bin/ksh
# bldsrv script file -- HP-UX
# Builds a Micro Focus COBOL stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# Compile the program.
cob +DAportable +z -cx $1.cbl

# Link the program.
```



```
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2gmf \
    -L$COBDIR/coblib -lcobol -lcrtn

# Copy the shared library to the sqllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv 的编译和连接选项	
编译选项:	
cob	COBOL 编译程序。
+DAportable	生成在 PA_RISC 1.x 和 2.0 工作站上兼容的代码。
+z	生成与位置无关的代码。
-cx	编译为目标模块。
连接选项:	
ld	使用连接程序连接编辑。
-b	创建一个共享库而不是通常的可执行文件。
-o \$1	指定可执行文件。
\$1.o	包括该程序的目标文件。
-L\$DB2PATH/lib	指定 DB2 运行期共享库的位置。例如: \$HOME/sqllib/lib
-ldb2	与 DB2 共享库连接。
-ldb2gmf	与 Micro Focus COBOL 的 DB2 异常处理程序库连接。
-L\$COBDIR/coblib	指定 COBOL 运行期库的位置。
-lcobol	连接 COBOL 库。
-lcrtn	连接 crtn 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv outsrv
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv outsrv database
```

此脚本文件将存储过程复制到 `sqllib/function` 目录中。

必要时, 将该存储过程设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了存储过程 `outsrv`，就可构建调用此存储过程的客户机应用程序 `outcli`。可使用 `bldapp` 脚本文件构建 `outcli`。有关详情，参考第174页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
outcli database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 `outsrv`，并在服务器数据库上执行同名的存储过程函数，然后将输出返回给客户机应用程序。

退出存储过程

当开发存储过程时，可使用如下语句退出存储过程：

```
move SQLZ-HOLD-PROC to return-code.
```

借助于此语句，存储过程将正确地返回到客户机应用程序。

第8章 构建 Linux 应用程序

Linux C	179	用户定义函数 (UDF)	189
DB2 CLI 应用程序	179	多线程应用程序	191
构建和运行嵌入式 SQL 应用程序	181	Linux C++	192
使用 DB2 API 的 DB2 CLI 应用程序	182	DB2 API 和嵌入式 SQL 应用程序	192
DB2 CLI 存储过程	182	构建和运行嵌入式 SQL 应用程序	194
DB2 API 和嵌入式 SQL 应用程序	184	嵌入式 SQL 存储过程	195
构建和运行嵌入式 SQL 应用程序	186	用户定义函数 (UDF)	197
嵌入式 SQL 存储过程	187	多线程应用程序	199

本章提供在 Linux 上构建应用程序的详细信息。在脚本文件中，以 db2 开始的命令是“命令行处理器”(CLP)命令。有关 CLP 命令的详情，参考 *Command Reference*。

有关 Linux 的最新 DB2 应用程序开发情况的更新信息，请访问以下的 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

Linux C

本节包括下列主题：

- DB2 CLI 应用程序
- DB2 CLI 存储过程
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 CLI 应用程序

sqllib/samples/cli 中的脚本文件 bldcli 包含构建 DB2 CLI 程序所需的命令。参数 \$1 指定源文件的名称。

这是唯一的必需参数，且是不包括嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

如果该程序包括嵌入式 SQL（根据扩展名 .sql 来判断），则调用脚本 embprep 来预编译该程序，生成一个扩展名为 .c 的程序文件。

```
#!/bin/ksh
# bldcli script file -- Linux
# Builds a CLI program with Linux C
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sql" ]]
then
    embprep $1 $2 $3 $4
fi

# Compile the error-checking utility.
cc -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -I$DB2PATH/include -c $1.c

# Link the program.
cc -o $1 $1.o utilcli.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

bldcli 的编译和连接选项

编译选项:

cc C 编译程序。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldcli 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-Wl,-rpath,\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sql1lib/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`, 输入:

```
bldcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名, 运行该可执行文件:

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldcli dbusemx database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldcli dbusemx database userid password
```

产生可执行文件 `dbusemx`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

sqllib/samples/cli 中的脚本文件 bldapi 包括使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 utilapi 实用程序文件（该文件包括创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 bldcli 脚本文件之间唯一的区别。有关 bldapi 和 bldcli 之间公共的编译和连接选项，请参阅第179页的『DB2 CLI 应用程序』。

要根据源文件 dbmconn.c 构建样本程序 dbmconn，输入：

```
bldapi dbmconn
```

产生可执行文件 dbmconn。可输入可执行文件名，运行该可执行文件：

```
dbmconn
```

DB2 CLI 存储过程

sqllib/samples/cli 中的脚本文件 bldclisp 包含构建 DB2 CLI 存储过程所需的命令。参数 \$1 指定源文件的名称。

```
#!/bin/ksh
# bldclisp script file -- Linux
# Builds a CLI stored procedure in Linux C.
# Usage: bldclisp <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the error-checking utility.
cc -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -I$DB2PATH/include -c $1.c

# Link the program.
cc -o $1 $1.o utilcli.o -shared -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

```
# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp 的编译和连接选项	
编译选项:	
cc	C 编译程序。
-I\$DB2PATH/include	指定 DB2 包含文件的位置。例如: \$HOME/sqlllib/include。
-c	只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。
连接选项:	
cc	使用编译程序作为连接程序的前端。
-o \$1	指定可执行文件。
\$1.o	包括该程序的目标文件。
utilcli.o	包括该实用程序的目标文件以便检查错误。
-shared	生成共享库。
-L\$DB2PATH/lib	指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。
-Wl,-rpath,\$DB2PATH/lib	指定运行时 DB2 共享库的位置。例如: \$HOME/sqlllib/lib。
-ldb2	连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `spserver.c` 构建样本程序 `spserver`, 输入:

```
bldclisp spserver
```

此脚本文件将共享库复制到 `sqlllib/function` 目录中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`, 就可构建存取该共享库的 CLI 客户机应用程序 `spclient`。

可使用脚本文件 `bldcli` 构建 `spclient`。有关详情, 参考第179页的『DB2 CLI 应用程序』。

要存取该共享库, 可输入以下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/c` 中的脚本文件 `bldapp` 包括构建 DB2 应用程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是唯一的必需参数, 且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库, 因此还须提供三个可选参数: 第二个参数 `$2` 指定您想连接的数据库的名称; 第三个参数 `$3` 指定数据库的用户 ID; 第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序, `bldapp` 将这些参数传送给预编译和联编脚本文件 `embprep`。如果未提供数据库名, 则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时, 才需要用户 ID 和口令参数。


```

#!/bin/ksh
# bldapp script file -- Linux
# Builds a C application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.c error-checking utility.
    cc -I$DB2PATH/include -c utilemb.c
else
    # Compile the utilapi.c error-checking utility.
    cc -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
cc -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
    # Link the program with utilemb.o.
    cc -o $1 $1.o utilemb.o -L$DB2PATH/lib \
        -Wl,-rpath,$DB2PATH/lib -ldb2
else
    # Link the program with utilapi.o.
    cc -o $1 $1.o utilapi.o -L$DB2PATH/lib \
        -Wl,-rpath,$DB2PATH/lib -ldb2
fi

```

bldapp 的编译和连接选项

编译选项:

- cc** C 编译程序。
- I\$DB2PATH/include**
指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include
- c** 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldapp 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

-o \$1 指定可执行文件。

\$1.o 指定目标文件。

utilemb.o

如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.o

如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: `$HOME/sql1lib/lib`。如果不指定 `-L` 选项, 则假定 `/usr/lib:/lib`。

-Wl,-rpath,\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: `$HOME/sql1lib/lib`。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.c` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。

要运行可执行文件, 输入可执行文件名:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名：
`updat`
2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：
`updat database`
3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 `ID` 和口令：
`updat database userid password`

嵌入式 SQL 存储过程

`sqllib/samples/c` 中的脚本文件 `blsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例中构建存储过程，所以不需要指定用户 `ID` 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# blsrv script file -- Linux
# Builds a C stored procedure
# Usage: blsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# Compile the program.
cc -I$DB2PATH/include -c $1.c

# Link the program and create a shared library
cc -shared -o $1 $1.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

blsrv 的编译和连接选项

编译选项:

cc C 编译程序。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

连接选项:

cc 使用编译程序作为连接程序的前端。

-shared

生成共享库。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-Wl,-rpath,\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sqllib/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如连接 `sample` 数据库, 输入:

```
blsrv spserver
```

如果连接另一个数据库, 还须输入该数据库名:

```
blsrv spserver database
```

此脚本文件将共享库复制到服务器的 `sqllib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了共享库 spserver，就可构建存取它的客户机应用程序 spclient。

可使用脚本文件 bldapp 构建 spclient。有关详情，参考第184页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用共享库中的存储过程，可输入以下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 sample，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 spserver，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

sqllib/samples/c 中的脚本文件 bldudf 包含构建 UDF 所需的命令。UDF 不包含嵌入式 SQL 语句。因此，要构建 UDF 程序，不必连接数据库或预编译和联编该程序。

参数 \$1 指定源文件名。此脚本文件将此源文件名用作共享库名。

```
#!/bin/ksh
# bldudf script file -- Linux
# Builds a C UDF library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the program.
cc -I$DB2PATH/include -c $1.c
```

```

# Link the program and create a shared library.
cc -o $1 $1.o -shared -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldudf 的编译和连接选项	
编译选项:	
cc	C 编译程序。
-I\$DB2PATH/include	指定 DB2 包含文件的位置。例如: \$HOME/sqlllib/include。
-c	只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。
连接选项:	
cc	使用编译程序作为连接程序的前端。
-o \$1	指定可执行文件。
\$1.o	包括该程序的目标文件。
-shared	生成共享库。
-L\$DB2PATH/lib	指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。
-Wl,-rpath,\$DB2PATH/lib	指定运行时 DB2 共享库的位置。例如: \$HOME/sqlllib/lib。
-ldb2	连接 DB2 库。
-ldb2apie	连接“DB2 API 引擎”库, 以便可使用 LOB 定位器。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到 `sqlllib/function` 目录中。

如果需要, 为该 UDF 设置文件方式, 以便 DB2 实例可运行它。

一旦构建了 `udfsrv`，就可构建调用它的客户机应用程序 `udfcli`。提供了此程序的 DB2 CLI 和嵌入式 SQL 版本。

可根据 `sqllib/samples/cli` 中的源文件 `udfcli.c`，使用脚本文件 `bldcli` 构建 DB2 CLI `udfcli` 程序。有关详情，参考第179页的『DB2 CLI 应用程序』。

可根据 `sqllib/samples/c` 中的源文件 `udfcli.sqc`，使用脚本文件 `bldapp` 构建嵌入式 SQL `udfcli` 程序。有关详情，参考第184页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF，可输入可执行文件名来运行样本调用应用程序：

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

使用 Linux C 编写的多线程应用程序需用 `-D_REENTRANT` 编译，并用 `-lpthread` 连接。

`sqllib/samples/c` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID，第四个参数 `$4` 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldmt script file -- Linux
# Builds a C multi-threaded embedded SQL program.
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# Compile the program.
cc -I$DB2PATH/include -c $1.c -D_REENTRANT

# Link the program.
cc -o $1 $1.o -lpthread -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

除了以上讨论的 `-D_REENTRANT` 和 `-lpthread` 选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第184页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据 `thdsrver.sqc` 源文件构建样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入：

```
thdsrver
```

Linux C++

本节包括下列主题：

- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数
- 多线程应用程序

DB2 API 和嵌入式 SQL 应用程序

`sqlllib/samples/cpp` 中的脚本文件 `bldapp` 包含构建样本 C++ 程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是唯一的必需参数，且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定数据库的用户 ID；第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序，`bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名，则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。


```

#!/bin/ksh
# bldapp script file -- Linux
# Builds a C++ application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.C error-checking utility.
    g++ -I$DB2PATH/include -c utilemb.C
else
    # Compile the utilapi.C error-checking utility.
    g++ -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
g++ -I$DB2PATH/include -c $1.C

if [[ -f $1".sqC" ]]
then
    # Link the program with utilemb.o
    g++ -o $1 $1.o utilemb.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
else
    # Link the program with utilapi.o
    g++ -o $1 $1.o utilapi.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
fi

```

bldapp 的编译和连接选项

编译选项:

g++ C++ 编译程序。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldapp 的编译和连接选项

连接选项:

g++ 使用编译程序作为连接程序的前端。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilemb.o

如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.o

如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-Wl,-rpath,\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sql1lib/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.C` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqC` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

注: 请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ 存储过程的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。此脚本文件将源文件名 `$1` 用作共享库名。

```
#!/bin/ksh
# bldsrv script file -- Linux
# Builds a C++ stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# Compile the program.
g++ -I$DB2PATH/include -c $1.C

# Link the program and create a shared library.
g++ -shared -o $1 $1.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

```
# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv 的编译和连接选项	
编译选项:	
g++	C++ 编译程序。
-I\$DB2PATH/include	指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include
-c	只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。
连接选项:	
g++	使用编译程序作为连接程序的前端。
-shared	生成共享库。
-o \$1	指定可执行文件。
\$1.o	包括该程序的目标文件。
-L\$DB2PATH/lib	指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。
-Wl,-rpath,\$DB2PATH/lib	指定运行时 DB2 共享库的位置。例如: \$HOME/sqllib/lib。
-ldb2	连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv spserver
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv spserver database
```

此脚本文件将共享库复制到服务器的 `sqllib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`, 就可构建调用共享库中的存储过程的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情, 参考第192页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用共享库中的存储过程, 可输入以下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数。 存储过程将输出返回给客户机应用程序。

用户定义函数 (UDF)

注: 请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ UDF 的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldudf` 包含构建 UDF 所需的命令。UDF 不包含嵌入式 SQL 语句。这意味着要构建 UDF 程序, 不必连接数据库来预编译和联编该程序。

参数 `$1` 指定源文件名。此脚本文件将此源文件名用作共享库名。

```
#!/bin/ksh
# bldudf script file -- Linux
# Builds a C++ UDF library
```

```

# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlllib

# Compile the program.
if [[ -f $1".c" ]]
then
g++ -I$DB2PATH/include -c $1.c

elif [[ -f $1".C" ]]
then
g++ -I$DB2PATH/include -c $1.C
fi

# Link the program.
g++ -o $1 $1.o -shared -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldudf 的编译和连接选项

编译选项:

g++ C++ 编译程序。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqlllib/include。

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldudf 的编译和连接选项

连接选项:

g++ 使用编译程序作为连接程序的前端。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-shared
生成共享库。

-L\$DB2PATH/lib
指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-Wl,-rpath,\$DB2PATH/lib
指定运行时 DB2 共享库的位置。例如: \$HOME/sqlllib/lib。

-ldb2 连接 DB2 库。

-ldb2apie
连接“DB2 API 引擎”库, 以便可使用 LOB 定位器。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到 `sqlllib/function` 目录中。

必要时, 将该 UDF 设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。可根据 `sqlllib/samples/cpp` 中的 `udfcli.sqC` 源文件, 使用脚本文件 `bldapp` 来构建 `udfcli` 程序。有关详情, 参考第192页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF, 可输入可执行文件名来运行样本调用应用程序:

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

使用 Linux C++ 编写的多线程应用程序需用 `-D_REENTRANT` 编译, 并用 `-lpthread` 连接。

sqllib/samples/cpp 中的脚本文件 bldmt 包括构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 \$1 指定源文件的名称。第二个参数 \$2 指定要连接的数据库名称。第三个参数 \$3 指定数据库的用户 ID，第四个参数 \$4 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 sample 数据库。

```
#!/bin/ksh
# bldmt script file -- Linux
# Builds a C++ multi-threaded embedded SQL program.
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlib

# If an embedded SQL program, precompile and bind it.
embprep $1 $2 $3 $4

# Compile the program.
g++ -D_REENTRANT -I$DB2PATH/include -c $1.C

# Link the program.
g++ -lpthread -o $1 $1.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

除了以上讨论的 `-D_REENTRANT` 和 `-lpthread` 选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第192页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sqC` 构建样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入：

```
thdsrver
```

第9章 构建 OS/2 应用程序

IBM VisualAge C++ OS/2 版的版本 3	201	使用编译程序	213
DB2 CLI 应用程序	202	嵌入式 SQL 应用程序	214
构建和运行嵌入式 SQL 应用程序	203	构建和运行嵌入式 SQL 应用程序	216
使用 DB2 API 的 DB2 CLI 应用程序	204	嵌入式 SQL 存储过程	217
DB2 CLI 存储过程	204	Micro Focus COBOL	219
DB2 API 和嵌入式 SQL 应用程序	206	使用编译程序	219
构建和运行嵌入式 SQL 应用程序	208	DB2 API 和嵌入式 SQL 应用程序	220
嵌入式 SQL 存储过程	209	构建和运行嵌入式 SQL 应用程序	221
用户定义函数 (UDF)	211	嵌入式 SQL 存储过程	222
IBM VisualAge C++ OS/2 版的版本 4.0	213	REXX	224
IBM VisualAge COBOL OS/2 版	213		

本章提供在 OS/2 上构建应用程序的详细信息。在命令文件中，以 db2 开始的命令是“命令行处理器”(CLP)命令。有关 CLP 命令的详情，可参考 *Command Reference*。

有关 OS/2 的最新 DB2 应用程序开发情况更新信息，请访问以下 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

注：OS/2 上的 16 位应用程序中不允许有包含用户定义 SQLDA 的复合 SQL 语句。

IBM VisualAge C++ OS/2 版的版本 3

本节包括下列主题：

- DB2 CLI 应用程序
- DB2 CLI 存储过程
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)

注：VisualAge C++ 编译程序用于 %DB2PATH\samples\c 和 %DB2PATH\samples\cpp 目录中提供的 C 和 C++ 样本程序。相同的命令文件也位于这两个目录中。它们包含根据文件扩展名接受 C 或 C++ 源文件的命令。

DB2 CLI 应用程序

%DB2PATH%\samples\cli 中的命令文件 bldcli 包含构建 DB2 CLI 程序所需的命令。参数 %1 指定源文件名。

这是唯一的必需参数，且是不包括嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

如果该程序包含嵌入式 SQL（根据扩展名 .sqc 来判断），则调用 embprep 命令文件来预编译该程序，生成一个扩展名为 .c 的程序文件。

```
@echo off
rem bldcli command file - OS/2
rem Builds a CLI program with IBM VisualAge C++.
rem Usage: bldcli prog_name [ db_name [ userid password ] ]

if exist "%1.sqc" call embprep %1 %2 %3 %4
if exist "%1.sqx" call embprep %1 %2 %3 %4
if "%1" == "" goto error

rem Compile the error-checking utility.
icc -C+ -O- -Ti+ utilcli.c

rem Compile the program.
if exist "%1.sqx" goto cpp
icc -C+ -O- -Ti+ %1.c
goto link_step
:cpp
icc -C+ -O- -Ti+ %1.cxx

rem Link the program.
:link_step
ilink /NOFREE /NOI /DEBUG /ST:64000 /PM:VIO %1.obj utilcli.obj,%1.exe,NUL,db2cli.lib;

goto exit
:error
echo Usage: bldcli prog_name [ db_name [ userid password ] ]
:exit
@echo on
```

bldcli 的编译和连接选项

编译选项:

- icc** IBM VisualAge C++ 编译程序。
- C+** 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。
- O-** 无优化。关闭优化使得使用调试程序更容易。
- Ti+** 生成调试程序信息

bldcli 的编译和连接选项

连接选项:

ilink 使用 ilink 连接程序连接编辑。

/NOFREE

无自由格式。

/NOI 不忽略大小写。强制使用区别大小写的标识符。

/DEBUG 包括调试信息。

/ST:64000

指定堆栈大小至少为 64 000。

/PM:VIO

允许程序在 OS/2 窗口中运行。

%1.obj 包括目标文件。

utilcli.obj

包括该实用程序的目标文件以便检查错误。

%1.exe 指定可执行文件。

NUL 接受缺省值。

db2cli.lib

连接 DB2 CLI 库。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`，输入：

```
bldcli tbinfo
```

产生可执行文件 `tbinfo.exe`。可输入可执行文件名（不带扩展名）来运行该可执行文件：

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`：

1. 如果与同一实例中的 `sample` 数据库连接，输入：

```
bldcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接，还须输入该数据库名：

```
bldcli dbusemx database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldcli dbusemx database userid password
```

产生可执行文件 `dbusemx.exe`。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名（不带扩展名）：

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

%DB2PATH%\samples\cli 中的命令文件 `bldapi` 包含使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 `utilapi` 实用程序文件（该文件包括创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 `bldcli` 命令文件之间唯一的区别。有关 `bldapi` 和 `bldcli` 之间公共的编译和连接选项，请参阅第202页的『DB2 CLI 应用程序』。

要根据源文件 `dbmconn.c` 构建样本程序 `dbmconn`，输入：

```
bldapi dbmconn
```

产生可执行文件 `dbmconn.exe`。可输入可执行文件名（不带扩展名）来运行该可执行文件：

```
dbmconn
```

DB2 CLI 存储过程

%DB2PATH%\samples\cli 中的命令文件 `bldclisp` 包含构建 CLI 存储过程所需的命令。此命令文件将存储过程构建为服务器上的 DLL。

参数 %1 指定源文件名。此命令文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldclisp command file - OS/2
rem Builds a CLI stored procedure using the IBM VisualAge C++ compiler.
rem Usage: bldclisp <prog_name>

if "%1" == "" goto error

rem Compile the error-checking utility.
icc -C+ -Ti+ -Ge- -Gm+ -W2 utilcli.c
rem Compile the program.
```

```

if exist "%1.cxx" goto cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.c
goto link_step
:cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.cxx

:link_step
rem Link the program and produce a DLL.
ilink /NOFREE /MAP /NOI /DEBUG /ST:64000 %1.obj utilcli.obj,%1.dll,,db2cli.lib,%1.def;

rem Copy the stored procedure DLL to the 'function' directory
copy %1.dll %DB2PATH%\function

goto exit
:error
echo Usage: bldclisp prog_name
:exit
@echo on

```

bldclisp 的编译和连接选项	
编译选项:	
icc	IBM VisualAge C++ 编译程序。
-C+	只执行编译; 不连接。此命令文件有单独的编译和连接步骤。
-Ti+	生成调试程序信息。
-Ge-	构建 .DLL 文件。使用静态连接的运行期库的版本。
-Gm+	连接多任务库。
-W2	输出警告、错误、严重的和不可恢复的错误信息。
连接选项:	
ilink	使用 ilink 连接程序连接编辑。
/NOFREE	无自由格式。
/MAP	生成映象文件。
/NOI	不忽略大小写。强制使用区别大小写的标识符。
/DEBUG	包括调试信息。
/ST:64000	指定堆栈大小至少为 64000。
%1.obj	包括目标文件。
%1.dll	创建动态链接库。
db2cli.lib	连接 DB2 CLI 库。
%1.def	模块定义文件。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 spserver.c 构建样本程序 spserver, 输入:

```
bldclisp spserver
```

此脚本文件将共享库复制到服务器的 %DB2PATH%\function 路径中。

接着在服务器上运行 `screate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf screate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`, 就可构建调用共享库中的存储过程的 CLI 客户机应用程序 `spclient`。

可使用命令文件 `bldcli` 构建 `spclient`。有关详情, 参考第202页的『DB2 CLI 应用程序』。

要存取该共享库, 可输入以下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

%DB2PATH%\samples\c 和 %DB2PATH%\samples\cpp 中的命令文件 `bldapi.cmd` 包含构建 DB2 应用程序所需的命令。

第一个参数 %1 指定源文件名。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 %2 指定您想连接的数据库的名称；第三个参数 %3 指定数据库的用户 ID；第四个参数 %4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编命令文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
@echo off
rem bldapp command file -- OS/2
rem Builds a VisualAge C++ application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

if exist "%1.sqx" goto embedded
if exist "%1.sqc" goto embedded
goto non_embedded

:embedded
rem Precompile and bind the program.
call embprep %1 %2 %3 %4
rem Compile the program.
if exist "%1.cxx" goto cpp_embedded
icc -c utilemb.c
icc -C+ -O- -Ti+ %1.c
goto link_embedded
:cpp_embedded
icc -c utilemb.cxx
icc -C+ -O- -Ti+ %1.cxx
goto link_embedded

:non_embedded
rem Compile the program.
if exist "%1.cxx" goto cpp
icc -c utilapi.c
icc -C+ -O- -Ti+ %1.c
goto link_non_embedded
:cpp
icc -c utilapi.cxx
icc -C+ -O- -Ti+ %1.cxx
goto link_non_embedded

rem Link the program.
:link_embedded
ilink /NOFREE /NOI /DEBUG /ST:64000 /PM:VIO %1.obj utilemb.obj,,db2api;
goto exit
:link_non_embedded
ilink /NOFREE /NOI /DEBUG /ST:64000 /PM:VIO %1.obj utilapi.obj,,db2api;
:exit
@echo on
```

bldapp 的编译和连接选项

编译选项:

icc IBM VisualAge C++ 编译程序。
-C+ 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
-O- 无优化。关闭优化使得使用调试程序更容易。
-Ti+ 生成调试程序信息

连接选项:

ilink 使用 ilink 连接程序连接编辑。
/NOFREE
无自由格式。
/NOI 不忽略大小写。强制使用区别大小写的标识符。
/DEBUG 包括调试信息。
/ST:64000
指定堆栈大小至少为 64000。
/PM:VIO
允许程序在 OS/2 窗口中运行。
utilemb.obj
如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。
utilapi.obj
如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。
db2api 连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.c` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client.exe`。

要运行可执行文件, 输入该可执行文件名 (不带扩展名):

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```


产生可执行文件 `updat.exe`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名 (不带扩展名):

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

`%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中的命令文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此命令文件将存储过程编译为服务器上的 DLL。

第一个参数 `%1` 指定源文件名。第二个参数 `%2` 指定想要连接的数据库的名称。因为必须在数据库所在的同一实例上构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

此命令文件使用源文件名 `%1` 作为 DLL 名。

```
@echo off
rem bldsrv command file -- OS/2
rem Builds a VisualAge C++ stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2

rem Compile the program.
if exist "%1.cxx" goto cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.c
goto link_step
:cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.cxx

:link_step
rem Link the program.
ilink /NOFREE /NOI /DEBUG /ST:64000 %1.obj,%1.dll,,db2api,%1.def;
```

```
rem Copy the stored procedure to the %DB2PATH%\function directory.
copy %1.dll %DB2PATH%\function
@echo on
```

bldsrv 的编译和连接选项	
编译选项:	
icc	IBM VisualAge C++ 编译程序。
-C+	只执行编译; 不连接。此命令文件有单独的编译和连接步骤。
-Ti+	生成调试程序信息。
-Ge-	构建 .DLL 文件。使用静态连接的运行期库的版本。
-Gm+	连接多任务库。
-W2	输出警告、错误、严重的和不可恢复的错误信息。
连接选项:	
ilink	使用 ilink 连接程序连接编辑。
/NOFREE	无自由格式。
/NOI	不忽略大小写。强制使用区别大小写的标识符。
/DEBUG	包括调试信息。
/ST:64000	指定堆栈大小至少为 64000。
%1.dll	创建动态链接库。
db2api	连接 DB2 库。
%1.def	模块定义文件。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 输入:

```
bldsrv spserver
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv spserver database
```

此命令文件将共享库复制到服务器的 `%DB2PATH%\function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`, 就可构建存取该共享库的客户机应用程序 `spclient`。

可使用命令文件 `bldapp` 构建 `spclient`。有关详情, 参考第206页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程, 可输入如下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

`%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中的命令文件 `bldudf` 包含构建 UDF 所需的命令。

UDF 不能包含嵌入式 SQL 语句。因此, 要构建 UDF 程序, 不必连接数据库来预编译和联编该程序。

此命令文件采用一个参数 `%1` 来指定源文件名。它使用源文件名 `%1` 作为 DLL 名。

```
@echo off
rem bldudf command file -- OS/2
rem Builds a VisualAge C++ user-defined function (UDF)
rem Usage: bldudf <prog_name>

if "%1" == "" goto error

rem Compile the program.
```

```

if exist "%1.cxx" goto cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.c
goto link_step
:cpp
rem icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.cxx

:link_step
rem Link the program.
ilink /NOFREE /MAP /NOI /DEBUG /ST:64000 %1.obj,%1.dll,,db2api db2apie,%1.def;

rem Copy the UDF to the %DB2PATH%\function directory
copy %1.dll %DB2PATH%\function

goto exit
:error
echo Usage: bldudf prog_name
:exit
@echo on

```

bldudf 的编译和连接选项

编译选项:

icc IBM VisualAge C++ 编译程序。

-C+ 只执行编译；不连接。此命令文件有单独的编译和连接步骤。

-Ti+ 生成调试程序信息。

-Ge- 构建 .DLL 文件。使用静态连接的运行期库的版本。

-Gm+ 连接多任务库。

-W2 输出警告、错误、严重的和不可恢复的错误信息。

连接选项:

ilink 使用 ilink 连接程序连接编辑。

/NOFREE 无自由格式。

/MAP 生成映象文件。

/NOI 不忽略大小写。强制使用区别大小写的标识符。

/DEBUG 包括调试信息。

/ST:64000 指定堆栈大小至少为 64000。

%1.dll 创建动态链接库。

db2api 连接 DB2 库。

db2apie 连接“DB2 API 引擎”库。

%1.def 模块定义文件。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`，输入：

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到服务器的 %DB2PATH%\function 路径中。

必要时，将该 UDF 设置为文件方式，以便客户机应用程序可存取它。

一旦构建了 udfsrv，就可构建调用它的客户机应用程序 udfcli。提供了此程序的 DB2 CLI 和嵌入式 SQL 版本。

可根据 %DB2PATH%\samples\cli 中的 udfcli.c 源文件，使用命令文件 bldcli.cmd 构建 DB2 CLI udfcli 程序。有关详情，参考第202页的『DB2 CLI 应用程序』。

可根据 %DB2PATH%\samples\c 中的源文件 udfcli.sqc，使用命令文件 bldapp 构建嵌入式 SQL udfcli 程序。有关详情，参考第206页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF，可输入可执行文件名（不带扩展名）来运行样本调用应用程序：

```
udfcli
```

调用应用程序从 udfsrv 库中调用 ScalarUDF 函数。

IBM VisualAge C++ OS/2 版的版本 4.0

VisualAge C++ 版本 4 编译程序的应用程序构建信息对于 AIX、OS/2 和 Windows 32 位操作系统是相同的。有关此信息，参阅第120页的『VisualAge C++ 版本 4.0』。

IBM VisualAge COBOL OS/2 版

本节包括下列主题：

- 使用编译程序
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程

使用编译程序

下面几点帮助您对 DB2 使用 IBM VisualAge COBOL 编译程序。

创建联编文件的解决方法

当使用 DB2 OS/2 版和 IBM Cobol 创建应用程序时，DB2 预编译程序经常创建联编文件失败。这是因为 OS/2 内的一个文件句柄限制。

对此编译程序的修正使 OS/2 允许在执行编译的机器上有更多的文件句柄。应将:

```
SET SHELLHANDLESINC=20
```

一行插入装有 DB2 OS/2 版的机器上的 CONFIG.SYS 文件中。或者, 在编译时可使用 NODATA 选项 (这是 IBM Cobol 选项)。

嵌入式 SQL 和 DB2 API 调用

如果开发包含嵌入式 SQL 和 DB2 API 调用的应用程序, 且使用的是 IBM VisualAge COBOL 编译程序, 应切记以下几点:

- 当使用命令行处理器命令 db2 prep 预编译应用程序时, 使用 target ibmcob 选项。
- 在源文件中不要使用制表符。
- 可在源文件中使用 PROCESS 和 CBL 关键字以设置编译选项。只能将这些关键字置于 8 至 72 列之间。
- 如果您的应用程序只包含嵌入式 SQL, 而不包含 DB2 API 调用, 则不需要使用 pgmname(mixed) 编译选项。如果使用 DB2 API 调用, 则必须使用 pgmname(mixed) 编译选项。
- 如果使用的是 IBM VisualAge COBOL 编译程序的“System/390 主机数据类型支持”功能部件, 则您的应用程序的 DB2 包含文件位于如下目录中:

```
%DB2PATH%\include\cobol_i
```

如果要使用提供的命令文件构建 DB2 样本程序, 必须将在命令文件中指定的包含文件路径更改为指向 cobol_i 目录而不是 cobol_a 目录。

如果未使用 IBM VisualAge COBOL 编译程序的“System/390 主机数据类型支持”功能部件, 或使用的是此编译程序的较早版本, 则您的应用程序的 DB2 包含文件位于如下目录中:

```
%DB2PATH%\include\cobol_a
```

按如下所示, 指定包括 .cbl 扩展名的 COPY 文件名:

```
COPY "sql.cbl".
```

嵌入式 SQL 应用程序

%DB2PATH%\samples\cobol 中的命令文件 bldapp.cmd 包含构建 DB2 应用程序所需的命令。

第一个参数 %1 指定源文件名。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 %2 指定您想连接的数据库的名称；第三个参数 %3 指定数据库的用户 ID；第四个参数 %4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编命令文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
@echo off
rem bldapp command file -- OS/2
rem Builds a VisualAge COBOL application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if exist "%1.sqb" goto prepbind
goto compile_step
:prepbind
call embprep %1 %2 %3 %4

:compile_step
rem Compile the checkerr error checking utility.
cob2 -c -g -qpgmname(mixed) -qlib -I%DB2PATH%\include\cobol_a checkerr.cbl

rem Compile the program.
cob2 -c -g -qpgmname(mixed) -qlib -I%DB2PATH%\include\cobol_a %1.cbl

rem Link the program.
ilink %1.obj checkerr.obj db2api.lib /ST:64000 /PM:VIO /NOI /DEBUG
@echo on
```

bldapp 的编译和连接选项

编译选项:

- cob2** IBM VisualAge COBOL 编译程序。
- c** 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。
- g** 包括调试信息。
- qpgmname(mixed)** 指示编译程序可用混合大小写的名称调用 (CALL) 库入口点。
- qlib** 指示编译程序处理 COPY 语句。
- Ipath** 指定 DB2 包含文件的位置。例如: `-I%DB2PATH%\include\cobol_a`。

bldapp 的编译和连接选项

连接选项:

ilink 使用 `ilink` 连接程序连接编辑。

checkerr.obj

包括错误检查实用程序的目标文件。

db2api.lib

连接 DB2 库。

/ST:64000

指定堆栈大小至少为 64000。

/PM:VIO

允许程序在 OS/2 窗口中运行。

/NOI 连接时不忽略大小写。

/DEBUG 包括调试信息。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `client.cbl` 构建非嵌入式 SQL 样本程序 `client`，输入：

```
bldapp client
```

产生可执行文件 `client.exe`。可输入可执行文件名（不带文件扩展名），来对 `sample` 数据库运行该可执行文件：

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 SQL 应用程序 `updat`：

1. 如果与同一实例中的 `sample` 数据库连接，输入：

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接，还须输入该数据库名：

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldapp updat database userid password
```

产生可执行文件 `updat.exe`。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名（不带文件扩展名）：

```
updat
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：

```
updat database
```


3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
updat database userid password
```

嵌入式 SQL 存储过程

%DB2PATH%\samples\cobol 中的命令文件 bldsrv 包含构建存储过程所需的命令。此命令文件将存储过程编译为服务器上的 DLL。

第一个参数 %1 指定源文件名。第二个参数 %2 指定想要连接的数据库的名称。因为必须在数据库所在的同一实例中构建存储过程，所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 sample 数据库。

此命令文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldsrv command file -- OS/2
rem Builds a VisualAge COBOL stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2

rem Compile the program.
cob2 -c -g -qpgmname(mixed) -qlib -I%DB2PATH%\include\cobol_a %1.cbl

rem Link the program.
ilink %1.obj checkerr.obj %1.def db2api.lib /ST:64000 /PM:VIO /NOI /DEBUG

rem Copy stored procedure to the %DB2PATH%\function directory.
copy %1.dll %DB2PATH%\function
@echo on
```

bldsrv 的编译和连接选项

编译选项:

- cob2** IBM VisualAge COBOL 编译程序。
- c** 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。
- g** 包括调试信息。
- qpgmname(mixed)** 指示编译程序可用混合大小写的名称调用 (CALL) 库入口点。
- qlib** 指示编译程序处理 COPY 语句。
- Ipath** 指定 DB2 包含文件的位置。例如： `-I%DB2PATH%\include\cobol_a`。

bldsrv 的编译和连接选项

连接选项:

ilink 使用 ilink 连接程序连接编辑。
checkerr.obj
包括错误检查实用程序的目标文件。
%1.def 模块定义文件。
db2api.lib
连接 DB2 库。
/ST:64000
指定堆栈大小至少为 64000。
/PM:VIO
允许程序在 OS/2 窗口中运行。
/NOI 连接时不忽略大小写。
/DEBUG 包括调试信息。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv outsrv
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv outsrv database
```

此命令文件使用与样本程序在同一目录中的模块定义文件 `outsrv.def` 来构建 DLL。此命令文件将存储过程 DLL `outsrv.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

必要时, 将该 DLL 设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了 DLL `outsrv`, 就可构建存取该 DLL 的客户机应用程序 `outcli`。可使用 `bldapp` 命令文件构建 `outcli`。有关详情, 参考第214页的『嵌入式 SQL 应用程序』。

要调用存储过程, 可输入如下命令运行客户机应用程序:

```
outcli database userid password
```

其中,

database

是要连接的数据库的名称。该名称可以是 `sample`、其远程别名或其他名称。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取 DLL `outsrv`，并在服务器数据库上执行大量同名的存储过程函数，然后将输出返回给客户机应用程序。

Micro Focus COBOL

本节包括下列主题：

- 使用编译程序
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程

使用编译程序

DB2 不支持与 Micro Focus COBOL 编译程序一起提供的 `link386` 连接程序。要连接 DB2 Micro Focus COBOL 程序，必须使用 IBM 编译程序产品中的 `ilink` 连接程序。本节讨论的脚本文件中使用的 `cbllink` 命令调用 `ilink` 连接程序。

当用包含嵌入式 SQL 和 DB2 API 调用的 Micro Focus COBOL 编译程序构建应用程序时，切记以下几点：

- 当使用命令行处理器命令 `db2 prep` 预编译应用程序时，应使用缺省 `target mfcob` 选项。
- 确保 `LIB` 环境变量指向 `%DB2PATH%\lib`，如下所示：

```
set LIB=%DB2PATH%\lib;%LIB%
```

- Micro Focus COBOL 的 DB2 COPY 文件驻留在 `%DB2PATH%\include\cobol_mf` 中。如下所示设置 `COBCPY` 环境变量以包括该目录：

```
set COBCPY=%DB2PATH%\include\cobol_mf;%COBCPY%
```

必须使用调用约定 8 来调用所有 DB2 应用程序设计接口。DB2 COBOL 预编译程序自动在 `SPECIAL-NAMES` 段中插入 `CALL-CONVENTION` 子句。如果 `SPECIAL-NAMES` 段不存在，DB2 COBOL 预编译程序按如下方式创建它：

```
Identification Division
Program-ID. "static".
special-names.
    call-convention 8 is DB2API.
```

而且，无论何时调用 DB2 API，预编译程序都会自动将符号 DB2API（用于标识调用约定）置于“call”关键字之后。例如，每当预编译程序从嵌入式 SQL 语句生成 DB2 API 运行期调用时，会发生此情况。

如果在未预编译的应用程序中调用 DB2 API，应以类似上面给定的方式在该应用程序中人工创建 SPECIAL-NAMES 段。如果直接调用 DB2 API，则需要要在“call”关键字之后人工添加 DB2API 符号。

DB2 API 和嵌入式 SQL 应用程序

%DB2PATH%\samples\cobol_mf 中的命令文件 bldapp 包含构建 DB2 应用程序所需的命令。

第一个参数 %1 指定源文件名。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 %2 指定您想连接的数据库的名称；第三个参数 %3 指定数据库的用户 ID；第四个参数 %4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编命令文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
@echo off
rem bldapp command file -- OS/2
rem Builds a Micro Focus COBOL application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if exist "%1.sqb" goto prepbind
goto compile_step
:prepbind
call embprep %1 %2 %3 %4

:compile_step
rem Compile the error-checking utility.
cobol checkerr.cbl;

rem Compile the program.
cobol %1.cbl;

rem Link the program.
cblllink %1.obj checkerr.obj db2api.lib db2gmf32.lib
@echo on
```

bldapp 的编译和连接选项
<p>编译选项:</p> <p>cobol Micro Focus COBOL 编译程序。</p>
<p>连接选项:</p> <p>cbllink 使用连接程序连接编辑。</p> <p>checkerr.obj 包括错误检查实用程序的目标文件。</p> <p>db2api.lib 连接 DB2 API 库。</p> <p>db2gmf32.lib 连接 M. F. COBOL 的 DB2 异常处理程序库。</p> <p>有关其他编译程序选项，参考编译程序文档。</p>

要根据源文件 `client.cbl` 构建非嵌入式 SQL 样本程序 `client`，输入：

```
bldapp client
```

产生可执行文件 `client.exe`。可输入可执行文件名（不带文件扩展名），来对 `sample` 数据库运行该可执行文件：

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 SQL 应用程序 `updat`：

1. 如果与同一实例中的 `sample` 数据库连接，输入：

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接，还须输入该数据库名：

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldapp updat database userid password
```

产生可执行文件 `updat.exe`。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名（不带文件扩展名）：

```
updat
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名：

```
updat database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
updat database userid password
```

嵌入式 SQL 存储过程

%DB2PATH%\samples\cobol_mf 中的命令文件 bldsrv 包含构建嵌入式 SQL 存储过程所需的命令。此命令文件将存储过程编译为服务器上的 DLL。

第一个参数 %1 指定源文件名。第二个参数 %2 指定想要连接的数据库的名称。因为必须在数据库所在的同一实例中构建存储过程，所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 sample 数据库。此命令文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldsrv command file -- OS/2
rem Builds a Micro Focus COBOL stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2

rem Compile the stored procedure.
cobol %1.cbl;

rem Link the stored procedure and create a shared library.
cbllink /d %1.obj db2api.lib db2gmf32.lib

rem Copy the stored procedure to the %DB2PATH%\function directory.
copy %1.dll %DB2PATH%\function
@echo on
```

bldsrv 的编译和连接选项
编译选项: cobol Micro Focus COBOL 编译程序。

blsrv 的编译和连接选项

连接选项:

cbllink

使用 Micro Focus COBOL 连接程序连接编辑。

/d 创建 .dll 文件。

db2api.lib

包括 DB2 API 库。

db2gmf32.lib

连接 M. F. COBOL 的 DB2 异常处理程序库。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`，如果连接 `sample` 数据库，则输入：

```
blsrv outsrv
```

如果连接另一个数据库，还须输入该数据库名：

```
blsrv outsrv database
```

连接程序使用用户未指定的缺省入口点。 `/d` 选项用于创建 `.dll` 文件以构建存储过程。此命令文件将存储过程 `DLL outsrv.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

必要时，将该 `DLL` 设置为文件方式，以便客户机应用程序可存取它。

一旦构建了 `DLL outsrv`，就可构建存取该 `DLL` 的客户机应用程序 `outcli`。可使用 `bldapp` 命令文件构建 `outcli`。有关详情，参考第220页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
outcli database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取 DLL outsrv，并在服务器数据库上执行大量同名的存储过程函数。然后将输出返回给该客户机应用程序。

REXX

不要编译或联编 REXX 程序。

在 OS/2 上，应用程序文件必须有 .cmd 扩展名。在创建之后，可从操作系统命令提示符处直接运行应用程序。

OS/2 REXX 程序必须包含开始于第一行第一列的注解，以便将它与批处理命令区分开来：

```
/* Any comment will do. */
```

可在目录 %DB2PATH%\samples\rexx 中找到 REXX 样本程序。要运行 REXX 样本程序 updat，输入：

```
updat
```

有关 REXX 和 DB2 的详情，参考 *Application Development Guide* 中的“Programming in REXX”一章。

第10章 构建 PTX 应用程序

ptx/C	225	用户定义函数 (UDF)	235
DB2 CLI 应用程序	225	多线程应用程序	237
构建和运行嵌入式 SQL 应用程序	227	ptx/C++	238
使用 DB2 API 的 DB2 CLI 应用程序	228	DB2 API 和嵌入式 SQL 应用程序	238
DB2 CLI 存储过程	228	构建和运行嵌入式 SQL 应用程序	240
DB2 API 和嵌入式 SQL 应用程序	231	嵌入式 SQL 存储过程	241
构建和运行嵌入式 SQL 应用程序	233	用户定义函数 (UDF)	243
嵌入式 SQL 存储过程	233	多线程应用程序	245

本章提供有关在 PTX 上使用 DB2 NUMA-Q 版构建应用程序的详细信息。在脚本文件中，以 db2 开始的命令是“命令行处理器”(CLP)命令。有关 CLP 命令的详情，可参考 *Command Reference*。

有关 PTX 的最新 DB2 应用程序开发情况的更新，请访问以下 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

ptx/C

本节包括下列主题：

- DB2 CLI 应用程序
- 使用 DB2 API 的 DB2 CLI 应用程序
- DB2 CLI 存储过程
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 CLI 应用程序

sqllib/samples/cli 中的脚本文件 bldcli 包含构建 DB2 CLI 程序所需的命令。参数 \$1 指定源文件的名称。

这是唯一的必需参数，且是不包含嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

如果该程序包含嵌入式 SQL（根据扩展名 .sql 来判断），则调用脚本 embprep 来预编译该程序，生成一个扩展名为 .c 的程序文件。

```
#!/bin/ksh
# bldcli script file -- PTX
# Builds a DB2 CLI program
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sql" ]]
then
    embprep $1 $2 $3 $4
fi

# Compile the error-checking utility.
cc -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -I$DB2PATH/include -c $1.c

# Link the program.
cc -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2
```

bldcli 的编译和连接选项

编译选项:

cc 使用 C 编译程序。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译，不连接。编译和连接是两个独立的步骤。

blcli 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

-o \$1 指定可执行程序。

\$1.o 包括该程序的目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`, 输入:

```
blcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名, 运行该可执行文件:

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
blcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
blcli dbusemx database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
blcli dbusemx database userid password
```

产生可执行文件 `dbusemx`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名称:

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

sqllib/samples/cli 中的脚本文件 bldapi 包含使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 utilapi 实用程序文件（该文件包含创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 bldcli 脚本文件之间唯一的区别。有关 bldapi 和 bldcli 之间公共的编译和连接选项，请参阅第225页的『DB2 CLI 应用程序』。

要根据源文件 dbmconn.c 构建样本程序 dbmconn，输入：

```
bldapi dbmconn
```

产生可执行文件 dbmconn。可输入可执行文件名，运行该可执行文件：

```
dbmconn
```

DB2 CLI 存储过程

sqllib/samples/cli 中的脚本文件 bldclisp 包含构建 DB2 CLI 存储过程所需的命令。参数 \$1 指定源文件的名称。

```

#!/bin/ksh
# bldclisp script file -- PTX
# Builds a DB2 CLI stored procedure
# Usage: bldclisp <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the error-checking utility.
cc -KPIC -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -KPIC -I$DB2PATH/include -c $1.c

# Link the program.
cc -G -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldclisp 的编译和连接选项

编译选项:

- cc** C 编译程序。
- KPIC** 为共享库生成与位置无关的代码。
- I\$DB2PATH/include**
指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。
- c** 只执行编译, 不连接。编译和连接是两个独立的步骤。

bldclisp 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

-G 生成共享库。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 spserver.c 构建样本程序 spserver, 输入:

```
bldclisp spserver
```

此脚本文件将共享库复制到服务器的 sql1lib/function 路径中。

接着在服务器上运行 spcreate.db2 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 spserver, 就可构建存取该共享库的 CLI 客户机应用程序 spclient。

可使用脚本文件 bldcli 构建 spclient。有关详情, 参考第225页的『DB2 CLI 应用程序』。

要存取该共享库, 可输入以下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数, 然后将输出返回给客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/c` 中的构建文件 `bldapp` 包含构建 DB2 API 和嵌入式 SQL 程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是唯一的必需参数, 且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库, 因此还须提供三个可选参数: 第二个参数 `$2` 指定您想连接的数据库的名称; 第三个参数 `$3` 指定数据库的用户 ID; 第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序, `bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名, 则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时, 才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- PTX
# Builds a C application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ] ]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlib

# if an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.c error-checking utility.
    cc -I$DB2PATH/include -c utilemb.c
else
    # Compile the utilapi.c error-checking utility.
    cc -I$DB2PATH/include -c utilapi.c
fi
```

```

# Compile the program.
cc -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  cc -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2
else
  # Link the program with utilapi.o
  cc -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2
fi

```

bldapp 的编译和连接选项

编译选项:

cc C 编译程序。

-I\$DB2PATH/include 指定 DB2 包含文件的位置。例如: \$HOME/sql1lib/include

-c 只执行编译, 不连接。编译和连接是两个独立的步骤。

连接选项:

cc 使用编译程序作为连接程序的前端。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

util.o 包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib 指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.c` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。

要运行可执行文件, 输入可执行文件名:

```
client
```


构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名称:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

`sqllib/samples/c` 中的脚本文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要指定用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldsrv script file -- PTX
# Builds a C stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
```

```

embprep $1 $2

# Compile the program.
cc -KPIC -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
cc -G -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldsrv 的编译和连接选项	
编译选项:	
cc	C 编译程序。
-KPIC	为共享库生成与位置无关的代码。
-I\$DB2PATH/include	指定 DB2 包含文件的位置。例如: <code>-I\$DB2PATH/include</code>
-c	只执行编译, 不连接。编译和连接是两个独立的步骤。
连接选项:	
cc	使用编译程序作为连接程序的前端。
-G	生成共享库。
-o \$1	指定可执行文件。
\$1.o	包括该程序的目标文件。
-L\$DB2PATH/lib	指定连接时 DB2 静态库和共享库的位置。例如: <code>\$HOME/sql1lib/lib</code> 。如果不指定 <code>-L</code> 选项, 则假定 <code>/usr/lib:/lib</code> 。
-ldb2	连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv spserver
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv spserver database
```

此脚本文件将共享库复制到服务器的 `sqllib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`, 就可构建存取它的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情, 参考第231页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用共享库中的存储过程, 可输入以下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`, 并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

`sqllib/samples/c` 中的脚本文件 `bldudf` 包含构建 UDF 所需的命令。UDF 不包含嵌入式 SQL 语句。因此, 要构建 UDF 程序, 不必连接数据库或预编译和联编该程序。

参数 `$1` 指定源文件名。此脚本文件将源文件名用作共享库名。

```

#! /bin/ksh
# bldudf script file -- PTX
# Builds a C user-defined function library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlllib

# Compile the program.
cc -KPIC -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
cc -G -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqlllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldudf 的编译和连接选项

编译选项:

cc C 编译程序。

-KPIC 为共享库生成与位置无关的代码。

-I\$DB2PATH/include
指定 DB2 包含文件的位置。例如: \$HOME/sqlllib/include。

-c 只执行编译, 不连接。编译和连接是两个独立的步骤。

连接选项:

cc 使用编译程序作为连接程序的前端。

-G 生成共享库。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-L\$DB2PATH/lib
指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-ldb2 连接 DB2 库。

-ldb2apie
连接“DB2 API 引擎”库, 以便可使用 LOB 定位器。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bludf udfsrv
```

此脚本文件将 UDF 复制到 sqllib/function 目录中。

必要时，将该 UDF 设置为文件方式，以便客户机应用程序可存取它。

一旦构建了 udfsrv，就可构建调用它的客户机应用程序 udfcli。提供了此程序的 DB2 CLI 和嵌入式 SQL 版本。

可根据 sqllib/samples/cli 中的源文件 udfcli.c 使用脚本文件 bldcli 构建 DB2 CLI udfcli 程序。有关详情，参考第225页的『DB2 CLI 应用程序』。

可根据 sqllib/samples/c 中的源文件 udfcli.sqc 使用脚本文件 bldapp 构建嵌入式 SQL udfcli 程序。有关详情，参考第231页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF，可输入可执行文件名运行样本调用应用程序：

```
udfcli
```

调用应用程序从 udfsrv 库中调用 ScalarUDF 函数。

多线程应用程序

使用 ptx/C 的多线程式应用程序需要使用 -Kthread 进行编译和连接。

sqllib/samples/c 中的脚本文件 bldmt 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 \$1 指定源文件的名称。第二个参数 \$2 指定要连接的数据库名称。第三个参数 \$3 指定数据库的用户 ID，第四个参数 \$4 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID和口令是可选的。如果未提供数据库名，则该程序使用缺省的 sample 数据库。

```
#!/bin/ksh
# bldmt script file -- PTX
# Builds a C multi-threaded embedded SQL program.
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# Compile the program.
```

```
cc -Kthread -I$DB2PATH/include -c $1.c  
  
# Link the program.  
cc -Kthread -o $1 $1.o -L$DB2PATH/lib -ldb2
```

除了以上讨论的 `-Kthread` 选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第231页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据 `thdsrver.sqc` 源文件构建样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入：

```
thdsrver
```

ptx/C++

本节包括下列主题：

- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/cpp` 中的构建文件 `bldapp` 包含构建 DB2 API 和嵌入式 SQL 程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是唯一的必需参数，且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定数据库的用户 ID；第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序，`bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名，则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```

#!/bin/ksh
# bldapp script file -- PTX
# Builds a C++ application program
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# if an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.C error-checking utility.
    c++ -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c utilemb.C
else
    # Compile the utilapi.C error-checking utility.
    c++ -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c utilapi.C
fi

# Compile the program.
c++ -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.C

if [[ -f $1".sqC" ]]
then
    # Link the program with utilemb.o
    c++ -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2 -lseq
else
    # Link the program with utilapi.o
    c++ -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2 -lseq
fi

```

bldapp 的编译和连接选项

编译选项:

- c++** C++ 编译程序。
- I\$DB2PATH/include**
 指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include
- D_RWSTD_COMPILE_INSTANTIATE=0**
 不实例化破坏波类。
- c** 只执行编译, 不连接。 编译和连接是两个独立的步骤。

bldapp 的编译和连接选项

连接选项:

c++ 使用编译程序作为连接程序的前端。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilemb.o

如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.o

如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: `$HOME/sql1lib/lib`。如果不指定 `-L` 选项, 则假定 `/usr/lib:/lib`。

-ldb2 连接 DB2 库。

-lseq 连接 Sequent 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.C` 构建 DB2 API 非嵌入式 SQL 样本程序, 输入:

```
bldapp client
```

产生可执行文件 `client`。

要运行可执行文件, 输入可执行文件名:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqC` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:


```
updat
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名称：

```
updat database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令：

```
updat database userid password
```

嵌入式 SQL 存储过程

注：请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ 存储过程的信息。

sqllib/samples/cpp 中的脚本文件 bldsrv 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 \$1 指定源文件的名称。这是唯一需要的参数。构建嵌入式 SQL 程序需要连接数据库连接，因此还需一个附加的可选参数 \$2，该参数指定您想连接的数据库的名称。如果未提供数据库名，则使用缺省 sample 数据库。因为必须在数据库所在的同一实例中构建存储过程，所以不需要用户 ID 和口令的参数。脚本文件 bldsrv 将参数传送给预编译和联编文件 embprep。

源文件名 \$1 被用作共享库名。

```
#!/bin/ksh
# bldsrv script file -- PTX
# Builds a C++ stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# Compile the program. First ensure it is coded with extern "C".
c++ -KPIC -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.C

# Link the program and create a shared library.
c++ -G -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1.so $DB2PATH/function/$1
```

blsrv 的编译和连接选项

编译选项:

- c++** C++ 编译程序。
- KPIC** 为共享库生成与位置无关的代码。
- I\$DB2PATH/include**
指定 DB2 包含文件的位置。例如: `-I$DB2PATH/include`
- D_RWSTD_COMPILE_INSTANTIATE=0**
不实例化破坏波类。
- c** 只执行编译, 不连接。本书假定编译和连接是两个独立的步骤。

连接选项:

- c++** 使用编译程序作为连接程序的前端。
- G** 生成共享库。
- o \$1** 指定可执行文件。
- \$1.o** 包括该程序的目标文件。
- L\$DB2PATH/lib**
指定连接时 DB2 静态库和共享库的位置。例如: `$HOME/sql1lib/lib`。如果不指定 `-L` 选项, 则假定 `/usr/lib:/lib`。
- ldb2** 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.sqC` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入:

```
blsrv spserver
```

如果连接另一个数据库, 还须输入该数据库名:

```
blsrv spserver database
```

此脚本文件将共享库复制到服务器的 `sql1lib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`，就可构建调用共享库中的存储过程的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第238页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用共享库中的存储过程，可输入以下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。 该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数。存储过程将输出返回给客户机应用程序。

用户定义函数 (UDF)

注：请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ UDF 的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldudf` 包含构建 UDF 所需的命令。UDF 不包含嵌入式 SQL 语句。因此，要构建 UDF 程序，不必连接数据库或预编译和联编该程序。

参数 `$1` 指定源文件名。此脚本文件将源文件名用作共享库名。

```

#!/bin/ksh
# bldudf script file -- PTX
# Builds a C++ user-defined function library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the program.
c++ -KPIC -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.c

# Link the program and create a shared library.
c++ -G -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1.so $DB2PATH/function/$1

```

bldudf 的编译和连接选项

编译选项:

- c++** C++ 编译程序。
- KPIC** 为共享库生成与位置无关的代码。
- I\$DB2PATH/include**
 指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。
- D_RWSTD_COMPILE_INSTANTIATE=0**
 不实例化破坏波类。
- c** 只执行编译, 不连接。本书假定编译和连接是两个独立的步骤。

bldudf 的编译和连接选项

连接选项:

c++ 使用编译程序作为连接程序的前端。

-G 生成共享库。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-ldb2 连接 DB2 库。

-ldb2apie

连接“DB2 API 引擎”库, 以便可使用 LOB 定位器。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到服务器的 `sqlllib/function` 路径中。

必要时, 将该 UDF 设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。可根据 `sqlllib/samples/cpp` 中的 `udfcli.sqc` 源文件, 使用脚本文件 `bldapp` 来构建 `udfcli` 程序。有关详情, 参考第238页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF, 可输入可执行文件名运行样本调用应用程序:

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

使用 `ptx/C++` 的多线程式应用程序需要使用 `-Kthread` 进行编译和连接。

`sqlllib/samples/cpp` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID, 第四个参数 `$4` 指定口令。只有第一个参数

即源文件名是必需的。数据库名、用户 ID和口令是可选的。 如果未提供数据库名，则该程序使用缺省的 sample 数据库。

```
#!/bin/ksh
# bldmt script file -- PTX
# Builds a C++ multi-threaded embedded SQL program
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# Compile the program.
c++ -Kthread -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.C

# Link the program.
c++ -Kthread -o $1 $1.o -L$DB2PATH/lib -ldb2 -lseq
```

除了以上讨论的 `-Kthread` 选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第238页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sqC` 构建样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入：

```
thdsrver
```

第11章 构建 Silicon Graphics IRIX 应用程序

MIPSpro C	248	用户定义函数 (UDF) 的客户机应用程序	255
DB2 CLI 应用程序	248	多线程应用程序	256
构建和运行嵌入式 SQL 应用程序	250	MIPSpro C++	257
使用 DB2 API 的 DB2 CLI 应用程序	251	DB2 API 和嵌入式 SQL 应用程序	257
存储过程的 DB2 CLI 客户机应用程序	251	存储过程的嵌入式 SQL 客户机应用程序	259
UDF 的 DB2 CLI 客户机应用程序	252	UDF 的嵌入式 SQL 客户机应用程序	259
DB2 API 和嵌入式 SQL 应用程序	252	多线程应用程序	260
构建和运行嵌入式 SQL 应用程序	254		
存储过程的嵌入式 SQL 客户机应用程序	255		

本章提供有关在 Silicon Graphics IRIX 上构建 DB2 应用程序的详细信息。在脚本文件中，以 db2 开始的命令是“命令行处理器”(CLP)命令。有关 CLP 命令的详情，参考 *Command Reference*。

有关 Silicon Graphics IRIX 上最新 DB2 应用程序开发情况的更新信息，请访问以下的 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

DB2 Silicon Graphics IRIX 版只用于客户机。要运行 DB2 应用程序并要构建 DB2 嵌入式 SQL 应用程序，您需要从客户机存取服务器上的 DB2 数据库。该服务器可能正运行着不同的操作系统。有关配置客户机至服务器通信的信息，参阅 *DB2 (UNIX 版) 快速入门*。

另外，因为您将从在不同操作系统上运行的远程客户机存取服务器上的数据库，因此您需要将数据库实用程序（包含 DB2 CLI）与该数据库联编。有关详情，参阅第40页的『联编』。

DB2 库支持

Silicon Graphics IRIX 提供三种独立且不兼容的对象类型：o32（缺省类型）、n32（新 32 位对象类型）和 64（64 位对象类型）。DB2 还不支持 64 对象类型，但支持 o32 和 n32 对象类型。

本操作系统有两个相互独立且不兼容的线程 API 版本：sproc 接口和 POSIX 线程 API。DB2 提供对 POSIX 线程 API 的支持。

使用 `sproc` 接口的应用程序可使用非线程版本 DB2 库 `libdb2`，这种版本不具有线程安全特性。使用 `sproc` 接口时务必小心，因为 `libdb2` 不具有 `sproc` 安全特性。

要具备此范围内的功能，DB2 提供了下列库支持：

lib/libdb2.so

无线程的 o32

lib/libdb2_th.so

具有 POSIX 线程的 o32

lib32/libdb2.so

无线程的 n32

lib32/libdb2_th.so

具有 POSIX 线程的 n32

要使用 n32 对象类型，必须使用 `-n32` 选项编译和连接程序，还要将程序与 `lib32/libdb2.so` 或 `lib32/libdb2_th.so` 库连接。要使用缺省的 o32 对象类型，必须将程序与 `lib/libdb2.so` 或 `lib/libdb2_th.so` 库连接，而不使用 `-n32` 选项。

注：要用本章中包含的构建文件构建 n32 对象类型应用程序，应取消指示的命令的注解。

MIPSpro C

本节说明如何使用 MIPSpro C 来构建使用下列种类的 DB2 接口的程序：

- DB2 CLI
- DB2 API
- 嵌入式 SQL

DB2 CLI 应用程序

`sqllib/samples/cli` 中的脚本文件 `bldcli` 包含构建 DB2 CLI 程序所需的命令。
参数 `$1` 指定源文件名

这是唯一的必需参数，且是不包含嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还需提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定该数据库的用户 ID，`$4` 指定口令。

如果该程序包含嵌入式 SQL（根据扩展名 .sql 来判断），则调用 embprep 脚本预编译该程序，生成一个带 .c 扩展名的文件。

```
#!/bin/ksh
# bldcli script file -- Silicon Graphics IRIX
# Builds a CLI program with MIPSpro C.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sql" ]]
then
    embprep $1 $2 $3 $4
fi

# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32

if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
    # Link with db2 n32 object type libraries.
    DB2_LIBPATH=$DB2PATH/lib32
else
    # Link with db2 o32 object type libraries.
    DB2_LIBPATH=$DB2PATH/lib
fi

# Compile the error-checking utility.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.c

# Link the program.
cc $IRIX_OBJECT_MODE -o $1 $1.o utilcli.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
```

bldcli 的编译和连接选项

编译选项:

cc 使用 C 编译程序。

\$IRIX_OBJECT_MODE

如果取消了 'IRIX_OBJECT_MODE=-n32' 的注解，则包含值 "-n32"；否则，不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。

bldcli 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

\$IRIX_OBJECT_MODE

如果取消了 'IRIX_OBJECT_MODE=-n32' 的注解, 则包含值 "-n32"; 否则, 不包含任何值。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2_LIBPATH

指定连接时 DB2 静态库和共享库的位置。对于 o32 对象类型, 它指向: \$DB2PATH/lib; 对于 n32 对象类型, 它指向: \$DB2PATH/lib32。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-rpath \$DB2_LIBPATH

指定运行时 DB2 共享库的位置。对于 o32 对象类型, 它指向: \$DB2PATH/lib; 对于 n32 对象类型, 它指向: \$DB2PATH/lib32。

-lm 与数学库连接。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`, 输入:

```
bldcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令来运行该可执行文件:

```
tbinfo database userid password
```

构建和运行嵌入式 SQL 应用程序

要根据源文件 `dbusemx.sqc` 构建 `dbusemx`, 应包括该数据库和该数据库所在实例的用户 ID 及口令的参数:

```
bldcli dbusemx database userid password
```

产生可执行文件 `dbusemx`。

要运行嵌入式 SQL 应用程序, 输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令:

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包含使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

sqllib/samples/cli 中的脚本文件 bldapi 包含用 DB2 API 构建 DB2 CLI 程序的命令。此文件在 utilapi 实用程序文件（该文件包含创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 bldcli 脚本文件之间唯一的区别。有关 bldapi 和 bldcli 之间公共的编译和连接选项，请参阅第248页的『DB2 CLI 应用程序』。

要根据源文件 dbmconn.c 构建样本程序 dbmconn，输入：

```
bldapi dbmconn
```

产生可执行文件 dbmconn。可输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令来运行该可执行文件：

```
dbmconn database userid password
```

存储过程的 DB2 CLI 客户机应用程序

存储过程是存取数据库并将信息返回给客户机应用程序的程序。存储过程在服务器上编译和存储。该服务器在另一个平台上运行。

要在 DB2 支持的平台服务器上构建 DB2 CLI 存储过程 spserver，参考本书中针对该平台的“构建应用程序”章节。有关 DB2 客户机可存取的其他服务器，参阅第6页的『支持的服务器』。

一旦构建了存储过程 spserver，就可根据源文件 spclient.c，使用脚本文件 bldcli 构建调用该存储过程的客户机应用程序 spclient。有关详情，参考第248页的『DB2 CLI 应用程序』。

可输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令来调用该存储过程：

```
spclient database userid password
```

客户机应用程序存取共享库 spserver，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

UDF 的 DB2 CLI 客户机应用程序

用户定义函数 (UDF) 是在服务器上编译和存储的您自己的标量函数和表函数。该服务器在另一个平台上运行。要在 DB2 支持的平台服务器上构建用户定义函数程序 `udfsrv`，参考本书中针对该平台的“构建应用程序”章节。有关 DB2 客户机可存取的其他服务器，参阅第6页的『支持的服务器』。

一旦构建了 `udfsrv`，可根据 `sqllib/samples/cli` 中的 `udfcli.c` 源文件，使用 DB2 CLI 脚本文件 `bldcli` 构建调用它的 DB2 CLI 客户机应用程序 `udfcli`。有关详情，参考第248页的『DB2 CLI 应用程序』。

要调用该 UDF 程序，输入可执行程序名、数据库名和该数据库所在实例的用户 ID 和口令来运行调用应用程序：

```
udfcli database userid password
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/c` 中的脚本文件 `bldapp` 包含构建 DB2 应用程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是唯一的必需参数，且是不包含嵌入式 SQL 的 DB2 API 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定数据库的用户 ID；第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序，`bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名，则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- Silicon Graphics IRIX
# Builds a C application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlib

# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32

if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
# Link with db2 n32 object type libraries.
DB2_LIBPATH=$DB2PATH/lib32
else
# Link with db2 o32 object type libraries.
DB2_LIBPATH=$DB2PATH/lib
fi
```

```

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
  embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  cc $IRIX_OBJECT_MODE -o $1 $1.o utilemb.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
else
  # Link the program with utilapi.o
  cc $IRIX_OBJECT_MODE -o $1 $1.o utilapi.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
fi

```

bldapp 的编译和连接选项

编译选项:

cc 使用 C 编译程序。

\$IRIX_OBJECT_MODE

如果取消了 'IRIX_OBJECT_MODE=-n32' 的注解, 则包含值 "-n32"; 否则, 不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sql/lib/include

-c 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

bldapp 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

\$IRIX_OBJECT_MODE

如果取消了 'IRIX_OBJECT_MODE=-n32' 的注解, 则包含值 "-n32"; 否则, 不包含任何值。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilemb.o

如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.o

如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

-L\$DB2_LIBPATH

指定连接时 DB2 静态库和共享库的位置。对于 o32 对象类型, 它指向: \$DB2PATH/lib; 对于 n32 对象类型, 它指向: \$DB2PATH/lib32。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-rpath \$DB2_LIBPATH

指定运行时 DB2 共享库的位置。对于 o32 对象类型, 它指向: \$DB2PATH/lib; 对于 n32 对象类型, 它指向: \$DB2PATH/lib32。

-lm 与数学库连接。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.c` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。

要运行该可执行文件, 输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令:

```
client database userid password
```

构建和运行嵌入式 SQL 应用程序

要根据源文件 `updat.sqc` 构建样本程序 `updat`, 应包括该数据库和该数据库所在实例的用户 ID 及口令的参数:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。要对 `sample` 数据库运行该可执行文件，输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令：

```
updat database userid password
```

存储过程的嵌入式 SQL 客户机应用程序

存储过程是存取数据库并将信息返回给客户机应用程序的程序。存储过程在服务器上编译和存储。该服务器在另一个平台上运行。

要在 DB2 支持的平台上构建嵌入式 SQL 存储过程 `spserver`，参考本书中针对该平台的“构建应用程序”章节。有关 DB2 客户机可存取的其他服务器，参阅第6页的『支持的服务器』。

一旦构建了存储过程 `spserver`，就可构建调用此存储过程的客户机应用程序。可根据源文件 `spclient.sqc`，使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第252页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用该存储过程，输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令来运行客户机应用程序：

```
spclient database userid password
```

客户机应用程序存取存储过程库 `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF) 的客户机应用程序

用户定义函数 (UDF) 是在服务器上编译和存储的您自己的标量函数和表函数。该服务器在另一个平台上运行。要在 DB2 支持的平台上构建用户定义函数程序 `udfsrv`，参考本书中针对该平台的“构建应用程序”章节。有关 DB2 客户机可存取的其他服务器，参阅第6页的『支持的服务器』。

一旦构建了 `udfsrv`，就可根据 `sqllib/samples/c` 中的 `udfcli.sqc` 源文件，使用脚本文件 `bldapp` 构建调用它的嵌入式 SQL 客户机应用程序 `udfcli`。有关详情，参考第252页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用该 UDF 程序，输入可执行程序名、数据库名和该数据库所在实例的用户 ID 和口令来运行调用应用程序：

```
udfcli database userid password
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

Silicon Graphics IRIX 上的多线程的应用程序需要使用 `-ldb2_th` 和 `-lpthread` 连接选项与 `o32` 或 `n32` 对象类型的 DB2 库的 POSIX 线程版本连接。

`sqlllib/samples/c` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID，第四个参数 `$4` 指定口令。

```
#!/bin/ksh
# bldmt script file -- Silicon Graphics IRIX
# Builds a C multi-threaded embedded SQL program
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlllib

# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32

if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
# Link with db2 n32 object type libraries.
DB2_LIBPATH=$DB2PATH/lib32
else
# Link with db2 o32 object type libraries.
DB2_LIBPATH=$DB2PATH/lib
fi

# Precompile and bind the program.
embprep $1 $2 $3 $4

# Compile the program.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.c

# Link the program.
cc $IRIX_OBJECT_MODE -o $1 $1.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2_th -lpthread
```

除了以上讨论的 `-ldb2_th` 和 `-lpthread` 连接选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第252页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sqc` 构建样本程序 `thdsrver`，应包括该数据库和该数据库所在实例的用户 ID 及口令的参数：

```
bldmt thdsrver database userid password
```

产生可执行文件 `thdsrver`。

要运行该可执行文件，输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令：

```
thdsrver database userid password
```

MIPSpro C++

本节包括下列主题:

- DB2 API 和嵌入式 SQL 应用程序
- 多线程应用程序

DB2 API 和嵌入式 SQL 应用程序

sqllib/samples/cpp 中的脚本文件 bldapp 包含构建 DB2 应用程序所需的命令。

第一个参数 \$1 指定源文件的名称。这是非嵌入式 SQL 应用程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库, 因此还提供了三个可选的参数: 第二个参数 \$2 指定您想连接的数据库的名称; 第三个参数 \$3 指定该数据库的用户 ID, 第四个参数 \$4 指定口令。

对于嵌入式 SQL 程序, bldapp 将这些参数传送给预编译和联编文件 embprep。

```
#!/bin/ksh
# bldapp script file -- Silicon Graphics IRIX
# Builds a C++ application program
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32

if [ "$IRIX_OBJECT_MODE" = "-n32" ]; then
    # Link with db2 n32 object type libraries.
    DB2_LIBPATH=$DB2PATH/lib32
else
    # Link with db2 o32 object type libraries.
    DB2_LIBPATH=$DB2PATH/lib
fi

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.C error-checking utility.
    CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilemb.C
else
    # Compile the utilapi.c error-checking utility.
    CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.C

if [[ -f $1".sqc" ]]
then
    # Link the program with utilemb.o
    CC $IRIX_OBJECT_MODE -o $1 $1.o utilemb.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
```

```

else
  # Link the program with utilapi.o
  CC $IRIX_OBJECT_MODE -o $1 $1.o utilapi.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
fi

```

bldapp 的编译和连接选项	
编译选项:	
CC	使用 C++ 编译程序。
\$IRIX_OBJECT_MODE	如果取消了 'IRIX_OBJECT_MODE=-n32' 的注解, 则包含值 "-n32"; 否则, 不包含任何值。
-I\$DB2PATH/include	指定 DB2 包含文件的位置。 例如: \$HOME/sql/lib/include
-c	只执行编译; 不连接。此脚本具有独立的编译和连接步骤。
连接选项:	
CC	使用编译程序作为连接程序的前端。
\$IRIX_OBJECT_MODE	如果取消了 'IRIX_OBJECT_MODE=-n32' 的注解, 则包含值 "-n32"; 否则, 不包含任何值。
-o \$1	指定可执行文件。
\$1.o	包括该程序的目标文件。
utilemb.o	如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。
utilapi.o	如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。
-L\$DB2_LIBPATH	指定连接时 DB2 静态库和共享库的位置。对于 o32 对象类型, 它指向: \$DB2PATH/lib; 对于 n32 对象类型, 它指向: \$DB2PATH/lib32。如果不指定 -L 选项, 则假定 /usr/lib:/lib。
-rpath \$DB2_LIBPATH	指定运行时 DB2 共享库的位置。对于 o32 对象类型, 它指向: \$DB2PATH/lib; 对于 n32 对象类型, 它指向: \$DB2PATH/lib32。
-lm	与数学库连接。
-ldb2	连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `updat.sqC` 构建样本程序 `updat`，应包括该数据库和该数据库所在实例的用户 ID 及口令的参数：

```
bldapp updat database userid password
```

产生可执行文件 `updat`。要运行该可执行文件，输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令：

```
updat database userid password
```

存储过程的嵌入式 SQL 客户机应用程序

存储过程是存取数据库并将信息返回给客户机应用程序的程序。存储过程在服务器上编译和存储。该服务器在另一个平台上运行。

要在 DB2 支持的平台服务器上构建嵌入式 SQL 存储过程 `spserver`，参考本书中针对该平台的“构建应用程序”章节。有关 DB2 客户机可存取的其他服务器，参考第6页的『支持的服务器』。

一旦构建了存储过程 `spserver`，就可构建调用此存储过程的客户机应用程序。可根据源文件 `spclient.sqC` 使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第257页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用该存储过程，输入可执行文件名、数据库名和该数据库所在实例的用户 ID 和口令来运行客户机应用程序：

```
spclient database userid password
```

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数。存储过程将输出返回给客户机应用程序。

UDF 的嵌入式 SQL 客户机应用程序

用户定义函数 (UDF) 是在服务器上编译和存储的您自己的标量函数。该服务器在另一个平台上运行。要在 DB2 支持的平台服务器上构建用户定义函数程序 `udfsrv`，参考本书中针对该平台的“构建应用程序”章节。有关 DB2 客户机可存取的其他服务器，参阅第6页的『支持的服务器』。

一旦构建了 `udfsrv`，就可根据 `sqllib/samples/cpp` 中的 `udfcli.sqC` 源文件，使用脚本文件 `bldapp` 构建调用它的嵌入式 SQL 客户机应用程序 `udfcli`。有关详情，参考第257页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用该 UDF 程序，输入可执行程序名、数据库名和该数据库所在实例的用户 ID 和口令来运行调用应用程序：

```
udfcli database userid password
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

Silicon Graphics IRIX 上的多线程应用程序需要使用 `-ldb2_th` 和 `-lpthread` 连接选项与 `o32` 或 `n32` 对象类型的 DB2 库的 POSIX 线程版本连接。

`sqllib/samples/cpp` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID，第四个参数 `$4` 指定口令。

```
#!/bin/ksh
# bldmt script file -- Silicon Graphics IRIX
# Builds a C++ multi-threaded embedded SQL program
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqlib

# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32

if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
# Link with db2 n32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib32
else
# Link with db2 o32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib
fi

# Precompile and bind the program.
embprep $1 $2 $3 $4

# Compile the program.
CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.C

# Link the program.
CC $IRIX_OBJECT_MODE -o $1 $1.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2_th -lpthread
```

除了以上讨论的 `-ldb2_th` 和 `-lpthread` 连接选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第257页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sqC` 构建样本程序 `thdsrver`，应包括该数据库和该数据库所在实例的用户 ID 及口令的参数：

```
bldmt thdsrver database userid password
```

产生可执行文件 `thdsrver`。

要对 `sample` 数据库运行该可执行文件，输入可执行文件名、数据库名和该数据库所在实例的用户 `ID` 和口令：

```
thdsrver database userid password
```

第12章 构建 Solaris 应用程序

SPARCompiler C	263	DB2 API 和嵌入式 SQL 应用程序	279
DB2 CLI 应用程序	264	构建和运行嵌入式 SQL 应用程序	281
构建和运行嵌入式 SQL 应用程序	266	嵌入式 SQL 存储过程	282
使用 DB2 API 的 DB2 CLI 应用程序	266	用户定义函数 (UDF)	285
DB2 CLI 存储过程	267	多线程应用程序	288
DB2 API 和嵌入式 SQL 应用程序	269	Micro Focus COBOL	289
构建和运行嵌入式 SQL 应用程序	272	使用编译程序	289
嵌入式 SQL 存储过程	272	DB2 API 和嵌入式 SQL 应用程序	289
用户定义函数 (UDF)	275	构建和运行嵌入式 SQL 应用程序	291
多线程应用程序	278	嵌入式 SQL 存储过程	292
SPARCompiler C++	279	退出存储过程	296

本章提供在 Solaris 操作环境中构建应用程序的详细信息。在脚本文件中，以 db2 开始的命令是“命令行处理器”(CLP)命令。有关 CLP 命令的详情，参考 *Command Reference*。

有关 Solaris 操作环境的最新 DB2 应用程序开发的更新，请访问以下的 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

注：

1. 由于在 Solaris 操作环境中实现线程的方法原因，在 DB2 构建文件和 makefile 的连接步骤中使用 `-mt` 多线程选项。这可能会稍微降低性能。如果要获得最佳性能，可尝试不用此选项连接应用程序，而使用非线程 `libdb2.so` 库进行连接。但是，如果不使用 `-mt` 开关，当运行应用程序时可能收到类似如下的信息。

```
libc internal error: _rmutex_unlock: rmutex not held
```

或者，应用程序可能挂起而不发出任何错误信息。

2. 要用本章中包含的构建文件构建 64 位应用程序，取消指示的命令的注解。

SPARCompiler C

本节包括下列主题：

- DB2 CLI 应用程序
- 使用 DB2 API 的 DB2 CLI 应用程序
- DB2 CLI 存储过程

- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 CLI 应用程序

sqllib/samples/cli 中的脚本文件 bldcli 包含构建 DB2 CLI 程序所需的命令。参数 \$1 指定源文件的名称。

这是不包含嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

如果该程序包含嵌入式 SQL（根据扩展名 .sql 来判断），则调用脚本 embprep 来预编译该程序，生成一个扩展名为 .c 的程序文件。

```
#!/bin/ksh
# bldcli script file -- Solaris
# Builds a DB2 CLI program.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sql" ]]
then
    embprep $1 $2 $3 $4
fi

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
fi

# Compile the error-checking utility.
cc $CFLAGS_64 -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc $CFLAGS_64 -I$DB2PATH/include -c $1.c

# Link the program.
cc $CFLAGS_64 -o $1 $1.o utilcli.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
```


bldcli 的编译和连接选项

编译选项:

cc 使用 C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

连接选项:

cc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-o \$1 指定可执行程序。

\$1.o 包括该程序的目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqllib/lib

-R\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sqllib/lib

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`, 输入:

```
bldcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名, 运行该可执行文件:

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldcli dbusemx database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldcli dbusemx database userid password
```

产生可执行文件 `dbusemx`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
dbusemx database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序, 以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

`sqlllib/samples/cli` 中的脚本文件 `bldapi` 包含使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 `utilapi` 实用程序文件 (该文件包含创建和卸下数据库所需的 DB2 API) 中编译和连接。这是此文件与 `bldcli` 脚本文件之间唯一的区别。有关 `bldapi` 和 `bldcli` 之间公共的编译和连接选项, 请参阅第264页的『DB2 CLI 应用程序』。

要根据源文件 `dbmconn.c` 构建样本程序 `dbmconn`, 输入:

```
bldapi dbmconn
```

产生可执行文件 `dbmconn`。可输入可执行文件名, 运行该可执行文件:

```
dbmconn
```

DB2 CLI 存储过程

sqllib/samples/cli 中的脚本文件 bldclisp 包含构建 DB2 CLI 存储过程所需的命令。参数 \$1 指定源文件的名称。

```
#!/bin/ksh
# bldclisp script file -- Solaris
# Builds a DB2 CLI stored procedure.
# Usage: bldclisp <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
fi

# Compile the error-checking utility.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c

# Link the program.
cc $CFLAGS_64 -G -o $1 $1.o utilcli.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp 的编译和连接选项

编译选项:

cc C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-Kpic 为共享库生成与位置无关的代码。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqlllib/include。

-c 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。

连接选项:

cc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilcli.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-R\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sqlllib/lib。

-ldb2 连接 DB2 库。

-G 生成共享库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 spserver.c 构建样本程序 spserver, 输入:

```
bldclisp spserver
```

该脚本文件将存储过程复制到服务器的 sqlllib/function 路径中。

接着在服务器上运行 spcreate.db2 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目，可使用以下命令卸下它们：

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`，就可构建调用该存储过程的 CLI 客户机应用程序 `spclient`。

可使用脚本文件 `bldcli` 构建 `spclient`。有关详情，参考第264页的『DB2 CLI 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/c` 中的脚本文件 `bldapp` 包含构建 DB2 应用程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定数据库的用户 ID；第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序, bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名, 则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时, 才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- Solaris
# Builds a C application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
fi

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
    embprep $1 $2 $3 $4
    # Compile the utilemb.c error-checking utility.
    cc $CFLAGS_64 -I$DB2PATH/include -c utilemb.c
else
    # Compile the utilapi.c error-checking utility.
    cc $CFLAGS_64 -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
cc $CFLAGS_64 -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
    # Link the program with utilemb.o
    cc $CFLAGS_64 -o $1 $1.o utilemb.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
else
    # Link the program with utilapi.o
    cc $CFLAGS_64 -o $1 $1.o utilapi.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
fi
```

bldapp 的编译和连接选项

编译选项:

cc C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

连接选项:

cc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilemb.o

如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.o

如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

-\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sqllib/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.c` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。

要运行可执行文件, 输入可执行文件名:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

`sqllib/samples/c` 中的脚本文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例上构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldsrv script file -- Solaris
# Builds a C stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
```



```

embprep $1 $2

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
fi

# Compile the program.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c

# Link the program and create a shared library
cc $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bidsrv 的编译和连接选项

编译选项:

cc C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-Kpic 为共享库生成与位置无关的代码。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: **-I\$DB2PATH/include**

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldsrv 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-G 生成共享库。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-R\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sql1lib/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv spserver
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv spserver database
```

此脚本文件将存储过程复制到 `sql1lib/function` 目录中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`，就可构建调用该存储过程的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第269页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。 该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

`sqllib/samples/c` 中的脚本文件 `bldudf` 包含构建 UDF 所需的命令。UDF 不包含嵌入式 SQL 语句。因此，要创建 UDF 程序，不必连接数据库或预编译并链接该程序。

参数 `$1` 指定源文件名 此脚本文件将源文件名用作共享库名。

```
#!/bin/ksh
# bldudf script file -- Solaris
# Builds a C UDF library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
```

```

fi

# Compile the program.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
cc $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldudf 的编译和连接选项

编译选项:

cc C 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解，则包含值 "-xarch=v9"；否则，不包含任何值。

-Kpic 为共享库生成与位置无关的代码。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。

-c 只执行编译；不连接。此脚本文件有单独的编译和连接步骤。

bldudf 的编译和连接选项

连接选项:

cc 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-R\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sqllib/lib。

-ldb2 连接 DB2 库。

-ldb2apie

连接“DB2 API 引擎”库, 以便可使用 LOB 定位器。

-G 生成共享库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到 `sqllib/function` 目录中。

必要时, 将该 UDF 设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。提供了此程序的 DB2 CLI 和嵌入式 SQL 版本。

可根据 `sqllib/samples/cli` 中的源文件 `udfcli.c`, 使用脚本文件 `bldcli` 构建 DB2 CLI `udfcli` 程序。有关详情, 参考第264页的『DB2 CLI 应用程序』。

可根据 `sqllib/samples/c` 中的源文件 `udfcli.sqc`, 使用脚本文件 `bldapp` 构建嵌入式 SQL `udfcli` 程序。有关详情, 参考第269页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF, 可输入可执行文件名运行样本调用应用程序:

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

在 Solaris 上使用 SPARCCompiler C 的多线程应用程序需要用 `-mt` 进行编译和连接。它将 `-D_REENTRANT` 传送至预处理程序，并将 `-lthread` 传送至连接程序。POSIX 线程也要求将 `-lpthread` 传送给连接程序。另外，使用编译程序选项 `-D_POSIX_PTHREAD_SEMANTICS` 允许函数的 POSIX 变体，如 `getpwnam_r()`。

`sqllib/samples/c` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID，第四个参数 `$4` 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldmt script file -- Solaris
# Builds a C multi-threaded embedded SQL program.
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
fi

# Compile the program.
cc $CFLAGS_64 -mt -D_POSIX_PTHREAD_SEMANTICS -I$DB2PATH/include -c $1.c

# Link the program.
cc $CFLAGS_64 -mt -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -lpthread
```

除了以上讨论的 `-mt`、`-D_POSIX_PTHREAD_SEMANTICS` 和 `-lpthread` 选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第269页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据 `thdsrver.sqc` 源文件构建样本程序 `thdsrver`，输入：

```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入：

```
thdsrver
```

SPARCompiler C++

本节包括下列主题：

- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)
- 多线程应用程序

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/cpp` 中的脚本文件 `bldapp` 包含构建 DB2 应用程序所需的命令。

第一个参数 `$1` 指定源文件的名称。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `$2` 指定您想连接的数据库的名称；第三个参数 `$3` 指定数据库的用户 ID；第四个参数 `$4` 指定口令。

对于嵌入式 SQL 程序，`bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名，则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- Solaris
# Builds a C++ application program.
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
fi
```

```

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
  embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  CC $CFLAGS_64 -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  CC $CFLAGS_64 -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
CC $CFLAGS_64 -I$DB2PATH/include -c $1.C

if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  CC $CFLAGS_64 -o $1 $1.o utilemb.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -mt
else
  # Link the program with utilapi.o
  CC $CFLAGS_64 -o $1 $1.o utilapi.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -mt
fi

```

bldapp 的编译和连接选项

编译选项:

CC C++ 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sql/lib/include

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldapp 的编译和连接选项

连接选项:

CC 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

utilemb.o

如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.o

如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-R\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sqllib/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.C` 构建非嵌入式 SQL DB2 API 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqc` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

注: 请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ 存储过程的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldsrv` 包含构建嵌入式 SQL 存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

此脚本文件将源文件名 `$1` 用作共享库名。

```
#!/bin/ksh
# bldsrv script file -- Solaris
# Builds a C++ stored procedure
# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
```

```

else
  CFLAGS_64=
fi

# Compile the program.
CC $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.C

# Link the program and create a shared library
CC $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -mt

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldsrv 的编译和连接选项

编译选项:

CC C++ 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-Kpic 为共享库生成与位置无关的代码。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: `-I$DB2PATH/include`

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldsrv 的编译和连接选项

连接选项:

CC 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-G 生成共享库。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-R\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sql1lib/lib。

-ldb2 连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.sqc` 构建样本程序 `spserver`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv spserver
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv spserver database
```

此脚本文件将共享库复制到服务器的 `sql1lib/function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了共享库 `spserver`，就可构建调用共享库中的存储过程的客户机应用程序 `spclient`。

可使用脚本文件 `bldapp` 构建 `spclient`。有关详情，参考第279页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用共享库中的存储过程，可输入以下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。 该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取共享库 `spserver`，并在服务器数据库上执行大量存储过程函数。存储过程将输出返回给客户机应用程序。

用户定义函数 (UDF)

注：请参阅第54页的『用 C++ 编写 UDF 和存储过程的考虑事项』中有关构建 C++ UDF 的信息。

`sqllib/samples/cpp` 中的脚本文件 `bldudf` 包含构建 UDF 所需的命令。UDF 不包含嵌入式 SQL 语句。因此，要创建 UDF 程序，不必连接数据库或预编译并链接该程序。

参数 `$1` 指定源文件名。此脚本文件将源文件名用作共享库名。

```
#!/bin/ksh
# bldudf script file -- Solaris
# Builds a C++ UDF library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
```

```

then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi

# Compile the program.
if [[ -f $1".c" ]]
then
  CC $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c
elif [[ -f $1".C" ]]
then
  CC $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.C
fi

# Link the program and create a shared library.
CC $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldudf 的编译和连接选项

编译选项:

CC C++ 编译程序。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-Kpic 为共享库生成与位置无关的代码。

-I\$DB2PATH/include

指定 DB2 包含文件的位置。例如: \$HOME/sqllib/include。

-c 只执行编译; 不连接。此脚本文件有单独的编译和连接步骤。

bldudf 的编译和连接选项

连接选项:

CC 使用编译程序作为连接程序的前端。

\$CFLAGS_64

如果取消了 'BUILD_64BIT=true' 的注解, 则包含值 "-xarch=v9"; 否则, 不包含任何值。

-G 生成共享库。

-o \$1 指定可执行文件。

\$1.o 包括该程序的目标文件。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sqlllib/lib。如果不指定 -L 选项, 则假定 /usr/lib:/lib。

-R\$DB2PATH/lib

指定运行时 DB2 共享库的位置。例如: \$HOME/sqlllib/lib。

-ldb2 连接 DB2 库。

-ldb2apie

连接“DB2 API 引擎”库, 以便可使用 LOB 定位器。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数程序 `udfsrv`, 输入:

```
bldudf udfsrv
```

此脚本文件将 UDF 复制到 `sqlllib/function` 目录中。

必要时, 将该 UDF 设置为文件方式, 以便客户机应用程序可存取它。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。可根据 `sqlllib/samples/cpp` 中的 `udfcli.sqC` 源文件, 使用脚本文件 `bldapp` 来构建 `udfcli` 程序。有关详情, 参考第279页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用 UDF, 可输入可执行文件名运行样本调用应用程序:

```
udfcli
```

调用应用程序从 `udfsrv` 库中调用 `ScalarUDF` 函数。

多线程应用程序

在 Solaris 上使用 SPARCompiler C++ 的多线程应用程序需要用 `-mt` 进行编译和连接。它将 `-D_REENTRANT` 传送到预处理程序，并将 `-lthread` 传送到连接程序。POSIX 线程也要求将 `-lpthread` 传送给连接程序。另外，使用编译程序选项 `-D_POSIX_PTHREAD_SEMANTICS` 允许函数的 POSIX 变体，如 `getpwnam_r()`。

`sqllib/samples/cpp` 中的脚本文件 `bldmt` 包含构建嵌入式 SQL 多线程程序所需的命令。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。第三个参数 `$3` 指定数据库的用户 ID，第四个参数 `$4` 指定口令。只有第一个参数即源文件名是必需的。数据库名、用户 ID 和口令是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

```
#!/bin/ksh
# bldmt script file -- Solaris
# Builds a C++ multi-threaded embedded SQL program
# Usage: bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2 $3 $4

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true

if [ "$BUILD_64BIT" != "" ]
then
    CFLAGS_64=-xarch=v9
else
    CFLAGS_64=
fi

# Compile the program.
CC $CFLAGS_64 -mt -D_POSIX_PTHREAD_SEMANTICS -I$DB2PATH/include -c $1.C

# Link the program.
CC $CFLAGS_64 -mt -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -lpthread
```

除了以上讨论的 `-mt`、`-D_POSIX_PTHREAD_SEMANTICS` 和 `-lpthread` 选项之外，在没有连接任何实用程序文件的情况下，其他编译和连接选项与嵌入式 SQL 脚本文件 `bldapp` 所用的相同。有关这些选项的信息，参阅第279页的『DB2 API 和嵌入式 SQL 应用程序』。

要根据源文件 `thdsrver.sqC` 构建样本程序 `thdsrver`，输入：


```
bldmt thdsrver
```

产生可执行文件 `thdsrver`。要对 `sample` 数据库运行该可执行文件，输入：

```
thdsrver
```

Micro Focus COBOL

本节包括下列主题：

- 使用编译程序
- DB2 API 和 DB2 嵌入式应用程序
- 嵌入式 SQL 存储过程

使用编译程序

如果开发包含嵌入式 SQL 和 DB2 API 调用的应用程序，且使用的是 Micro Focus COBOL 编译程序，则切记以下几点：

- 当使用命令行处理器命令 `db2 prep` 预编译应用程序时，使用 `target mfcob` 选项（缺省项）。
- 为了使用内部预编译程序前端、运行期解释程序或动画制作程序调试程序，可执行由 Micro Focus 提供的 `mkrts` 命令，将“DB2 类属 API”入口点添加到 Micro Focus 运行期模块 `rts32` 中，如下所示：

1. 作为超级用户注册。
2. 带以下目录中提供的自变量执行 `mkrts` 命令：

```
/opt/IBMdb2/V7.1/lib/db2mkrts.args
```

- 在 Micro Focus COBOL 环境变量 `COBCPY` 中必须包括 DB2 COBOL COPY 文件目录。`COBCPY` 环境变量指定 `COPY` 文件的位置。Micro Focus COBOL 的 DB2 COPY 文件驻留在该数据库实例目录下的 `sqllib/include/cobol_mf` 中。

要包括此目录，输入：

```
export COBCPY=$COBCPY:$HOME/sqllib/include/cobol_mf
```

注：应在 `.profile` 文件中设置 `COBCPY`。

DB2 API 和嵌入式 SQL 应用程序

`sqllib/samples/cobol_mf` 中的脚本文件 `bldapp` 包含构建 DB2 应用程序所需的命令。

第一个参数 \$1 指定源文件的名称。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 \$2 指定您想连接的数据库的名称；第三个参数 \$3 指定数据库的用户 ID；第四个参数 \$4 指定口令。

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
#!/bin/ksh
# bldapp script file -- Solaris
# Builds a Micro Focus COBOL application program
# Usage: bldapp [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
    embprep $1 $2 $3 $4
fi

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY

# Compile the checkerr.cbl error-checking utility.
cob -cx checkerr.cbl

# Compile the program.
cob -cx $1.cbl

# Link the program.
cob -x $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

bldapp 的编译和连接选项

编译选项:

- cob** Micro Focus COBOL 编译程序。
- cx** 编译为目标模块。

bldapp 的编译和连接选项

连接选项:

cob 使用编译程序作为连接程序的前端。

-x 指定可执行程序。

\$1.o 包括该程序的目标文件。

checkerr.o

包括该实用程序的目标文件以便检查错误。

-L\$DB2PATH/lib

指定连接时 DB2 静态库和共享库的位置。例如: \$HOME/sql1lib/lib。

-ldb2 连接 DB2 库。

-ldb2gmf

连接 Micro Focus COBOL 的 DB2 异常处理程序库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `client.cbl` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client`。可以输入如下命令, 对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入该数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名:

```
updat database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名以及该数据库实例的用户 ID 和口令：

```
updat database userid password
```

嵌入式 SQL 存储过程

注：

1. 在 Solaris 上构建 Micro Focus 存储过程之前，运行以下命令：

```
db2stop
db2set DB2LIBPATH=$LD_LIBRARY_PATH
db2set DB2ENVLIST="COBDIR LD_LIBRARY_PATH"
db2set
db2start
```

确保使用命令 `db2stop` 停止数据库。发出最后一个 `db2set` 命令检查设置：确保 `DB2LIBPATH` 和 `DB2ENVLIST` 设置正确。

2. 某些在 Solaris 上使用的最新版本的 Micro Focus COBOL 编译程序，不能用来创建静态连接的存储过程。因此，已修改 `makefile` 和脚本文件 `bldsrv`，以便可创建动态连接的存储过程。

为使远程客户机应用程序可以成功地调用此动态连接的存储过程，在执行该存储过程之前，需要在该存储过程所驻留的服务器上调用 Micro Focus COBOL 例程 `cobinit()`。完成此操作的封装器程序是在 `makefile` 或脚本文件 `bldsrv` 的执行期间创建的。然后将它与存储过程代码连接，以构成存储过程的共享库。因为使用此封装器程序，所以要使客户机应用程序调用名为 `x` 的存储过程，它必须调用 `x_wrap` 而不是 `x`。

本节后面会解释该封装器程序的详情。

`sqllib/samples/cobol_mf` 中的脚本文件 `bldsrv` 包含构建存储过程所需的命令。此脚本文件将存储过程编译为可由客户机应用程序调用的共享库。

第一个参数 `$1` 指定源文件的名称。第二个参数 `$2` 指定要连接的数据库名称。因为必须在数据库所在的同一实例上构建存储过程，所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

此脚本文件将源文件名 `$1` 用作共享库名。

```
#!/bin/ksh
# bldsrv script file -- Solaris
# Builds a Micro Focus COBOL stored procedure
```

```

# Usage: bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
embprep $1 $2

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY

# Compile the program.
cob -cx $1.cbl

# Create the wrapper program for the stored procedure.
wrapsrv $1

# Link the program creating shared library $1 with main entry point ${1}_wrap
cob -x -o $1 ${1}_wrap.c $1.o -Q -G -L$DB2PATH/lib -ldb2 -ldb2gmf

# Copy the shared library to the sqllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function

```

bldsrv 的编译和连接选项

编译选项:

cob	COBOL 编译程序。
-cx	编译为目标模块。

bldsrv 的编译和连接选项

连接选项:

cob 使用编译程序来连接编辑。

-x 产生可执行程序。

-o \$1 指定可执行程序。

\${1}_wrap.c
指定封装器程序。

\$1.o 指定该程序的目标文件。

-Q

-G

-L\$DB2PATH/lib

指定 DB2 运行期共享库的位置。例如: \$HOME/sql1lib/lib。如果不指定 -L 选项, 则编译程序假定如下路径: /usr/lib:/lib。

-ldb2 连接 DB2 库。

-ldb2gmf

与 Micro Focus COBOL 的 DB2 异常处理程序库连接。

有关其他编译程序选项, 参考编译程序文档。

封装器程序 `wrapsrv` 可使 Micro Focus COBOL 例程 `cobinit()` 刚好在执行该存储过程之前被调用。以下显示的是它的内容。

```

#!/bin/ksh
# wrapsrv script file
# Creates the wrapper program for Micro Focus COBOL stored procedures
# Usage: wrapsrv <stored_proc>

# Note: The client program calls "<stored_proc>_wrap" not "<stored_proc>"

# Create the wrapper program for the stored procedure.
cat << WRAPPER_CODE > ${1}_wrap.c
#include <stdio.h>
void cobinit(void);
int $1(void *p0, void *p1, void *p2, void *p3);

int main(void)
{
    return 0;
}

int ${1}_wrap(void *p0, void *p1, void *p2, void *p3)
{
    cobinit();
    return $1(p0, p1, p2, p3);
}
WRAPPER_CODE

```

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`，如果连接 `sample` 数据库，则输入：

```
bldsrv outsrv
```

如果连接另一个数据库，还须输入该数据库名：

```
bldsrv outsrv database
```

该脚本文件将存储过程复制到服务器的 `sqllib/function` 路径中。

必要时，将该存储过程设置为文件方式，以便客户机应用程序可存取它。

一旦构建了存储过程 `outsrv`，就可构建调用此存储过程的客户机应用程序 `outcli`。可使用 `bldapp` 脚本文件构建 `outcli`。有关详情，参考第289页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
outcli database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 `outsrv`，并在服务器数据库上执行同名的存储过程函数，然后将输出返回给客户机应用程序。

退出存储过程

当您开发存储过程时，可使用如下语句退出存储过程：

```
move SQLZ-HOLD-PROC to return-code
```

借助于此语句，存储过程将正确地返回到客户机应用程序。当本地 COBOL 客户机应用程序调用该存储过程时，这一点尤其重要。

第13章 构建 Windows 32 位操作系统的应用程序

Microsoft Visual Basic	299	构建和运行嵌入式 SQL 应用程序	320
ActiveX 数据对象 (ADO)	299	使用 DB2 API 的 DB2 CLI 应用程序	320
远程数据对象 (RDO)	300	DB2 CLI 存储过程	321
对象链接和嵌入 (OLE) 自动化	302	DB2 API 和嵌入式 SQL 应用程序	323
OLE 自动化 UDF 和存储过程	302	构建和运行嵌入式 SQL 应用程序	325
Microsoft Visual C++	302	嵌入式 SQL 存储过程	326
ActiveX 数据对象 (ADO)	303	用户定义函数 (UDF)	329
对象链接和嵌入 (OLE) 自动化	303	IBM VisualAge C++ 版本 4.0	331
OLE 自动化 UDF 和存储过程	304	IBM VisualAge COBOL	331
DB2 CLI 应用程序	304	使用编译程序	331
构建和运行嵌入式 SQL 应用程序	306	DB2 API 和嵌入式 SQL 应用程序	332
使用 DB2 API 的 DB2 CLI 应用程序	307	构建和运行嵌入式 SQL 应用程序	333
DB2 CLI 存储过程	307	嵌入式 SQL 存储过程	334
DB2 API 和嵌入式 SQL 应用程序	310	Micro Focus COBOL	336
构建和运行嵌入式 SQL 应用程序	312	使用编译程序	336
嵌入式 SQL 存储过程	312	DB2 API 和嵌入式 SQL 应用程序	337
用户定义函数 (UDF)	315	构建和运行嵌入式 SQL 应用程序	338
IBM VisualAge C++ 版本 3.5	318	嵌入式 SQL 存储过程	339
DB2 CLI 应用程序	318	Object REXX	341

本章提供在 Windows 32 位操作系统上构建应用程序的详细信息。在批处理文件中，以 db2 开始的命令是“命令行处理器”(CLP) 命令。如果需要有关 DB2 命令的更多信息，参考 *Command Reference*。

有关 Windows 32 位操作系统的最新 DB2 应用程序开发情况的更新信息，请访问以下的 Web 页面：

<http://www.ibm.com/software/data/db2/udb/ad>

注：

1. Windows 32 位操作系统上的所有应用程序，包含嵌入式 SQL 和非嵌入式 SQL，必须在 DB2 命令窗口中构建，而不能在操作系统命令提示符处构建。
2. 程序中使用的任何包括变量 %DB2PATH% 的路径名应引在引号中，如 "%DB2PATH%\function"，如版本 7.1 在 Windows 32 位操作系统上 DB2 的缺省安装为 \Program Files\sql1lib，它包含一个空格。如果不使用引号，会得到这样的一条错误信息：“命令语法错误”。在本章中，这样的路径名如果作为命令或代码示例的一部分来提供，则只把它引在引号中。

WCHARTYPE CONVERT 预编译选项

WCHARTYPE 预编译选项采用多字节格式或使用 `wchar_t` 数据类型的宽字符格式处理图形数据。有关此选项的更多信息可在 *Application Development Guide* 中找到。

对于 DB2 Windows 32 位操作系统版本，用 Microsoft Visual C++ 编译程序编译的应用程序支持 WCHARTYPE CONVERT 选项。但是，如果您的应用程序将数据插入一个 DB2 数据库，而数据所用的代码页与该数据库所用的代码页不同，则在使用此编译程序时不要选择 CONVERT 选项。在这种情况下 DB2 通常会执行代码页转换；但 Microsoft C 运行期环境不处理特定双字节字符的字符替代。这可能导致运行期转换错误。

用 IBM VisualAge C++ 编译程序编译的应用程序不支持 WCHARTYPE CONVERT 选项。对于此编译程序，对 WCHARTYPE 使用缺省选项 NOCONVERT。使用 NOCONVERT 选项，应用程序和数据库管理程序之间不发生隐式字符转换。图形主变量中的数据作为未改变的“双字节字符集”(DBCS) 字符发送到数据库管理程序并从数据库管理程序接收。

如果需要将图形数据由宽字符格式转换到多字节格式，则使用 `wcstombs()` 函数。例如：

```
wchar_t widechar[200];
wchar_t mb[200];
wcstombs((char *)mb,widechar,200);

EXEC SQL INSERTINTO TABLENAME VALUES(:mb);
```

类似地，可使用 `mbstowcs()` 函数从多字节格式转换到宽字符格式。

如果您的应用程序与 C 运行期库静态联编，则不要从应用程序发出 `setlocale()` 调用，因为这可能导致 C 运行期转换错误。如果应用程序与 C 运行期库动态联编，则可使用 `setlocale()`。对存储过程情况也如此。

对象链接和嵌入数据库 (OLE DB) 表函数

DB2 支持 OLE DB 表函数。对于这些函数，除创建 CREATE FUNCTION DDL 外不需要进行应用程序构建。DB2 在 `%DB2PATH%\samples\oledb` 目录中提供了 OLE DB 表函数样本文件。它们是“命令行处理器”(CLP) 文件。可用下列步骤构建它们：

1. `db2 connect to database_name`
2. `db2 -t -v -f file_name.db2`
3. `db2 terminate`

其中 `database_name` 是要连接的数据库，而 `file_name` 是 CLP 文件的名称，扩展名为 `.db2`。

有关 OLE DB 表函数的完整说明，参阅 *Application Development Guide*。

Microsoft Visual Basic

注：DB2 AD 客户机 Windows 32 位操作系统版没有为 Microsoft Visual Basic 提供预编译程序。

本节包括下列主题：

- ActiveX 数据对象 (ADO)
- 远程数据对象 (RDO)
- 对象链接和嵌入 (OLE) 自动化

ActiveX 数据对象 (ADO)

“ActiveX 数据对象” (ADO) 用来编写一个应用程序，以便通过 OLE DB 提供者存取和操纵数据库服务器中的数据。ADO 的主要优点是高速、易用、低内存消耗以及占用磁盘少。

要在 Microsoft Visual Basic 中使用 ADO，需要建立对 ADO 类型库的引用。执行下列各项操作：

1. 从“项目”菜单选择“引用”
2. 选择“Microsoft ActiveX 数据对象 <version_number> 库”的校验框
3. 单击“确认”。

其中 <version_number> 是 ADO 库的当前版本。

一旦完成此操作，则可通过“VBA 对象浏览器”和“IDE 编辑器”存取 ADO 对象、方法和特性。

一个完整的 Visual Basic 程序包括表单和其他图形元素，您需要在 Visual Basic 环境中查看它。以下是作为一个程序一部分的 Visual Basic 命令，该程序用来存取编目在 ODBC 中的 DB2 sample 数据库：

建立连接：

```
Dim db As Connection
Set db = New Connection
```

设置本地游标库提供的客户机端游标：

```
db.CursorLocation = adUseClient
```

设置提供器以便 ADO 可使用“Microsoft ODBC 驱动程序”，不用用户 ID / 口令打开数据库“sample”；即使用当前用户：

```
db.Open "SAMPLE"
```

创建一个记录集：

```
Set adoPrimaryRS = New Recordset
```

使用 select 语句填充该记录集：

```
adoPrimaryRS.Open "select EMPNO, LASTNAME, FIRSTNAME, MIDINIT, EDLEVEL, JOB  
from EMPLOYEE Order by EMPNO", db
```

此时，程序员可使用 ADO 方法存取数据，如移动到下一个记录集：

```
adoPrimaryRS.MoveNext
```

删除记录集中的当前记录：

```
adoPrimaryRS.Delete
```

程序员还可执行下列操作来存取个别字段：

```
Dim Text1 as String  
Text1 = adoPrimaryRS!LASTNAME
```

DB2 在 %DB2PATH%\samples\ADO\VB 目录中提供了 Visual Basic ADO 样本程序。

远程数据对象 (RDO)

“远程数据对象” (RDO) 提供一个通过 ODBC 存取远程数据源的信息模型。RDO 提供一个对象集，它使得与数据库连接、执行查询和存储过程、处理结果和落实服务器更改更容易。RDO 是为存取远程 ODBC 关系数据源而专门设计的，有了它，使用 ODBC 更容易，而不必进行复杂的应用程序编程，它也是通过 ODBC 驱动程序存取关系数据库的主要途径。RDO 通过“开放式数据库链接” (ODBC) API 和驱动程序管理器实现一个薄代码层，该代码层使用最少的工作站资源来建立连接、创建结果集和游标，执行复杂的过程。

要在 Microsoft Visual Basic 中使用 RDO，需要建立对 Visual Basic 项目的引用。执行下列各项操作：

1. 从“项目”菜单选择“引用”
2. 选择“Microsoft 远程数据对象 <version_number>”的校验框
3. 单击“确认”。

其中 <version_number> 是当前的 RDO 版本。

一个完整的 Visual Basic 程序包括表单和其他图形元素，您需要在 Visual Basic 环境中查看它。以下是作为一个 DB2 程序一部分的 Visual Basic 命令，该程序与 sample 数据库连接，打开一个记录集，然后从 EMPLOYEE 表中选择所有列，将雇员姓名逐个显示到一个信息窗口中：

```
Dim rdoEn As rdoEngine
Dim rdoEv As rdoEnvironment
Dim rdoCn As rdoConnection
Dim Cnct$
Dim rdoRS As rdoResultset
Dim SQLQueryDB As String
```

指定连接字符串：

```
Cnct$ = "DSN=SAMPLE;UID=;PWD=;"
```

设置 RDO 环境：

```
Set rdoEn = rdoEngine
Set rdoEv = rdoEn.rdoEnvironments(0)
```

与数据库连接：

```
Set rdoCn = rdoEv.OpenConnection("", , , Cnct$)
```

对记录集指定 SELECT 语句：

```
SQLQueryDB = "SELECT * FROM EMPLOYEE"
```

打开记录集，执行查询：

```
Set rdoRS = rdoCn.OpenResultset(SQLQueryDB)
```

未到达记录集末尾时，在信息框中一次显示表中一个雇员的 LASTNAME 和 FIRSTNAME：

```
While Not rdoRS.EOF
MsgBox rdoRS!LASTNAME &", " &rdoRS!FIRSTNAME
```

移动到记录集中的下一行：

```
rdoRS.MoveNext
Wend
```

关闭程序：

```
rdoRS.Close
rdoCn.Close
rdoEv.Close
```

DB2 在 %DB2PATH%\samples\RDO 目录中提供了 Visual Basic RDO 样本程序。

对象链接和嵌入 (OLE) 自动化

本节描述使用 Microsoft Visual Basic 的“对象链接和嵌入”(OLE) 自动化 UDF, 并介绍如何存取存储过程的样本 OLE 自动化控制器。

由于 OLE 与语言无关, 因此通过列举 OLE 自动化服务器的方法, 并将这些方法作为 UDF 向 DB2 注册, 可用任何语言实现 OLE 自动化 UDF 和存储过程。支持 OLE 自动化服务器开发的应用程序开发环境包括下列语言的某些版本: Microsoft Visual Basic、Microsoft Visual C++、Microsoft Visual J++、Microsoft FoxPro、Borland Delphi、Powersoft PowerBuilder 和 Micro Focus COBOL。同时, 适当地为 OLE 封装的 Java bean 对象(例如, 用 Microsoft Visual J++) 可通过 OLE 自动化存取。

需要参考适当的应用程序开发环境文档以获取开发 OLE 自动化服务器的更多信息。有关使用 OLE 自动化进行 DB2 程序设计的详情, 参阅 *Application Development Guide*。

OLE 自动化 UDF 和存储过程

Microsoft Visual Basic 支持创建 OLE 自动化服务器。通过向 Visual Basic 项目添加类模块创建 Visual Basic 新对象类型。通过向类模块添加公共子过程创建方法。可将这些公用过程作为 OLE 自动化 UDF 和存储过程注册到 DB2。参考 Microsoft Visual Basic 手册 *Creating OLE Servers, Microsoft Corporation, 1995* 和 Microsoft Visual Basic 提供的 OLE 样本, 以获取关于创建和构建 OLE 服务器的文档。

DB2 提供在 Microsoft Visual Basic 中使用的 OLE 自动化 UDF 和存储过程的自包含样本, 它们位于目录 %DB2PATH%\samples\ole\msvb 中。有关构建和运行 OLE 自动化 UDF 和存储过程样本的详情, 请参阅 %DB2PATH%\samples\ole 中的 README 文件。

Microsoft Visual C++

本节包括下列主题:

- ActiveX 数据对象 (ADO)
- 对象链接和嵌入 (OLE) 自动化
- DB2 CLI 应用程序
- DB2 CLI 存储过程
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)

注: Visual C++ 编译程序用于在 %DB2PATH%\samples\c 和 %DB2PATH%\samples\cpp 目录中提供的 C 和 C++ 样本程序。已在这两目录中放置了相同的批处理文件。它们包含根据文件扩展名接受 C 或 C++ 源文件的命令。除前两个主题“ActiveX 数据对象” (ADO) 和“对象链接与嵌入 (OLE) 自动化”中的批处理文件之外, 本节中其它的批处理文件用于演示构建程序。

ActiveX 数据对象 (ADO)

一旦做了以下更改, 使用 Visual C++ 的 DB2 ADO 程序可以象一般 C++ 程序一样编译。

要让 C++ 源程序作为 ADO 程序运行, 可将下面的 import 语句放在源程序文件开始处:

```
#import "C:\program files\common files\system\ado\msado<VERSION NUMBER>.dll" \  
no_namespace \  
rename( "EOF", "adoEOF")
```

其中 <VERSION NUMBER> 是 ADO 库的版本号。

当编译程序时, 用户需要验证 msado<VERSION NUMBER>.dll 在指定的路径中。另一个方法是将 C:\program files\common files\system\ado 添加到环境变量 LIBPATH 中, 然后在源文件中使用这个较短的 import 语句:

```
#import <msado<VERSION NUMBER>.dll> \  
no_namespace \  
rename( "EOF", "adoEOF")
```

这是 DB2 样本程序 BLOBAccess.dsp 中使用的方法。

有了此 IMPORT 语句, DB2 程序就可存取 ADO 库。现在, 可象编译任何其他程序一样编译 Visual C++ 程序。如果还使用另一个程序设计接口, 如 DB2 API 或 DB2 CLI, 可参考本章中适当章节以获取构建程序的其他信息。

DB2 在 %DB2PATH%\samples\ADO\VC 目录中提供了 Visual C++ ADO 样本程序。

对象链接和嵌入 (OLE) 自动化

本节描述使用 Microsoft Visual C++ 的“对象链接和嵌入” (OLE) 自动化 UDF, 以及存储过程的样本 OLE 自动化控制器。

由于 OLE 与语言无关, 因此通过列举 OLE 自动化服务器的方法, 并将这些方法作为 UDF 向 DB2 注册, 可用任何语言实现 OLE 自动化 UDF 和存储过程。支持 OLE 自动化服务器开发的应用程序开发环境包括下列语言的某些版本: Microsoft Visual Basic、Microsoft Visual C++、Microsoft Visual J++、Microsoft

FoxPro、Borland Delphi、Powersoft PowerBuilder 和 Micro Focus COBOL。同时，适当地为 OLE 封装的 Java bean 对象（例如，用 Microsoft Visual J++）可通过 OLE 自动化存取。

需要参考适当的应用程序开发环境文档以获取开发 OLE 自动化服务器的更多信息。有关使用 OLE 自动化进行 DB2 程序设计的详情，参阅 *Application Development Guide*。

OLE 自动化 UDF 和存储过程

Microsoft Visual C++ 支持创建 OLE 自动化服务器。可使用“Microsoft 基础类”和“Microsoft 基础类应用程序向导”实现服务器，或作为 Win32 应用程序来实现服务器。服务器可以是 DLL 或 EXE。参考 Microsoft Visual C++ 文档和 Microsoft Visual C++ 提供的 OLE 样本以获取进一步的信息。有关构建 DB2 的 Visual C++ UDF 的信息，参阅第315页的『用户定义函数 (UDF)』。有关使用 DB2 CLI 构建 Visual C++ 存储过程的信息，参阅第307页的『DB2 CLI 存储过程』。有关构建 DB2 的 Visual C++ 嵌入式 SQL 存储过程的信息，参阅第312页的『嵌入式 SQL 存储过程』。

DB2 提供在 Microsoft Visual C++ 中使用的 OLE 自动化 UDF 和存储过程的自包含样本，它们位于目录 %DB2PATH%\samples\ole\msvc 中。有关构建和运行 OLE 自动化 UDF 和存储过程样本的详情，请参阅 %DB2PATH%\samples\ole 中的 README 文件。

DB2 CLI 应用程序

%DB2PATH%\samples\cli 中的批处理文件 bldmcli.bat 包含构建 DB2 CLI 程序所需的命令。

参数 %1 指定源文件名。

这是唯一的必需参数，且是不包含嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 %2 指定您想连接的数据库的名称；第三个参数 %3 指定数据库的用户 ID；第四个参数 %4 指定口令

如果该程序包含嵌入式 SQL（根据扩展名 .sqc 或 .sqx 来判断），则调用批处理文件 embprep 来预编译该程序，分别生成一个扩展名为 .c 或 .cxx 的程序文件。

```
@echo off
rem bldmcli batch file - Windows 32-bit Operating Systems
rem Builds a CLI program with Microsoft Visual C++.
rem Usage: bldmcli prog_name [ db_name [ userid password ]]
```



```

if exist "%1.sqc" call embprep %1 %2 %3 %4
if exist "%1.sqx" call embprep %1 %2 %3 %4

rem Compile the error-checking utility.
cl -Z7 -Od -c -W1 -D_X86=1 -DWIN32 utilcli.c

rem Compile the program.
if exist "%1.sqx" goto cpp
cl -Z7 -Od -c -W1 -D_X86=1 -DWIN32 %1.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx

rem Link the program.
:link_step
link -debug:full -debugtype:cv -OUT:%1.exe %1.obj utilcli.obj db2cli.lib
@echo on

```

bldmcli 的编译和连接选项

编译选项:

- cl** Microsoft Visual C++ 编译程序。
- Z7** 生成 C7 格式的 CodeView 信息。
- Od** 禁用优化。关闭优化使得使用调试程序更容易。
- c** 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。
- W1** 设置警告级。
- D_X86_=1**
 Windows 32 位操作系统在基于 Intel 的计算机上运行所需的编译程序选项。
- DWIN32**
 Windows 32 位操作系统需要的编译程序选项。

bldmcli 的编译和连接选项

连接选项:

link 使用 32 位连接程序来连接编辑。

-debug:full
包括调试信息。

-debugtype:cv
指示调试程序类型。

-OUT:%1.exe
指定可执行文件。

%1.obj 包括目标文件。

utilcli.obj
包括该实用程序的目标文件以便检查错误。

db2cli.lib
连接 DB2 CLI 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`, 输入:

```
bldmcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名, 运行该可执行文件:

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldmcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接, 还须输入数据库名:

```
bldmcli dbusemx database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldmcli dbusemx database userid password
```

产生可执行文件 `dbusemx`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名称:

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

sqllib/samples/cli 中的脚本文件 bldmapi 包含使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 utilapi 实用程序文件（该文件包含创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 bldmcli 批处理文件之间唯一的区别。有关 bldmapi 和 bldmcli 之间公共的编译和连接选项，请参阅第304页的『DB2 CLI 应用程序』。

要根据源文件 dbmconn.c 构建样本程序 dbmconn，输入：

```
bldmapi dbmconn
```

产生可执行文件 dbmconn。可输入可执行文件名，运行该可执行文件：

```
dbmconn
```

DB2 CLI 存储过程

%DB2PATH%\samples\cli 中的批处理文件 bldmclis.bat 包含构建 CLI 存储过程所需的命令。此批处理文件将存储过程构建为服务器上的 DLL。

参数 %1 指定源文件名。此批处理文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldmclis.bat file - Windows 32-bit Operating Systems
rem Builds a CLI stored procedure using the Microsoft Visual C++ compiler.
rem Usage: bldmclis prog_name

if "%1" == "" goto error

rem Compile the program.
if exist "%1.cxx" goto cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c utilcli.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx utilcli.c

rem Link the program.
:link_step
link -debug:full -debugtype:cv -dll -out:%1.dll %1.obj utilcli.obj db2cli.lib -def:%1.def

rem Copy the stored procedure DLL to the 'function' directory
```

```
copy %1.dll "%DB2PATH%\function"

goto exit
:error
echo Usage: bldmclis prog_name
:exit
@echo on
```

bldmclis 的编译和连接选项	
编译选项:	
c1	Microsoft Visual C++ 编译程序。
-Z7	生成 C7 格式的 CodeView 信息。
-Od	禁用优化。关闭优化使得使用调试程序更容易。
-c	只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
-W2	设置警告级。
-D_X86_=1	Windows 32 位操作系统在基于 Intel 的计算机上运行所需的编译程序选项。
-DWIN32	Windows 32 位操作系统需要的编译程序选项。
连接选项:	
link	使用 32 位连接程序来连接编辑。
-debug:full	包括调试信息。
-debugtype:cv	指示调试程序类型。
-OUT:%1.dll	构建 .DLL 文件。
%1.obj	包括目标文件。
utilcli.obj	包括该实用程序的目标文件以便检查错误。
db2cli.lib	连接 DB2 CLI 库。
-def:%1.def	使用模块定义文件。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `spserver.c` 构建存储过程 `spserver`, 输入:

```
bldmclis spserver
```

此批处理文件使用模块定义文件 `spserver.def` 构建存储过程，该模块定义文件与 CLI 样本程序位于同一目录中。此批处理文件将存储过程 DLL `spserver.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库：

```
db2 connect to sample
```

如果存储过程先前已经编目，可使用以下命令卸下它们：

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`，就可构建调用该存储过程的 CLI 客户机应用程序 `spclient`。

可使用脚本文件 `bldmcli` 构建 `spclient`。有关详情，参考第304页的『DB2 CLI 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

%DB2PATH%\samples\c 和 %DB2PATH%\samples\cpp 中的批处理文件 bldmapp.bat 包含构建嵌入式 SQL 程序所需的命令。

第一个参数 %1 指定源文件名。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个附加的可选参数：第二个参数 %2 指定您想连接的数据库的名称；第三个参数 %3 指定数据库的用户 ID；第四个参数 %4 指定口令

对于嵌入式 SQL 程序，bldmapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
@echo off
rem bldmapp.bat -- Windows 32-bit operating systems
rem Builds a Microsoft Visual C++ application program
rem Usage: bldmapp prog_name [ db_name [ userid password ] ]

if exist "%1.sqx" goto embedded
if exist "%1.sqc" goto embedded
goto non_embedded

:embedded
rem Precompile and bind the program.
call embprep %1 %2 %3 %4
rem Compile the program.
if exist "%1.cxx" goto cpp_emb
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c utilemb.c
goto link_embedded
:cpp_emb
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx utilemb.cxx
rem Link the program.
:link_embedded
link -debug:full -debugtype:cv -out:%1.exe %1.obj utilemb.obj db2api.lib
goto exit

:non_embedded
rem Compile the program.
if exist "%1.cxx" goto cpp_non
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c utilapi.c
goto link_non_embedded
:cpp_non
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx utilapi.cxx
rem Link the program.
:link_non_embedded
link -debug:full -debugtype:cv -out:%1.exe %1.obj utilapi.obj db2api.lib
:exit
@echo on
```

bldmapp 的编译和连接选项

编译选项:

- c1** Microsoft Visual C++ 编译程序。
- Z7** 生成 C7 格式的 CodeView 信息。
- Od** 禁用优化。关闭优化使得使用调试程序更容易。
- c** 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
- W2** 设置警告级。
- D_X86_=1**
Windows 32 位操作系统在基于 Intel 的计算机上运行所需的编译程序选项。
- DWIN32**
Windows 32 位操作系统需要的编译程序选项。

连接选项:

- link** 使用 32 位连接程序来连接编辑。
- debug:full**
包括调试信息。
- debugtype:cv**
指示调试程序类型。
- out:%1.exe**
指定文件名
- %1.obj** 包括目标文件
- utilemb.obj**
如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。
- utilapi.obj**
如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。
- db2api.lib**
连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据 %DB2PATH%\samples\c 中的源文件 `client.c`, 或根据 %DB2PATH%\samples\cpp 中的源文件 `client.cxx` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldmapp client
```

产生可执行文件 `client.exe`。可输入可执行文件名（不带扩展名）来运行该可执行文件：

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从 `%DB2PATH%\samples\c` 中的 C 源文件 `updat.sqc`，或 `%DB2PATH%\samples\cpp` 中的 C++ 源文件 `updat.sqx` 构建嵌入式 SQL 应用程序 `updat`：

1. 如果与同一实例中的 `sample` 数据库连接，输入：

```
bldmapp updat
```

2. 如果与同一实例中的另一个数据库连接，还须输入数据库名：

```
bldmapp updat database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldmapp updat database userid password
```

产生可执行文件 `updat.exe`。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名：

```
updat
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名称：

```
updat database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令：

```
updat database userid password
```

嵌入式 SQL 存储过程

`%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中的批处理文件 `bldmsrv.bat` 包含构建嵌入式 SQL 存储过程所需的命令。此批处理文件将存储过程构建为服务器上的 DLL。

第一个参数 `%1` 指定源文件名。第二个参数 `%2` 指定想要连接的数据库的名称。因为必须在数据库所在的同一实例上构建存储过程，所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 `sample` 数据库。

此批处理文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldmsrv.bat -- Windows 32-bit operating systems
rem Builds a Microsoft Visual C++ stored procedure
rem Usage: bldmsrv prog_name [ db_name ]

rem Precompile and bind the program.
call embprep %1 %2

rem Compile the program.
if exist "%1.cxx" goto cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx

:link_step
rem Link the program.
link -debug:full -debugtype:cv -out:%1.dll -dll %1.obj db2api.lib -def:%1.def

rem Copy the stored procedure DLL to the 'function' directory
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldmsrv 的编译和连接选项

编译选项:

- cl** Microsoft Visual C++ 编译程序。
- Z7** 生成 C7 格式的 CodeView 信息。
- Od** 禁用优化。
- c** 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。
- W2** 输出警告、错误、严重的和不可恢复的错误信息。
- D_X86_=1**
Windows 32 位操作系统在基于 Intel 的计算机上运行所需的编译程序选项。
- DWIN32**
Windows 32 位操作系统需要的编译程序选项。

bldmsrv 的编译和连接选项

连接选项:

link 使用连接程序连接编辑。

-debug:full
包括调试信息。

-debugtype:cv
指示调试程序类型。

-out:%1.dll
构建 .DLL 文件。

%1.obj 包括目标文件。

db2api.lib
连接 DB2 库。

-def:%1.def
模块定义文件。

有关其他编译程序选项，参考编译程序文档。

要根据 C 源文件 `spserver.sqc` 或 C++ 源文件 `spserver.sqx` 构建 `spserver` 存储过程 DLL，输入：

```
bldmsrv spserver
```

如果连接另一个数据库，还须输入该数据库名：

```
bldmsrv spserver database
```

此批处理文件使用模块定义文件 `spserver.def` 构建 DLL，该模块定义文件与样本程序位于同一目录中。此批处理文件将该 DLL `spserver.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库：

```
db2 connect to sample
```

如果存储过程先前已经编目，可使用以下命令卸下它们：

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`，就可构建调用它的客户机应用程序 `spclient`。

可使用脚本文件 `bldmapp` 构建 `spclient`。有关详情，参考第310页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程 DLL `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

`%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中的批处理文件 `bldmudf` 包含构建 UDF 所需的命令。

UDF 不能包含嵌入式 SQL 语句。因此，要构建 UDF 程序，不必连接数据库来预编译和联编该程序。

此批处理文件用参数 `%1` 来指定源文件名。它使用源文件名 `%1` 作为 DLL 名。

```
@echo off
rem bldmudf.bat -- Windows 32-bit operating systems
rem Builds a Microsoft Visual C++ user-defined function (UDF).
rem Usage: bldmudf udf_prog_name

if "%1" == "" goto error

rem Compile the program.
if exist "%1.cxx" goto cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx

:link_step
rem Link the program.
link -debug:full -debugtype:cv -dll -out:%1.dll %1.obj db2api.lib db2apie.lib -def:%1.def
```

```
rem Copy the UDF DLL to the 'function' directory
copy %1.dll "%DB2PATH%\function"

goto exit
:error
echo Usage: bldmudf prog_name
:exit
@echo on
```

bldmudf 的编译和连接选项

编译选项:

- c1** Microsoft Visual C++ 编译程序。
- Z7** 生成 C7 格式的 CodeView 信息。
- Od** 禁用优化。
- c** 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
- W2** 输出警告、错误、严重的和不可恢复的错误信息。
- D_X86_=1**
Windows 32 位操作系统在基于 Intel 的计算机上运行所需的编译程序选项。
- DWIN32**
Windows 32 位操作系统需要的编译程序选项。

bldmudf 的编译和连接选项

连接选项:

link 使用连接程序连接编辑。

-debug:full
包括调试信息。

-debugtype:cv
指示调试程序类型。

-dll 创建 DLL。

-out:%1.dll
构建 .DLL 文件。

%1.obj 包括目标文件。

db2api.lib
连接 DB2 库。

db2apie.lib
连接“DB2 API 引擎”库。

-def:%1.def
模块定义文件。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `udfsrv.c` 构建用户定义函数 `udfsrv`，输入：

```
bldmudf udfsrv
```

此批处理文件使用模块定义文件 `udfsrv.def` 构建用户定义函数，该模块定义文件与样本程序位于同一目录中。此批处理文件将用户定义函数 DLL `udfsrv.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

一旦构建了 `udfsrv`，就可构建调用它的客户机应用程序 `udfcli`。提供了此程序的 DB2 CLI 以及嵌入式 SQL C 和 C++ 版本。

可根据 `%DB2PATH%\samples\cli` 中的源文件 `udfcli.c`，使用批处理文件 `bldmcli` 构建 DB2 CLI `udfcli` 程序。有关详情，参考第304页的『DB2 CLI 应用程序』>。

可根据 `%DB2PATH%\samples\c` 中的源文件 `udfcli.sqc`，使用批处理文件 `bldmapp` 构建嵌入式 SQL C `udfcli` 程序。有关详情，参考第310页的『DB2 API 和嵌入式 SQL 应用程序』。

可根据 %DB2PATH%\samples\cpp 中的源文件 udfcli.sqx, 使用批处理文件 bldmapp 构建嵌入式 SQL C++ udfcli 程序。有关详情, 参考第310页的『DB2 API 和嵌入式 SQL 应用程序』。

要运行 UDF, 输入:

```
udfcli
```

调用应用程序从 udfsrv DLL 中调用 ScalarUDF 函数。

IBM VisualAge C++ 版本 3.5

本节包括下列主题:

- DB2 CLI 应用程序
- DB2 CLI 存储过程
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程
- 用户定义函数 (UDF)

注: VisualAge C++ 编译程序用于 %DB2PATH%\samples\c 和 %DB2PATH%\samples\cpp 目录中提供的 C 和 C++ 样本程序。已在这两目录中放置了相同的批处理文件。它们包含根据文件扩展名接受 C 或 C++ 源文件的命令。

DB2 CLI 应用程序

%DB2PATH%\samples\cli 中的批处理文件 bldvcli.bat 包含用 IBM VisualAge C++ 构建 DB2 CLI 程序所需的命令。

参数 %1 指定源文件名。

这是不包含嵌入式 SQL 的 CLI 程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库, 因此还须提供三个可选参数: 第二个参数 %2 指定您想连接的数据库的名称; 第三个参数 %3 指定数据库的用户 ID; 第四个参数 %4 指定口令

如果该程序包含嵌入式 SQL (根据扩展名 .sqc 或 .sqx 来判断), 则调用批处理文件 embprep 来预编译该程序, 分别生成一个扩展名为 .c 或 .cxx 的程序文件。

```
@echo off
rem bldvcli batch file - Windows 32-bit Operating Systems
rem Builds a CLI program with IBM VisualAge C++.
rem Usage: bldvcli prog_name

if exist "%1.sqc" call embprep %1 %2 %3 %4
```

```

if exist "%1.sqx" call embprep %1 %2 %3 %4

rem Compile the error-checking utility.
icc -c -Ti -W1 /I"%DB2PATH%\include" utilcli.c

rem Compile the program.
if exist "%1.sqx" goto cpp
icc -c -Ti -W1 /I"%DB2PATH%\include" %1.c
goto link_step
:cpp
icc -c -Ti -W1 /I"%DB2PATH%\include" %1.cxx

rem Link the program.
:link_step
ilink /MAP /DEBUG /ST:32000 /PM:VIO %1.obj utilcli.obj db2cli.lib
@echo on

```

bldvcli 的编译和连接选项

编译选项:

icc IBM VisualAge C++ 编译程序。

-c 只执行编译；不连接。本书假定编译和连接是两个独立的步骤。

-Ti 生成调试程序信息。

-W1 输出警告、错误、严重的和不可恢复的错误信息。

连接选项:

ilink 使用资源连接程序连接编辑。

/MAP 生成映象文件。

/DEBUG 包括调试信息。

/ST:32000
指定堆栈大小至少为 32 000。

/PM:VIO
允许程序在窗口或全屏幕中运行。

%1.obj 包括目标文件。

utilcli.obj
包括该实用程序的目标文件以便检查错误。

db2cli.lib
连接 DB2 CLI 库。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `tbinfo.c` 构建样本程序 `tbinfo`，输入：

```
bldvcli tbinfo
```

产生可执行文件 `tbinfo`。可输入可执行文件名，运行该可执行文件：

```
tbinfo
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `dbusemx.sqc` 构建嵌入式 SQL 应用程序 `dbusemx`：

1. 如果与同一实例中的 `sample` 数据库连接，输入：

```
bldvcli dbusemx
```

2. 如果与同一实例中的另一个数据库连接，还须输入数据库名：

```
bldvcli dbusemx database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 ID 和口令：

```
bldvcli dbusemx database userid password
```

产生可执行文件 `dbusemx`。

有三种方法运行此嵌入式 SQL 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名：

```
dbusemx
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名称：

```
dbusemx database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令：

```
dbusemx database userid password
```

使用 DB2 API 的 DB2 CLI 应用程序

DB2 包括使用 DB2 API 创建和卸下数据库的 CLI 样本程序，以演示如何在多个数据库上使用 CLI 函数。第22页的表7中的 CLI 样本程序的说明指示使用 DB2 API 的样本。

`sqllib/samples/cli` 中的脚本文件 `bldvapi` 包含使用 DB2 API 构建 DB2 CLI 程序所需的命令。此文件在 `utilapi` 实用程序文件（该文件包含创建和卸下数据库所需的 DB2 API）中编译和连接。这是此文件与 `bldvcli` 批处理文件之间唯一的区别。有关 `bldvapi` 和 `bldvcli` 之间公共的编译和连接选项，请参阅第318页的『DB2 CLI 应用程序』。

要根据源文件 `dbmconn.c` 构建样本程序 `dbmconn`，输入：

```
bldvapi dbmconn
```

产生可执行文件 `dbmconn`。可输入可执行文件名，运行该可执行文件：

DB2 CLI 存储过程

%DB2PATH%\samples\cli 中的批处理文件 bldvclis.bat 包含构建 CLI 存储过程所需的命令。此批处理文件将存储过程构建为服务器上的 DLL。

参数 %1 指定源文件名。此批处理文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldvclis.bat file - Windows 32-bit Operating Systems
rem Builds a CLI stored procedure using the IBM VisualAge C++ compiler
rem Usage: bldvclis prog_name

if "%1" == "" goto error

rem Compile the program.
if exist "%1.cxx" goto cpp
icc -c+ -Ti -Ge- -Gm+ -Wl %1.c utilcli.c
goto link_step
:cpp
icc -c+ -Ti -Ge- -Gm+ -Wl %1.cxx utilcli.c

:link_step
rem Import the library and create an export file.
rem The function name in the .def file must be decorated to be consistent
rem with the function name in the .map file. Typically, this is done by
rem prepending "_" and appending "@" and the number of bytes of arguments,
rem as in: "@16". In spserverva.def, for example, the IBM VisualAge C++
rem compiler requires "EXPORTS _outlanguage@16" and not "EXPORTS outlanguage".
ilib /GI %1va.def

rem Link the program and produce a DLL.
ilink /ST:64000 /PM:VIO /MAP /DLL %1.obj utilcli.obj %1va.exp db2cli.lib

rem Copy the stored procedure DLL to the 'function' directory
copy %1.dll "%DB2PATH%\function"

goto exit
:error
echo Usage: bldvclis prog_name
:exit
@echo on
```

bldvclis 的编译和连接选项

编译选项:

- icc** IBM VisualAge C++ 编译程序。
- c+** 只执行编译; 不连接。此批处理文件有单独的编译和连接步骤。
- Ti** 生成调试程序信息。
- Ge-** 构建 .DLL 文件。使用静态连接的运行期库的版本。
- Gm+** 连接多任务库。
- W1** 输出警告、错误、严重的和不可恢复的错误信息。

连接选项:

- ilink** 使用资源连接程序连接编辑。
- /ST:64000**
指定堆栈大小至少为 64 000。
- /PM:VIO**
允许程序在窗口或全屏幕中运行。
- /MAP** 生成映象文件。
- /DLL** 构建 .DLL 文件。
- %1.obj** 包括目标文件。
- %1.exp** 包括 VisualAge 调出文件。
- db2cli.lib**
连接 DB2 CLI 库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `spserver.c` 构建 `spserver` 存储过程, 输入:

```
bldvclis spserver
```

此批处理文件使用模块定义文件 `spserverva.def` 构建存储过程, 该模块定义文件与 CLI 样本程序位于同一目录中。此批处理文件将存储过程 DLL `spserver.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

接着在服务器上运行 `spcreate.db2` 脚本文件以编目存储过程。首先连接数据库:

```
db2 connect to sample
```

如果存储过程先前已经编目, 可使用以下命令卸下它们:

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们:

```
db2 -td@ -vf spcreate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时, 将该共享库设置为文件方式, 以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`, 就可构建调用该存储过程的 CLI 客户机应用程序 `spclient`。

可使用批处理文件 `bldvcli` 构建 `spclient`。有关详情, 参考第318页的『DB2 CLI 应用程序』。

要调用存储过程, 可输入如下命令运行样本客户机应用程序:

```
spclient database userid password
```

其中,

database

是要连接的数据库的名称。 该名称可能是 `sample`, 或它的别名, 或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 `spserver`, 并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

DB2 API 和嵌入式 SQL 应用程序

`%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中的批处理文件 `bldvapp.bat` 包含构建 DB2 应用程序所需的命令。

第一个参数 `%1` 指定源文件名。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库, 因此还须提供三个可选参数: 第二个参数 `%2` 指定您想连接的数据库的名称; 第三个参数 `%3` 指定数据库的用户 ID; 第四个参数 `%4` 指定口令

对于嵌入式 SQL 程序, `bldvapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名, 则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时, 才需要用户 ID 和口令参数。

```

@echo off
rem bldvapp.bat -- Windows 32-bit operating systems
rem Builds a VisualAge C++ application program
rem Usage: bldvapp prog_name [ db_name [ userid password ]]

if exist "%1.sqx" goto embedded
if exist "%1.sqc" goto embedded
goto non_embedded

:embedded
rem Precompile and bind the program.
call embprep %1 %2 %3 %4
rem Compile the program.
if exist "%1.cxx" goto cpp_emb
icc -c -Ti -W1 %1.c utilemb.c
goto link_embedded
:cpp_emb
icc -c -Ti -W1 %1.cxx utilemb.cxx
rem Link the program.
:link_embedded
ilink /MAP /DEBUG /ST:32000 /PM:VIO %1.obj utilemb.obj db2api.lib
goto exit

:non_embedded
rem Compile the program.
if exist "%1.cxx" goto cpp_non
icc -c -Ti -W1 %1.c utilapi.c
goto link_non_embedded
:cpp_non
icc -c -Ti -W1 %1.cxx utilapi.cxx
rem Link the program.
:link_non_embedded
ilink /MAP /DEBUG /ST:32000 /PM:VIO %1.obj utilapi.obj db2api.lib
:exit
@echo on

```

bldvapp 的编译和连接选项

编译选项:

- icc** IBM VisualAge C++ 编译程序。
- c** 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
- Ti** 生成调试程序信息。
- W1** 输出警告、错误、严重的和不可恢复的错误信息。

bldvapp 的编译和连接选项

连接选项:

ilink 使用资源连接程序连接编辑。

/MAP 生成映象文件。

/DEBUG 包括调试信息。

/ST:32000
指定堆栈大小至少为 32 000。

/PM:VIO
允许程序在窗口或全屏幕中运行。

%1.obj 包括目标文件。

utilemb.obj
如果是嵌入式 SQL 程序, 应包括嵌入式 SQL 实用程序目标文件以便检查错误。

utilapi.obj
如果不是嵌入式 SQL 程序, 应包括 DB2 API 实用程序目标文件以便检查错误。

db2api.lib
连接 DB2 库。

有关其他编译程序选项, 参考编译程序文档。

要根据 %DB2PATH%\samples\c 中的源文件 `client.c`, 或根据 %DB2PATH%\samples\cpp 中的源文件 `client.cxx` 构建 DB2 API 非嵌入式 SQL 样本程序 `client`, 输入:

```
bldvapp client
```

产生可执行文件 `client.exe`。可输入可执行文件名 (不带扩展名) 来运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从 %DB2PATH%\samples\c 中的 C 源文件 `updat.sqc` 或 %DB2PATH%\samples\cpp 中的 C++ 源文件 `updat.sqx` 构建嵌入式 SQL 应用程序 `updat`:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldvapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入数据库名:

```
bldvapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldvapp updat database userid password
```

产生可执行文件 `updat.exe`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:

```
updat
```

2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名称:

```
updat database
```

3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令:

```
updat database userid password
```

嵌入式 SQL 存储过程

`%DB2PATH%\samples\c` 和 `%DB2PATH%\samples\cpp` 中的批处理文件 `bldvsrv.bat` 包含构建嵌入式 SQL 存储过程所需的命令。此批处理文件将存储过程编译为一个 DLL, 并将它存储在服务器上。

第一个参数 `%1` 指定源文件名。第二个参数 `%2` 指定想要连接的数据库的名称。因为必须在数据库所在的同一实例中构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

此批处理文件使用源文件名 `%1` 作为 DLL 名。

```
@echo off
rem bldvsrv.bat -- Windows 32-bit operating systems
rem Builds a VisualAge C++ stored procedure
rem Usage: bldvsrv prog_name [ db_name ]

rem Precompile and bind the program.
call embprep %1 %2

rem Compile the program.
if exist "%1.cxx" goto cpp
icc -c+ -Ti -Ge- -Gm+ -Wl %1.c
goto link_step
:cpp
icc -c+ -Ti -Ge- -Gm+ -Wl %1.cxx

:link_step
rem Import the library and create a definition file.
rem The function name in the .def file must be decorated to be consistent
```

```
rem with the function name in the .map file. Typically, this is done by
rem prepending "_" and appending "@" and the number of bytes of arguments,
rem for example, "@16". In spservvva.def, the IBM VisualAge C++ compiler requires
rem "EXPORTS _outlanguage@16" and not "EXPORTS outlanguage".
ilib /GI %1va.def
```

```
rem Link the program and produce a DLL.
ilink /ST:64000 /PM:VIO /MAP /DLL %1.obj %1va.exp db2api.lib
```

```
rem Copy the Stored Procedure DLL to the 'function' directory.
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldvsrv 编译和连接选项	
编译选项:	
icc	IBM VisualAge C++ 编译程序。
-c+	只执行编译; 不连接。此批处理文件有单独的编译和连接步骤。
-Ti	生成调试程序信息。
-Ge-	构建 .DLL 文件。使用静态连接的运行期库的版本。
-Gm+	连接多任务库。
-W1	输出警告、错误、严重的和不可恢复的错误信息。
连接选项:	
ilink	使用资源连接程序连接编辑。
/ST:64000	指定堆栈大小至少为 64 000。
/PM:VIO	允许程序在窗口或全屏幕中运行。
/MAP	生成 MAP 文件。
/DLL	构建 .DLL 文件。
%1.obj	包括目标文件。
%1va.exp	VisualAge 调出文件。
db2api.lib	连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据 C 源文件 spserver.sqc 或 C++ 源文件 spserver.sqx 构建存储过程 DLL spserver, 输入:

```
bldvsrv spserver
```

如果连接另一个数据库，还须输入该数据库名：

```
bldmsrv spserver database
```

此批处理文件使用模块定义文件 `spserverva.def` 构建存储过程，该模块定义文件与样本程序位于同一目录中。此批处理文件将存储过程 DLL `spserver.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

接着在服务器上运行 `screate.db2` 脚本文件以编目存储过程。首先连接数据库：

```
db2 connect to sample
```

如果存储过程先前已经编目，可使用以下命令卸下它们：

```
db2 -td@ -vf spdrop.db2
```

然后可使用以下命令编目它们：

```
db2 -td@ -vf screate.db2
```

最后停止并重新启动数据库以便可识别新的共享库。必要时，将该共享库设置为文件方式，以便 DB2 实例可存取它。

一旦构建了存储过程 `spserver`，就可构建调用它的客户机应用程序 `spclient`。

可使用脚本文件 `bldvapp` 构建 `spclient`。有关详情，参考第323页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
spclient database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程 DLL `spserver`，并在服务器数据库上执行大量存储过程函数。然后将输出返回给该客户机应用程序。

用户定义函数 (UDF)

%DB2PATH%\samples\c 和 %DB2PATH%\samples\cpp 中的批处理文件 bldvudf 包含构建 UDF 所需的命令。

UDF 不能包含嵌入式 SQL 语句。因此，要构建 UDF 程序，不必连接数据库或预编译及联编该程序。

参数 %1 指定源文件名。此批处理文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldvudf.bat -- Windows 32-bit operating systems
rem Builds a VisualAge C++ user-defined function (UDF)
rem Usage: bldvudf program_name

if "%1" == "" goto error

rem Compile the program.
if exist "%1.cxx" goto cpp
icc -Ti -c+ -Ge- -Gm+ -W1 %1.c
goto link_step
:cpp
icc -Ti -c+ -Ge- -Gm+ -W1 %1.cxx

:link_step
rem Generate an import library and export file using a definition file.
rem Function(s) in the .def file are prepended with an underscore, and
rem appended with the @ sign and number of bytes of arguments (in decimal).
rem Parameters of less than four bytes are rounded up to four bytes.
rem Structure size is rounded up to a multiple of four bytes.
rem For example, function fred prototyped as: "int fred(int, int, short);"
rem would appear as: "_fred@12" in the .def file.
rem These decorated function names can also be found in %1.map
rem after running the following ilink command without %1va.exp.
ilib /gi %1va.def

rem Link the program to a dynamic link library
ilink /ST:64000 /PM:VIO /MAP /DLL %1.obj %1va.exp db2api.lib db2apie.lib

rem Copy the UDF DLL to the 'function' directory.
copy %1.dll "%DB2PATH%\function"

goto exit
:error
echo Usage: bldvudf prog_name
:exit
@echo on
```

bldvudf 的编译和连接选项

编译选项:

- icc** IBM VisualAge C++ 编译程序。
- Ti** 生成调试程序信息。
- c+** 只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
- Ge-** 构建 .DLL 文件。使用静态连接的运行期库的版本。
- Gm+** 连接多任务库。
- W1** 输出警告、错误、严重的和不可恢复的错误信息。

连接选项:

- ilink** 使用资源连接程序连接编辑。
- /ST:64000**
指定堆栈大小至少为 64000。
- /PM:VIO**
允许程序在窗口或全屏幕中运行。
- /MAP** 生成 MAP 文件。
- /DLL** 构建 .DLL 文件。
- %1.obj** 包括目标文件。
- %1va.exp**
包括 VisualAge 调出文件。
- db2api.lib**
连接 DB2 库。
- db2apie.lib**
连接“DB2 API 引擎”库。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `udf.c` 构建用户定义函数 `udfsrv`, 输入:

```
bldvudf udfsrv
```

此批处理文件使用模块定义文件 `udfsrv.def` 构建用户定义函数, 该模块定义文件与样本程序位于同一目录中。此批处理文件将用户定义函数 DLL `udfsrv.dll` 复制到服务器的 `%DB2PATH%\function` 路径中。

一旦构建了 `udfsrv`, 就可构建调用它的客户机应用程序 `udfcli`。提供了此程序的 DB2 CLI 以及嵌入式 SQL C 和 C++ 版本。

可根据 %DB2PATH%\samples\cli 中的源文件 udfcli.c，使用批处理文件 bldvcli 构建 DB2 CLI udfcli 程序。有关详情，参考第318页的『DB2 CLI 应用程序』。

可根据 %DB2PATH%\samples\c 中的源文件 udfcli.sqc，使用批处理文件 bldvapp 构建嵌入式 SQL C udfcli 程序。有关详情，参考第323页的『DB2 API 和嵌入式 SQL 应用程序』。

可根据 %DB2PATH%\samples\cpp 中的源文件 udfcli.sqx，使用批处理文件 bldvapp 构建嵌入式 SQL C++ udfcli 程序。有关详情，参考第323页的『DB2 API 和嵌入式 SQL 应用程序』。

要运行 UDF，输入：

```
udfcli
```

调用应用程序从 udfsrv DLL 中调用 ScalarUDF 函数。

IBM VisualAge C++ 版本 4.0

VisualAge C++ 版本 4 编译程序的应用程序构建信息对于 AIX、OS/2 和 Windows 32 位操作系统而言是相同的。有关此信息，参阅第120页的『VisualAge C++ 版本 4.0』。

IBM VisualAge COBOL

本节包括下列主题：

- 使用编译程序
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程

使用编译程序

如果开发包含嵌入式 SQL 和 DB2 API 调用的应用程序，且使用的是 IBM VisualAge COBOL 编译程序，应切记以下几点：

- 当使用命令行处理器命令 db2 prep 预编译应用程序时，使用 target ibmcob 选项（缺省项）。
- 在源文件中不要使用制表符。
- 可在源文件中使用 PROCESS 和 CBL 关键字以设置编译选项。只能将这些关键字置于 8 至 72 列之间。

- 如果您的应用程序只包含嵌入式 SQL，而不包含 DB2 API 调用，则不需要使用 `pgmname(mixed)` 编译选项。如果使用 DB2 API 调用，则必须使用 `pgmname(mixed)` 编译选项。
- 如果您使用 IBM VisualAge COBOL 编译程序的“System/390 主机数据类型支持”功能部件，则应用程序的 DB2 包含文件在以下目录中：

```
%DB2PATH%\include\cobol_i
```

如果要使用提供的批处理文件构建 DB2 样本程序，必须将在批处理文件中指定的包含文件路径更改为指向 `cobol_i` 目录而不是 `cobol_a` 目录。

如果不使用 IBM VisualAge COBOL 编译程序的“System/390 主机数据类型支持”功能部件，或者使用此编译程序的较早版本，则应用程序的 DB2 包含文件在以下目录中：

```
%DB2PATH%\include\cobol_a
```

按如下所示，指定包括 `.cbl` 扩展名的 COPY 文件名：

```
COPY "sql.cbl".
```

DB2 API 和嵌入式 SQL 应用程序

`%DB2PATH%\samples\cobol` 中的批处理文件 `bldapp.bat` 包含构建 DB2 应用程序所需的命令。

第一个参数 `%1` 指定源文件名。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 `%2` 指定您想连接的数据库的名称；第三个参数 `%3` 指定数据库的用户 ID；第四个参数 `%4` 指定口令

对于嵌入式 SQL 程序，`bldapp` 将这些参数传送给预编译和联编文件 `embprep`。如果未提供数据库名，则使用缺省 `sample` 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
@echo off
rem bldapp.bat -- Windows 32-bit operating systems
rem Builds a VisualAge COBOL application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if not exist "%1.sqb" goto compile_step
call embprep %1 %2 %3 %4

:compile_step
rem Compile the error-checking utility.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" checkerr.cbl
```

```

rem Compile the program.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" %1.cbl

rem Link the program.
cob2 %1.obj checkerr.obj db2api.lib
@echo on

```

bldapp 的编译和连接选项	
编译选项:	
cob2	IBM VisualAge COBOL 编译程序。
-qpgmname(mixed)	指示编译程序可用大小写混合的名称调用 (CALL) 库入口点。
-c	只执行编译; 不连接。本书假定编译和连接是两个独立的步骤。
-qlib	指示编译程序处理 COPY 语句。
-Ipath	指定 DB2 包含文件的位置。例如: <code>-I"%DB2PATH%\include\cobol_a"</code> 。
checkerr.cbl	编译错误检查实用程序。
连接选项:	
cob2	使用编译程序来连接编辑。
checkerr.obj	包括错误检查实用程序的目标文件。
db2api.lib	连接 DB2 库。
有关其他编译程序选项, 参考编译程序文档。	

要根据源文件 `client.cbl` 构建非嵌入式 SQL 样本程序 `client`, 输入:

```
bldapp client
```

产生可执行文件 `client.exe`。可输入可执行文件名 (不带扩展名), 来对 `sample` 数据库运行该可执行文件:

```
client
```

构建和运行嵌入式 SQL 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 SQL 应用程序:

1. 如果与同一实例中的 `sample` 数据库连接, 输入:

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接, 还须输入数据库名:

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接, 还须输入该数据库实例的用户 ID 和口令:

```
bldapp updat database userid password
```

产生可执行文件 `updat`。

有三种方法运行此嵌入式 SQL 应用程序:

1. 如果存取同一实例中的 `sample` 数据库, 只须输入可执行文件名:
`updat`
2. 如果存取同一实例中的另一个数据库, 输入可执行文件名和数据库名称:
`updat database`
3. 如果存取另一个实例中的数据库, 输入可执行文件名、数据库名称以及该数据库实例的用户 ID 和口令:

`updat database userid password`

嵌入式 SQL 存储过程

`%DB2PATH%\samples\cobol` 中的批处理文件 `bldsrv.bat` 包含构建嵌入式 SQL 存储过程所需的命令。此批处理文件将存储过程编译为服务器上的 DLL。

第一个参数 `%1` 指定源文件名。第二个参数 `%2` 指定想要连接的数据库的名称。因为必须在数据库所在的同一实例上构建存储过程, 所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名, 则该程序使用缺省的 `sample` 数据库。

此批处理文件使用源文件名 `%1` 作为 DLL 名。

```
@echo off
rem bldsrv.bat -- Windows 32-bit operating systems
rem Builds a VisualAge COBOL stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2

rem Compile the stored procedure.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" %1.cb1

rem Link the stored procedure and create a shared library.
ilib /no1 /gi:%1 %1.obj
ilink /free /no1 /dll db2api.lib %1.exp %1.obj iwzrwin3.obj

rem Copy stored procedure to the %DB2PATH%\function directory.
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldsrv 的编译和连接选项

编译选项:

cob2 IBM VisualAge COBOL 编译程序。

-qpgmname(mixed)

指示编译程序可用大小写混合的名称调用 (CALL) 库入口点。

-c 只执行编译; 不连接。此批处理文件有单独的编译和连接步骤。

-qlib 指示编译程序处理 COPY 语句。

-Ipath 指定 DB2 包含文件的位置。例如: `-I"%DB2PATH%\include\cobol_a"`。

连接选项:

ilink 使用 IBM VisualAge COBOL 连接程序。

/free 自由格式。

/no1 无标志。

/dll 创建具有源程序名的 DLL。

db2api.lib

连接 DB2 库。

%1.exp 包括调出文件。

%1.obj 包括该程序的目标文件。

iwzrwin3.obj

包括 IBM VisualAge COBOL 提供的目标文件。

有关其他编译程序选项, 参考编译程序文档。

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`, 如果连接 `sample` 数据库, 则输入:

```
bldsrv outsrv
```

如果连接另一个数据库, 还须输入该数据库名:

```
bldsrv outsrv database
```

该脚本文件将存储过程复制到服务器的 `sqllib/function` 路径中。

必要时, 将此存储过程设置为文件方式, 以便 DB2 实例可运行它。

一旦构建了存储过程 `outsrv`, 就可构建调用此存储过程的客户机应用程序 `outcli`。可使用批处理文件 `bldapp` 构建 `outcli`。有关详情, 参考第332页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程, 可输入如下命令运行样本客户机应用程序:

```
outcli database userid password
```

其中,

database

是要连接的数据库的名称。该名称可以是 `sample`、其远程别名或其他名称。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取存储过程库 `outsrv`，并在服务器数据库上执行同名的存储过程函数，然后将输出返回给客户机应用程序。

Micro Focus COBOL

本节包括下列主题：

- 使用编译程序
- DB2 API 和嵌入式 SQL 应用程序
- 嵌入式 SQL 存储过程

使用编译程序

如果开发包含嵌入式 SQL 和 DB2 API 调用的应用程序，且使用的是 Micro Focus 编译程序，应切记以下几点：

- 当使用命令行处理器命令 `db2 prep` 预编译应用程序时，使用 `target mfcob` 选项（缺省项）。
- 确保 `LIB` 环境变量指向 `%DB2PATH%\lib`，如下所示：

```
set LIB="%DB2PATH%\lib;%LIB%"
```

- Micro Focus COBOL 的 DB2 COPY 文件驻留在 `%DB2PATH%\include\cobol_mf` 中。为包括该目录，应设置 `COBCPY` 环境变量，如下所示：

```
set COBCPY="%DB2PATH%\include\cobol_mf;%COBCPY%"
```

必须使用调用约定 74 来调用所有 DB2 应用程序设计接口。DB2 COBOL 预编译程序自动在 `SPECIAL-NAMES` 段中插入 `CALL-CONVENTION` 子句。如果 `SPECIAL-NAMES` 段不存在，DB2 COBOL 预编译程序按如下方式创建它：

```
Identification Division
Program-ID. "static".
special-names.
    call-convention 74 is DB2API.
```

而且，无论何时调用 DB2 API，预编译程序都会自动将符号 `DB2API`（用于标识调用约定）置于 `"call"` 关键字之后。例如，每当预编译程序从嵌入式 SQL 语句生成 DB2 API 运行期调用时，会发生此情况。

如果在未预编译的应用程序中调用 DB2 API，应以类似上面给定的方式在该应用程序中人工创建 SPECIAL-NAMES 段。如果正直接调用 DB2 API，则将需要在 "call" 关键字之后人工添加 DB2API 符号。

DB2 API 和嵌入式 SQL 应用程序

%DB2PATH%\samples\cobol_mf 中的批处理文件 bldapp 包含构建 DB2 应用程序所需的命令。

第一个参数 %1 指定源文件名。这是不包含嵌入式 SQL 的程序所需的唯一参数。构建嵌入式 SQL 程序需要连接数据库，因此还须提供三个可选参数：第二个参数 %2 指定您想连接的数据库的名称；第三个参数 %3 指定数据库的用户 ID；第四个参数 %4 指定口令

对于嵌入式 SQL 程序，bldapp 将这些参数传送给预编译和联编文件 embprep。如果未提供数据库名，则使用缺省 sample 数据库。仅当构建程序所在的实例不同于数据库所在的实例时，才需要用户 ID 和口令参数。

```
@echo off
rem bldapp.bat -- Windows 32-bit operating systems
rem Builds a Micro Focus Cobol application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if not exist "%1.sqb" goto compile_step
call embprep %1 %2 %3 %4

:compile_step
rem Compile the error-checking utility.
cobol checkerr.cbl;

rem Compile the program.
cobol %1.cbl;

rem Link the program.
cbllink -l %1.obj checkerr.obj db2api.lib
@echo on
```

bldapp 的编译和连接选项

编译选项:

cobol Micro Focus COBOL 编译程序。

bldapp 的编译和连接选项

连接选项:

cbllink

使用连接程序连接编辑。

-l 连接 `lcobol` 库。

checkerr.obj

连接错误检查实用程序的目标文件。

db2api.lib

连接 `DB2 API` 库。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `client.cbl` 构建非嵌入式 `SQL` 样本程序 `client`，输入：

```
bldapp client
```

产生可执行文件 `client.exe`。可输入可执行文件名（不带扩展名），来对 `sample` 数据库运行该可执行文件：

```
client
```

构建和运行嵌入式 `SQL` 应用程序

有三种方法可从源文件 `updat.sqb` 构建嵌入式 `SQL` 应用程序：

1. 如果与同一实例中的 `sample` 数据库连接，输入：

```
bldapp updat
```

2. 如果与同一实例中的另一个数据库连接，还须输入数据库名：

```
bldapp updat database
```

3. 如果与另一个实例中的数据库连接，还须输入该数据库实例的用户 `ID` 和口令：

```
bldapp updat database userid password
```

产生可执行文件 `updat.exe`。

有三种方法运行此嵌入式 `SQL` 应用程序：

1. 如果存取同一实例中的 `sample` 数据库，只须输入可执行文件名（不需输入扩展名）：

```
updat
```

2. 如果存取同一实例中的另一个数据库，输入可执行文件名和数据库名称：

```
updat database
```

3. 如果存取另一个实例中的数据库，输入可执行文件名、数据库名称以及该数据库实例的用户 `ID` 和口令：

嵌入式 SQL 存储过程

%DB2PATH%\samples\cobol_mf 中的批处理文件 bldsrv 包含构建嵌入式 SQL 存储过程所需的命令。此批处理文件将存储过程编译为服务器上的 DLL。

第一个参数 %1 指定源文件名。第二个参数 %2 指定想要连接的数据库的名称。因为必须在数据库所在的同一实例上构建存储过程，所以不需要用户 ID 和口令的参数。

只有第一个参数即源文件名是必需的。数据库名是可选的。如果未提供数据库名，则该程序使用缺省的 sample 数据库。

此批处理文件使用源文件名 %1 作为 DLL 名。

```
@echo off
rem bldsrv.bat -- Windows 32-bit operating systems
rem Builds a Micro Focus Cobol stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2

rem Compile the stored procedure.
cobol %1.cbl /case;

rem Link the stored procedure and create a shared library.
cbllink /d %1.obj db2api.lib

rem Copy the stored procedure to the %DB2PATH%\function directory.
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldsrv 的编译和连接选项

编译选项:

cobol Micro Focus COBOL 编译程序。

/case 防止外部符号转换为大写。

bldsrv 的编译和连接选项

连接选项:

cbllink

使用 Micro Focus COBOL 连接程序连接编辑。

/d 创建 .dll 文件。

db2api.lib

连接 DB2 API 库。

有关其他编译程序选项，参考编译程序文档。

要根据源文件 `outsrv.sqb` 构建样本程序 `outsrv`，如果连接 `sample` 数据库，则输入：

```
bldsrv outsrv
```

如果连接另一个数据库，还须输入该数据库名：

```
bldsrv outsrv database
```

此脚本文件将 DLL 复制到服务器的 `sqllib/function` 路径中。

必要时，将该 DLL 设置为文件方式，以便客户机应用程序可存取它。

一旦构建了 DLL `outsrv`，就可构建调用它的客户机应用程序 `outcli`。可使用批处理文件 `bldapp` 构建 `outcli`。有关详情，参考第337页的『DB2 API 和嵌入式 SQL 应用程序』。

要调用存储过程，可输入如下命令运行样本客户机应用程序：

```
outcli database userid password
```

其中，

database

是要连接的数据库的名称。该名称可能是 `sample`，或它的别名，或另一个数据库名。

userid 是有效的用户 ID。

password

是有效的口令。

客户机应用程序存取 DLL `outsrv`，并在服务器数据库上执行同名的存储过程函数。然后将输出返回给该客户机应用程序。

Object REXX

Object REXX 是 REXX 语言的面向对象版本。面向对象扩充已添加到经典的 REXX 中，但其现存的函数和指令未改变。Object REXX 解释程序是其先前版本的增强版本，它还支持：

- 类、对象和方法
- 信息传送和多态
- 单继承和多继承

Object REXX 完全符合经典的 REXX。在本节中，只要提到 REXX，都指所有 REXX 版本（包括 Object REXX）。

不要预编译或联编 REXX 程序。

在 Windows NT 上，REXX 程序不必以注解开始。然而，为便于移植，建议每个 REXX 程序的第一行第一列以注解开始。这将使程序与其他平台的批处理命令区分开来：

```
/* Any comment will do. */
```

可在目录 %DB2PATH%\samples\rexx 中找到 REXX 样本程序。要运行 REXX 样本程序 updat，执行下列操作：

1. 如果服务器上的数据库管理程序尚未运行，则输入以下命令启动它：

```
db2start
```

2. 输入：

```
rexx updat.cmd
```

有关 REXX 和 DB2 的详情，参考 *Application Development Guide* 中的“Programming in REXX”一章。

附录A. 关于数据库管理程序实例

DB2 支持在同一机器上的多个数据库管理程序实例。数据库管理程序实例有它自己的配置文件、目录和数据库。

每个数据库管理程序实例可管理几个数据库。但是，一个给定的数据库只属于一个实例。图1说明这种关系。

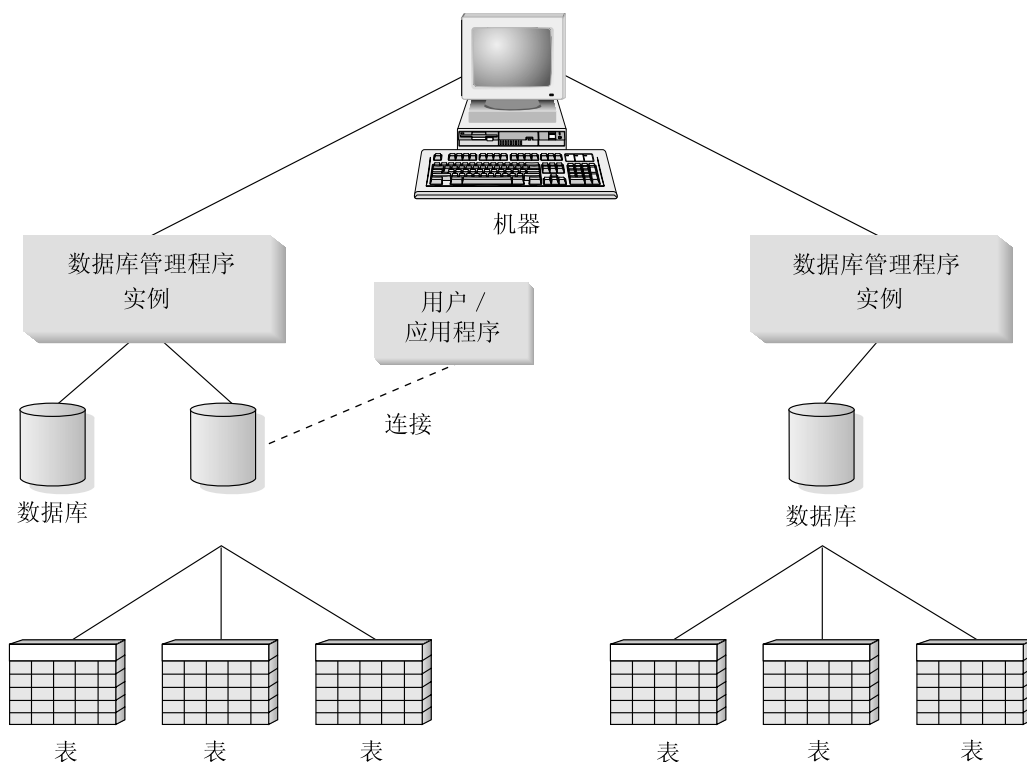


图 1. 数据库管理程序实例

数据库管理程序实例增大了灵活性，使您在同一机器上可有多个数据库环境。例如，可有一个数据库管理程序实例用于开发环境，而另一个实例用于生产环境。

当使用 UNIX 服务器时，您可以在不同的数据库管理程序实例上有不同的 DB2 版本。例如，您可以有一个数据库管理程序实例运行 DB2 通用数据库版本 6.1，而

另一个运行 DB2 通用数据库版本 7.1。但是在同一版本级别内，只支持一个发行版和一个修订版级别。例如，DB2 版本 5.0 和 DB2 版本 5.2 不能在 UNIX 服务器上并存。

当使用 OS/2、Windows NT 和 Windows 2000 服务器时，您必须在每个数据库管理程序实例上有相同的 DB2 版本、发行版和修订版级别。不能让一个数据库管理程序实例运行 DB2 通用数据库版本 6.1，而另一个实例运行 DB2 通用数据库版本 7.1。

您需要知道所用的每个实例的下列信息：

实例名 对于 UNIX 平台，这是在创建数据库管理程序实例时指定的有效的用户名。

对于 OS/2、Windows NT 和 Windows 2000，这是由最多 8 个字符组成的字母数字字符串。在安装期间会为您创建一个名为 "DB2" 的实例。

实例目录

实例所在的主目录。

对于 UNIX 平台，实例目录是 \$HOME/sql1lib，其中 \$HOME 是实例拥有者的主目录。

对于 OS/2、Windows NT 和 Windows 2000，实例目录是 %DB2PATH%\instance_name。变量 %DB2PATH% 确定将 DB2 安装在何处。根据 DB2 所在的驱动器，%DB2PATH% 指向 drive:\sql1lib。

在 OS/2、Windows NT 和 Windows 2000 上的实例路径的创建是基于：

%DB2PATH%\%DB2INSTANCE%（例如，C:\SQLLIB\DB2）

或者，如果定义了 DB2INSTPROF，则基于：

%DB2INSTPROF%\%DB2INSTANCE%（例如，C:\PROFILES\DB2）

在 OS/2、Windows NT 和 Windows 2000 上使用 DB2INSTPROF 环境变量，以支持在客户机只有读取存取权的网络驱动器上运行 DB2。在此情况下，将设置 DB2 指向 drive:\sql1lib，并将设置 DB2INSTPROF 指向本地路径（例如，C:\PROFILES），它将包含所有针对实例的信息（如目录和配置），因为 DB2 需要这些文件的更新权限。

有关创建和管理数据库管理程序实例的信息，参考适合您平台的快速入门一书。

附录B. 迁移应用程序

当您将安装的版本 2 或更高版本的 DB2、“DB2 客户机应用程序使能器”或 DB2 Software Developer's Kit 升级至“DB2 通用数据库版本 7.1”时，您的数据库和节点目录将自动迁移。要从 DB2 版本 1 迁移，首先必须迁移至“DB2 通用数据库版本 5”。然后可从版本 5 迁移至版本 7.1。要迁移现存的数据库，使用管理指南中所描述的工具。

注：

1. **HP-UX**。如果您要将 DB2 从 HP-UX 版本 10 或更早版本迁移至 HP-UX 版本 11，则您的 DB2 程序必须用 HP-UX 版本 11 上的 DB2 重新预编译（如果这些程序包括嵌入式 SQL），且必须重新编译。这包括所有 DB2 应用程序、存储过程、用户定义函数和用户出口程序。另外，在 HP-UX 版本 11 上编译的 DB2 程序可能无法在 HP-UX 版本 10 或更早版本上运行。在 HP-UX 版本 10 上编译和运行的 DB2 程序可以与 HP-UX 版本 11 服务器远程连接。
2. **Linux**。DB2 不支持从“DB2 通用数据库 Linux 版的版本 5.2 (Beta 版)”迁移。
3. **Micro Focus COBOL**。用 DB2 版本 2.1.1 或更早版本预编译和用 Micro Focus COBOL 编译的任何现存应用程序，都应使用 DB2 的当前版本重新预编译，再用 Micro Focus COBOL 重新编译。如果用先前版本的 IBM 预编译程序构建的这些应用程序未重新预编译，则当发生异常终止时数据库可能被破坏。

注：下面内容以及“问题”和“条件”两节仅适用于 UNIX 平台。

如果有 DB2 版本 1、DB2 版本 2、DB2 版本 5 或 DB2 版本 6.1 的应用程序，并希望它们既可以在先前版本的数据库实例中运行，又可在同一个机器上的 DB2 版本 7.1 实例中运行，则可能需要对环境做某些更改。要确定做何更改，须回答下列问题，然后复查“条件”一节，以了解是否有任何条件适合于您的情况。

使用 AIX 系统来解释提出的观点。相同的概念也适用于其他 UNIX 平台，但是详情可能不同，如环境变量和特定的命令。如果您不熟悉操作系统的这些详情，请参阅管理指南或 *DB2 (UNIX 版) 快速入门* 一书中“计划安装”一章的“从先前 DB2 版本迁移”一节。

问题

问题 1: 先前 DB2 版本的应用程序是如何与 DB2 客户机运行期库 (例如, AIX 上的 libdb2.a) 连接的?

要确定可执行文件的嵌入式共享库搜索路径, 使用下列系统命令之一:

AIX /usr/bin/dump -H *executable_filename*

HP-UX

/usr/bin/chatr *executable_filename*

Linux /usr/bin/objdump -p *executable_filename*

PTX /usr/bin/dump -Lv *executable_filename*

Silicon Graphics IRIX

/bin/elfdump -Lv *executable_filename*

Solaris

/usr/bin/dump -Lv *executable_filename*

其中 *executable_filename* 是应用程序可执行文件的名称。

以下是 “DB2 版本 1 AIX 版” 应用程序的样本转储列表:

dbcat:

Loader Section

Loader Header Information			
VERSION#	#SYMtableENT	#RELOCent	LENidSTR
0x00000001	0x00000012	0x00000029	0x00000064
#IMPfilID	OFFfidSTR	LENstrTBL	OFFstrTBL
0x00000004	0x000003bc	0x00000077	0x00000420

Import File Strings

INDEX	PATH	BASE	MEMBER
0	/usr/lpp/db2_01_01_0000/lib:/usr/lpp/xlC/lib:/usr/lib:/lib		
1		libc.a	shr.o
2		libC.a	shr.o
3		libdb2.a	shr.o

第 0 (零) 行显示可执行文件为查找与它连接的共享库所搜索的目录路径。第 1、2 和 3 行显示该应用程序所连接的共享库。

取决于该应用程序的构建方式，您可能看到下列路径：

`/usr/lpp/db2_01_01_0000/lib`、`INSTHOME/sql/lib/lib` (其中 `INSTHOME` 是该数据库实例拥有者的主目录)，或只是 `/usr/lib:/lib` 组合。

问题 2: 在您的系统上，DB2 运行期库是如何配置的？

当安装 DB2 版本 1、2、5、6.1 或 7.1 中的任何一个时，有一个可选步骤，用于创建从系统缺省共享库路径 `/usr/lib` 至 DB2 安装路径 (包含 DB2 客户机运行期库) 的符号链路。

不同 DB2 版本的安装路径如下：

版本 1

`/usr/lpp/db2_01_01_0000/lib`

版本 2

`/usr/lpp/db2_02_01/lib`

版本 5

`/usr/lpp/db2_05_00/lib`

版本 6.1

`/usr/lpp/db2_06_01/lib`

版本 7.1

`/usr/lpp/db2_07_01/lib`

所有这些版本的运行期共享库的名称都为 `libdb2.a`。

在任何一个时刻，这些库中只有一个版本可以是缺省值。DB2 提供了此缺省值，这样当您构建应用程序时，它就不会取决于 DB2 的某一特定版本。

问题 3: 您是否在环境中指定了不同的搜索路径？

在 AIX 上可使用 `LIBPATH` 环境变量，在 HP-UX 上可使用 `SHLIB_PATH`，在 Linux、PTX、Silicon Graphics IRIX 和 Solaris 可使用 `LD_LIBRARY_PATH` 覆盖已编入您应用程序中的共享库搜索路径。

注：对于 Silicon Graphics IRIX 上的 n32 对象类型应用程序，使用 `LD_LIBRARYN32_PATH` 环境变量来覆盖它。

您可以使用在问题 1 的回答中给出的适用于您平台的适当系统命令来查看库搜索路径。

条件

一旦有了上述问题的答案，可能就需要对环境做一些更改。阅读以下列出的条件。如果其中一个条件适合您的情况，则进行必要的更改。

条件 1: 如果一个版本 6.1 应用程序从 AIX 缺省共享库路径 `/usr/lib/libdb2.a` 装入共享库，和

- 如果存在从 `/usr/lib/libdb2.a` 至 `/usr/lpp/db2_05_00/lib/libdb2.a` 的符号链路，并且数据库服务器是“DB2 通用数据库版本 7.1 AIX 版”，则执行下列操作之一：

- 更改此符号链路以指向：

```
/usr/lpp/db2_07_01/lib/libdb2.a
```

DB2 (UNIX 版) 快速入门 包含有关设置库之间的链路的信息。作为超级用户，您可以使用“`db2ln`”命令来更改链路，如下所示：

```
/usr/lpp/db2_07_01/cfg/db2ln
```

- 设置 `LIBPATH` 环境变量以指向 `/usr/lpp/db2_07_01/lib` 或 `INSTHOME/sql/lib/lib`，其中 `INSTHOME` 是 DB2 版本 7.1 实例拥有者的主目录。
- 配置从应用程序（客户机）实例至服务器实例的 TCP/IP 连接。参考 *安装和配置补遗*，以获得有关配置 TCP/IP 的信息。
- 如果存在从 `/usr/lib/libdb2.a` 至 `/usr/lpp/db2_07_01/lib/libdb2.a` 的符号链路，且该数据库服务器是 DB2 版本 6.1，则配置从应用程序（客户机）实例至服务器实例的 TCP/IP 连接。参考 *安装和配置补遗*，以获得有关配置 TCP/IP 的信息。

条件 2: 如果版本 6.1 应用程序从 DB2 版本 6.1 实例拥有者的 `$HOME` 路径 (`$HOME/sql/lib/lib/libdb2.a`) 装入共享库，且该数据库服务器是“DB2 通用数据库版本 7.1 AIX 版”，则执行下列操作之一：

- 将应用程序实例迁移至与该数据库服务器实例相同的版本。
- 设置 `LIBPATH` 环境变量以指向 `/usr/lpp/db2_07_01/lib` 或 `INSTHOME/sql/lib/lib`，其中 `INSTHOME` 是 DB2 版本 7.1 实例拥有者的主目录。
- 配置从应用程序（客户机）实例至服务器实例的 TCP/IP 连接。参考 *安装和配置补遗*，以获得有关配置 TCP/IP 的信息。

条件 3: 如果版本 6.1 应用程序从 DB2 版本 6.1 安装路径 (`/usr/lpp/db2_06_01/lib/libdb2.a`) 装入共享库，且该数据库服务器是“DB2 通用数据库版本 7.1 AIX 版”，则执行下列操作之一：

- 设置 LIBPATH 环境变量以指向 /usr/lpp/db2_07_01/lib 或 INSTHOME/sql1ib/lib, 其中 INSTHOME 是数据库实例拥有者的主目录。
- 配置从应用程序 (客户机) 实例至服务器实例的 TCP/IP 连接。参考安装和配置补遗, 以获得有关配置 TCP/IP 的信息。

条件 4: 如果版本 6.1 应用程序从 “DB2 通用数据库版本 7.1 AIX 版” 的安装路径 (/usr/lpp/db2_07_01/lib/libdb2.a) 装入共享库, 且该数据库服务器是 DB2 版本 6.1, 则配置从应用程序 (客户机) 实例至服务器实例的 TCP/IP 连接。参考安装和配置补遗, 以获得有关配置 TCP/IP 的信息。

其他迁移考虑事项

当您开发应用程序时, 应考虑下列几点。它们将有助于使您的应用程序更易于移植:

- 在 UNIX 上, 在应用程序中仅使用缺省路径 /usr/lib:/lib。在 OS/2 和 Windows 32 位操作系统上, 通过进行下列设置确保 LIB 环境变量指向 %DB2PATH%\lib:

```
set LIB=%DB2PATH%\lib;%LIB%
```

还要在缺省路径和您所使用的 DB2 版本之间创建符号链路。确保该链路连至您的应用程序所需的最低级别的 DB2。参考适合您平台的快速入门一书, 以获得有关设置链路的信息。

- 如果您的应用程序需要一个特定版本的 DB2, 则编码您的应用程序中指定该 DB2 版本的路径。例如, 如果您的 AIX 应用程序需要 DB2 版本 5, 则编码 /usr/lpp/db2_05_00/lib。一般情况下, 您不必这样做。
- 当您正在构建用于生产而不是用于内部开发的应用程序时, 该应用程序中的路径不应指向实例拥有者的目录 (在 UNIX 上, 该目录为 sql1ib/lib; 在 OS/2 和 Windows 32 位操作系统上, 该目录为 %DB2PATH%\lib)。这就使得应用程序非常依赖于特定的用户名和环境。
- 一般情况下, 不要在 Windows 32 位操作系统上使用 LIBPATH 环境变量或 LIB 环境变量, 来改变特定环境中的搜索路径。该变量会替换在该环境中运行的应用程序中指定的搜索路径。应用程序可能无法查找到所需要的库或文件。
- 在 “DB2 通用数据库版本 6.1 和版本 7.1” 中, 具有字符串语义的所有字符数组项目都有 char 类型, 而不是其他变体, 如无符号的 char。使用 “DB2 通用数据库版本 6.1 或版本 7.1” 编码的任何应用程序都应该遵循此惯例。

如果您有使用无符号 char 的 DB2 版本 1 应用程序, 则编译程序可能会发出警告或产生错误, 这是因为版本 1 应用程序中的无符号 char 与版本 6.1 或版本 7.1 函数原型中的 char 之间的类型冲突。如果发生这种情况, 则使用编译程序选项 -DSQLOLDCHAR 来消除该问题。

- 有关“DB2 通用数据库版本 7.1”和 DB2 的先前版本之间不兼容性的列表，参考 *SQL Reference*。有关“DB2 通用数据库版本 7.1”和 DB2 的先前版本之间 API 不兼容性的列表，参考 *Administrative API Reference*。

附录C. 问题确定

当构建或运行应用程序时，可能遇到下列几类问题：

- 客户机或服务器问题，如在构建期间或运行应用程序时连接数据库失败。
- 操作系统问题，如在构建期间找不到文件。
- 在构建期间的编译程序选项问题。
- 在构建期间或运行应用程序时的语法和编码问题。

可使用下列信息资源解决这些问题：

构建文件

对于构建问题，如连接数据库、预编译、编译、连接和联编时发生的问题，可使用本书中列出的构建文件来查看有效的命令行处理器命令和编译程序选项。

编译程序文档

参考此文档，可获得有关构建脚本文件中未涉及的编译程序选项问题的信息。

Application Development Guide

参考 *Application Development Guide*，以获得语法和其他编码问题的信息。

CLI Guide and Reference

参考 *CLI Guide and Reference*，以获得与 CLI 程序相关的语法、CLI 跟踪设施、配置关键字和编码问题的信息。

SQL Reference

参考 *SQL Reference*，以获得有关 SQL 语句和函数的语法信息。

SQLCA 数据结构

如果应用程序发出 SQL 语句或调用数据库管理程序 API，则它必须检查 SQLCA 数据结构以获取错误情况。

SQLCA 数据结构将错误信息置于 SQLCODE 和 SQLSTATE 字段中返回。在执行每个 SQL 语句后以及在调用大多数数据库管理程序 API 后，数据库管理程序会更新该结构。

应用程序可检索并打印错误信息，或将错误信息显示在屏幕上。有关详情，参考 *Application Development Guide*。

联机出错信息

DB2 的不同部件，包括数据库管理程序、数据库管理实用程序、安装和配

置过程以及命令行处理器，都生成联机错误信息。这些信息中的每一个都具有唯一的前缀，且前缀后带四位或五位信息号码。一个字母跟在该信息号之后，指示错误的严重程度。

可以使用命令行处理器输入以下命令，来查看该信息的帮助：

```
db2 "? xxxnnnn"
```

其中，xxx 是信息前缀，nnnn 是信息号。要加上双引号。

有关 DB2 出错信息的完整列表和说明，参阅信息参考。

诊断工具和错误记录

为不能使用其他信息源解决的构建或运行期问题提供了这些工具。诊断工具包括跟踪设施、系统记录和信息记录等等。DB2 根据优先级和发生点将错误和警告情况置于错误记录中。有关详情，参考 *Troubleshooting Guide*。还有一个专门用来调试 CLI 程序的 CLI 跟踪设施。有关详情，参考 *CLI Guide and Reference*。

附录D. 使用 DB2 资料库

DB2 通用数据库由联机帮助、书籍（PDF 和 HTML）和 HTML 格式的样本程序组成。本节描述所提供的信息以及如何访问这些信息。

要存取联机产品信息，可以使用“信息中心”。有关详情，参见第366页的『用“信息中心”存取“信息”』。可以查看任务信息、DB2 书籍、疑难解答信息、样本程序和 Web 上的 DB2 信息。

DB2 PDF 文件和打印的书籍

DB2 信息

下表将 DB2 书籍分为四个类别:

DB2 指南和参考信息

这些书籍包含所有平台的公共 DB2 信息。

DB2 安装和配置信息

这些书籍是针对特定平台上的 DB2 的。例如，有分别针对 OS/2 平台、Windows 平台和基于 UNIX 的平台上 DB2 的快速入门书籍。

HTML 格式的跨平台样本程序

这些样本是与“应用程序开发客户机”一起安装的样本程序的 HTML 版本。样本仅供参考，并不替代实际程序。

发行说明

这些文件包含 DB2 书籍中未能包括的最新信息。

HTML 格式的安装手册、发行说明和教程可直接在产品 CD-ROM 上看到。大部分书籍在产品 CD-ROM 上都有 HTML 格式以便查看，而在 DB2 出版物 CD-ROM 上则有 Adobe Acrobat (PDF) 格式以便查看和打印。还可从 IBM 订购打印的副本；参见第362页的『订购打印书籍』。下表列示了可订购的书籍。

在 OS/2 和 Windows 平台上，可在 `sqllib\doc\html` 目录下安装 HTML 文件。DB2 信息被翻译成各种语言；但是，并非所有的信息都有每一种语言的翻译版本。每当信息不能以某种特定语言表示出来时，就会提供英语信息

在 UNIX 平台上，可在 `doc/%L/html`（其中 `%L` 表示本国语言环境）目录下安装多种语言版本的 HTML 文件。有关详情，参考适当的快速入门书籍。

您可以各种方法来获取 DB2 书籍并存取信息:

- 第365页的『查看联机信息』
- 第369页的『搜索联机信息』
- 第362页的『订购打印书籍』
- 第361页的『打印 PDF 书籍』

表 18. DB2 信息

名称	说明	书号	HTML 目录
		PDF 文件名	
DB2 指南和参考信息			
管理指南	<p>管理指南: 计划提供数据库概念的概述、有关设计问题 (如逻辑和物理数据库设计) 的信息, 以及高可用性的讨论。</p> <p>管理指南: 实现提供有关实现问题 (如实现设计、存取数据库、审核、备份和恢复) 的信息。</p> <p>管理指南: 性能提供有关数据库环境以及应用程序性能评估和调整的信息。</p> <p>在北美, 可使用书号 SBOF-8934 来订购三卷英文版的管理指南。</p>	<p>SB84-0219 db2d1x70</p> <p>SB84-0218 db2d2x70</p> <p>SB84-0243 db2d3x70</p>	db2d0
<i>Administrative API Reference</i>	描述 DB2 应用程序设计接口 (API) 以及您可以用来管理数据库的数据结构。此书还说明如何在应用程序中调用 API。	SC09-2947 db2b0x70	db2b0
应用程序构建指南	提供环境设置信息和关于如何在 Windows、OS/2 和基于 UNIX 的平台上编译、链接和运行 DB2 应用程序的逐步指导。	SB84-0220 db2axx70	db2ax
<i>APPC, CPI-C, and SNA Sense Codes</i>	提供关于使用DB2 通用数据库产品时可能遇到的 APPC、CPI-C 和 SNA 检测码的一般信息。	无书号 db2apx70	db2ap
仅有 HTML 格式的版本。			

表 18. DB2 信息 (续)

名称	说明	书号	HTML 目录
		PDF 文件名	
<i>Application Development Guide</i>	说明如何开发使用嵌入式 SQL 或 Java (JDBC 和 SQLJ) 来存取 DB2 数据库的应用程序。讨论主题包括在分区环境或联合体系统中编写存储过程、编写用户定义函数、创建用户定义类型、使用触发器和开发应用程序。	SC09-2949 db2a0x70	db2a0
<i>CLI Guide and Reference</i>	说明如何开发使用“DB2 调用层接口”(一个与 Microsoft ODBC 规范兼容的可调用 SQL 接口)来存取 DB2 数据库的应用程序。	SC09-2950 db2l0x70	db2l0
<i>Command Reference</i>	说明如何使用“命令行处理器”，并描述用来管理数据库的 DB2 命令。	SC09-2951 db2n0x70	db2n0
<i>Connectivity Supplement</i>	提供有关以下各项的设置和参考信息：如何将作为 DRDA 应用程序请求器的 DB2 AS/400 版、DB2 OS/390 版、DB2 MVS 版、DB2 VM 版与 DB2 通用数据库服务器配合使用。此书还详述了如何将 DRDA 应用服务器与 DB2 Connect 应用程序请求器配合使用。 仅有 HTML 和 PDF 格式。	无书号 db2h1x70	db2h1
<i>Data Movement Utilities Guide and Reference</i>	说明如何使用 DB2 实用程序(如调入、调出、装入、自动装入程序和 DPROP)来使数据移动易于进行。	SC09-2955 db2dmx70	db2dm
数据仓库中心管理指南	提供有关如何使用“数据仓库中心”构建和维护数据仓库的信息。	SB84-0226 db2ddx70	db2dd
<i>Data Warehouse Center Application Integration Guide</i>	提供帮助程序员将应用程序与“数据仓库中心”和“信息目录管理程序”集成的信息。	SC26-9994 db2adx70	db2ad
<i>DB2 Connect 用户指南</i>	提供 DB2 Connect 产品的概念、程序设计以及一般用法信息。	SB84-0221 db2c0x70	db2c0
<i>DB2 Query Patroller Administration Guide</i>	提供 DB2 Query Patroller 系统的操作概述、特定操作和管理信息以及管理图形用户界面实用程序的任务信息。	SC09-2958 db2dwx70	db2dw

表 18. DB2 信息 (续)

名称	说明	书号	HTML 目录
		PDF 文件名	
<i>DB2 Query Patroller</i> 用户指南	描述如何使用 DB2 Query Patroller 的工具和功能。	SB84-0222	db2ww
		db2wwx70	
词汇表	提供 DB2 及其部件中使用的术语的定义。 有 HTML 格式可用且在 <i>SQL Reference</i> 中。	无书号	db2t0
		db2t0x70	
<i>Image, Audio, and Video Extenders</i> 管理和程序设计	提供有关 DB2 Extender 的一般信息, 有关 Image、Audio 和 Video (IAV) Extender 的管理和配置的信息, 以及有关使用 IAV Extender 进行程序设计的信息。它包括参考信息、诊断资料 (带有信息) 和样本。	SB84-0247	dmbu7
		dmbu7x70	
<i>Information Catalog Manager Administration Guide</i>	提供有关管理信息目录的指南。	SC26-9995	db2di
		db2dix70	
<i>Information Catalog Manager Programming Guide and Reference</i>	提供“信息目录管理程序”的体系结构接口的定义。	SC26-9997	db2bi
		db2bix70	
信息目录管理程序用户指南	提供有关使用“信息目录管理程序”用户界面的信息。	SB84-0227	db2ai
		db2aix70	
安装和配置补遗	指导您了解计划、安装和设置特定于平台的 DB2 客户机。此补遗还包含关于联编、设置客户机和服务器通信、DB2 GUI 工具、DRDA AS、分布式安装、配置分布式请求和存取多机种数据源的信息。	GB84-0127	db2iy
		db2iyx70	
信息参考	列出由 DB2、信息目录管理程序和数据仓库中心发出的信息和代码, 并描述应执行的操作。 在北美, 您可订购两卷英文版的信息参考 (使用书号 SBOF-8932)。	第 1 卷 GB84-0216	db2m0
		db2m1x70	
		第 2 卷 GB84-0217	
		db2m2x70	
<i>OLAP Integration Server Administration Guide</i>	说明如何使用“OLAP 集成服务器”的“管理程序”部件。	SC27-0787	n/a
		db2dpx70	

表 18. DB2 信息 (续)

名称	说明	书号	HTML 目录
		PDF 文件名	
<i>OLAP Integration Server Metaoutline User's Guide</i>	说明如何使用标准“OLAP 元轮廓”接口（而非通过使用“元轮廓辅助程序”）创建和填充 OLAP 元轮廓。	SC27-0784 db2upx70	n/a
<i>OLAP Integration Server Model User's Guide</i>	说明如何使用标准“OLAP 模型接口”（而非使用“模型辅助程序”）来创建 OLAP 模型。	SC27-0783 db2lpx70	n/a
<i>OLAP Setup and User's Guide</i>	提供 OLAP Starter Kit 的配置和设置信息。	SC27-0702 db2ipx70	db2ip
<i>OLAP Spreadsheet Add-in User's Guide for Excel</i>	描述如何使用 Excel 电子表格程序来分析 OLAP 数据。	SC27-0786 db2epx70	db2ep
<i>OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3</i>	描述如何使用 Lotus 1-2-3 电子表格程序来分析 OLAP 数据。	SC27-0785 db2tpx70	db2tp
<i>Replication Guide and Reference</i>	提供随 DB2 提供的“IBM 复制”工具的计划、配置、管理和用法信息。	SC26-9920 db2e0x70	db2e0
<i>Spatial Extender 用户指南和参考</i>	提供关于空间扩展程序的安装、配置、管理、程序设计和疑难解答的信息。还提供空间数据概念的重要说明，并提供空间扩展程序特定的参考资料（信息和 SQL）。	SB84-0249 db2sbx70	db2sb
<i>SQL 入门</i>	介绍 SQL 概念，并提供许多构造和任务的示例。	SB84-0223 db2y0x70	db2y0
<i>SQL Reference, 第 1 卷和第 2 卷</i>	描述 SQL 语法、语义和语言规则。此书还包括关于发行版间的不兼容性、产品限制和目录视图的信息。 在北美，可使用书号 SBOF-8933 来订购两卷英文版的 <i>SQL Reference</i> 。	第 1 卷 SC09-2974 db2s1x70 第 2 卷 SC09-2975 db2s2x70	db2s0
<i>System Monitor Guide and Reference</i>	描述如何收集关于数据库和数据库管理程序的各种信息。此书说明如何利用信息来了解数据库活动、提高性能和确定问题的原因。	SC09-2956 db2f0x70	db2f0

表 18. DB2 信息 (续)

名称	说明	书号	HTML 目录
		PDF 文件名	
<i>Text Extender</i> 管理和程序设计	提供有关 DB2 Extender 的一般信息, 有关 Text Extender 的管理和配置的信息, 以及有关使用 Text Extender 进行程序设计的信息。它包括参考信息、诊断资料 (带有信息) 和样本。	SB84-0248 desu9x70	desu9
<i>Troubleshooting Guide</i>	帮助您确定错误源、从问题中恢复并向 “DB2 客户服务” 咨询以使用诊断工具。	GC09-2850 db2p0x70	db2p0
新增内容	描述 DB2 通用数据库 (版本 7) 中的新特性、函数和增强功能。	SB84-0224 db2q0x70	db2q0
DB2 安装和配置信息			
<i>DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings</i>	提供 OS/2 和 Windows 32 位操作系统上的 DB2 Connect 企业版的计划、迁移、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2953 db2c6x70	db2c6
<i>DB2 Connect Enterprise Edition for UNIX Quick Beginnings</i>	提供基于 UNIX 的平台上的 DB2 Connect 企业版的计划、迁移、安装、配置和任务信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2952 db2cyx70	db2cy
<i>DB2 Connect</i> 个人版快速入门	提供 OS/2 和 Windows 32 位操作系统上的 DB2 Connect 个人版的计划、迁移、安装、配置和任务信息。此书还包含所有受支持的客户机的安装和设置信息。	GB84-0212 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings Linux 版	在进行所有受支持的 Linux 分发时, 提供 “DB2 Connect 个人版” 的计划、安装、迁移和配置信息。	GC09-2962 db2c4x70	db2c4
<i>DB2 DataLinks</i> 管理程序快速入门	提供 “DB2 DataLinks Manager AIX 版” 和 Windows 32 位操作系统的计划、安装、配置和任务信息。	GB84-0211 db2z6x70	db2z6
<i>DB2</i> 扩充企业版 UNIX 版快速入门	提供在基于 UNIX 的平台上的 DB2 扩充企业版的计划、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0209 db2v3x70	db2v3

表 18. DB2 信息 (续)

名称	说明	书号	HTML 目录
		PDF 文件名	
<i>DB2 Enterprise - Extended Edition for Windows Quick Beginnings</i>	提供 DB2 扩充企业版 Windows 32 位操作系统版的计划、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2963 db2v6x70	db2v6
<i>DB2 (OS/2 版) Quick Beginnings</i>	提供 OS/2 操作系统上的 DB2 通用数据库的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2968 db2i2x70	db2i2
<i>DB2 (UNIX 版) 快速入门</i>	提供在基于 UNIX 的平台上的 DB2 通用数据库的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0214 db2ixx70	db2ix
<i>DB2 Windows 版快速入门</i>	提供 Windows 32 位操作系统上的 DB2 通用数据库的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0215 db2i6x70	db2i6
<i>DB2 个人版快速入门</i>	提供 OS/2 和 Windows 32 位操作系统上的“DB2 通用数据库个人版”的计划、安装、迁移和配置信息。	GB84-0213 db2i1x70	db2i1
<i>DB2 Personal Edition Quick Beginnings Linux 版</i>	在进行所有受支持的 Linux 分发时，提供“DB2 通用数据库个人版”的计划、安装、迁移和配置信息。	GC09-2972 db2i4x70	db2i4
<i>DB2 Query Patroller 安装指南</i>	提供有关 DB2 Query Patroller 的安装信息。	GB84-0208 db2iwx70	db2iw
<i>DB2 Warehouse Manager 安装指南</i>	提供仓库代理程序、仓库变换器和“信息目录管理程序”的安装信息。	GB84-0122 db2idx70	db2id
HTML 格式的跨平台样本程序			

表 18. DB2 信息 (续)

名称	说明	书号	HTML 目录
		PDF 文件名	
HTML 格式的样本程序	为所有受 DB2 支持的平台上的程序设计语言提供 HTML 格式的样本程序。提供的样本程序仅供参考。并非所有样本都有所有程序设计语言的版本。HTML 样本仅当安装了“DB2 应用程序开发客户机”时才可用。 有关这些程序的详情，参考应用程序构建指南。	无书号	db2hs
发行说明			
<i>DB2 Connect</i> 发行说明	提供 DB2 书籍中未能包括的最新信息。	参见注释 2。	db2cr
<i>DB2</i> 安装注释	提供 DB2 书籍中未能包括的最新安装特定信息。	仅在产品 CD-ROM 上提供。	
<i>DB2</i> 发行说明	提供 DB2 书籍中未能包括的、有关所有 DB2 产品和功能部件的最新信息。	参见注释 2。	db2ir

注:

1. 文件名第六个位置的字符 *x* 指示书籍的语言版本。例如，文件名 db2d0e70 标识英语版本的管理指南，而文件名 db2d0f70 标识同一本书的法语版本。下列字母用在文件名的第六个位置以指示语言版本:

语言	标识符
巴西葡萄牙语	b
保加利亚语	u
捷克语	x
丹麦语	d
荷兰语	q
英语	e
芬兰语	y
法语	f
德语	g
希腊语	a
匈牙利语	h
意大利语	i
日语	j
韩国语	k
挪威语	n
波兰语	p

葡萄牙语	v
俄语	r
简体中文	c
斯洛文尼亚语	l
西班牙语	z
瑞典语	s
繁体中文	t
土耳其语	m

2. DB2 书籍中未能包括的最新信息以 HTML 格式在“发行说明”中提供，或作为 ASCII 文件提供。在“信息中心”中和产品 CD-ROM 上都提供了 HTML 版本。要查看 ASCII 文件：

- 在基于 UNIX 的平台上，参见 Release.Notes 文件。此文件位于 DB2DIR/Readme/%L 目录中，其中 %L 表示本国语言环境名，而 DB2DIR 表示：
 - 在 AIX 上，是 /usr/lpp/db2_07_01
 - 在 HP-UX、PTX、Solaris 和 Silicon Graphics IRIX 上，是 /opt/IBMdb2/V7.1
 - 在 Linux 上，是 /usr/IBMdb2/V7.1。
- 在其它平台上，参见 RELEASE.TXT 文件。此文件在安装产品的目录中。在 OS/2 平台上，还可双击 **IBM DB2** 文件夹，然后双击**发行说明**图符。

打印 PDF 书籍

如果想要书籍的打印副本，则可打印 DB2 出版物 CD-ROM 上的 PDF 文件。使用 Adobe Acrobat 读入程序，可打印整本书籍或特定范围内的页。有关库中每本书的文件名，参见第354页的表18。

可从 Adobe Web 站点（网址 <http://www.adobe.com>）获取 Adobe Acrobat 读入程序的最新版本。

这些 PDF 文件包括在 DB2 出版物 CD-ROM 上，文件扩展名为 PDF。要存取这些 PDF 文件：

1. 插入 DB2 出版物 CD-ROM。在基于 UNIX 的平台上，安装 DB2 出版物 CD-ROM。参考快速入门一书以了解安装过程。
2. 启动 Acrobat 读入程序。
3. 从下列位置之一打开期望的 PDF 文件：
 - 在 OS/2 和 Windows 平台上：
 - x:\doc\language* 目录，其中 *x* 表示 CD-ROM 驱动器而 *language* 表示两个字符的国家代码，它表示您所用的语言（例如，EN 表示英语）。

- 在基于 UNIX 的平台上:

CD-ROM 上的 `/cdrom/doc/%L` 目录, 其中 `/cdrom` 表示 CD-ROM 的安装点而 `%L` 表示期望的本国语言环境的名称。

还可从 CD-ROM 将 PDF 文件复制至本地或网络驱动器并从该处读取它们。

订购打印书籍

可通过使用销售单 (SBOF) 书号单本地或成套地订购打印的 DB2 书籍 (仅限北美)。要订购书籍, 与 IBM 授权经销商或市场代表联系, 或致电 1-800-879-2755 (美国) 或 1-800-IBM-4YOU (加拿大)。还可从 Publications Web 页 (网址为 <http://www.elink.ibm.link.ibm.com/pbl/pbl>) 订购这些书籍。

有两套书籍。SBOF-8935 提供了“DB2 仓库管理程序”的参考和用法信息。SBOF-8931 提供了所有其他“DB2 通用数据库”产品和功能部件的参考和用法信息。每个 SBOF 的内容列示在下表中:

表 19. 订购打印书籍

SBOF 号	包括的书籍
SBOF-8931	<ul style="list-style-type: none"> • 管理指南: 计划 • 管理指南: 实现 • 管理指南: 性能 • 管理 API 参考 • 应用程序构建指南 • 应用程序开发指南 • CLI Guide and Reference • 命令参考 • Data Movement Utilities Guide and Reference • 数据仓库中心管理指南 • Data Warehouse Center Application Integration Guide • DB2 Connect 用户指南 • 安装和配置补遗 • Image, Audio, and Video Extenders 管理和程序设计 • 信息参考, 第 1 卷和第 2 卷 • OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL 入门 • SQL Reference, 第 1 卷和第 2 卷 • System Monitor Guide and Reference • Text Extender 管理和程序设计 • Troubleshooting Guide • 新增内容
SBOF-8935	<ul style="list-style-type: none"> • Information Catalog Manager Administration Guide • 信息目录管理程序用户指南 • Information Catalog Manager Programming Guide and Reference • Query Patroller 管理指南 • Query Patroller 用户指南

DB2 联机文档

存取联机帮助

随所有 DB2 部件都附带提供了联机帮助。下表描述了各种类型的联机帮助。

帮助类型	内容	如何存取...
命令帮助	说明命令行处理器中命令的语法。	<p>从命令行处理器，以交互方式输入： <code>? command</code></p> <p>其中 <i>command</i> 表示一个关键字或整个命令。</p> <p>例如，<code>? catalog</code> 显示所有 CATALOG 命令的帮助，而 <code>? catalog database</code> 显示 CATALOG DATABASE 命令的帮助。</p>
客户机配置辅助程序帮助	说明您可在窗口或笔记本中执行的任务。此帮助包括您需要知道的概述和前提条件信息，并描述如何使用窗口或笔记本控件。	从窗口或笔记本，单击 帮助 按钮或按 F1 键。
命令中心帮助		
控制中心帮助		
数据仓库中心帮助		
事件分析程序帮助		
信息目录管理程序帮助		
卫星管理中心帮助		
脚本中心帮助		
信息帮助	描述信息的起因以及您应该执行的任何操作。	<p>从命令行处理器，以交互方式输入： <code>? XXXnnnnn</code></p> <p>其中 <i>XXXnnnnn</i> 表示有效的信息标识符。</p> <p>例如，<code>? SQL30081</code> 显示关于 SQL30081 信息的帮助。</p> <p>要每次查看一屏信息帮助，可输入： <code>? XXXnnnnn 尚有</code></p> <p>要在文件中保存信息帮助，可输入： <code>? XXXnnnnn > filename.ext</code></p> <p>其中 <i>filename.ext</i> 表示想要保存信息帮助的文件。</p>

帮助类型	内容	如何存取...
SQL 帮助	说明 SQL 语句的语法。	<p>从命令行处理器，以交互方式输入：</p> <pre>help statement</pre> <p>其中，<i>statement</i> 表示 SQL 语句。</p> <p>例如，<code>help SELECT</code> 显示有关 SELECT 语句的帮助。</p> <p>注：在基于 UNIX 的平台上，SQL 帮助不可用。</p>
SQLSTATE 帮助	说明 SQL 状态及类代码。	<p>从命令行处理器，以交互方式输入：</p> <pre>? sqlstate 或 ? class code</pre> <p>其中，<i>sqlstate</i> 表示有效的五位 SQL 状态，而 <i>class code</i> 表示该 SQL 状态的头两位。</p> <p>例如，<code>? 08003</code> 显示 08003 SQL 状态的帮助，而 <code>? 08</code> 显示 08 类代码的帮助。</p>

查看联机信息

此产品中的书籍为超文本标记语言 (HTML) 软拷贝格式。软拷贝格式使您可搜索或浏览信息，并提供访问相关信息的超文本链接。它还使得在站点间共享库更容易。

可使用遵循 HTML 版本 3.2 规范的任何浏览器来查看联机书籍或样本程序。

要查看联机书籍或样本程序：

- 如果正在运行 DB2 管理工具，则使用“信息中心”。
- 从浏览器，单击**文件** → **打开页**。打开的页中包含 DB2 信息的描述和至 DB2 信息的链接：

- 在基于 UNIX 的平台上，打开以下页：

```
INSTHOME/sql1lib/doc/%L/html/index.htm
```

其中 *%L* 表示本国语言环境名称

- 在其它平台上，打开以下页：

```
sql1lib\doc\html\index.htm
```

该路径位于安装了 DB2 的驱动器上。

如果尚未安装“信息中心”，则可通过双击 **DB2 信息** 图符来打开该页。视您正在使用的系统不同，图符在主产品文件夹中或在“Windows 开始”菜单中。

安装 Netscape 浏览器

如果还未安装 Web 浏览器，则可从产品包装箱中的 Netscape CD-ROM 安装 Netscape。要获取如何安装它的详细指导，执行：

1. 插入 Netscape CD-ROM。
2. 安装 CD-ROM（仅限于在基于 UNIX 的平台上）。参考快速入门一书以了解安装过程。
3. 有关安装指导，参考 CDNAV *nn.txt* 文件，其中 *nn* 表示两字符语言标识符。该文件位于 CD-ROM 的根目录下。

用“信息中心”存取“信息”

“信息中心”提供对 DB2 产品信息的快速存取。在所有装有 DB2 管理工具的平台，都提供了“信息中心”。

可通过双击“信息中心”图符来打开“信息中心”。视正在使用的系统的不同，该图符在主产品文件夹的“信息”文件夹中，或在 Windows 的开始菜单中。

还可通过使用工具栏和 DB2 Windows 平台上的帮助菜单来存取“信息中心”。

“信息中心”提供了六种类型的信息。单击适当的标签来查看提供给该类型的主题。

任务	可使用 DB2 执行的关键任务。
参考	DB2 参考信息，如关键字、命令以及 API。
书籍	DB2 书籍。
疑难解答	错误信息类别及其恢复操作。
样本程序	随“DB2 应用程序开发客户机”一起提供的样本程序。如果未安装“DB2 应用程序开发客户机”，则不显示此标签。
Web	万维网（WWW）上的 DB2 信息。要存取此信息，必须从系统连接至 Web。

当选择其中一个列表中的项时，“信息中心”启动一个查看器来显示信息。视所选择的信息种类的不同，查看器可能是系统帮助查看器、编辑器或 Web 浏览器。

“信息中心”提供了查找功能部件，因此您不用浏览这些列表就能查找特定主题。

对于全文本搜索，请遵循“信息中心”中指向搜索 DB2 联机信息搜索表格的超文本链接。

HTML 搜索服务器通常是自动启动的。如果 HTML 信息中的搜索不起作用，则可能必须使用下列其中一个方法来启动搜索服务器：

在 Windows 上

单击**开始**并选择程序 → **IBM DB2** → **信息** → **启动 HTML 搜索服务器**。

在 OS/2 上

双击 **DB2 OS/2** 版文件夹，然后双击**启动 HTML 搜索服务器**图符。

如果在搜索 HTML 信息时遇到任何其它问题，可参考发行说明。

注：搜索功能在 Linux、PTX 和 Silicon Graphics IRIX 环境中不可用。

使用 DB2 向导

向导通过让您一次一步地完成每一个任务来协助您完成特定管理任务。可通过控制中心和客户机配置辅助程序来获取向导。下表列出了这些向导并描述了它们的用途。

注：“创建数据库”、“创建索引”、“配置多站点更新”和“性能配置”向导对分区数据库环境可用。

向导	帮助您...	如何存取...
添加数据库	在客户机工作站上编目数据库。	从“客户机配置辅助程序”单击添加。
备份数据库	确定、创建并调度应急计划。	从“控制中心”，用鼠标右键单击想要备份的数据库并选择 备份 → 数据库 （使用向导）。
配置多站点更新	配置多站点更新、分布式事务或两阶段落实。	从“控制中心”，用鼠标右键单击 数据库 文件夹并选择 多站点更新 。
创建数据库	创建数据库并执行一些基本配置任务。	从“控制中心”，用鼠标右键单击 数据库 文件夹，并选择 创建 → 数据库 （使用向导）。
创建表	选择基本数据类型并创建表的主关键字。	从“控制中心”，用鼠标右键单击 表 图符，并选择 创建 → 表 （使用向导）。
创建表空间	创建新的表空间。	从“控制中心”，用鼠标右键单击 表空间 图符，并选择 创建 → 表空间 （使用向导）。

向导	帮助您...	如何存取...
创建索引	建议对于所有查询要创建和卸下哪些索引。	从“控制中心”，用鼠标右键单击索引图符，并选择 创建 → 索引（使用向导） 。
性能配置	通过更新配置参数来调整数据库性能以满足您的业务需求。	从“控制中心”，用鼠标右键单击想要调整的数据库并选择 使用向导配置性能 。 对于分区数据库环境，从“数据库分区”视图，用鼠标右键单击想要调整的首个数据库分区并选择 使用向导配置性能 。
复原数据库	在故障之后恢复数据库。它帮助您了解要使用的备份及要重放的纪录。	从“控制中心”，用鼠标右键单击想要复原的数据库并选择 复原 → 数据库（使用向导） 。

设置文档服务器

在缺省情况下，DB2 信息安装在本地系统上。这表示需要存取 DB2 信息的每个人都必须安装相同的文件。要将 DB2 信息存储在单个位置中，执行下列步骤：

1. 将所有文件和子目录从本地系统上的 `\sqllib\doc\html` 复制至 Web 服务器。每一本书都有其自己的子目录，该子目录包含构成该书的所有必需的 HTML 和 GIF 文件。确保目录结构仍相同。
2. 配置 Web 服务器以查找新位置中的文件。有关信息，可参考**安装和配置补遗**中的 NetQuestion 附录。
3. 如果正在使用“信息中心”的 Java 版本，可为所有 HTML 文件指定基本的 URL。您应将该 URL 用于书籍列表。
4. 当能够查看书籍文件时，可将经常查看的主题做成书签。您可能想把下列各页做成书签：
 - 书籍列表
 - 经常使用的书籍的目录
 - 经常引用的文章，如 ALTER TABLE 主题
 - 搜索格式

有关如何从中央机器处理 DB2 通用数据库联机文档文件的信息，参考**安装和配置补遗**中的 NetQuestion 附录。

搜索联机信息

要查找 HTML 文件中的信息，使用下列方法之一：

- 在顶部框中单击**搜索**。使用搜索格式来查找特定的主题。此功能在 Linux、PTX 和 Silicon Graphics IRIX 环境中不可用。
- 在顶部框中单击**索引**。使用索引来查找书中的特定主题。
- 显示帮助或 HTML 书籍的目录或索引，然后使用 Web 浏览器的查找功能查找书中的特定主题。
- 使用 Web 浏览器的书签功能来快速返回至特定的主题。
- 使用“信息中心”的搜索功能来查找特定的主题。参见第366页的『用“信息中心”存取“信息”』以获取详情。

附录E. 注意事项

IBM 可能未在所有国家中提供本文档中讨论的产品、服务或功能部件。关于您所在区域目前可用的产品及服务的信息，请向当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并不说明或暗示只能使用 IBM 的产品、程序或服务。凡是同等功能的产品、程序或服务，只要不侵犯 IBM 的知识产权，都可以用来替代 IBM 产品、程序或服务。当然，评估和验证非 IBM 产品、程序或服务均由用户自行负责。

本文档的议题可能涉及 IBM 的某些专利或正在申请中的专利的应用。提供本文档，并不表示允许您使用这些专利。您可以将许可证查询以书面形式寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

关于双字节 (DBCS) 许可证查询的信息，请与您所在国家的 IBM 知识产权部门联系，将查询以书面形式寄往：

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

以下段落不适用于英国与其它当地法律不允许这种供应方式的国家：国际商用机器公司『按原样』出版此书，不做任何明确或暗示的担保，包括但不限于有关非伪造、商业性或符合特殊目的的隐含保证。一些地区在某些事务中不允许否认拒绝明确或暗示的担保，因此本条款可能不适合您。

本信息中可能有技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些信息将包含在本书新的版本中。IBM 可以随时对本书中说明的产品和/或程序进行改进和/或改动，而不必通知您。

此信息中对非 IBM Web 站点的任何引用仅是为了方便起见，而不以任何方式为那些 Web 站点作保证。那些 Web 站点的资料并非此 IBM 产品资料的一部分，使用那些 Web 站点的风险由您自己承担。

对于您所提供的任何信息，IBM 有权利以任何她认为适当的方式使用或散发，而不必对您负任何责任。

为了以下目的：(1) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换 (2) 允许对已经交换的信息进行相互使用，而希望获取本程序有关信息的合法用户请与下列地址联系：

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

只要遵守适当的条款和条件，包括某些情形下的一定数量的付款，都可获取这方面的信息。

这些信息中描述的特许程序及其所有可用的特许资料，按 IBM 客户协议、IBM 国际程序许可证协议或任何等价的协议中的条款，由 IBM 提供。

此处包含的所有性能数据都是在受控环境中确定的。因此，在其他操作环境中获得的结果可能与之相差很大。某些测量可能是在开发级的系统上进行的，不能保证这些测量方法在通用系统上同样可用。此外，某些测量方法可能是通过外推法归纳来估计的。实际结果可能会有所不同。此文档的用户应针对他们的特定环境验证数据是否适用。

涉及非 IBM 产品的信息可从这些产品的供应商、其发行公告或其它公众可用源得到。IBM 未测试这些产品，因此不能确认性能的精确度、兼容性或其它对非 IBM 产品的索赔赔偿要求等。有关非 IBM 产品功能方面的问题可向它们的供应商提出。

所有关于 IBM 未来方向或意向的声明都可能随时更改或撤消，而不作任何通知，并且仅代表发展目标。

此信息包含了用于日常商业处理的数据和报表的示例。为了尽可能完整地说明问题，这些示例中包含了个人、公司、品牌和产品的名称。所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址相似，纯属巧合。

版权许可证：

本信息中可能包含用源语言编写的示例应用程序，它们说明了各种不同的操作平台上的程序设计技术。您可以为了开发、使用、市场营销或分发应用程序(这些应用程序遵守编写这些示例程序的操作平台的应用程序接口)的目的，以任何形式复

制、修改和分发这些示例程序，不用向 IBM 付费。这些例子未经所有条件下的完整测试。因此，IBM 不能保证或暗示其可靠性、可用性或这些程序的功能。

这些样本程序或任何派生产品的每个副本或任何部分必须包含如下的版权公告：

©（您的公司名称）（年度）。此代码各部分派生自“IBM 公司样本程序”。© Copyright IBM Corp. _输入年份_。All rights reserved.

注册商标

以星号 (*) 标出的下列术语是 IBM 公司在美国和 / 或其他国家的商标。

ACF/VTAM	IBM
AISPO	IMS
AIX	IMS/ESA
AIX/6000	LAN DistanceMVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	OS/2
BookManager	OS/390
CICS	OS/400
C Set++	PowerPC
C/370	QBIC
DATABASE 2	QMF
DataHub	RACF
DataJoiner	RISC System/6000
DataPropagator	RS/6000
DataRefresher	S/370
DB2	SP
DB2 Connect	SQL/DS
DB2 Extender	SQL/400
DB2 OLAP Server	System/370
DB2 Universal Database	System/390
Distributed Relational	SystemView
Database Architecture	VisualAge
DRDA	VM/ESA
eNetwork	VSE/ESA
Extended Services	VTAM
FFST	WebExplorer
First Failure Support Technology	WIN-OS/2

下列各项是其他公司的商标或注册商标：

Microsoft、Windows、和 Windows NT 是 Microsoft 公司的商标或注册商标。

Java 或所有基于 Java 的商标和标志以及 Solaris 是 Sun Microsystems 公司在美国和 / 或其他国家的商标。

Tivoli 和 NetView 是 Tivoli Systems 公司在美国和 / 或其他国家的商标。

UNIX 是经 X/Open 有限公司唯一许可的在美国和 / 或其它国家的注册商标。

以双星号 (**) 标出的其他公司、产品或服务名, 可能是其他公司的商标或服务标志。

索引

[A]

安装

Netscape 浏览器 366

[B]

包含样本程序的目录 11

备份数据库向导 367

本书的结构 3

本书的目录 3

编目样本数据库 37

编译程序

受支持的版本 7

问题 351

标志, 关于 SQL 92 和 MVS 一致性
4

[C]

操作系统

AIX 7

HP-UX 8

Linux 9

OS/2 9

PTX 9

Silicon Graphics IRIX 10

Solaris 10

Windows 32 位 11

操作系统问题 351

查看

联机信息 365

程序设计接口

嵌入式 SQL 1

DB2 API 1

DB2 CLI 1

Java 嵌入式 SQL (SQLJ) 1

Java 数据库链接 (JDBC) 1

出版物, 相关的 1

创建表空间向导 367

创建表向导 367

创建数据库向导 367

创建样本数据库 37

存储过程

和 AIX 上的 CALL 语句 93

将 SPARCompiler C++ 用于

Solaris 282

使用 HP-UX C 154, 159

使用 HP-UX C++ 167

使用 IBM C Set++ AIX 版 113

使用 IBM COBOL Set AIX 版
138

使用 Linux C 187

使用 Linux C 构建 CLI 182

使用 Linux C++ 195

在 AIX 上使用 IBM C 104

在 AIX 上使用 IBM C 构建
CLI 98

在 AIX 上使用 VisualAge

C++ 130

在 HP-UX 上使用 Micro Focus
COBOL 176

在 OS/2 上使用 VisualAge C++
版本 3 构建嵌入式 SQL 209

在 OS/2 上使用 VisualAge C++
版本 3 构建 CLI 204

在 PTX 上使用 ptx/C 233

在 PTX 上使用 ptx/C 构建
CLI 228

在 PTX 上使用 ptx/C++ 241

在 Solaris 上使用 SPARCompiler
C 272

在 Solaris 使用 SPARCompiler C
构建 CLI 267

在 Windows 上使用 Visual C++
的嵌入式 SQL 312

在 Windows 上使用 Visual C++
构建 CLI 307

在 Windows 上使用 VisualAge
3.5 C++ 构建 CLI 321

在 Windows 上使用 VisualAge
COBOL 334

存储过程 (续)

在 Windows 上使用 VisualAge
C++ 3.5 326

在 Windows 上用于嵌入式 SQL
Micro Focus COBOL 339

在 Windows 上用 Visual Basic 实
现 OLE 自动化 302

在 Windows 上用 Visual C++ 实
现 OLE 自动化 304

AIX 入口点 92

AIX 上的 Micro Focus
COBOL 144

AIX 上的 VisualAge C++ 124

AIX 上 Micro Focus COBOL 的
封装器程序 146

C++ 考虑事项 54

Java JDBC 70

Java JDBC 客户机应用程序 69

Java SQLJ 76

Java SQLJ 客户机应用程序 75

OS/2 上的 Micro Focus

COBOL 222

Silicon Graphics IRIX DB2 CLI
客户机应用程序 251

Silicon Graphics IRIX MIPSpro C
嵌入式 SQL 客户机应用程序
255

Silicon Graphics IRIX MIPSpro
C++ 嵌入式 SQL 客户机应用程
序 259

Solaris 上的 Micro Focus
COBOL 292

Solaris 上 Micro Focus COBOL
的封装器程序 294

VisualAge COBOL OS/2 版 217

存储过程的 bldsrv 脚本文件

使用 HP-UX C 159

使用 Linux C 187

在 PTX 上使用 ptx/C 233

在 PTX 上使用 ptx/C++ 241

存储过程的 bldsrv 脚本文件 (续)
 在 Solaris 上使用 SPARCompiler
 C 272
存储过程构建器
 在 DB2 AD 客户机中的支持 4
 作为数据库工具 xxii
错误检查实用程序 48
错误信息和错误记录 351

[D]

打印 PDF 书籍 361
代码样本, 包括在 DB2 AD 客户机中
 4
多线程应用程序
 关于 54
 使用 HP-UX C 163
 使用 HP-UX C++ 172
 使用 Linux C 191
 使用 Linux C++ 199
 在 AIX 上使用 IBM C 109
 在 AIX 上使用 IBM C
 Set++ 119
 在 PTX 上使用 ptx/C 237
 在 PTX 上使用 ptx/C++ 245
 在 Silicon Graphics IRIX 上使用
 MIPSpro C 256
 在 Silicon Graphics IRIX 上使用
 MIPSpro C++ 260
 在 Solaris 上使用 SPARCompiler
 C 278
 在 Solaris 上使用 SPARCompiler
 C++ 288

[F]

发行说明 361
服务器
 配置通信协议 36
 启动通信 36
 受支持的 6
 问题 351
复原向导 368

[G]

工具
 在 DB2 AD 客户机中 4

工具 (续)
 诊断 351
 构建文件, 关于 44
 关于本书 1
 关于 DB2 AD 客户机 4

[H]

环境
 设置 OS/2 32
 设置 UNIX 33
 设置 Windows 34

[J]

记录, 错误 351

[K]

客户机问题 351

[L]

联编样本数据库 37
联机帮助 363
联机错误信息 351
联机信息
 查看 365
 搜索 369

[N]

您需要的背景知识 3

[P]

配置多站点更新向导 367
配置通信协议 36
配置文件
 使用 VisualAge C++ 版本 4 120
 使用 VisualAge C++ OS/2 版
 213
 使用 VisualAge C++ Windows 版
 331
 AIX 上的 api.icc 127
 AIX 上的 clis.icc 124

配置文件 (续)
 AIX 上的 cli.icc 121
 AIX 上的 emb.icc 128
 AIX 上的 stp.icc 130
 AIX 上的 udf.icc 133

[Q]

迁移应用程序 345
前提条件
 编译程序 7
 操作系统 7
 环境设置 31
 您需要的程序设计知识 3
前缀, 错误信息 351
嵌入式 SQL
 构建应用程序 51
 样本程序 11
嵌入式 SQL 的 bldapp 脚本文件
 使用 HP-UX C 157
 使用 Linux C 184
 使用 Linux C++ 192
 在 PTX 上使用 ptx/C 231
 在 PTX 上使用 ptx/C++ 238
 在 Silicon Graphics IRIX 上使用
 MIPSpro C 252
 在 Solaris 上使用 SPARCompiler
 C 269
 Solaris 上的 Micro Focus
 COBOL 289

[R]

如何使用本书 3
软件, 受支持的 7

[S]

设置环境 31
设置文档服务器 368
实例名和主目录 343
使用本书 1
示例文本, 使用 3
受支持的编译程序版本 7
书籍 353, 362
书, 关于本 1

数据库管理程序实例

创建 31

关于 343

谁应使用本书 3

搜索

联机信息 366, 369

索引向导 367

[T]

添加数据库向导 367, 368

通信, 服务器上启用 36

[W]

文档, 相关的 1

问题确定 351

[X]

相关出版物 1

向导

备份数据库 367

创建表 367

创建表空间 367

创建数据库 367

复原数据库 368

配置多站点更新 367

索引 367

添加数据库 367, 368

完成任务 367

性能配置 368

斜体字, 使用 3

信息中心 366

信息, 联机错误 351

性能配置向导 368

[Y]

样本程序

含嵌入式 SQL 51

跨平台 359

列出 11

HTML 359

样本数据库, 创建 37

应用程序

嵌入式 SQL 51

应用程序 (续)

DB2 CLI 51

Java 57

Java JDBC 69

Java SQLJ 75

应用程序开发 (DB2 AD) 客户机, 关于 DB2 4

用户定义函数 (UDF)

关于 54

和 AIX 上的 CREATE

FUNCTION 语句 94

和 AIX 上的 EXTERNAL NAME

子句 94

将 SPARCompiler C++ 用于

Solaris 285

使用 HP-UX C 161

使用 HP-UX C++ 170

使用 IBM C Set++ AIX 版 116

使用 Linux C 189

使用 Linux C++ 197

在 AIX 上使用 IBM C 107

在 AIX 上使用 VisualAge

C++ 133

在 OS/2 上使用的 VisualAge C++

版本 3 211

在 PTX 上使用 ptx/C 235

在 PTX 上使用 ptx/C++ 243

在 Solaris 上使用 SPARCompiler

C 275

在 Windows 上使用 Visual

C++ 315

在 Windows 上使用 VisualAge

C++ 3.5 329

在 Windows 上用 Visual Basic 实

现 OLE 自动化 302

在 Windows 上用 Visual C++ 实

现 OLE 自动化 304

AIX 入口点 92

C++ 考虑事项 54

Java 79

Java JDBC 客户机应用程序 70

Java SQLJ 客户机应用程序 76

Silicon Graphics IRIX DB2 CLI

客户机应用程序 252

用户定义函数 (UDF) (续)

Silicon Graphics IRIX MIPSpro C

嵌入式 SQL 客户机应用程序

255

Silicon Graphics IRIX MIPSpro

C++ 嵌入式 SQL 客户机应用程

序 259

用于检查错误的实用程序 48

用于 HP-UX C++ 存储过程的 bldsrv

脚本文件 167

用于 HP-UX C++ UDF 的 bldudf 脚

本文件 170

用于 IBM C Set++ AIX 版存储过程

的 bldsrv 脚本文件 113

用于 IBM C Set++ AIX 版 UDF 的

bldudf 脚本文件 116

用于 IBM COBOL Set AIX 版存储过

程的 bldsrv 脚本文件 138

用于 Linux C++ 存储过程的 bldsrv

脚本文件 195

用于 Linux C++ UDF 的 bldudf 脚本

文件 197

用于 Solaris 存储过程的

SPARCompiler C++ 的 bldsrv 脚本

文件 282

用于 Solaris UDF 的 SPARCompiler

C++ 的 bldudf 脚本文件 285

语法问题 351

语言标识符

书籍 360

语言, 受支持的 7

预编译程序

包括在 DB2 AD 客户机中 4

远程服务器连接 31

远程数据对象 (RDO)

在 DB2 AD 客户机中的支持 4

在 Windows 上使用 Visual

Basic 300

[Z]

在服务器上启用通信 36

在 AIX 上用于 C 存储过程的 bldsrv

脚本文件 104

在 AIX 上用于 C UDF 的 bldudf 脚

本文件 107

在 DB2 AD 客户机中启用预编译程序支持的 API 4
在 DB2 AD 客户机中支持的 Microsoft ODBC 4
诊断工具 351
主机服务器, 创建 39
主目录, 实例 343
最新信息 361

[特别字符]

“命令行处理器”(CLP) 文件 11

A

ActiveX 数据对象
在 DB2 AD 客户机中的支持 4
在 Windows 上使用 Visual Basic 299
在 Windows 上使用 Visual C++ 303
AIX 上的 bldcli 脚本文件 95
AIX 上的 bldclisp 脚本文件 98
AIX 上的 CALL 语句和存储过程 93
AIX 上用于 C 的 bldapp 脚本文件 101
AIX 上用于 C++ 的 bldapp 脚本文件 110
AIX 上用于 Micro Focus COBOL 存储过程的 bldsrv 脚本文件 144
AIX 上用于 Micro Focus COBOL 存储过程的 wrapsrv 脚本文件 146
AIX/6000, 受支持的版本 7
API, DB2
关于 50
Applet
一般要点 80
Java 57
Java JDBC 68
Java SQLJ 74
AS/400 服务器, 创建 39

C

CALL CLP 命令 87
calludf 样本程序 51

checkerr.cbl, 用于 COBOL 错误检查 48
CLI 中的静态 SQL xii
CLI, DB2
存储过程的 Silicon Graphics IRIX 客户机应用程序 251
关于 51
静态 SQL xii
问题确定 351
样本程序 11
在 AIX 上使用 VisualAge C++ 编写的存储过程 124
在 AIX 上使用 VisualAge C++ 编写的应用程序 121
AIX 存储过程 98
AIX 应用程序 95
HP-UX 存储过程 154
HP-UX 应用程序 152
Linux 存储过程 182
Linux 应用程序 179
OS/2 VisualAge 版本 3 存储过程 204
OS/2 VisualAge 版本 3 应用程序 202
PTX 存储过程 228
PTX 应用程序 225
Silicon Graphics IRIX 应用程序 248
Solaris 存储过程 267
Solaris 应用程序 264
UDF 的 Silicon Graphics IRIX 客户机应用程序 252
VisualAge 3.5 Windows 版存储过程 321
VisualAge 3.5 Windows 版应用程序 318
Windows 存储过程 307
Windows 应用程序 304
CLP 样本文件 11
COBOL 编译程序
安装和运行 92
使用 IBM COBOL Set AIX 版编译程序 135
使用 VisualAge COBOL OS/2 版 213
受支持的版本 7

COBOL 编译程序 (续)
在 Windows 上使用 VisualAge COBOL 331

CREATE FUNCTION 语句和 UDF 94

C++
UDF 和存储过程 54
C/C++ 编译程序, 受支持的版本 7

D

DB2 库
查看联机信息 365
打印 PDF 书籍 361
订购打印书籍 362
联机帮助 363
设置文档服务器 368
书籍的语言标识符 360
搜索联机信息 369
向导 367
信息中心 366
最新信息 361

DB2 资料库

结构 353
书籍 353

DB2 AD 客户机提供的开发环境 4

DB2 AD 客户机中的包含文件 4

DB2 AD 客户机中的“命令行处理器”(CLP) 4

DB2 CLI, 关于 51

db2sampl, 用于创建样本数据库 37

DFTDBPATH, 使用它来指定缺省路径 37

Domino Go 80

E

expsamp 程序, 用来调出表 39

EXTERNAL NAME 子句和 UDF 94

F

FORTAN 编译程序, 受支持的版本 7

H

HP-UX 上的 bldcli 脚本文件 152

HP-UX 上的 bldclisp 脚本文件 154

HP-UX 上用于 Micro Focus COBOL
存储过程的 bldsrv 脚本文件 176
HP-UX 上用于 Micro Focus COBOL
的 bldapp 脚本文件 174
HP-UX C++ 的 bldapp 脚本文件
165
HP-UX, 受支持的版本 8
HTML
 样本程序 359

I

IBM COBOL Set AIX 版的 bldapp
脚本文件 136

J

Java

 构建文件 71
 构建 JDBC 存储过程 70
 构建 JDBC 应用程序 69
 构建 JDBC Applet 68
 构建 SQLJ 程序 71
 构建 SQLJ 存储过程 76
 构建 SQLJ 应用程序 75
 构建 SQLJ Applet 74
 构建 UDF 79
 关于 49
 设置 AIX 环境 58
 设置 HP-UX 环境 59
 设置 Linux 环境 60
 设置 Silicon Graphics IRIX 环境
 63
 设置 Solaris 环境 65
 设置 Windows 环境 66
 设置 OS/2 环境 61
 样本程序 11, 68
 在 DB2 AD 客户机中的支持 4
 支持平台 7
 DB2 Applet 的一般要点 80
 JDBC 存储过程的客户机应用程序
 69
 OS/2 的 HPFS 驱动器 68
 UDF 的 JDBC 客户机应用程序
 70
Java SQLJ 的 bldsqlj 构建文件 71
Java SQLJ 的 bldsqljs 构建文件 76

JDBC

 程序 68
 存储过程的客户机应用程序 69
 构建存储过程 70
 构建应用程序 69
 构建 Applet 68
 在 DB2 AD 客户机中的支持 4
 DB2 JDBC 支持 57
 UDF 的客户机应用程序 70

L

Linux 上的 bldcli 脚本文件 179
Linux 上的 bldclisp 脚本文件 182
Linux, 受支持的版本 9
Lotus Domino Go 80

M

makefile

 关于 46
 用于 Java 68
Micro Focus COBOL
 安装和运行 92
 在 AIX 上使用编译程序 141
 在 HP-UX 上使用编译程序 173
 在 OS/2 上使用编译程序 219
 在 Solaris 上使用编译程序 289
 在 Windows 上使用编译程序 336
 支持平台 7
 AIX 上用于存储过程的封装器程序
 146
 OS/2 上的 DB2 API 连接调用约
 定 8 219
 OS/2 上的 DB2API.lib 219
 Solaris 上存储过程的封装器程序
 294
 Windows 上的 DB2 API 连接调用
 约定 74 336
 Windows 上的 DB2API.lib 336
Microsoft Windows 32 位, 受支持的
 版本 11

N

Netscape 浏览器
 安装 366

O

Object REXX

 在 Windows NT 上运行程序 341

ODBC

 和受支持的服务器 6
 在 DB2 AD 客户机中支持 4

OLE 样本程序 11

OLE 自动化

 在 DB2 AD 客户机中的支持 4
 在 Windows 上使用 Visual
 Basic 302
 在 Windows 上使用 Visual
 C++ 303
 Windows 上的 Visual Basic 存储
 过程 302
 Windows 上的 Visual Basic
 UDF 302
 Windows 上的 Visual C++ 存储过
 程 304
 Windows 上的 Visual C++
 UDF 304

OLE DB 表函数

 在 DB2 AD 客户机中的支持 4
 在 Windows 上使用 298

ORG 表, 创建并调出 39

OS/2 上的命令文件

 用于 Micro Focus COBOL 存储过
 程的 bldsrv 222
 用于 Micro Focus COBOL 的
 bldapp 220
 用于 VisualAge COBOL 存储过程
 的 bldsrv 217
 用于 VisualAge COBOL 的
 bldapp 214
 用于 VisualAge C++ 存储过程的
 bldsrv 209
 用于 VisualAge C++ 的
 bldapp 206
 用于 VisualAge C++ UDF 的
 bldudf 211
Java SQLJ 的 bldsqlj 71
Java SQLJ 的 bldsqljs 76
VisualAge C++ 的 bldcli 202
VisualAge C++ 的 bldclisp 204
OS/390 服务器, 创建 39

outcli 样本程序 51
outsrv 样本程序 51

P

PDF 361
PTX 上的 bldcli 脚本文件 225
PTX 上的 bldclisp 脚本文件 228
PTX, 受支持的版本 9

R

REXX
 在 AIX 上设置和运行程序 148
 在 AIX 上受支持的版本 7
 在 DB2 AD 客户机中的支持 4
 在 OS/2 上运行程序 224
 在 Windows NT 上运行程序 341
REXX 程序中的注解 224, 341

S

Silicon Graphics IRIX 上的 bldcli 脚本文件 248
Silicon Graphics IRIX 上 MIPSpro C++ 的 bldapp 脚本文件 257
Silicon Graphics IRIX, 受支持的版本 10
SmartGuide
 向导 367
Solaris 上的 bldcli 脚本文件 264
Solaris 上的 bldclisp 脚本文件 267
Solaris 上 Micro Focus COBOL 存储过程的 bldsrv 脚本文件 292
Solaris 上 Micro Focus COBOL 存储过程的 wrpsrv 脚本文件 294
Solaris 上 SPARCompiler C++ 的 bldapp 脚本文件 279
Solaris, 受支持的版本 10
SPECIAL-NAMES 段 219, 336
SQL 过程
 和 CLP CALL 命令 87
 和 CREATE PROCEDURE 语句 87
 设置环境 83
SQLCA 数据结构 351
SQLJ
 存储过程 76

SQLJ (续)

 存储过程的客户机应用程序 75
 构建程序 71
 构建应用程序 75
 在 DB2 AD 客户机中的支持 4
 Applet 74
 bldsqlj 构建文件 71
 bldsqljs 构建文件 76
 DB2 SQLJ 支持 57
 UDF 的客户机应用程序 76
STAFF 表, 创建并调出 39

U

UDF 的 bldudf 脚本文件
 使用 HP-UX C 161
 使用 Linux C 189
 在 PTX 上使用 ptx/C 235
 在 PTX 上使用 ptx/C++ 243
 在 Solaris 上使用 SPARCompiler C 275
udf 样本程序 51
updat 样本程序 51
utilapi.c, 用于 C 错误检查 48
utilapi.c, 用于 CLI 错误检查 48
utilapi.C, 用于 C++ 错误检查 48
utilcli.c, 用于 CLI 错误检查 48
utilemb.sqc, 用于 C 错误检查 48
utilemb.sqC, 用于 C++ 错误检查 48

W

web 服务器 80
Windows 上的批处理文件
 IBM VisualAge COBOL 的
 bldapp 332
 Java SQLJ 的 bldsqlj 71
 Java SQLJ 的 bldsqljs 76
 Micro Focus COBOL 存储过程的
 bldsrv 339
 Micro Focus COBOL 的
 bldapp 337
 Visual C++ 存储过程的
 bldmsrv 312
 Visual C++ 的 bldcli 304
 Visual C++ 的 bldmapp 310
 Visual C++ 的 bldmclis 307

Windows 上的批处理文件 (续)

 Visual C++ UDF 的 bldmudf 315
 VisualAge COBOL 存储过程的
 bldsrv 334
 VisualAge C++ 存储过程的
 bldvsrv 326
 VisualAge C++ 的 bldvapp 323
 VisualAge C++ 的 bldvcli 318
 VisualAge C++ 的 bldvclis 321
 VisualAge C++ 3.5 UDF 的
 bldvudf 329
Windows 32 位, 受支持的版本 11
Windows NT 的 CONVERT 选项 297
Windows NT 的 mbstowcs() 函数 297
Windows NT 的 NOCONVERT 选项 297
Windows NT 的 setlocale() 函数 297
Windows NT 的 WCHARTYPE
 CONVERT 预编译选项 297
Windows NT 的 wcstombs() 函数 297
Windows NT 宽字符格式 297
Windows NT 上 IBM VisualAge
 COBOL 存储过程的 bldsrv 批处理
 文件 334

与 IBM 联系

如果有技术问题，请在与“DB2 客户支持中心”联系之前复查并执行 *Troubleshooting Guide* 所建议的操作。本指南对您可以收集哪些信息以使“DB2 客户支持中心”更好地为您服务提出了建议。

要获取信息或订购任何“DB2 通用数据库”产品，与当地分支机构的 IBM 代表联系，或与任何特许 IBM 软件经销商联系。

您如果住在美国，请致电下列其中一个号码：

- 1-800-237-5511，可获得客户支持
- 1-888-426-4343，可了解所提供的服务项目

产品信息

您如果住在美国，请致电下列其中一个号码：

- 1-800-IBM-CALL (1-800-426-2255) 或 1-800-3IBM-OS2 (1-800-342-6672)，可订购产品或获取一般信息。
- 1-800-879-2755，可订购出版物。

<http://www.ibm.com/software/data/>

DB2 万维网网页提供关于新闻、产品说明、培训计划等等的当前 DB2 信息。

<http://www.ibm.com/software/data/db2/library/>

“DB2 产品和服务技术库”可供您访问常见问题、修订、书籍以及最新的 DB2 技术资料。

注：此资料可能只有英文版。

<http://www.elink.ibm.com/pbl/pbl/>

International Publications Ordering Web 站点提供关于如何订购书籍的信息。

<http://www.ibm.com/education/certify/>

IBM Web 站点中的“专业认证程序”提供各种 IBM 产品（包括 DB2）的认证测试信息。

<ftp.software.ibm.com>

以匿名形式注册。可在目录 /ps/products/db2 中找到有关 DB2 和许多其他产品的演示程序、修订、信息和工具。

comp.databases.ibm-db2, bit.listserv.db2-l

这些 Internet 新闻组可供用户来讨论使用 DB2 产品的经验。

On CompuServe: GO IBMDB2

输入此命令来访问 IBM DB2 系列论坛。这些论坛支持所有的 DB2 产品。

有关如何在美国以外的地区与 IBM 联系的信息，参见 IBM Software Support Handbook 的附录 A。要存取此文档，访问以下 Web 页面：<http://www.ibm.com/support/>，然后选择该页面底部附近的 IBM Software Support Handbook 链接。

注：在某些国家，IBM 特许经销商应与他们的经销商支持机构联系，而不是与“IBM 支持中心”联系。



Part Number: CT7XXSC

Printed in China

SB84-0220-00



CT7XXSC

