DB2 通用数据库

Image Extender、Audio Extender 和 Video Extender 管理和编程

版本 8

S152-0598-00

DB2 通用数据库

Image Extender、Audio Extender 和 Video Extender 管理和编程

版本 8

S152-0598-00

在使用本资料及其支持的产品之前，请阅读*声明*中的一般信息。

第一版，2003 年 6 月

本文档包含 IBM 的专利信息。它在许可证协议下提供，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。

可以在线方式或通过您当地的 IBM 代表订购 IBM 出版物。

- 要以在线方式订购出版物，可访问“IBM 出版物中心”（IBM Publications Center），网址为 www.ibm.com/shop/publications/order。
- 要查找您当地的 IBM 代表，可访问“IBM 全球联系人目录”（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide。

要从美国或加拿大的 DB2 营销机构订购 DB2 出版物，请拨打电话 1-800-IBM-4YOU（426-4968）。

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 可以它认为合适的任何方式使用或分发此信息，而无须对您承担任何责任。

© Copyright International Business Machines Corporation 1998, 2003. All rights reserved.

目录

© Copyright IBM Corp. 1998, 2003

图

© Copyright IBM Corp. 1998, 2003

表

© Copyright IBM Corp. 1998, 2003

关于本书

本书描述如何使用 DB2 Extender 来准备和维护用来处理图像、音频或视频数据的 DB2® 数据库。它还描述了如何使用 DB2 Extender 提供的用户定义函数（UDF）和应用程序编程接口（API）来存取和处理这些类型的数据。通过将 UDF 合并到程序的 SQL 语句中，并合并 API，您可以存取非传统数据（如图像和视频剪辑）和传统的数字数据和字符数据。

本书中提到的“DB2”指的就是 DB2 UDB。

谁应使用本书

本书面向熟悉 DB2 管理概念、工具和技术的 DB2 数据库管理员。

本书还面向熟悉 SQL 并熟悉可用于 DB2 应用程序的一种或多种编程语言的 DB2 应用程序员。

本书是为将使用 DB2 Image Extender、Audio Extender 和 Video Extender 的人准备的。使用 Text Extender 的人员应参见*《DB2 Text Extender 管理和编程》*。

使用本书的方法

本书的结构如下:

『第 1 部分: 介绍』

此部分给出 DB2 Extender 的概述。若不熟悉用 DB2 Extender 进行管理或编程，则阅读此部分。

『第 2 部分: 管理图像、音频和视频数据』

此部分描述如何准备和维护用来处理图像、音频和视频数据的 DB2 数据库。若需要管理包含图像、音频或视频数据的 DB2数据库，则阅读此部分。

『第 3 部分: 对图像、音频或视频数据进行编程』

此部分描述如何使用 DB2 Extender UDF 和 API 来请求对图像、音频或视频数据的操作。若需要在 DB2 应用程序中存取和处理图像、音频或视频数据，则阅读此部分。

『第 4 部分: 参考』

此部分介绍 DB2 Extender UDF、API、管理命令的参考信息以及诊断信息，如消息和代码。若您熟悉 DB2 Extender 概念和任务，但需要关于特定的 DB2 Extende UDF、API、命令、消息或代码的信息，则阅读此部分。

『附录』

附录描述:

- 如何设置环境变量，DB2 Extender 使用这些环境变量来查找文件并标识图像、音频和视频对象的显示程序或播放程序。
- 如何安装和使用随 Extender 一起提供的样本程序和媒体文件

© Copyright IBM Corp. 1998, 2003

特定平台信息

DB2 Extender 可与“DB2 通用数据库”的单分区数据库环境一起使用，或与“DB2 通用数据库扩充企业版”的多分区数据库环境一起使用。

本书包含关于在任一环境中使用 DB2 Extender 的信息。仅与在“DB2 通用数据库扩充企业版”的多分区环境中使用 Extender 相关的信息标记为“**仅限于 EEE**”。仅与在“DB2通用数据库”的单分区环境中使用 Extender 相关的信息标记为“**仅限于非 EEE**”。未标记为与特定环境相关的信息适用于两个环境。

突出显示约定

本书使用下列约定:

粗体 粗体文本用来指示新术语的定义。

斜体 斜体指示将用值替换的变量参数，或强调文本中使用的字。

大写 大写字母指示:
- 数据类型
- 目录名
- 字段名
- API 调用
- 命令
- 关键字
- 变量名称

示例 示例文本指示系统信息或输入的值。示例文本也用于编码示例。

如何阅读语法图

在整本书中，命令和 SQL 语法是使用语法图描述的。阅读语法图的方法如下所述:

- 顺着线路从左到右、从上到下阅读语法图。

  ►►── 符号指示语句开始。

  ──► 符号指示语句语法在下一行上继续。

  ►── 符号指示语句继续前一行。

  ──►◄ 符号指示语句结束。

- 必需的项出现在水平线（主路径）上。

```
►►─required item──────────────────────────────────────────►◄
```

- 可选的项出现在主路径下面。

```
►►──┬──────────────────────────────────────────────────────►◄
    └─optional item─┘
```

- 若可从两个或多个项中进行选择，则它们出现在堆栈中。

  若*必须*选择其中一个项，则堆栈的其中一个项出现在主路径上。

```
►►──┬─required choice1─┬──────────────────────────────────►◄
    └─required choice2─┘
```

  若这些项都不是选项，则整个堆栈出现在主路径下面。

堆栈上方的双箭头指示可选择堆栈中的多个项。

- 关键字以大写形式出现（例如，`/DB2IMAGE:`）。它们必须完全按显示的形式来拼写。变量以小写形式出现（例如，`srcpath`）。它们代表语法中用户提供的名称或值。
- 若显示标点符号、圆括号、算术运算符或其它这样的符号，则必须作为语法的一部分输入它们。

相关信息

DB2 通用数据库 V8

DB2 Universal Database Quick Beginnings Version 8 for DB2 Servers (GC09–4836), for DB2 Clients (GC09–4832), for DB2 Connect Personal Edition (GC09–4834), for DB2 Personal Edition (GC09–4838), and IBM Data Links Manager (GC09–4829–00)。这些书籍描述如何在适当的平台上计划、安装、配置和迁移“DB2 通用数据库”。

IBM DB2 Universal Database Administration Guide Version 8 Planning (SC09–4822), Performance (SC09–4821), and Implementation (SC09–4820)。这些书籍描述如何设计和实现 DB2 数据库。

IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1, Version 8 (SC09–4849)。本书描述如何开发使用“DB2 调用层接口”来存取 DB2 数据库的应用程序，“DB2 调用层接口”是符合 Microsoft ODBC 规范的可调用 SQL 接口。

IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2, Version 8 (SC09–4850)。本书描述如何开发使用“DB2 调用层接口”来存取 DB2 数据库的应用程序，“DB2 调用层接口”是符合 Microsoft ODBC 规范的可调用 SQL 接口。

IBM DB2 Universal Database Command Reference Version 8 (SC09–4829)。本书描述如何使用 DB2 命令行处理器，并给出关于 DB2 命令的参考信息。

DB2 通用数据库 Text Extender

《DB2 通用数据库 Text Extender 管理和编程 V8》，S152-0597。本书描述如何管理 DB2 数据库的文本数据。它还描述如何使用 DB2 Text Extender 提供的应用程序编程接口来存取和处理文本数据。

DB2 通用数据库 XML Extender

《DB2 通用数据库 XML Extender 管理和编程》。本书描述如何管理 DB2 数据库的 XML 文档。它还描述如何使用 DB2 XML Extender 提供的应用程序编程接口来存取和处理 XML 文档和数据。

DB2 通用数据库 Spatial Extender

《Spatial Extender 用户指南和参考》，SB84-0249。本书提供有关 Spatial Extender 的安装、配置、管理、编程和故障排除的信息。还提供了空间数据概念的重要描述，并提供了特定于 Spatial Extender 的参考信息（消息和 SQL）。

DB2 通用数据库 z/OS 版 Image Extender、Audio Extender 和 Video Extender

DB2 Universal Database for z/OS Version 6 Image, Audio, and Video Extenders Administration and Programming, SC26-9650。本书描述如何管理用于图像、音频和视频数据的 DB2 z/OS 版数据库服务器。它还描述如何使用用户定义函数和 DB2 z/OS 版 Image Extender、Audio Extender 和 Video Extender 提供的应用程序编程接口来存取和处理图像、音频和视频数据。

DB2 通用数据库 z/OS 版 Text Extender

DB2 Universal Database for z/OS Version 6 Text Extender Administration and Programming, SC26-9651。本书描述如何管理用于文本数据的 DB2 z/OS 版数据库服务器。它还描述如何使用用户定义函数和 DB2 z/OS 版 Text Extender 提供的应用程序编程接口来存取和处理文本数据。

如何发送意见

您的反馈能帮助 IBM 提供高质量的信息。请将您对本书或其它 DB2 Extender 文档的意见发送给我们。可使用以下方法来提供意见:

- 通过电子邮件将您的意见发送至 ctscrcf@cn.ibm.com。请务必提供书名、书的部件号、产品版本以及（如果可能的话）您有意见的文本的特定位置（如页码或表号）。

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 可以它认为合适的任何方式使用或分发此信息，而无须对您承担任何责任。

有关与 IBM 联系的更多信息，参见第 509 页的『与 IBM 联系』。

第 1 部分 介绍

© Copyright IBM Corp. 1998, 2003

第 1 章 介绍

管理 Extender 服务器

DB2 Extender 在 DB2 客户机 / 服务器环境中运行。此环境由数据库服务器和一个或多个远程数据库客户机组成。DB2 Extender 服务在服务器上运行。在存取它们之前，必须先启动它们。

设置环境之后，可从客户机停止并重新启动 Extender 服务。可从客户机或服务器获取 Extender 的状态。

仅限于 EEE：在多分区环境中，还可添加和删除数据库分区。

建立 Extender 环境

从服务器上的操作系统命令行，输入 DMBSTART 命令以启动 Extender 服务:

```
dmbstart
```

DMBSTART 命令为允许存放 Extender 数据的所有数据库启动 Extender 服务。此命令还可启动 DB2（若它不在运行之中）。需要 SYSADM、SYSCTRL 或 SYSMAINT 权限才可运行此命令。在 AIX 上，必须登录为 Extender 实例所有者。

现在，若 C 语言应用程序建立与数据库的连接，该应用程序就可通过 API 存取 Extender 服务。同样，若要使用 db2ext 命令行，必须与要使用的数据库连接。db2ext 命令行需要不同于 DB2 命令行使用的连接的单独连接。

打开客户机上的 db2ext 命令行处理器，并运行 DB2 Extender 的 CONNECT 命令。在下列示例中，该命令连接至 PERSONNL 数据库。它使用 ANPASS 密码存取带 ANITAS 限定符的表:

```
connect to personnl user anitas using anpass
```

仅限于 EEE：若正在分区数据库环境中使用 DB2 Extender，则 DMBSTART 命令将在对实例定义的所有数据库分区服务器中启动 Extender 服务。若仅想在一个数据库分区服务器上启动 Extender 服务，在命令中指定要启动的节点即可。以下示例显示若要在节点号 2 上启动 Extender 服务，将要输入什么内容。

```
dmbstart nodenum 2
```

仅限于 EEE：在启动单个数据库分区服务器之前，必须在该节点上启动 DB2。

现在可运行列示在第 411 页的第 15 章，『客户机的管理命令』中的 DB2 Extender 命令的剩余部分。

添加和删除数据库分区（仅限于 EEE）

为了在分区数据库环境中使用 Extender，对 Extender 定义的分区必须与对 DB2 定义的分区相匹配。DMBSTART 命令在当前实例定义的每个节点上启动 Extender 服务器。服务器将自动检测它在其上运行的节点是否是最近创建的，并执行任何必需的初始化。若从 DB2 中删除节点，则必须手工删除与该节点相关联的 Extender 文件。

© Copyright IBM Corp. 1998, 2003

添加和删除分区

有关添加和删除分区的 DB2 命令的更多信息，参考 *IBM DB2 Universal Database Enterprise Extended-Edition Quick Beginnings*。

下列步骤是添加数据库分区所必需的:

1. 使用命令 DB2NCRT 或命令 DB2START ADDNODE 创建 DB2 的分区。
2. 使用 Extender 命令 DMBSTART NODENUM 创建 Extender 的分区。
3. 使用 DB2 命令 REDISTRIBUTE NODEGROUP 再分发 DB2 数据以利用新节点配置。
4. 使用 Extender 命令 REDISTRIBUTE NODEGROUP 再分发 Extender 数据以利用新节点配置。

下列步骤是删除数据库分区所必需的:

1. 使用 DB2 命令 REDISTRIBUTE NODEGROUP 再分发 DB2 数据，以从要删除的分区中除去它。
2. 使用 Extender 命令 REDISTRIBUTE NODEGROUP 再分发 Extender 数据，以从要删除的分区中除去它。
3. 通过使用 DB2 命令 DB2NDROP 或命令 DB2STOP DROP 删除 DB2 的分区。
4. 使用 Extender 命令 DMBSTART NODENUM 删除 Extender 的分区。
5. 手工除去与删除的分区相关联的 Extender 文件。

数据库分区的 Extender 数据在名为 node*num* 的子目录中，其中 *num* 是与该数据库分区相对应的节点号。该子目录在 DB2MMDATAPATH 环境变量的值指定的目录中。要除去删除的数据库分区的 Extender 数据，删除适当的 node*num* 子目录及其下面的所有子目录。（有关 DB2MMDATAPATH 的更多信息，参见第 476 页的『如何使用 DB2MMDATAPATH 环境变量（仅限于 EEE）』。）

停止和启动 Extender 服务器

停止使用 Extender 服务的应用程序时，在明确停止服务器之前，或在服务器再循环之前，该服务器仍是活动的。可通过在服务器上，从操作系统的命令行输入 DMBSTOP 命令停止所有 Extender 服务器。

要从客户机停止和重新启动 Extender 服务，从 db2ext 命令行运行 STOP SERVER 和 START SERVER 命令。这些命令停止和启动当前数据库的 Extender 服务。

仅限于 EEE: 在分区数据库环境中，可使用 DMBSTART 来启动对实例定义的所有数据库分区服务器，或只启动单个数据库分区服务器。不带任何参数的 DMBSTART 将启动所有数据库分区服务器。若仅想启动一个数据库分区服务器，在命令中指定要启动的节点即可，如下所示:

```
dmbstart nodenum 2
```

一旦在某个节点上启动了服务器，则必须将服务器重新连接至数据库。使用 Extender 命令 RECONNECT SERVER，如下所示:

```
reconnect server at nodenum 2
```

仅限于 EEE: 若正在分区数据库环境中使用 DB2 Extender，不带任何参数的 DMBSTOP 将停止对实例定义的所有数据库分区服务器。若只想停止一个数据库分区服务器，则必须先断开数据库与该服务器的连接。使用 Extender 命令 DISCONNECT SERVER，如下所示:

```
disconnect server at nodenum 2
```

然后可运行 DMBSTOP，并在命令中指定要停止的节点。以下示例显示为了在节点号 2 上停止 Extender 服务，将要从服务器的命令行中输入什么内容。

```
dmbstop nodenum 2
```

仅限于 EEE: 除非数据库以维护方式运行，否则不要在特定节点上运行 DMBSTOP。另外，您需要确保断开此节点时，将不会触发任何 Extender 活动。否则，可能会遇到意外的行为。

显示服务器状态

在服务器中，可用 DMBSTAT 命令显示 Extender 服务器状态。例如，以下命令列示已启用的数据库，以及 Extender 是否已启动且在运行。在运行此命令之前，连接服务器。

```
dmbstat
```

在客户机中，可使用 GET SERVER STATUS 命令获取数据库的 Extender 服务器的状态。例如，以下命令列示 personnl 数据库的状态:

```
get server status personnl
```

创建和管理多个服务器实例

可以创建和使用 DB2 Extender 服务器的多个实例。如果创建了 DB2 服务器的多个实例，则应创建 DB2 Extender 服务器的多个实例。DB2 Extender 服务器的每个实例都与 DB2 服务器的一个实例相关联，且具有相同的名称。您还可以列示系统上可用的 DB2 Extender 服务器实例、并行运行多个实例和除去实例。

创建多个 DB2 Extender 服务器实例

安装 DB2 Extender 时，将创建原始或缺省 DB2 Extender 实例，并且命名为与缺省 DB2 实例相同。在 Windows 上，缺省 DB2 Extender 实例名为 DB2。在 UNIX 上，缺省 DB2 Extender 实例与初始缺省 DB2 实例同名。要创建 DB2 Extender 服务器的附加实例，您必须具有 SYSADMIN 权限，并且，在 UNIX 上，您必须具有 root 用户权限。

使用 DMBICRT 命令来创建 DB2 Image Extender、Audio Extender 或 Video Extender 服务器的附加实例。如果要为 DB2 实例 DEVINST 创建 DB2 Extender 服务器实例，则在操作系统命令行上输入:

```
dmbicrt  devinst
```

当您执行 DMBICRT 命令时，将为该实例创建一个子目录，并将该实例添加至由 DB2 Extender 维护的实例列表。

仅限于 EEE:

- 在 Windows 中，缺省 DB2 Extender 服务器实例名为 DB2MPP。
- 当使用 DMBICRT 来创建 DB2 Image Extender、Audio Extender 或 Video Extender 服务器的附加实例时，您必须指定 DB2 Extender 用于分区数据库环境中的各种操作

的目录。这是 UNIX 中的 DB2MMDATAPATH 环境变量以及 Windows 中的注册表项中指定的目录。它必须是一个共享目录，且必须存在于该实例的所有节点上。

- 在 Windows 中，您还必须指定要使用的 TCP/IP 端口的范围；在 UNIX 中，必须将端口范围添加至 /etc/services 文件（参见第 9 页的『DMBICRT』）。

列示实例

使用 DMBILIST 命令来列示系统上可用的所有 DB2 Extender 服务器实例。要找出哪一个实例是活动的，请输入以下命令:

```
echo  %DB2INSTANCE%    （在 Windows 中）

echo $DB2INSTANCE      （在 UNIX 中）
```

并行运行多个实例

要并行运行 DB2 Extender 服务器的多个实例，请执行下列步骤:

在 Windows 中

从命令行:

1. 通过输入以下命令，将 DB2INSTANCE 变量设置为要启动的实例的名称:

   ```
   set db2instance=instanceName
   ```

2. 启动 Extender 服务。

在 UNIX 中

1. 作为实例所有者或具有该实例的系统管理权限的用户注册。
2. 设置环境。
3. 启动数据库管理器。

设置当前实例

当运行命令来对实例启动或停止服务时，这些命令将应用于当前实例。通过将 DB2INSTANCE 变量设置为实例名，您可以指定要使用 DB2 Extender 服务器的哪一个实例。

除去实例

要除去 DB2Extender 的实例，请执行下列步骤:

1. 停止所有当前正在使用该实例的应用程序。
2. 使用 DMBSTOP 和 db2ext TERMINATE 命令来停止 Extender 服务和所有 db2ext 命令行处理器会话。
3. 备份 DB2 Extender 实例目录中您要保存的文件，如 QBIC 目录文件。删除实例时，此目录中的文件将被除去。
4. 对要删除的实例输入 DMBIDROP 命令。例如，要删除 DEVINST 实例，输入:

   ```
   dmbidrop devinst
   ```

使用 DMBIDROP 命令除去 DB2 Extender 的实例并不会除去相关联的 DB2 实例。您必须单独地除去相关联的 DB2 实例。即使删除与 DB2 Extender 实例相关联的 DB2 实例，也不会除去 DB2 Extender 实例。然而，您不能使用它。

迁移实例

在 UNIX 系统上，在您安装新版本的 DB2 UDB 和 DB2 Extender 之后，应该迁移 DB2 Extender 实例。

要迁移用先前版本创建的现有 DB2 Extender 实例:

1. 迁移与 DB2 Extender 实例相关联的 DB2 UDB 实例。
2. 输入 DMBIMIGR 命令来迁移实例。例如，要迁移 OLDINST 实例，输入:

```
dmbimigr  oldinst
```

清除管理支持表

使用 DB2 Extender 时，管理支持表中最后可能会累积过时的条目。某人可能删除了媒体文件，但未删除数据库中对它的引用。删除过时的元数据可改进性能并收回存储空间。

使用 API: 图 1 中的样本代码清除 ANITAS 拥有的所有用户表的图像数据。它包括一些错误检查代码。完整的样本程序在客户机上 SAMPLES 子目录中的 API.C 文件中。

```
/*---- query database using DBiAdminReorgMetadata API ----*/
step="DBiAdminReorgMetadata API";
rc = DBiAdminReorgMetadata("anitas");
if (rc < 0) {
        printf("%s: %s FAILED!\n", argv[0], step);
        printMsg(rc);
        DBiGetError(&sqlcode, errorMsgText);
        printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
        fail = TRUE;
} else if (rc > 0) {
        printf("%s: %s, warning detected.\n", argv[0], step);
        printMsg(rc);
        DBiGetError(&sqlcode, errorMsgText);
        printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else
        printf("%s: %s PASSED\n\n", argv[0], step);

/*---- end of query using DBiAdminReorgMetadata API ----*/
```

图 1. 清除管理支持表的样本代码

使用 db2ext 命令行:

```
reorg database user anitas for db2image
```

若您不是 DBA，但具有 CONTROL 权限，可使用 DBxReorgMetadata API 或 REORG 命令来清除您拥有的表的元数据。

服务器的管理命令

本章中的命令是在服务器的操作系统的命令行上运行的。不要从 DB2 命令行或 db2ext 命令行运行它们。每当关机并重新启动服务器系统时，请运行 DMBSTART 命令。

清除元数据

仅限于 EEE: 还可在多分区数据库环境中发出 DMBSTART 和 DMBSTOP 服务器命令。当在多分区数据库环境中发出服务器命令时，除非包括节点号（在此情况下，该命令仅适用于指定的节点），否则，该命令适用于所有节点。

仅限于 EEE: 不能在多分区环境中运行 DMBSTAT 命令。可通过运行客户机命令 GET SERVER STATUS ALL 在多分区环境中检查服务器状态。

DMBICRT

图像	音频	视频
X	X	X

创建 DB2 Extender 实例。如果 DB2 有多个实例，则应创建多个 DB2 Extender 服务器实例。在 UNIX 上，您在安装 DB2 Extender 客户机时创建客户机实例；创建客户机实例将设置环境，以便使用 DB2 Extender。

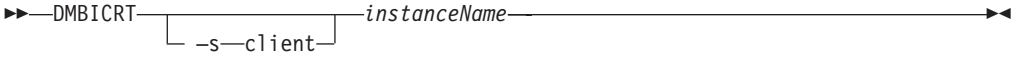
授权

SYSADM

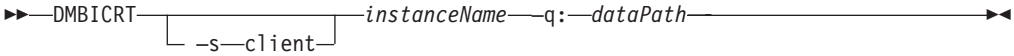
在 UNIX 上，您必须具有 root 用户权限。

命令语法

在非分区数据库环境中:



在 UNIX 中的分区数据库环境中:



在 Windows 中的分区数据库环境中:

命令参数

instanceName

现有 DB2 实例的名称。如果不存在具有此名称的 DB2 实例，则将询问您是否要创建它。

–s client

指定创建限于客户机的实例。当使用此参数时，*instanceName* 是客户机的用户标识。创建客户机实例将为客户机设置环境。（**仅限于 UNIX**）

dataPath

共享目录或文件系统的名称；该目录必须存在于所有节点上。在 UNIX 中，这是在 DB2MMDATAPATH 环境变量中设置的，在 Windows 中，这是在注册表中设置的。（**仅限于 EEE**）

start_port，**end_port**

要使用的 TCP/IP 端口的范围。端口范围必须等于或大于您正在使用的节点数。端口号被写至 Windows 注册表。（**仅限于 Windows EEE**）

例

在非分区数据库环境中为 DB2 实例 DEVINST 创建 DB2 Extender 服务器的实例:

```
dmbicrt  devinst
```

用法注释

DMBICRT 命令为实例所用的文件创建 DB2 Extender 实例目录。此目录名为:

- *install_directory*\INSTANCE*instance_name*，其中，*install_directory* 是安装了 DB2 Extender 的目录（**Windows 和 OS/2**）

- *INSTHOME*/dmb，其中 *INSTHOME* 是实例所有者的主目录（**UNIX**）

如果您使用 DMBICRT 命令时不存在具有指定名称的 DB2 实例，则将提示您创建它。

仅限于 EEE:

虽然您可以用任何参与节点的 root 用户标识运行 DMBICRT，但建议您使用同一节点来创建所有 DB2 Extender 服务器实例。该节点应该就是用来创建 DB2 实例以及 DB2 实例目录驻留所在的节点。如果使用另一节点来创建 DB2 Extender 服务器实例，则存储在任何节点上的实例列表都可能不完整。

在 UNIX 中，指定为 *dataPath* 的共享目录或文件系统作为 DB2MMDATAPATH 环境变量的值保存在 *$INSTHOME*/dmb/dmbprofile 中，在 Windows 中，保存在以下注册表键中:

```
\\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2 Extenders\PROFILES
      \instance_name\DB2MMDATAPATH
```

在 UNIX 上，必须在创建实例之前将端口范围添加至 /etc/services 文件。使用以下语法将两条目添加至该文件:

– DMB_*instance_name start_port*

– DMB_*instance_name*_END *end_port*

对于分区数据库环境中的所有节点，此范围必须足够大。

必须在创建 DB2 Extender 服务器实例之前创建 DB2 实例。

要在分区数据库环境中创建 Text Extender 实例，请使用 TXTICRT 命令，如《*DB2 通用数据库 Text Extender 管理与编程*》中所述。

DMBIDROP

图像	音频	视频
X	X	X

删除 DB2 Extender 实例。

授权

SYSADM

在 UNIX 上，您必须具有 root 用户权限。

命令语法

命令参数

instanceName

您想要删除的 DB2 Extender 实例的名称。

例

删除名为 DEVINST 的 DB2 Extender 服务器实例:

```
dmbidrop  devinst
```

用法注释

在运行此命令之前:

- 停止所有使用该实例的应用程序，并停止所有 db2ext 命令行处理器。
- 停止 Extender 服务。

DMBIDROP 命令除去 DB2 Extender 实例目录，从实例列表中除去该实例条目，并除去关于该实例的其它信息。

DMBIDROP 命令只除去 DB2 Extender 实例，它不除去与其相关联的 DB2 实例。必须显式地删除 DB2 实例。

即使删除与 DB2 Extender 服务器实例相关联的 DB2 实例，也不会删除 DB2 Extender 服务器实例。然而，您不能使用它。

（**仅限于 EEE**）如果删除 DB2 Extender 实例，在 UNIX 中，必须从 /etc/services 文件中除去该实例的起始和结束端口条目，在 Windows 中，必须从 \WINNT\system32\drivers\etc\Services 文件中除去这些条目。这些条目是 DMB_*instanceName*mpp 和 DMB_*instanceName*mpp_END。

DMBILIST

图像	音频	视频
X	X	X

列示 DB2 Extender 的所有实例。

授权

无。

命令语法

命令参数

无。

例

列示 DB2 Extender 实例:

```
dmbilist
```

DMBIMIGR

图像	音频	视频
X	X	X

（仅限于 **UNIX**）将 DB2 Extender 实例从先前发行版迁移至当前发行版。

授权

您必须具有 root 用户权限。

命令语法

命令参数

instanceName

您想要迁移的 DB2 Extender 实例的名称。

例

迁移名为 OLDINST 的 DB2 Extender 实例:

```
dmbimigr  oldinst
```

用法注释

在运行此命令之前:

- 必须已安装 DB2 Extender 的当前发行版。
- 必须迁移相关联的 DB2 实例。

对每个 DB2 Extender 实例运行一次 DMBIMIGR。使用 DMBILIST 来列示实例。

DMBSTART

图像	音频	视频
X	X	X

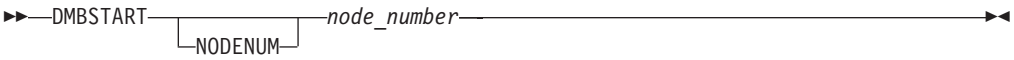
对 Extender 实例启动所有 Extender 服务。

仅限于 EEE: 若指定了节点，则该命令仅在该节点处启动 Extender 服务。DMBSTART 还在每个节点上启动“节点创建 / 删除”功能。“节点创建 / 删除”功能进行检查，以确保对 DB2 定义的节点与对 Extender 定义的节点相匹配。若不匹配，则“节点创建 / 删除”功能按要求添加或除去节点。

授权

SYSADM

命令语法



命令参数

node_number

要在该处启动 Extender 服务的节点。（**仅限于 EEE**）

例

启动 Extender 服务:

```
dmbstart
```

在节点号 2 上启动 Extender 服务:

```
dmbstart nodenum 2
```

用法注释

运行此命令:

- 当作为实例所有者登录时（在 AIX、HP-UX 或 Solaris 上）。
- 在 DB2INSTANCE 环境变量与要启动的实例相同的窗口中（在 Windows NT 上）。
- 每当关机并重新启动服务器系统时。

在单分区环境中，如果 DB2 实例没有在运行，DMBSTART 将启动它。

仅限于 EEE:

在多分区环境中，DMBSTART 不启动 DB2 实例。在分区环境中运行 DMBSTART 之前，必须启动 DB2。

如果 DMBSTART 失败，则进行下列检查:

– 确保 DBD2MMDATAPATH 变量的值正确。

– 确保该变量中的共享目录或文件系统存在，且在每个节点上都可存取。

DMBSTAT

图像	音频	视频
X	X	X

显示已启用的数据库，以及那些数据库的 Extender 服务是否已启动且在运行。

授权

无

命令语法

命令参数

无。

例

显示 Extender 服务的状态:

```
dmbstat
```

DMBSTOP

图像	音频	视频
X	X	X

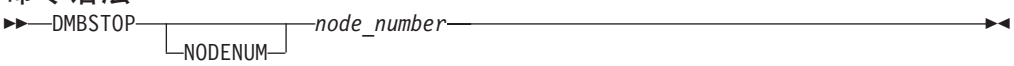
停止 Extender 实例的 Extender 服务。

仅限于 EEE: 若指定节点，则 DMBSTOP 仅在该节点上停止 Extender 服务。

授权

SYSADM

命令语法



命令参数

node_number

要在该处停止 Extender 服务的节点。（**仅限于 EEE**）

例

停止 Extender 服务:

```
dmbstop
```

在节点号 2 上停止 Extender 服务:

```
dmbstop nodenum 2
```

用法注释

运行此命令:

- 当作为实例所有者登录时（在 AIX、HP-UX 或 Solaris 上）。
- 在 DB2INSTANCE 环境变量与要停止的实例相同的窗口中（在 Windows NT 上）。

DMBSTOP 不停止 DB2 实例。

仅限于 EEE: 除非正以维护方式操作，否则不要在特定节点上运行 DMBSTOP。另外，您需要确保断开此节点时，将不会触发任何 Extender 活动。否则，可能会遇到意外的行为。

在分区数据库系统中再分发 Extender 数据（仅限于 EEE）

“DB2 扩充企业版”允许添加和删除分区数据库环境中的数据库分区服务器（也称为节点）。在添加这些节点之后（或删除这些节点之前），可再分发现有数据以利用新的配置。

您必须执行两个步骤来再分发 Extender 数据。首先，必须再分发 DB2 数据。然后，可再分发 DB2 Extender 数据。

再分发 DB2 数据

在再分发数据之前，必须使用 DB2 命令 REDISTRIBUTE NODEGROUP 来再分发 DB2 数据。

有关再分发 DB2 数据的更多信息，*参考《DB2 管理指南》*。

再分发 Extender 数据

再分发 DB2 数据之后，就已准备好再分发 Extender 数据了。输入 Extender 命令 REDISTRIBUTE NODEGROUP 来启动 Extender 数据再分发。

```
redistribute nodegroup
```

REDISTRIBUTE NODEGROUP 命令再分发音频、图像和视频 Extender 数据，以及 QBIC 特征数据，并将其放在相应的用户数据所在的节点上。

若再分发过程返回一个错误，则可重新运行该命令。根据命令响应提供的指示信息，可以重新运行该命令（带或不带 CONTINUE 参数）。此选项指导系统从停止的地方继续，而不是从头开始。在 DB2 的 REDISTRIBUTE NODEGROUP 之后首次运行 REDISTRIBUTE NODEGROUP 命令时，不能使用 CONTINUE 参数。

要维护数据完整性，一次再分发一个节点组。在启动另一再分发之前，等待一个节点组完成再分发。

在使用此命令之前，必须连接数据库。

您需要 SYSADM 或 DBADM 权限才可运行此命令。

检测视频画面切换

本章描述如何用 DB2 Video Extender API 来检测视频剪辑中的画面切换。这些 API 在除 Windows 3.1 之外的所有 DB2 Video Extender 平台中都可用。仅支持 MPEG-1 格式的视频剪辑的视频画面切换检测。

什么是视频画面切换?

想象有一间电视工作室，它将节目录制在录象带上以供随后播放。最近，该工作室已开始使用 Video Extender 来将其视频磁带剪辑存储在 DB2 数据库中。这使工作室工作人员有机会查询传统类型的关于他们的节目的信息，以及查看节目的剪辑。

工作室将会喜欢预览视频剪辑这一选项。他们希望可以查看称为故事板的可视摘要。第 18 页的图 2 中显示了故事板的一个示例。查看故事板可帮助工作室雇员了解录象的要点，而不必查看整个视频。它还可帮助雇员确定该视频是否就是他们需要的视频

（例如，是否值得下载和查看）。对于工作室来说，此需求非常重要。查看故事板（而不是整个视频）可大大缩短下载和查看时间。有关按这种方式使用视频画面切换检测能力的更多信息，参见第 29 页的『存储关于视频中所有镜头的信息』。

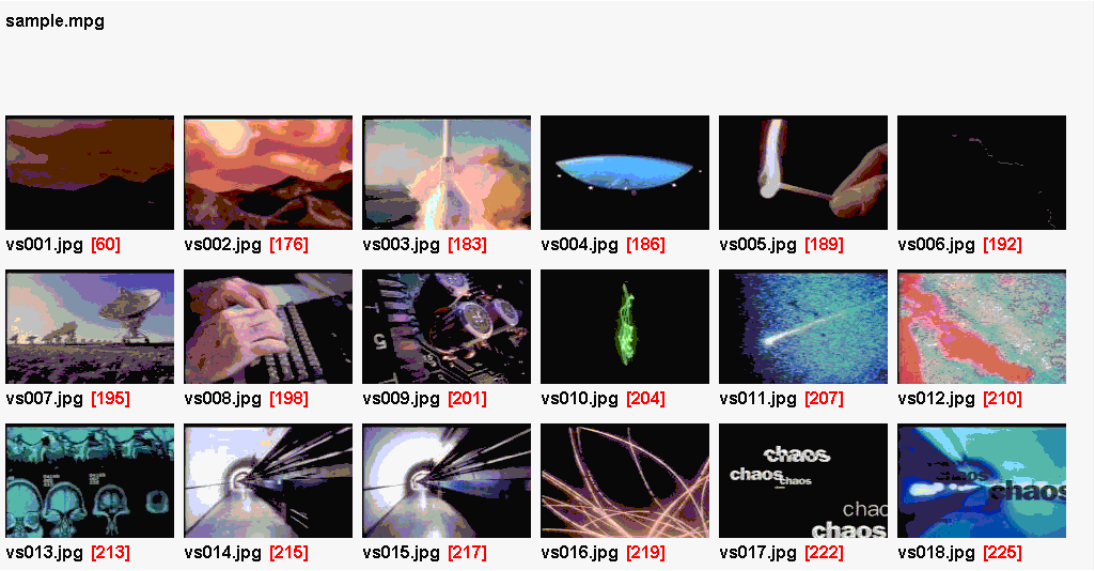


图 2. 视频故事板. 代表性帧概述了视频的内容和流动。

工作室计划使用 Video extender 的视频画面切换检测能力来为故事板捕捉代表性帧。

视频画面切换是视频剪辑中的一点，在该处，两个连续帧之间有显著的差异。例如，如果摄影机在录像时更改其视点，就会发生画面切换。两次画面切换之间的帧构成一个**镜头**。

当 Video Extender 检测到画面切换时[1]，它记录相关联的镜头的数据。该数据包括镜头的起始帧号、镜头的结束帧号，以及镜头中代表性帧的号码。镜头数据还包括代表性帧的像素内容。

查找和使用画面切换

Video Extender 提供了一组可用来查找视频剪辑中的镜头或帧的应用程序编程接口。在找到镜头或帧之后，可访问关于它的数据，如镜头的开始和结束帧号，或帧的像素内容。然后，可将此信息传送给程序，供它进行进一步的处理。例如，可将帧的内容传送至能够显示它的程序。

Video Extender 还提供了将镜头数据存储在**镜头目录**中的 API。镜头目录可以放在数据库或文件中。可以存取文件中的镜头目录，也可以存取数据库中镜头目录的只读视图。

镜头目录文件包含用于下列数据的字段:

- 镜头目录名
- 控制 Video Extender 检测镜头的方式的值，例如，镜头中的最小帧数

1. 视频画面切换检测码包括由美国加州大学伯克利分校开发的 MPEG 译码器，并经过波士顿大学的多媒体通信实验室（Multimedia Communication Laboratory）的修改。

- 控制将多少帧以及哪些帧存储为镜头的代表性帧的值
- 镜头号
- 起始帧号
- 结束帧号
- 代表性帧的号码
- 包含代表性帧的内容的文件的名称

数据库中镜头目录的视图包含用于下列数据的列:

- 镜头句柄
- 视频表名
- 视频列名
- 视频句柄
- 视频文件名
- 起始帧号
- 结束帧号
- 代表性帧的号码
- 代表性帧的数据
- 注释

可存取镜头目录文件中的数据，如果镜头目录在数据库中，则可查询数据。在显示故事板时，代表性帧信息特别有用。另外，若镜头目录在数据库中，则还可以将镜头数据与其它表中的相关数据相汇合。例如，视频工作室中的人员可在数据库中创建镜头目录 他们可将目录数据与包含视频剪辑的表，以及关于剪辑的信息汇合在一起。通过此方法，他们将能够使用单个查询来检索剪辑及关于该剪辑的商业信息，以及标识剪辑中的镜头。

镜头检测数据结构

与镜头检测相关的数据存储在镜头检测头文件 dmbshot.h 中包括的那些结构中。许多镜头检测 API 都要求您指向其中一个或多个结构。其中一些结构用来包含 Video Extender 用作输入的数据。例如，镜头控制结构包含控制镜头检测的信息。Video Extender 使用大多数结构来存储它从视频中检索到的数据。例如，视频帧数据结构包含帧的像素内容。

用于镜头检测的结构有 DBvIOType、DBvShotControl、DBvShotType、DBvFrameData 和 DBvStoryboardCtrl。

DBvIOType: DBvIOType 数据结构包含关于视频的数据，诸如它的格式、尺寸和帧的数目。此数据结构定义如下:

```
typedef struct {

   FILE *hFile;                    /* file handle for the video */
   char vhandle[255];              /* video handle (if from database)
   char vtable[255];               /* video table name (if from database) */
   char vcolumn[255];              /* video column name (if from database) */
   char vFile[255];                /* name of video file */
   char idxFile[255];              /* name of index file */
   char isIdx;                     /* 1 if the index exists, 0 otherwise */
   char isInDb;                    /* 1 if from DB, 0 if from file */
   int format;                     /* Format of the video */
```

```
   unsigned long dx, dy;           /* Dimensions of the video */
   unsigned long totalFrames;      /* TotalFrames in the video */
   unsigned long markFrame;        /* used by shot detection */
   unsigned long currentFrame;     /* The current video frame */
   DBvFrameData fd;                /* Frame data for current frame */
   DBvDCFrameData fdDc;            /* Frame data for DC images */
   unsigned char BGRValid;         /* reserved */
   unsigned short usDeviceID;      /* reserved */
   unsigned long hwnd;             /* reserved */
   int videoReset;                 /* Flag if video is opened or seeked */
   int firstshot;                  /* Used internally to indicate the first call */
   void *reserved                  /* reserved */

} DBvIOType;
```

DBvShotControl: DBvShotControl 数据结构包含用来控制镜头检测的信息，如检测方法。此数据结构定义如下:

```
typedef struct {

  unsigned long reserved;
  unsigned long method;           /* detection method */

    #define DETECT_CORRELATION  0x00000001
    #define DETECT_HISTOGRAM    0x00000002
    #define DETECT_CORRHIST     0x00000003
    #define DETECT_CORRHISTDISS 0x00000004


  int normalCorrValue;            /* Correlation threshold */
  int sceneCutSkipXY;             /* reserved */
  int CorrHistThresh;             /* Histogram threshold */
  int DissThresh;                 /* Dissolve threshold */
  int DissCacheSize;              /* Dissolve cache size */
  int DissNumCaches;              /* Dissolve cache number */
  int minShotSize;                /* Minimum frames in a shot */

} DBvShotControl;
```

表 1 描述了 DBvShotControl 中的每个字段及其允许的设置和缺省设置。要将这些字段初始化为缺省值，按照第 22 页的『初始化镜头检测数据结构中的值』中的描述来使用 DBvInitShotControl API。

DBvShotControl 设置取决于视频的类型: 数字化视频中的画面切换随视频的内容和格式的不同而有很大变化。另外，画面切换算法的准确性随视频的不同而有所变化。与更细微类型的更改，或整体色彩内容保持不变的更改相比，对整体帧外观有明显差异且清晰定义的画面切换的检测更准确。虽然对于大多数应用程序，缺省 DBvShotControl 字段设置就已足够，但您可能需要调整这些设置以减少出错或未能检测的情况。

表 1. DBvShotControl 字段

字段	含义
方法	标识 Video Extender 用来检测画面切换的方法。可选择下列其中一种方法: DETECT_CORRELATION。比较两个连续帧中的像素数。若差异超过相关阈值，将检测画面切换。 DETECT_HISTOGRAM。比较两个连续帧的直方图值。直方图值测量一个帧中色彩的分配。若差异超过直方图阈值，将检测画面切换。 DETECT_CORRHIST。使用相关方法来标识可能的画面切换，然后对标记为可能的画面切换的帧使用直方图方法。若超过直方图阈值，将检测画面切换。 DETECT_CORRHISTDISS。与 DETECT_CORRHIST 相同，但检查附加帧的渐稳。 缺省方法是 DETECT_CORRHIST。
normalCorrValue	0 至 100 的整数值，指定相关阈值。这将给出两个帧中的像素之间相关系数的最小值。值 0 表示总是检测下一帧的画面切换。值 100 表示仅当两个帧的所有像素均不相同时，才检测画面切换。缺省值是 60。
sceneCutSkipXY	保留。
CorrHistThresh	0 至 100 的整数值，指定直方图阈值。这将测量连续帧的直方图值之间的差异。值 0 表示仅当两个帧的直方图值完全不同时，才检测画面切换。值 100 表示总是检测下一帧的画面切换。缺省值是 10。
DissThresh	0 至 100 的整数值，指定渐稳测试阈值 这将测量一个帧中必须通过渐稳测试才会检测到渐稳的像素的百分比。值 0 表示总是检测帧的渐稳。值 100 表示仅当帧中的所有像素都通过渐稳测试时，才会检测到渐稳。缺省值是 15。
DissCacheSize	一个整数值，指定渐稳测试的倾斜部分中使用的帧数。缺省值是 4。
DissNumCaches	一个整数值，指定渐稳测试的一致性部分中使用的帧数。缺省值是 7。
minShotSize	一个整数值，指定镜头的最小帧数。为了要检测到某个镜头，它必须最少要有与最小数一样多的帧。缺省值是 5。

DBvShotType: DBvShotType 数据结构包含关于镜头的信息，诸如起始帧号、结束帧号和代表性帧的号码；以及指向代表性帧的像素内容的指针。此数据结构定义如下:

```
typedef struct {

  unsigned long startFrame;       /* starting frame number */
  unsigned long endFrame;         /* ending frame number */
  unsigned long repFrame;         /* representative frame number */
  DBvFrameData fd;                /* data for representative shot */
  unsigned long dx;               /* frame data width in pixels */
  unsigned long dy;               /* frame data height in pixels */
  char *comment;                  /* shot remark */

} DBvShotType;
```

DBvFrameData: DBvFrameData 数据结构包含帧的像素内容。此数据结构定义如下:

```
typedef struct                     /* video frame data */
{
  /* MPEG 1 pixels */
  unsigned char *luminance;        /* Luminance pixel plane (black and white) */
  unsigned char *Cr;               /* Cr pixel plane */
```

```
  unsigned char *Cb;               /* Cb pixel plane */
  unsigned char *reserved;

  } DBvFrameData;
```

DBvStoryboardCtrl: DBvStoryboardCtrl 数据结构包含一些值，用于控制将镜头中哪些以及多少代表性帧存储在视频目录中。参见第 30 页的『构建故事板』以获取如何使用这些值的描述。此数据结构定义如下:

```
typedef struct {

  int thresh1;                    /* threshold for small to medium scenes */
  int thresh2;                    /* threshold for medium to large scenes */
  int delta;                      /* offset used for representative frames */

} DBvStoryboardCtrl;
```

表 2 描述了 DBvStoryboardCtrl 中的每个字段及其缺省设置。要将这些字段初始化为缺省值，按照『初始化镜头检测数据结构中的值』中的描述来使用 DBvInitStoryboardCtrl API。

DBvStoryboardCtrl 设置取决于视频的类型: 对于不同类型的视频，哪些以及多少代表性帧对于故事板来说是优化的可能有所会有多不同。虽然对于许多类型的视频来说，缺省的 DBvStoryboardCtrl 类型设置工作得很好，但您可能想使用这些设置来测试视频子集。然后，对这些设置进行适当调整，再为更大范围的一组视频构建故事板。

表 2. DBvStoryboardCtrl 字段

字段	含义
thresh1	标识短镜头的阈值。所含帧数小于 thresh1 的值的镜头是短镜头。若已编目，则短镜头的信息将包括一个代表性帧（中间的帧）。 缺省值是 90。若将 thresh1 的值设置为 -1，则将认为某个镜头是短镜头（不管它的实际长度是多少）。
thresh2	标识中镜头到大镜头的阈值。所含帧数等于或小于 thresh2 的值，但至少达到 thresh1 的值的镜头认为是中镜头。若已编目，则中镜头的信息将包括两个代表性帧。代表性帧的位置由 delta 字段的值控制。将所含帧数大于 thresh2 的值的镜头认为是长镜头。若已编目，则长镜头的信息将包括三个代表性帧。第一个和最后一个代表性帧的位置由 delta 字段的值控制。第二个帧是中间的帧。 缺省值是 150。若将 thresh2 的值设置为 -1，则将认为某个镜头是短镜头（不管它的实际长度是多少）。
delta	标识用于代表性帧的偏移。对于中等帧和长帧，第一个代表性帧位于从镜头的开始算起，经过 delta 中帧数的偏移处。最后一个代表性帧位于从帧的末尾算起，经过 delta 中的帧数的偏移处。 缺省值是 5。

初始化镜头检测数据结构中的值: DBvShotControl 数据结构中的值控制镜头检测。DBvStoryboardCtrl 数据结构中的值控制故事板的构建。可明确地为这些数据结构中的字段指定值。另外，还可以将这些结构中的值初始化为缺省值。参见第 21 页的表 1 以获取 DBvShotControl 数据结构中的缺省值。参见表 2 以获取 DBvStoryboardCtrl 数据结构中的缺省值。

使用 DBvInitShotControl API 来初始化 DBvShotControl 数据结构中的值。在使用此 API 时，需要指定镜头控制结构。例如，下列语句将 DBvShotControl 结构中的字段初始化为缺省值:

```
DBvShotControl    shotCtrl;

rc=DBvInitShotControl(
              shotCtrl);          /* pointer to shot control structure */
```

使用 DBvInitStoryboardCtrl API 来初始化 DBvStoryboardCtrl 数据结构中的值。在使用此 API 时，需要指定故事板控制结构。例如，下列语句将 DBvStoryboardCtrl 结构中的字段初始化为缺省值:

```
DBvStoryboardCtrl    sbCtrl;

rc=DBvInitStoryboardCtrl(
              sbCtrl);          /* pointer to storyboard control structure */
```

获取镜头或帧

可使用 Video Extender 从视频中获取镜头或帧。在获取镜头或帧之前，必须打开视频进行镜头检测。Video Extender 使用索引来存取帧和镜头。在获取镜头或帧之前，必须创建视频的索引。

在打开视频并创建索引之后，就可获取视频中的下一镜头或帧，或通过帧号获取特定帧。Video Extender 可处理 MPEG-1 格式的视频剪辑。如果计划将检索到的帧与要求 RGB 格式的程序配合使用，则可以通过使用 Video Extender API 将帧转换为该格式。

打开视频进行镜头检测: 使用 DBvOpenFile API 来打开存储在文件中的视频。该文件必须可以从客户机进行存取。使用 DBvOpenHandle API 来打开存储在数据库表中的视频。该应用程序必须首先连接到数据库。若视频存储在数据库表中，则 Video Extender 将视频复制至一个临时文件。该临时文件位于客户机环境变量 DB2VIDEOTEMP 中指定的目录中。打开视频将对它初始化，以进行镜头检测。Video Extender 将设置一个指向视频起始处（即帧 0）的指针。

在使用任一 API 时，都需要指向用来包含指向视频数据结构（DBvIOType）的指针的区域。作为对 API 调用的响应，Video Extender 分配此结构，并使用此结构来存储关于视频的信息。此结构还指向包含当前帧的像素内容的帧数据结构（DBvFrameData）。有关这些结构的描述，参见第 19 页的『镜头检测数据结构』。对于 DBvOpenFile API，还需要指定视频文件的名称。对于 DBvOpenHandle API，还需要指定视频句柄。

例如，下列语句打开存储在文件中的视频进行镜头检测:

```
DBvIOType    *videoptr;

rc=DBvOpenFile (
       &videoptr,                        /* pointer to video structure pointer */
       "/employee/video/rsmith.mpg");    /* video file */
```

下列语句打开存储在数据库表中的视频进行镜头检测:

```
EXEC SQL BEGIN DECLARE SECTION;
char Vid_hdl[251];
EXEC SQL END DECLARE SECTION;

DBvIOType    *videoptr;

EXEC SQL SELECT VIDEO INTO :Vid_hdl
FROM EMPLOYEE
WHERE NAME="Anita Jones";
```

```
rc=DBvOpenHandle(
       &videoptr,                        /* pointer to video structure pointer */
       vid_hdl);                         /* video handle*/
```

创建视频的索引: Video Extender 使用索引来存取视频中的帧和镜头。需要创建视频的索引（MPEG 格式未提供帧和镜头的索引），才能从视频中获取镜头或帧。索引将帧号映射至组成 MPEG-1 视频的位流。

可通过使用 DBvCreateIndexFromVideo API 或 DBvCreateIndex API 来创建视频的索引。但是，若已使用 DBvOpenFile API 或 DBvOpenHandle API 打开视频进行镜头检测，则不必明确创建索引；Video Extender 将自动地为您创建索引。（参见第 23 页的『打开视频进行镜头检测』以获取关于如何打开视频的信息）。

当（明确或自动地）创建索引时，DB2 Video Extender 尝试将索引存储在视频文件所在的路径中。它首先尝试将索引文件存储为 fname.ext.idx，其中 fname 是视频文件的名称，而 ext 是视频文件的扩展名。若该尝试失败，则 Video Extender尝试将该文件存储为 fname.idx，并存储在视频文件所在的目录中。若该尝试失败，则它尝试将索引文件存储在本地目录中，首先存储为 fname.ext.idx，然后存储为 fname.idx。

当文件打开时，Video Extender 按以下次序寻找索引文件:

1. 索引文件的可写版本，优先于只读版本。
2. 与视频文件在同一路径中的索引文件，优先于当前目录。
3. 名为 fname.ext.idx 的索引，优先于名为 name fname.idx 的索引，其中 fname 是视频文件的名称，而 ext 是视频文件的扩展名。

例如，若为名为 myvideo.mpg 的视频文件创建了索引，则 Video Extender 将首先在视频文件的路径中寻找名为 myvideo.mpg.idx 的可写索引。

使用 DBvCreateIndexFromVideo API 时，指定 DBvIOType 数据结构。Video Extender 将索引文件的名称存储在该结构中。有关此结构的描述，参见第 19 页的『镜头检测数据结构』。例如，下列语句为先前已打开进行镜头检测的视频创建索引:

```
DBvIOType    *video;

rc=DBvCreateIndexFromVideo(
       video);                  /* pointer to video structure */
```

当使用 DBvCreateIndex API 时，指定视频文件的名称。Video Extender 将索引存储在文件中（与视频位于同一目录）。例如，下列语句创建视频文件（先前未打开进行镜头检测的文件）的索引:

```
rc=DBvCreateIndex(
       "/Employee/video/rsmith.mpg");    /* video file */
```

还可以确定是否存在视频的索引。使用 DBvIsIndex API 来检查索引。若不存在视频的索引，则此 API 将状态变量设置为 0，若存在视频的索引，则设置为 1。例如，下列语句检查视频文件的索引是否存在。

```
short  *status

rc=DBvIsIndex(
       "/Employee/video/rsmith.mpg",     /* video file */
       &status);                         /* status indicator */
```

备份视频索引: 备份视频索引文件，以防需要恢复它。该文件位于安装 Video Extender 的目录中。

获取帧: 可获取视频中的当前帧。还可将当前帧设置为特定帧号。使用 DBvGetFrame API 来获取视频中的当前帧。使用 DBvSetFrameNumber API 来将当前帧设置为特定帧号。

当使用 DBvGetFrame API 时，指定视频结构例如，下列语句获取视频中的当前帧:

```
DBvIOType      *video;

rc=DBvGetFrame(
       video);                 /* pointer to video structure */
```

使用 DBvSetFrameNumber API 时，指定视频结构和要设置成当前帧的帧的号码。例如，下列语句将当前帧设置为帧号 85，然后获取该帧:

```
DBvIOType      *video;

rc=DBvSetFrameNumber(
       video,                  /* pointer to video structure */
       85);                    /* frame number */

rc=DBvGetFrame(
       video);                 /* pointer to video structure */
```

输出时，DBvSetFrameNumber API 将复位 DBvIOType 结构中的 currentFrame 字段。DBvGetFrame API 将帧的像素内容放入 DBvFrameData 结构中。有关这些结构的描述，参见第 19 页的『镜头检测数据结构』。

获取镜头: 使用 DBvDetectShot API 来获取视频中的下一镜头。使用 DBvDetectShot API 时，需要指向下列数据结构:

- 视频（DBvIOType）
- 镜头控制（DBvShotControl）
- 镜头类型（DBvShotType）

还需要指向搜索的起始帧。Video Extender 将从视频中的该点开始搜索下一镜头。

作为此 API 的结果，Video Extender 设置 shotDetected 标志，并指向下一镜头的起始帧及其帧数据。若将 shotDetected 标志设置为 1，则表示已检测到镜头。在这种情况下，Video Extender:

- 将 DBvIOType 中的 currentFrame 字段设置为下一镜头的起始帧
- 将下一镜头的起始帧的数据放入 DBvIOType 中的 fd 字段中
- 设置 DBvShotType 以包含起始帧号、结束帧号、代表性帧的号码、代表性帧数据和下一镜头的注释

若将 shotDetected 标志设置为 0，则表示未检测到镜头。在这种情况下，Video Extender 返回一代码，指示到达视频的末尾。

有关这些结构的描述，参见第 19 页的『镜头检测数据结构』。

例如，下列语句请求视频中的下一镜头:

```
DBvIOType      *video;
long start_frame = 1;
char shotDetected = 0;
```

```
DBvShotControl    shotCtrl;
DBvShotType    shot;

shotCtrl→method=DETECT_CORRHIST
shotCtrl→normalCorrValue=60;
shotCtrl→sceneCutSkipXY=1;
shotCtrl→CorrHistThresh=10;
shotCtrl→DissThresh=10;
shotCtrl→DissCacheSize=4;
shotCtrl→DissNumCaches=7;
shotCtrl→minShotSize=0;

rc=DBvDetectShot(
       video,              /* pointer to video structure */
       start_frame,        /* starting frame for search */
       &shotDetected,      /* shot detected flag */
                           /* 1=detected, 0=not detected */
       shotCtrl,           /* pointer to shot control structure */
       &shot);             /* pointer to shot type structure */
```

转换检索到的帧的格式: MPEG-1 帧的内容使用 YUV 格式，该格式包括关于帧的亮度像素平面、Cr 像素平面和 Cb 像素平面的信息。 若要编辑视频帧，您可能会发现将帧由 YUV 格式转换为 RGB 格式很方便。Video Extender 提供了 DBvFrameDatato24BitRGB API 来将检索到的 MPEG-1 帧由 YUV 格式转换为 24 位 RGB 格式。要使用此 API，首先必须分配目标缓冲区。

当发出此 API 时，需要指向目标缓冲区和要转换的帧数据。还需要指定帧的高度和宽度。（可从帧的 DBvIOType 结构中获取帧的数据、高度和宽度。）例如，下列语句将 MPEG-1 帧转换为 24 位 RGB 格式:

```
char RGB[18000];
DBvIOType      *video;
DBvFrameData   fd;

rc=DBvGetNextFrame(
       video);                 /* pointer to video structure */

fd=video.fd
dx=video.dx
dy=video.dy

rc=DBvFrameDataTo24BitRGB (
       RGB,                     /* pointer to target buffer */
       &fd,                     /* pointer to frame data */
       dx,                      /* frame width */
       dy);                     /* frame height */
```

关闭视频文件: 使用 DBvClose API 来关闭已打开进行镜头检测的视频文件。当使用此 API 时，指定指向文件的视频结构的指针。

例如，下列语句关闭已打开进行镜头检测的视频文件:

```
DBvIOType    *video;

rc=DBvClose (video);
```

显示检索到的帧: 检索到的 MPEG-1 帧的内容使用 YUV 格式；这不是大多数图像显示程序可显示的格式。要显示检索到的视频帧，需要将其置于图像显示程序可以理解的格式，如 BMP 格式。例如，要显示 MPEG-1 帧:

1. 使用 DBvFrameDatato24BitRGB API 来将检索到的 MPEG-1 帧的格式由 YUV 格式转换为 24 位 RGB 格式。参见第 26 页的『转换检索到的帧的格式』以获取关于使用 DBvFrameDatato24BitRGB API 的信息。
2. 将适当的标题追加至转换后的帧。例如，BMP 格式需要一个包括诸如图像宽度和高度之类的信息的标题。
3. 将帧内容（及其标题）复制至文件。
4. 使用 DBiBrowse API 来显示该文件。有关使用 DBiBrowse API 的信息，参见第 121 页的『使用显示或播放 API』。

编目镜头

可将关于镜头的信息存储在镜头目录中。Video Extender 提供了一些 API 来:

- 创建并管理数据库中的镜头目录。可使用这些 API 来:
  - 在数据库中创建镜头目录
  - 在镜头目录中存储关于单个镜头的信息
  - 在镜头目录中存储关于视频中所有镜头的信息
  - 更改镜头目录中存储的镜头信息
  - 合并镜头目录中的镜头信息
  - 从镜头目录中删除镜头信息
  - 从数据库中删除镜头目录
- 创建镜头目录文件，并在其中存储关于视频中的所有镜头的信息。提供了一个 API，用于创建目录文件并用镜头数据填充它。可存取并处理镜头目录文件中的数据，但未提供 API 来执行该操作。

已编目的镜头提供故事板的输入：在镜头目录中存储镜头信息之后（无论该目录是在数据库中还是在文件中），可在与镜头相关的应用程序中使用该信息。例如，可创建一个应用程序，它获取视频中所有镜头的代表性帧，并在故事板中加以显示。

只需创建数据库的镜头目录：仅当您想让目录驻留在数据库中时，才需要创建镜头目录。当您存储视频中镜头的数据，并指示要在文件中进行输出时，Video Extender 自动创建镜头目录文件。

在创建目录之前（仅数据库）： 在数据库中创建和使用目录之前，必须:

- 发出 SQLConnect 调用。DB2 数据库中的镜头目录由一组表组成。必须先使用 SQLConnect 调用连接到数据库，再在数据库中创建镜头目录并对其执行操作。（SQLConnect 是一种 “DB2 调用层接口” 调用。）此调用返回一个连接句柄，您需要在管理镜头目录的 API 中指定它。
- 启动图像数据的数据库。在数据库中创建镜头目录之前，必须启用 DB2Image 数据类型的数据库。除了存储在镜头目录中的其它信息之外，Video Extender 还存储已编目的每一镜头的代表性帧数据。代表性帧数据的数据类型是 DB2Image。

创建镜头目录（仅数据库）： 使用 DBvCreateShotCatalog API 在数据库中创建镜头目录。（当存储镜头的数据，并指示要在文件中进行输出时，Video Extender 将自动创建镜头目录文件）。该目录由存储与镜头相关的信息的多个表组成。可通过使用 SQL 来查询表的视图。第 28 页的表 3 显示了视图中的各列。

表 3. 镜头目录视图中的列

列名	数据类型	描述
SHOTHANDLE	CHAR(36)	镜头句柄
VIDEOHANDLE	VARCHAR(254)	视频句柄。仅当视频是用 DBvOpenHandle API 打开的时，此列才包含值。
VIDEOTABLE	VARCHAR(254)	包含视频的表。仅当视频是用 DBvOpenHandle API 打开的时，此列才包含值。
VIDEOCOLUMN	VARCHAR(254)	包含视频的表列。仅当视频是用 DBvOpenHandle API 打开的时，此列才包含值。
VIDEOFILE	VARCHAR(254)	视频文件名。仅当视频是用 DBvOpenFile API 打开的时，此列才包含值。
STARTFRAME	INTEGER	起始帧号
ENDFRAME	INTEGER	结束帧号
REPFRAME	INTEGER	代表性帧的号码
REPFRAMEDATA	DB2IMAGE	代表性帧的数据
COMMENTS	LONG VARCHAR	注释

您可灵活地决定在数据库中创建多少镜头目录，以及在每个镜头目录中存储哪些镜头的信息。可创建一个目录来存储许多视频的镜头信息、将每个视频的镜头信息存储在独立目录中、或将视频中多个镜头的信息存储在多个目录中。

使用此 API 时，需要指定目录的名称。超过 16 个字符的名称将被截断。还需要指定 SQLConnect 调用返回给数据库的数据库连接句柄。例如,下列语句创建一个名为 hotshots 的镜头目录:

```
SQLHDBC  hdbc;

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvCreateShotCatalog(
                "hotshots",          /* shot catalog name */
                hdbc);               /* database connection handle */
```

镜头目录视图名为 MMDBSYS.SVcatname，其中 catname 是镜头目录的名称。例如，hotshots 目录的其中一个视图名为 MMDBSYS.SVHOTSHOTS。

存储关于单个镜头的信息（仅数据库）： 使用 DBvInsertShot API 将关于单个镜头的信息存储在镜头目录中。仅当镜头目录位于数据库中时，才可存储视频中单个镜头的信息。存储在目录中的信息包括:

- 镜头句柄
- （存储在表中的视频剪辑的）视频表名
- （存储在表中的视频剪辑的）视频列名
- （存储在表中的视频剪辑的）视频句柄
- （存储在表中的视频剪辑的）视频文件名
- 起始帧号

- 结束帧号
- 代表性帧的号码
- 代表性帧的数据

但是，不存储对镜头的注释。有关如何将注释添加至所存储的镜头信息的描述，参见第 32 页的『指定镜头的注释（仅数据库）』。

使用 DBvInsertShot API 时，需要指定镜头目录名并指定指向镜头的指针。设置指向镜头的指针的一种方法是获取下一个镜头，如第 25 页的『获取镜头』中所述。还需要指定 SQLConnect 调用返回给数据库的数据库连接句柄。例如，下列语句获取帧 1 之后的下一镜头，然后将关于该镜头的信息存储在名为 hotshots 的镜头目录中:

```
SQLHDBC  hdbc;
SQLHENV  henv;
DBvIOType      *video;
long start_frame = 1;
char shotDetected = 0;
DBvShotControl    shotCtrl;
DBvShotType    shot;

shotCtrl→method=DETECT_CORRHIST
shotCtrl→normalCorrValue=60;
shotCtrl→sceneCutSkipXY=1;
shotCtrl→CorrHistThresh=10;
shotCtrl→DissThresh=10;
shotCtrl→DissCacheSize=4;
shotCtrl→DissNumCaches=7;
shotCtrl→minShotSize=0;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvDetectShot(
       video,
       start_frame,
       &shotDetected,
       &shotCtrl,
       &shot)

rc=DBvInsertShot (
        "hotshots",             /*shot catalog name*/
       shot,                      /*pointer to shot*/
        hdbc);                  /*database connection handle*/
```

存储关于视频中所有镜头的信息: 使用 DBvBuildStoryboardTable API 或 DBvBuildStoryboardFile API 在镜头目录中存储视频中所有镜头的信息。DBvBuildStoryboardTable API 将信息存储在驻留在数据库中的镜头目录中。DBvBuildStoryboardFile API 创建一个镜头目录文件，并在该文件中存储镜头信息。

对于任一 API，源视频既可以在数据库表中也可以在文件中。

使用任一 API 时，都需要:

- 指定镜头目录名称。
- 指向视频结构。
- 指向 DBvShotControl 数据结构。

- 指向 DBvStoryboardCtrl 数据结构。此数据结构中的值控制将哪些以及多少视频帧存储为镜头目录中的代表性帧。有关设置这些值的更多信息，参见『构建故事板』。

还需要指定 SQLConnect 调用返回给数据库的数据库连接句柄（仅对于 DBvBuildStoryboardTable API）。

例如，下列语句在镜头目录中存储视频中所有镜头的信息。镜头目录位于数据库中。

```
SQLHDBC  hdbc;
SQLHENV  henv;
DBvIOType      *video;
DBvShotControl    shotCtrl;
DBvStoryBoardCtrl    sbCtrl;

sbCtrl→thresh1=50
sbCtrl→thresh2=500;
sbCtrl→delta=20;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvBuildStoryboardTable (
        "hotshots",             /*shot catalog name*/
       video,                     /*pointer to video structure*/
       shotCtrl,                  /*pointer to shot control structure*/
       sbctrl,                    /*pointer to storyboard control structure*/
        hdbc);                  /*database connection handle*/
```

下列语句创建一个镜头目录文件，并在该文件中存储视频中所有镜头的信息。

```
DBvIOType      *video;
DBvShotControl    shotCtrl;
DBvStoryBoardCtrl    sbCtrl;

sbCtrl→thresh1=50
sbCtrl→thresh2=500;
sbCtrl→delta=20;

rc=DBvBuildStoryboardFile (
       "hotshots",               /*shot catalog file name*/
       video,                    /*pointer to video structure*/
       shotCtrl,                 /*pointer to shot control structure*/
       sbctrl);                  /*pointer to storyboard control structure*/
```

构建故事板: 正如其名称所暗示的，DBvBuildStoryboardTable API 和 DBvBuildStoryboardFile API 对于存储要在故事板中使用的信息来说特别有用。**故事板**是视频的可视摘要。可通过显示在镜头目录中存储的视频的代表性帧来创建故事板。

DBvBuildStoryboardTable API 和 DBvBuildStoryboardFile API 存储镜头的一个或多个代表性帧。在 DBvStoryboardCtrl 结构中指定的值，控制为镜头存储的代表性帧的数目以及将使用哪些帧。参见第 19 页的『镜头检测数据结构』以获取 DBvStoryboardCtrl 结构的定义。第 31 页的图 3说明如何使用 DBvStoryboardCtrl 字段中的值。

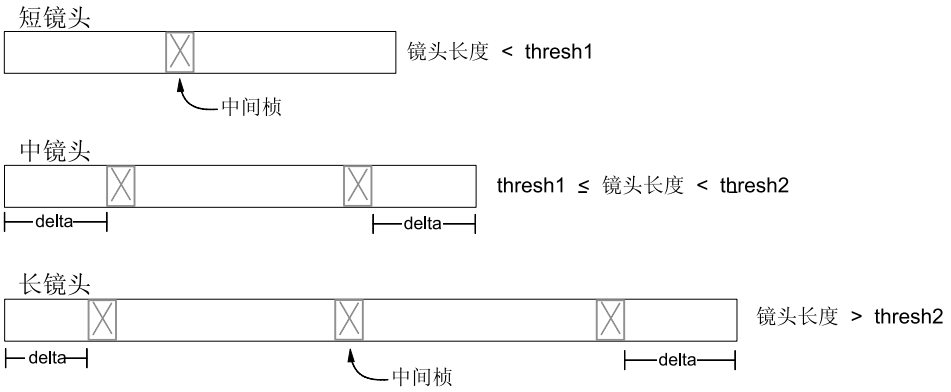


图 3. 如何使用 DBvStoryboardCtrl 结构中的值

如图 3 中所示:

- 对于短镜头，仅存储一个代表性帧。短镜头中帧的数目小于 DBvStoryboardCtrl 数据结构中的 thresh1 值。代表性帧是镜头的中间帧。
- 对于中镜头，将存储两个代表性帧。中镜头中帧的数目大于或等于 DBvStoryboardCtrl 数据结构中的 thresh1 值，且小于或等于 thresh2 值。DBvStoryboardCtrl 数据结构中的 delta 值确定从视频起始处到第一个代表性帧的帧数。delta 值还确定从第二个代表性帧到视频末尾的帧数。
- 对于长镜头，将存储三个代表性帧。长镜头中帧的数目大于 DBvStoryboardCtrl 数据结构中的 thresh2 值。DBvStoryBoardCtrl 数据结构中的 delta 值确定从视频起始处到第一个代表性帧的帧数。第二个代表性帧是视频的中间帧。从第三个代表性帧到视频末尾的距离由 delta 值确定。

若将 thresh1 或 thresh2 值设置为 -1，则可将任何镜头当作短镜头来处理。在此情况下，仅在镜头目录中存储镜头的一个代表性帧，即中间帧。

除了 DBvStoryboardCtrl 数据结构中的值之外，DBvShotControl 数据结构中的许多字段还会影响对故事板中后续屏幕存储哪些代表性帧。例如，DBvShotControl 数据结构中的 CorrHistThresh、normalcorrValue 和 minShotSize 字段指定镜头检测的阈值，因而影响将在视频的故事板中显示什么帧。使用 DBvBuildStoryBoardTable API 和 DBvBuildStoryBoardFile API 来存储将在故事板中使用的镜头信息时，可以首先使用 DBvStoryBoardCtrl 和 DBvShotControl 数据结构的初始设置来执行测试运行。然后，可通过更改这些数据结构的各种字段中的值来调整结果。

显示故事板: 可创建程序来显示故事板。通过存取镜头目录中存储的视频的代表性帧来做到这一点。若使用 DBvBuildStoryboardFile API 来存储视频的镜头，则镜头目录文件指向代表性帧的 GIF 文件。可使用浏览器或适当的显示程序来显示这些 GIF 文件。

若使用 DBvBuildStorybBoardTable API 来存储视频的镜头，（存储在数据库中的）镜头目录包含代表性帧的数据。可以在镜头目录视图中存取代表性帧数据（参见第 28 页的表 3 以获取视图的描述）。代表性帧数据使用 YUV 格式；这不是大多数图像显示程序可显示的格式。要显示代表性帧，可使用 DBvFrameDatato24BitRGB API 来转换帧数据，如第 26 页的『显示检索到的帧』中所述。然后，可使用浏览器或适当的显示程序来显示代表性帧。

故事板样本程序: SAMPLES 子目录包含两个样本程序，它们演示构建和显示视频的故事板。`makesf.exe` 文件中的样本程序使用 DBvBuildStoryBoardFile API 来创建镜头

目录文件，并在该文件中存储镜头数据。另一个样本程序 makehtml.exe 存取镜头目录文件，并创建 HTML 页来供 Web 浏览器显示。

指定镜头的注释（仅数据库）： 可指定要与镜头的其它信息一起存储在镜头目录中的注释。使用 DBvSetShotComment API 来指定注释。

使用此 API 时，需要指定存储注释的镜头目录的名称、正在添加其注释的镜头的句柄，以及注释。还需要指定 SQLConnect 调用返回给数据库的数据库连接句柄。例如，下列语句在名为 hotshots 的镜头目录中添加镜头注释（从帧号 85 开始）：

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

EXEC SQL SELECT SHOTHANDLE INTO :shothandle
  FROM MMDBSYS.SVHOTSHOTS
  WHERE STARTFRAME=85;

rc=DBvSetShotComment (
        "hotshots",             /*shot catalog name*/
        shothandle,                     /*shot handle*/
        "shot of beach at sunset",       /*comment*/
        hdbc);                  /*database connection handle*/
```

更改所存储的镜头信息（仅数据库）： 可更改镜头目录中存储的镜头信息。要执行该操作，使用 DBvUpdateShot API。将替换信息放入 DBvShotType 结构。还需要为剩余字段指定信息（即使它们不变）。当使用 DBvUpdateShot API 时，指定目录的名称并指定指向 DBvShotType 结构的指针。还需要指定 SQLConnect 调用返回给数据库的数据库连接句柄。

更改镜头的信息时，可选择更改与该信息一起存储的注释（若有的话）。若想更改注释，则在 DBvShotType 结构中指定它。若想保留旧注释，则在 DBvShotType 结构中指定空值。

例如，下列语句更改名为 hotshots 的目录中存储的镜头信息；该镜头从帧号 85 开始:

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];
DBvShotType    shot;
DBvFrameData  fd110;

/* get shot handle */

EXEC SQL SELECT SHOTHANDLE INTO :shothandle
  FROM MMDBSYS.SVHOTSHOTS
  WHERE STARTFRAME=85;

/* change shot attribute */

shot.startFrame=110;
shot.endFrame=200;
shot.repframe=110;
shot.fd=fd110;
shot.comment=NULL;

/* update shot information */
```

```
SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvUpdateShot (
        "hotshots",             /*shot catalog name*/
        shot,                           /*shot information*/
        hdbc);                  /*database connection handle*/
```

合并镜头目录中的镜头信息（仅数据库）： 可以合并镜头目录中存储的两个镜头的信息。在合并镜头信息时，通过标识第一个镜头和第二个镜头来指示合并次序。将第一个镜头的起始帧号存储为合并后镜头的起始帧号。将第一个和第二个镜头之间最大帧的号码存储为合并后镜头的结束帧号。合并将存储的第一个镜头的信息替换为合并后镜头的信息；将从镜头目录中删除存储的第二个帧的信息。

使用 DBvMergeShots API 来合并镜头目录中两个镜头的信息。当使用此 API 时，指定镜头目录名，后跟要合并的第一个和第二个帧的句柄。还需要指定 SQLConnect 调用返回给数据库的数据库连接句柄。例如，下列语句合并名为 hotshots 的目录中存储的两个镜头的信息；第一个镜头开始于帧 85，第二个镜头开始于帧 210:

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle1[37];
char shothandle2[37];

EXEC SQL SELECT SHOTHANDLE INTO :shothandle1
  FROM MMDBSYS.SVHOTSHOTS1
  WHERE STARTFRAME=85;

EXEC SQL SELECT SHOTHANDLE INTO :shothandle2
  FROM MMDBSYS.SVHOTSHOTS2
  WHERE STARTFRAME=210;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvMergeShots (
        "hotshots",             /*shot catalog name*/
        shothandle1,                    /*shot handle for first shot*/
        shothandle2,                    /*shot handle for second shot*/
        hdbc);                  /*database connection handle*/
```

从镜头目录中删除镜头信息（仅数据库）： 使用 DBvDeleteShot API 从镜头目录中删除关于镜头的信息。使用此 API 时，指定镜头目录名，后跟镜头句柄。还需要指定 SQLConnect 调用返回给数据库的数据库连接句柄。例如，下列语句删除名为 hotshots 的镜头目录中关于镜头的信息（该镜头从帧号 85 开始）：

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];

EXEC SQL SELECT shothandle INTO :shothandle
  FROM mmdbsys.svhotshots
  WHERE startframe=85;

rc=DBvDeleteShot (
        "hotshots",             /*shot catalog name*/
        shothandle,                     /*shot handle*/
        hdbc);                  /*database connection handle*/
```

删除镜头目录（仅数据库）： 使用 DBvDeleteShotCatalog API 来删除镜头目录。在使用此 API 时，指定要删除的镜头目录的名称，并指定 SQLConnect 调用返回给数据库的连接句柄。例如，下列语句删除名为 hotshots 的镜头目录:

```
SQLHDBC  hdbc;
SQLHENV  henv;

rc=DBvDeleteShotCatalog (
        "hotshots",             /*shot catalog name*/
        hdbc);                  /*database connection handle*/
```

第 2 章 概述

DB2（DB2）通用数据库（UDB）是功能强大的面向对象的数据库管理器。它存储并保护传统数字和字符数据以及大型的复杂对象（LOB）。DB2 Extender 帮助您利用 DB2 的面向对象的功能。Extender 为图像、音频、视频和文本对象定义单值数据类型和特殊函数。这样，Extender 节省了在应用程序中定义这些数据类型和函数的时间和精力。可通过 SQL 使用这些数据类型和函数。为此，Extender 向应用程序提供了对任何或所有这些类型的数据以及传统数字和字符数据的单一存取点。另外，Extender 向应用程序提供了搜索信息的新方法。例如，应用程序可使用颜色或纹理的可视示例来通过图像的可视特征搜索图像。

利用 DB2

DB2 Extender 利用 DB2 的面向对象功能。特别是，借助 DB2，您可以:

- 将最多 2G 字节的 LOB 存储在 DB2 数据库中。
- 为这些大的复杂的对象定义单值数据类型。使用这些用户定义类型（UDT）来标识对象表示的数据类型，如图像或音频。
- 定义可对用户定义数据类型请求的特定函数。例如，可定义函数来计算图像中的颜色数，或获取音频的采样速率。以与其它 SQL 函数相同的方法在 SQL 语句中请求这些用户定义函数（UDF）。

DB2 Extender 创建图像、音频、视频和文本对象的 UDT 和 UDF。对于您完成下列各项，UDT 和 UDF 可能会起到重要帮助作用:

- 开发应用程序。因为 Extender 定义数据类型和函数，所以不必在您的应用程序中定义它们。
- 确保一致性。同一组的 Extender UDT 和 UDF 可用于所有应用程序。这提供了现成的一致性级别，否则，在处理大对象的应用程序之间获得这样的一致性级别是很困难的。
- 创建功能强大的查询。因为请求 UDF 的方法与请求其它 SQL 函数的方法相同，所以应用程序可包括多数据类型查询。一个 SQL 语句可同时存取图像、音频、视频和文本对象以及传统的数字和字符数据。可在嵌入式 SQL 语句以及 “DB2 调用层接口”（DB2 CLI）调用中指定 UDF 和 UDT。

因为可将 Extender 处理的对象存储在 DB2 数据库中，所以对于存储在数据库中的传统数据类型的那些对象来说，同一安全性、完整性和恢复保护都适用。

另外，DB2 Extender 利用 “DB2 通用数据库扩充企业版” 的分区数据库环境。分区允许应用程序使用对于单个计算机来说太大的数据库。分区还允许 SQL 操作并行执行，从而提高 SQL 查询或实用程序的速度。

功能强大的搜索信息的新方法

在搜索信息方面，DB2 Extender 向应用程序提供了很大的灵活性。应用程序可搜索与存储在数据库中的传统类型的数据相关联的对象。例如，可通过音频剪辑的描述或录制日期来搜索它们。应用程序还可通过这些对象的固有特征（如视频剪辑的播放时间）来搜索它们。Extender 自动确定并存储这些特征，以供搜索时使用。

© Copyright IBM Corp. 1998, 2003

新的搜索方法

应用程序甚至可按内容来搜索图像。想象一下使用可视示例来搜索图像的应用程序。借助这种应用程序，用户可选择示例图像，并让应用程序查找颜色或纹理类似于示例中的颜色或纹理的其它图像。借助 DB2 Extender 按图像内容进行查询（QBIC）的功能，可创建以这种可视方式搜索图像的应用程序。

DB2 Extender

DB2 Extender 分别由 Image Extender、Audio Extender、Video Extender 和 Text Extender 组成。

本书涉及 Image Extender、Audio Extender 和 Video Extender。除非另有声明，否则本书中对 “Extender” 或 “DB2 Extender” 的任何进一步的引用都指的是 Image Extender、Audeo Extender 和 Video Extender。有关 Text Extender 的信息，参见*《Text Extender 管理与编程》*。关于 XML Extender 的信息，参见*《XML Extender 管理与编程》*。

SDK 和运行时环境

DB2 Extender 安装程序包提供了 Software Developers Kit（SDK）以及客户机和服务器运行时环境。您可以在安装了 DB2 Extender SDK 的客户机或服务器上开发 DB2 Extender 应用程序。

可以在包括 DB2 Extender 客户机运行时代码和服务器运行时代码的服务器中运行 DB2 Extender 应用程序。（客户机运行时代码是在您安装服务器运行时代码时自动安装的。）您还可以在安装了 DB2 Extender 客户机运行时代码的客户机上运行 DB2 Extender 应用程序。若从客户机运行 Extender 应用程序，则您需要确保可与服务器相连。

使用 Extender

可在 DB2 应用程序中请求 Extender UDF，也可以使用 DB2 命令行处理器交互地请求它们。

Extender 还提供了下列应用程序编程接口（API）：

- 用来准备和维护用于图像、音频和视频数据的数据库的管理 API。
- 用来显示图像和播放视频和音频剪辑的显示和播放 API。
- 用来准备图像并请求按内容查询的 QBIC API。（还可通过 UDF 请求内容搜索。）
- 用来标识基于视频中的画面切换的帧序列的视频镜头检测 API。

DB2 Extender 还提供了用来发出管理命令的命令行处理器。为了区分 Extender 提供的命令行处理器与 DB2 提供的命令行处理器，我们将把前者称为 “db2ext 命令行处理器”，将后者称为 “DB2 命令行处理器”。

示例

广告发布代理维护其广告的 DB2 数据库。过去，该代理存储关于每个广告活动的数字和字符数据，如客户的名称和广告完成日期。通过安装 DB2 UDB 和 DB2 Extender，该代理现在还在数据库中存储广告内容。这包括印刷广告的图像、电视广告的录像和广播广告的录音。如图 4 所示，所有相关的广告信息在一个名为 ADS 的数据库表中。

图 4. 多媒体数据库表. 该表包含图像、音频和视频数据以及传统数据类型。显示视频、音频和图像。

示例 1: 按特征检索视频

广告发布代理的帐户管理员需要查看在 1997 年为 IBM 创建的视频广告，但仅想看持续时间是 30 秒或更短的广告。

第 38 页的图 5 显示了存取视频的查询。注意查询中名为 `Filename` 和 `Duration` 的 Video Extender UDF。

示例

```
SELECT FILENAME(ADS_VIDEO)
       FROM ADS
WHERE CLIENT='IBM' AND
SHIP_DATE>='01/01/1997' AND
DURATION(ADS_VIDEO) <=30
```

图 5. 存取视频的查询

此查询返回期望的视频的文件名。然后，帐户管理员打开自己喜欢的视频播放程序，并播放每个视频文件的内容。

图 5 是一个帐户管理员可交互发出的查询示例。更常见的是，帐户管理员将使用应用程序来查找并播放视频。例如，图 6 显示了用 C 编码的这种应用程序的一些关键元素。该应用程序在名为 hvVid_fname 的 DB2 主变量中检索视频文件名。还要注意该应用程序使用名为 DBvPlay 的播放 API 来播放视频。

```
#include <dmbvideo.h>

int count = 0;

EXEC SQL BEGIN DECLARE SECTION;
char hvClient[30];                      /*client name*/
char hvCampaign[30];                    /*campaign name*/
char hvSdate[8];                        /*ship date*/
char hvVid_fname [251]                  /*video file name*/
EXEC SQL END DECLARE SECTION;


EXEC SQL DECLARE c1 CURSOR FOR
     SELECT CLIENT, CAMPAIGN, SHIP_DATE, FILENAME(ADS_VIDEO)
       FROM ADS
       WHERE CLIENT='IBM' AND
         SHIP_DATE≥'01/01/1997' AND
         DURATION(ADS_VIDEO)≤30
FOR FETCH ONLY;
```

图 6. 存取并播放视频的应用程序 (1/2)

```
EXEC SQL OPEN c1;
for (;;) {
        EXEC SQL FETCH c1 INTO :hvClient, :hvCampaign,
                         :hvSdate, :hvVid_fname;

         if (SQLCODE != 0)
                   break;

    printf("\nRecord %d:\n", ++count);
    printf("Client = '%s'\n", hvClient);
    printf("Campaign = '%s'\n", hvCampaign);
    printf("Sdate = '%s'\n", hvSdate);

    rc=DBvPlay(NULL,MMDB_PLAY_FILE,hvVid_fname,MMDB_PLAY_WAIT);
}
EXEC SQL CLOSE c1;
```

图 6. 存取并播放视频的应用程序 (2/2)

示例 2: 按内容搜索图像

广告发布代理的图形演示者正在为客户开发新的印刷广告。该演示者想在广告的背景中使用特定形状的蓝色，并想在代理创建印刷广告之前查看是否已使用了该颜色。要那样做，图形演示者运行按内容搜索图像的应用程序。这些图像存储在数据库表中（参见第 37 页的图 4）。该应用程序要求用户提供可视示例，即，表明感兴趣的颜色的图像。该应用程序然后分析示例中的颜色，并查找其颜色与示例最匹配的图像。

图 7 显示了可视示例和检索到的与其颜色最匹配的图像。

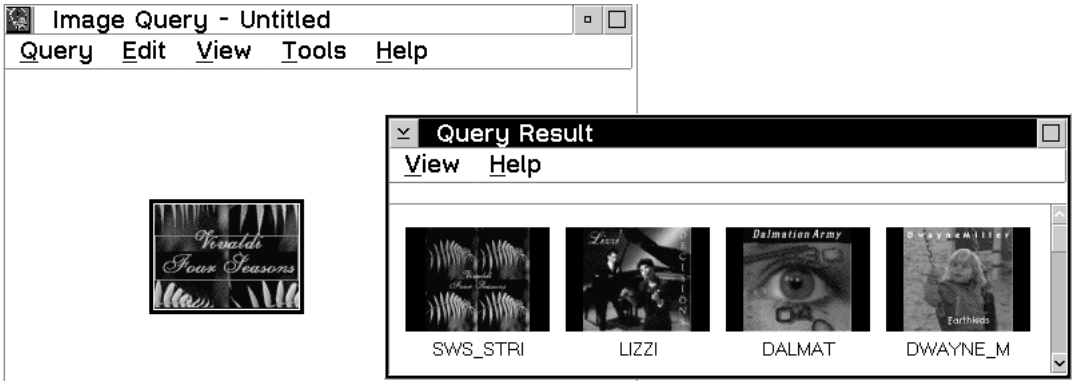


图 7. 按内容搜索图像. 使用可视示例来按平均颜色搜索图像。

第 40 页的图 8 显示应用程序的一些关键元素。注意应用程序使用名为 QbQueryCreate 的 QBIC API 来创建 QBIC 查询，使用 QbQueryAddFeature 和 QbQuerySetFeatureData 来将颜色选择添加至查询，使用 QbQuerySearch 来发出查询，并使用 QbQueryDelete 来删除查询。应用程序还使用名为 DBiBrowse 的图形 API 来显示检索到的图像。

示例

```
#include <dmbqbqpi.h>

#define   MaxQueryReturns        10

static  SQLHENV    henv;
static  SQLHDBC    hdbc;
static  SQLHSTMT   hstmt;
static  SQLRETURN  rc;

void main(int argc, char* argv[])
{
 char           line[4000];
 char*          handles[MaxQueryReturns];
 QbQueryHandle  qHandle=0;
 QbResult       results[MaxQueryReturns];
SQLINTEGER     count;
 SQLINTEGER     resultType=qbiArray;

 SQlAllocEnv(&henv);
 SQLAllocConnect(henv, &hdbc);
 rc = SQLConnect(hdbc, (SQLCHAR*)"qtest", SQL_NTS,
              (SQLCHAR*)"", SQL_NTS, (SQLCHAR*)"", SQL_NTS);

 if (argc !=2) {
 printf("usage: query colorname\n");
  exit(1);
 }

 QbImageSource is;
      is.type = qbiSource_AverageColor;

 /* run the get color subroutine */
 getColor(argv[1], is.average.Color);

 QbQueryCreate(&qhandle);
 QbQueryAddFeature(qhandle, "QbColorFeatureClass");
 QbQuerySetFeatureData(qhandle, "QbColorFeatureClass",&is);
 QbQuerySearch(qhandle, "ADS", "ADS_IMAGE", 10, 0, resultType
                           &count, results);
 for (int j = 0; j <count; j++) {
 printf(j,":\n");

 DBiBrowse("usr/local/bin/xv %s", MMDB_PLAY_HANDLE, handles[j],
               MMDB_PLAY_WAIT);
 }
```

图 8. 按内容搜索图像的应用程序 (1/2)

```
QbQueryDelete(qhandle);

 SQLDisconnect(hdbc);
 SQLFreeConnect(hdbc);
 SQLFreeEnv(henv);
}
```

图 8. 按内容搜索图像的应用程序 (2/2)

操作环境

DB2 Extender V7 在客户机／服务器环境中与 DB2 通用数据库 V7.1（或更高版本）一起运作。受支持的平台所需的最低版本和发行版级别与 “DB2 通用数据库 V7.1” 所需的最低版本和发行版级别相同。

受支持的客户机平台是: OS/2、AIX、Windows NT® 和更高版本、Windows 95、Windows 98、Solaris 操作环境和 HP-UX。

受支持的服务器是: OS/2、AIX、Windows NT 和更高版本、Solaris 操作环境和 HP-UX。

第 42 页的图 9 显示了受支持的平台。

另一 DB2 Extender 产品 “DB2 通用数据库 z/OS 版 Extender” 支持 z/OS 客户机和服务器。有关 “DB2 通用数据库 z/OS 版 Extender” 的更多信息，参见 *DB2 Universal Database for z/OS Image, Audio, and Video Extenders Administration and Programming* 或 *DB2 Universal Database for z/OS Text Extender Administration and Programming*。

DB2 Extender 可在单分区数据库环境中操作。

仅限于 EEE: Extender 还可在下列平台上的多分区数据库环境中操作: AIX、Solaris 操作环境和 Windows NT 和更高版本。

要在多分区数据库环境中运作，DB2 Extender 必须与 “DB2 通用数据库扩充企业版” 配合使用。

示例

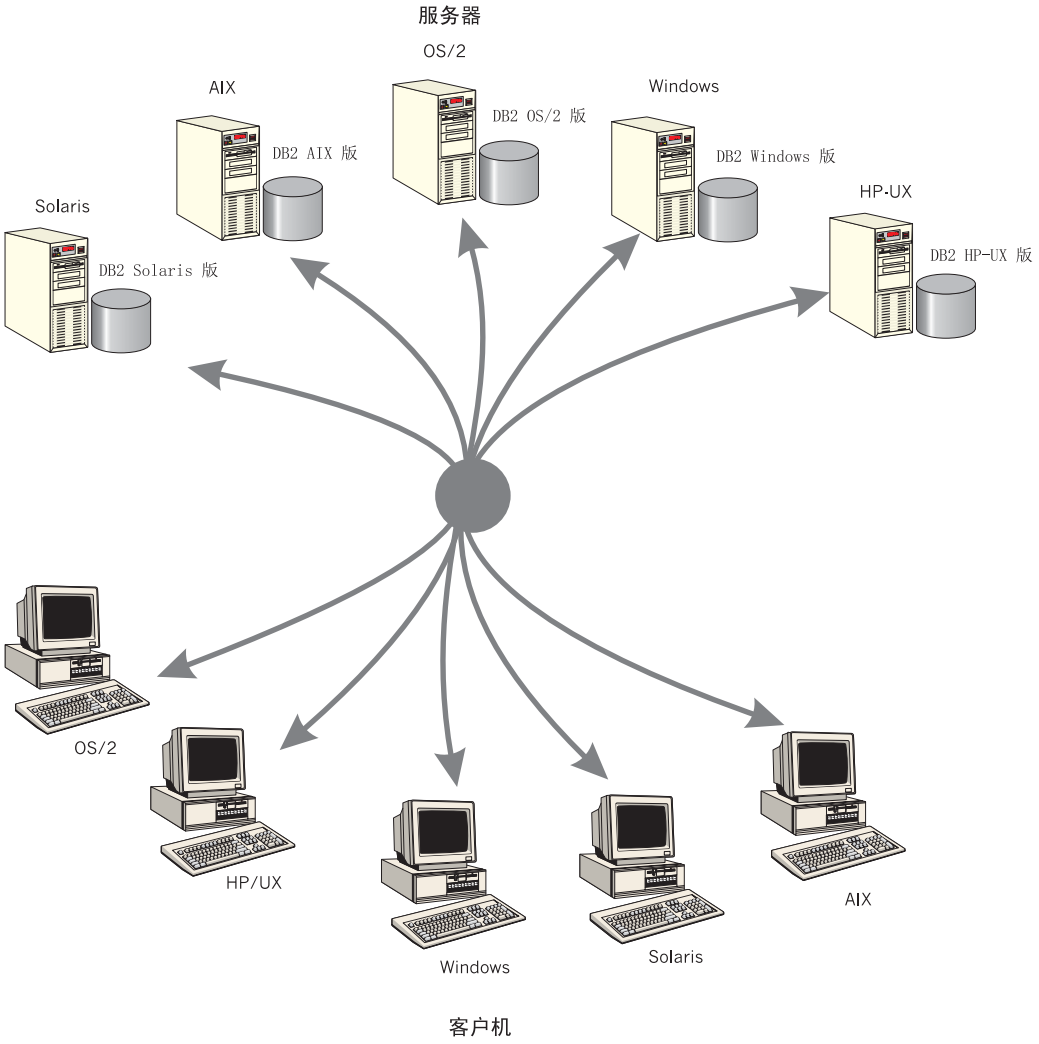


图 9. DB2 Extender 平台

第 3 章 DB2 Extender 概念

本章描述在使用 DB2 Extender 之前，需要了解的概念。

主题	参见
面向对象的概念	页 43
Extender 数据结构	页 46
分区数据库概念	页 50
Extender 安全性和恢复	页 52

有关面向对象的概念的更多信息，参见 *DB2 Application Development Guide*。

面向对象的概念

DB2 支持**面向对象**，它是这样一个概念：任何东西，无论是真实的还是虚拟的，都可在应用程序中表示成由一组操作和数据值组成的对象。例如，文档可以由文档数据以及可以对文档执行的操作（例如，填充、发送和打印）所组成的文档对象来表示。视频剪辑可以由视频对象来表示，视频对象由数据和操作（例如，播放视频剪辑或查找特定的视频帧）组成。就如现实生活中的对象，表示对象具有属性。例如，可对视频对象给出诸如压缩类型和采样速率之类的属性。

可按类型来将对象分组。同一类型的对象具有相同的属性，且行为方式相同，即，它们与相同的操作相关联。例如，若将视频类型定义为具有压缩类型属性，该视频类型的所有对象都具有该属性。若可播放视频类型的某个对象，则可播放视频类型的所有对象。

DB2 对面向对象的支持允许您将对象类型的实例存储在表的列中，并通过 SQL 语句中的函数对它们进行操作。例如，可将视频对象存储在表列中，并使用 SQL 函数对它们进行操作。另外，可在应用程序之间共享存储的对象的属性和行为。对于相同的对象类型，所有应用程序“看到”同一组属性和行为。

典型情况下，视频对象大而复杂。图像和音频对象亦如此。作为它支持面向对象的一部分，DB2 允许将大对象（LOB）存储在数据库中。它还提供了通过用户定义类型（UDT）、用户定义函数（UDF）和触发器来定义和处理 LOB 的方法。

大对象

DB2 允许以下列形式将**大对象**（LOB）存储在数据库中:

- 二进制大对象（BLOB）
- 字符大对象（CLOB）
- 双字节字符大对象（DBCLOB）

BLOB 是二进制字符串。图像、音频和视频对象以 BLOB 形式存储在数据库中。CLOB 是由带有相关代码页的单字节字符构成的字符串。此数据类型用于包含单字节字符的文本对象。DBCLOB 是由带有相关代码页的双字节字符构成的字符串。此数据类型用于使用了双字节字符的文本对象。

© Copyright IBM Corp. 1998, 2003

每个 LOB 的长度最多可以是 2G 字节；但是 DB2 允许每个表有许多 LOB 列。每行最多可存储 24G 字节的 LOB 空间，每个表最多可存储 4 太字节的 LOB 空间。

由于其大小，LOB 的内容不直接存储在用户的表中。每个 LOB 由表中的大对象描述符标识。该描述符用来存取存储在磁盘上别处的大对象。

DB2 Extender 增加了灵活性：将 LOB 的内容保存在文件中，并在数据库中指向它。当使用 DB2 Extender 来存储对象时，进行此指定。

用户定义类型

图像、视频、音频对象在数据库以 BLOB 形式表示。**用户定义类型**（UDT），也称为**单值类型**，提供了区分不同 BLOB 的方法。例如，可对图像对象创建 UDT，并对音频对象创建另一个 UDT。尽管图像和音频对象都是作为 BLOB 来存储的，但是它们仍被作为不同于 BLOB 并且互不相同的类型来对待。

用 SQL CREATE DISTINCT TYPE 语句创建 UDT。例如，假设您正在开发处理地图上的地理特征的应用程序。为地图对象创建名为 map 的单值类型，如下所示:

```
CREATE DISTINCT TYPE map AS BLOB (1M)
```

在内部，地图类型图像表示成长度为 1M 字节的 BLOB，但视为单值对象类型。

可象 SQL 内置类型那样使用 UDT 来描述存储在表的列中的数据。在以下示例中，创建表时，将列设计成存放地图类型的数据:

```
CREATE TABLE places
  (locid     INTEGER NOT NULL,
   location  CHAR (50),
   grid      map)
```

每个 DB2 Extender 创建其类型的 UDT，即，图像、音频和视频。

用户定义函数

用户定义函数（UDF）是创建 SQL 函数，并因此添加至随 DB2 一起提供的内置函数集的方法。尤其是，可创建专门对图像、音频和视频执行操作的 UDF。例如，可创建 UDF 来获取视频的压缩格式，并返回音频的采样速率。这提供了一种方法来定义特殊类型的对象的行为。例如，用为视频类型创建的函数来表达视频对象的行为，用为图像类型创建的函数来表达图像对象的行为。

用 SQL CREATE FUNCTION 语句创建 UDF。该语句指定此 UDF 所适用的数据类型。例如，以下语句创建名为 map_scale 的 UDF，它计算地图的比例尺。注：该 UDF 将地图标识为可对其应用的数据类型。实现函数的代码是用 C 语言编写的，在 EXTERNAL NAME 子句中标识:

```
CREATE FUNCTION map_scale (map)
  RETURNS SMALLINT
  EXTERNAL NAME 'scale!map'
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NO SQL
DETERMINISTIC
  NO EXTERNAL ACTION
```

可象使用内置函数那样在 SQL 语句中使用 UDF。在以下示例中， SQL SELECT 语句中使用 map_scale UDF 来返回存储在名为 grid 的表列中的地图的比例尺:

```
SELECT map_scale (grid)
  FROM places
  WHERE location='SAN JOSE, CALIFORNIA'
```

每个 DB2 Extender 都为其类型创建一组 UDF，即，特定于图像、特定于音频和特定于视频的 UDF。可在 SQL 语句中使用这些 UDF 来请求 Extender 功能，如将图像存储在表中、获取视频的帧速率，或添加关于音频的注释。

UDF 和 UDT 名

DB2 函数的全名是 *schema-name.function-name*， 其中，*schema-name* 是为 SQL 对象提供逻辑分组的标识符。DB2 Extender UDF 的模式名是 MMDBSYS。MMDBSYS 模式名也是 DB2 Extender UDT 的限定符。

可在引用 UDF 或 UDT 的任何位置使用全名。例如，MMDBSYS.CONTENT 标识模式名为 MMDBSYS，函数名为 CONTENT 的 UDF。MMDBSYS.DB2IMAGE 标识模式是 MMDBSYS，单值类型名是 DB2IMAGE 的 UDT。引用 UDF 或 UDT 时，也可省略模式名；在此情况下， DB2 使用函数路径来确定想要的函数或单值数据类型。

函数路径

函数路径是模式名的有序列表。DB2 使用列表中模式名的次序来解析对函数和单值数据类型的引用。可通过指定 SQL 语句 SET CURRENT FUNCTION PATH 来指定函数路径。这将在 CURRENT FUNCTION PATH 专用寄存器中设置函数路径。

对于 DB2 Extender，明智的选择是将 mmdbsys 模式添加至函数路径。这允许您输入 DB2 Extender UDF 和 UDT 名而不必对它们加上前缀 mmdbsys。以下是将 mmdbsys 模式添加至函数路径的示例:

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
```

若登录为 mmdbsys，则不要添加 mmdbsys 作为函数路径中的第一个模式: 若用 mmdbsys 用户标识登录，则函数路径中的第一个模式设置为 mmdbsys。之后，若尝试用 SET CURRENT FUNCTION PATH 语句将函数路径中的第一个模式设置为 mmdbsys，则函数路径将以两个 mmdbsys 模式开始 — 而这是错误的状态。

函数名重载

可**重载**函数名。这表示多个 UDF 即使在同一模式中，也可具有同一名称。但是，两个函数不能具有同一**特征符**。特征符是连接有所有函数参数的已定义数据类型的限定函数名。

触发器

触发器定义更改表时激活的一组操作。触发器可用来执行下列操作，例如: 验证输入数据、为最近插入的行生成一个值、从其它表中读取数据以进行交叉引用、或者将数据写入其它表以进行审查。通常将触发器用来进行完整性检查，或强制执行商务规则。

使用 SQL CREATE TRIGGER 语句创建触发器。以下语句创建触发器来强制关于部件库存的事务规则。当手上的数目少于最大库存数的 10% 时，该触发器再次订购部件。

```
CREATE TRIGGER reorder
       AFTER UPDATE OF on_hand, max_stocked ON parts
       REFERENCING NEW AS n_row
       FOR EACH ROW MODE DB2SQL
```

```
WHEN (n_row.on_hand < 0.10 * n_row.max_stocked)
BEGIN ATOMIC
       VALUES(issue_ship_request(n_row.max_stocked -
                                 n_row.on_hand,
                                 n_row.partno));
END
```

DB2 Extender 创建并维护管理支持表以记录关于存储在数据库中的图像、音频和视频的信息。（参见『管理支持表』以获取关于这些表的更多信息。）当对数据库插入、更新或删除图像、音频或视频数据时，Extender 使用触发器来更新这些表。

Extender 数据结构

Image Extender、Audio Extender 和 Video Extender 创建并使用管理支持表和句柄来存储并存取图像、音频和视频数据。Image Extender 还创建和使用 QBIC 目录来按内容存取图像。Video Extender 还使用索引文件和镜头目录来存取关于视频中的画面切换的信息。

管理支持表

管理支持表也称为元数据表，它包含 Extender 处理对图像、音频和视频对象的用户请求所需的信息。管理支持表中的信息通常称为“元数据”。

如第 47 页的图 10 所述，某些管理支持表标识为 Extender 启用的用户表和列。这些表引用其它管理支持表，那些管理支持表是为了存放关于启用列中的对象的属性信息而创建的。在这些表中，Extender 维护关于特定 Extender 定义的数据类型的独有属性的信息，以及关于 Extender 数据类型的公共属性的信息。例如，Image Extender 维护关于图像的宽度、高度和图像中的颜色数的信息，以及关于图像、音频和视频对象的公共属性的信息，诸如将该对象导入数据库的人或上次更新该对象的人的身份证号码。

管理支持表还可包含以 BLOB 格式存储的对象的内容。此外，可将对象存放在文件中，并由管理支持表引用。例如，可将视频剪辑以 BLOB 形式存储在管理支持表中，或存放在该表引用的文件中。

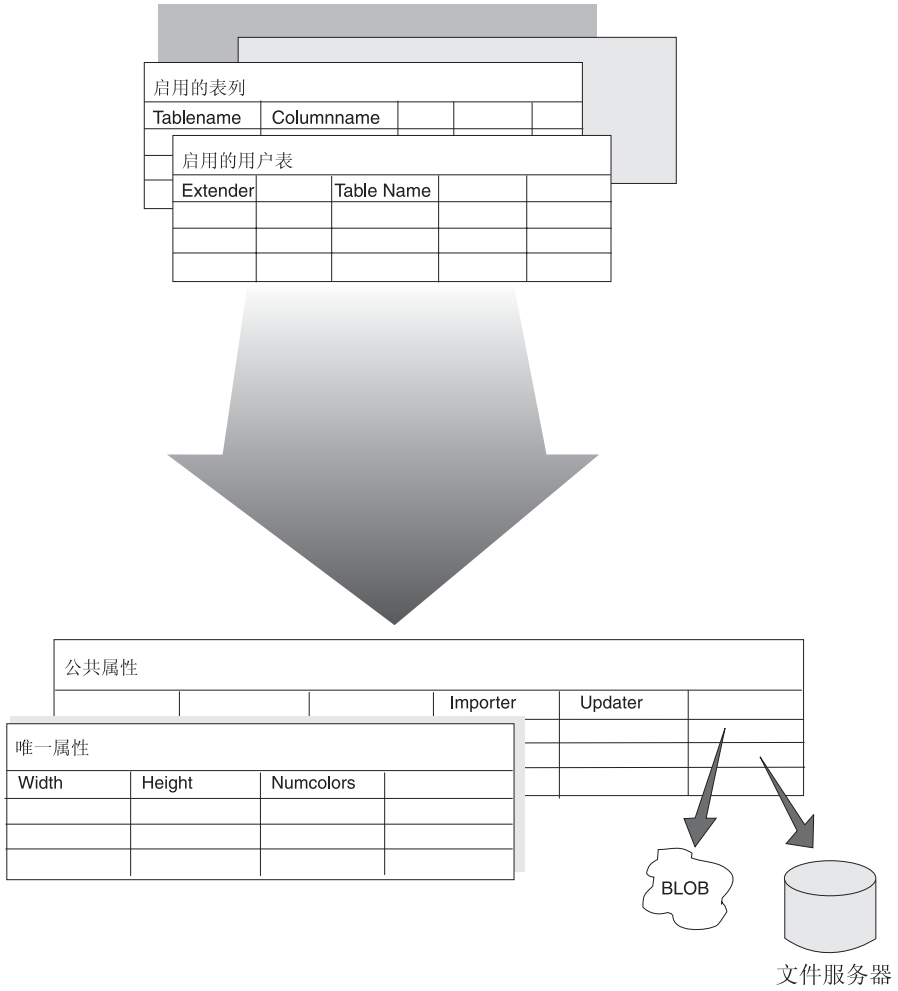


图 10. 管理支持表

句柄

当将图像、音频或视频对象存储在用户表中时，实际上并未将该对象存储在表中。相反，Extender 创建被称为**句柄**的字符串来代表该对象，并将句柄存储在表中。Extender 将对象存储在管理支持表中，或者，若将对象的内容存放在文件中，则在管理支持表中存储文件标识符。它还将对象的属性和句柄存储在管理支持表中。用这种方法，Extender 可将存储在用户表中的句柄与存储在管理支持表中的对象信息相链接。第 48 页的图 11 阐述为用户表中的两个图像存储的信息。

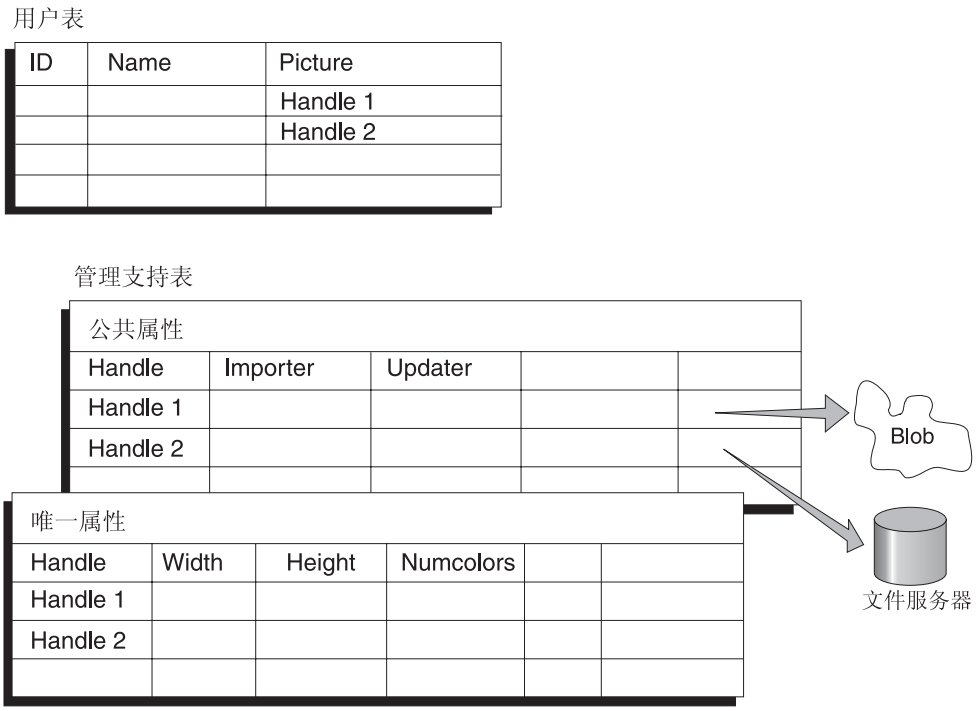


图 11. 句柄

QBIC 目录

QBIC 目录是一组存放关于图像的可视特征的数据的文件。Image Extender 使用此数据来按内容搜索图像。

对于要使其可用于“按内容搜索”的用户表中的每个图像列，创建 QBIC 目录。创建 QBIC 目录时，标识要让 Image Extender 对其进行分析、存储以及稍后查询数据的特征。还可在创建目录之后对 QBIC 目录添加或删除特征。

QBIC 目录可存放下列图像特征的数据:

平均颜色 图像中所有像素的颜色值的总和除以图像中的像素数。（像素 是可对其指定颜色和亮度的最小图像元素。）例如，若图像的 50% 由蓝色像素组成，另外 50% 为红色像素，则图像的平均颜色值是紫色。平均颜色用来搜索具有显著颜色的图像。若图像的颜色很显著，平均颜色将类似于显著颜色。

直方图颜色 根据 64 色频谱来测量图像中颜色的所占比重。对于 64 种颜色的每一种，直方图颜色标识图像中具有该颜色的像素的百分比。例如，图像的直方图颜色可能是 40% 白色像素、50% 蓝色和 10% 红色；图像中没有像素具有直方图频谱中的剩余颜色。直方图颜色用来搜索具有各种颜色的图像。

位置颜色 图像的指定区域中的像素的平均颜色值。例如，图像右上角可能会显示亮黄色太阳；此图像区域的位置颜色是亮黄色。位置颜色用来搜索特定区域中具有显著颜色的图像。

纹理 测量图像的粗糙度、反差和方向性。粗糙度指示图像中重复条目的大

小（例如，卵石与巨石）。反差标识图像中的亮度变化（亮与暗）。方向性指示图像中的方向是显著（如垂直方向的尖桩篱笆），还是不显著（如沙地的图像）。纹理用来搜索具有特定模式的图像。

要使图像可用于“按内容搜索”，编目该图像。编目图像时，Image Extender 通过计算图像的特征值分析图像，并将这些值存储在 QBIC 目录中。

按内容搜索图像时，您的查询标识搜索的一个或多个特征（如平均颜色）、每个特征的源（如示例图像）和目标编目图像集。Image Extender 计算源的特征值，并将其与目标图像的编目特征值作比较。然后，它计算得分，得分指示目标图像的特征值与源的类似程度。

可让 Image Extender 返回特定最类似于源的图像。Image Extender 将返回每个图像的句柄和图像得分。还可让 Image Extender 仅返回单个图像的得分。

视频索引

视频索引是一个文件， Video Extender 使用该文件来查找视频剪辑中的特定镜头或帧。

Video Extender 可检测到视频中的画面切换。**画面切换**是视频剪辑中的一个点，在该处，两个连续帧之间有明显的差异。例如，若摄象机在记录视频时更改其摄象点，就会发生这种情况。两次画面切换之间的帧构成一个**镜头**。

可使用 Video Extender 的画面检测能力来查找视频剪辑中的镜头，甚至个别帧。要做到这一点，Extender 需要对镜头或帧的信息创建索引。创建的索引信息存储在**索引文件**中。

镜头目录

镜头目录用来存储关于视频剪辑中的镜头的数据。可将镜头目录存储在数据库或文件中。

存储在文件中的镜头目录包含下列与镜头相关的数据:

- 镜头目录文件名
- 控制 Video Extender 检测镜头的方式的值，例如，镜头中的最小帧数
- 控制将多少帧，以及哪些帧，存储为镜头的代表性帧的值
- 镜头号
- 起始帧号
- 结束帧号
- 代表性帧的号码
- 包含代表性帧的内容的文件的名称

可存取镜头目录文件中的数据，或存取存储在数据库中镜头目录的视图。视图包含下列与镜头相关的数据的列:

- 镜头句柄
- 视频表名
- 视频列名
- 视频句柄
- 视频文件名

- 起始帧号
- 结束帧号
- 代表性帧的号码
- 代表性帧的数据
- 注释

分区数据库概念（仅限于 EEE）

DB2 Extender 可与“DB2 扩充企业版”一起操作，并以此方式来利用“DB2 扩充企业版”提供的分区数据库支持。

分区数据库是分布在两个或多个独立机器上的数据库。对于最终用户和应用程序开发人员，该数据库看起来象是单个机器上的单个数据库。分区允许应用程序有效地使用仅仅因为太大而导致一台机器不能处理的数据库。

分区数据库有两个或多个分区组成。每个分区都由其自己的**数据库分区服务器**管理。数据库分区服务器包括数据库管理器及其管理的数据和系统资源集。通常，会为每个机器指定一个数据库分区服务器。但是，单个机器上有可能有多个数据库分区服务器。每个数据库分区服务器存放整个数据库的一部分。数据库分区服务器有时也称为**节点**。

如第 51 页的图 12 所示，可对数据库分区进行逻辑分组，并对其指定名称。每组数据库分区称为**节点组**。例如，定义节点组允许将应用程序查询限制为对选择的数据库进行分区，从而提高事务速度。节点组可以只包含一个数据库分区，也可包含多个数据库分区。若节点组包含多个数据库分区，则它被称为**多分区节点组**。对多分区节点组命名的所有数据库分区必须驻留在同一数据库中。

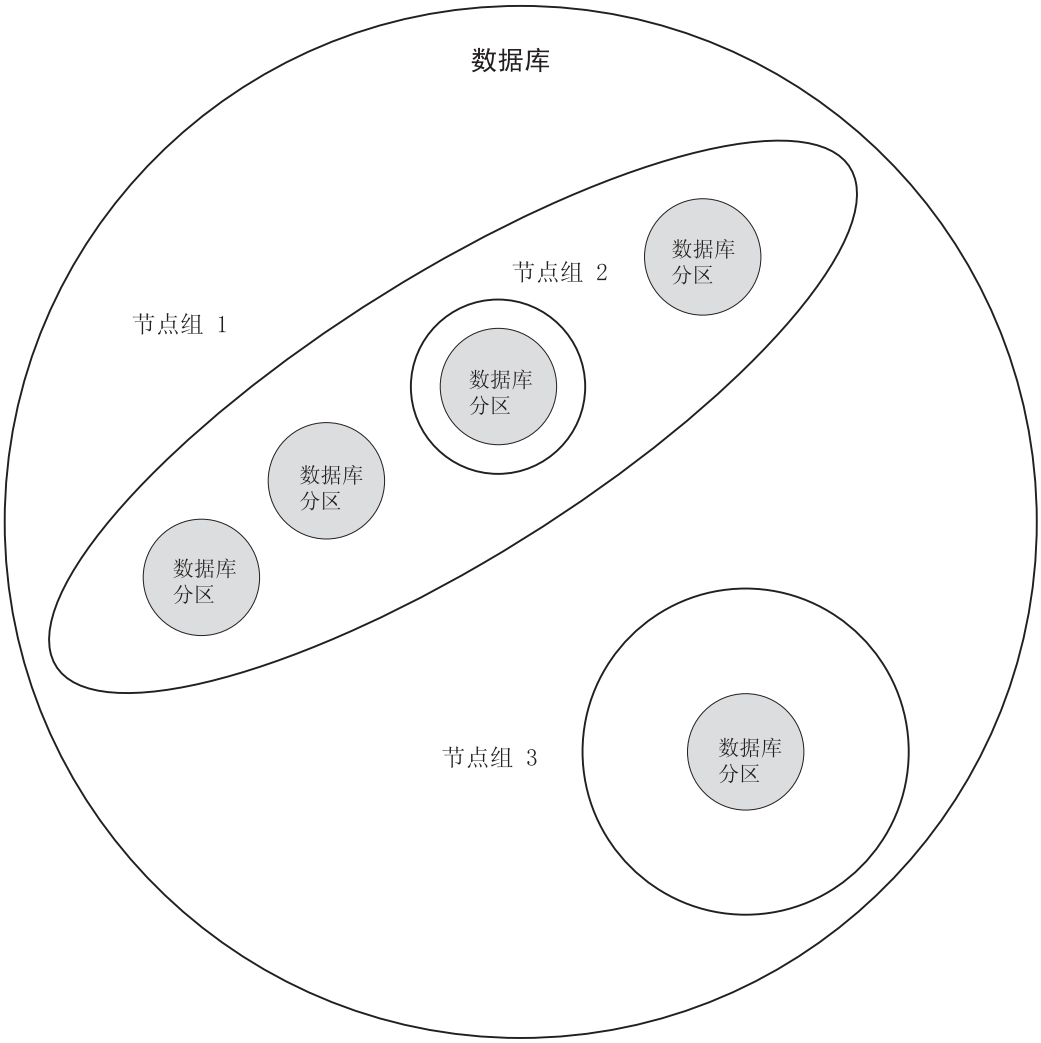


图 12. 数据库中的节点组

在分区数据库系统中使用 Extender 意味着您可:

- 通过在多个分区之间分发数据来减少输入 / 输出和处理瓶颈。
- 通过添加更多机器并在其中再分发数据以增加数据库大小。

并行处理

分区数据库可使用多个 CPU 来满足信息请求。检索和更新请求被自动分配到子请求中, 并在每台机器上的数据库分区服务器上并行执行。

作为分区数据库系统中的处理功能的说明, 假设单分区数据库中有 100,000,000 个记录要扫描。这个扫描将要求单个数据库管理器搜索 100,000,000 个记录。现在假设这些记录平均地分布在 20 个数据库分区服务器上; 每个数据库管理器仅必须扫描 5,000,000 个记录。若每个数据库管理管理都同时并以同一速度扫描, 则完成扫描所需的时间应是单分区系统处理此任务所需时间的 5%。

可放大性

随着数据库大小的增大，可将数据库分区服务器添加至数据库系统以改进性能，此处理称为**放大**数据库系统。

放大数据库时，添加了数据库分区服务器，它又将数据库分区添加至数据库系统中的每个现有数据库。然后，可将新数据库分区分配至该数据库的现有节点组。最后，可在该节点组中重新分配数据以使用新的数据库分区。

在分区数据库环境中使用 DB2 Extender

通过在分区数据库环境中使用 DB2 Extender，可采用尤其支持 LOB 处理的功能部件。因为可将数据库分布在许多机器上，所以可将 LOB 的大储存库（每个的长度可达 2G 字节）存储在一个数据库中。

并且，DB2 Extender 参与“DB2 扩充企业版”管理的 SQL 操作的并行处理。当“DB2 扩充企业版”并行运行查询时，还在个别数据库分区上并行运行查询中的任何 DB2 Extender UDF。

安全性和恢复

为 DB2 数据库中以 BLOB 形式存储的图像、音频和视频对象提供了与传统数字和字符数据相同的安全性和恢复保护。对于元数据表中为这些对象存储的信息来说，情况亦如此。用户必须具有必需的特权才可选择、插入或更新对象。

用户发出 UDF 来从用户表选择、插入、更新或删除对象。要执行请求的操作，UDF 必须能够存取（必要时必须能够更新）存放对象属性信息的管理支持表。若用户对用户表具有适当的特权，则 Extender 允许 UDF 对管理支持表执行这些操作。

某些与 Extender 相关的管理操作需要 DBADM 权限。参见第 217 页的第 14 章，『应用程序编程接口』以了解 DB2 Extender 管理 API 所需的权限。参见第 411 页的第 15 章，『客户机的管理命令』以了解 DB2 Extender 管理命令所需的权限。

当将图像、音频或视频的内容存储在从数据库中引用的文件中时，DB2 保护对象的元数据。该文件必须在 PUBLIC 可以读取（即，所有用户均可读取）的目录中。

可像备份和恢复 DB2 中的其它数据那样备份和恢复 BLOB 和元数据。可使用非 DB2 工具备份和恢复存储在文件中的对象内容。并且，可以使用非 DB 工具来备份和恢复 QBIC 目录和视频索引。有关备份 QBIC 目录的信息，参见页 127。有关备份视频索引的信息，参见页 25。

第 4 章 Extender 如何工作

DB2 Extender 进行大量工作来处理图像、音频和视频数据请求。阐述 Extender 如何工作的好方法是当您使用 Extender 时，检查它们做了些什么工作。本章描述的脚本包括 Image 和 Audio Extender。它讨论了用户执行的操作和 Extender 如何响应。

Extender 脚本

公司的人事部门想在 DB2 AIX 版中创建包括每个雇员的照片的人事数据库。

带有照片的“数据库”： 如图 13 显示， 数据库中的雇员表将包含每个雇员的身份证号码和姓名以及雇员的照片。

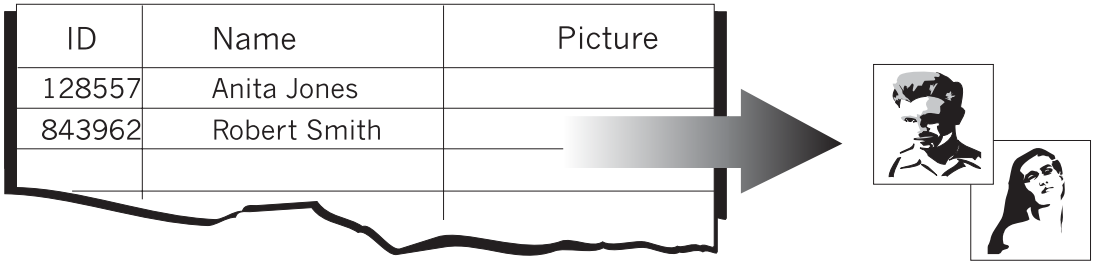


图 13. 雇员表

要准备该人事数据库以进行图像处理，系统管理员（即具有 SYSADM 权限的人员）从启动 Extender 服务开始。然后，系统管理员创建数据库并启用它，以供 Image Extender 使用。

数据库管理员（DBA），或具有等价权限的人员，通过 Image Extender 创建雇员表，然后启用它和雇员照片列。

带有声音的“数据库”： 在为图像处理准备好人事数据库和雇员表之后，人事部决定将每个雇员的录音添加至数据库。这在第 54 页的图 14 中显示。

© Copyright IBM Corp. 1998, 2003

脚本

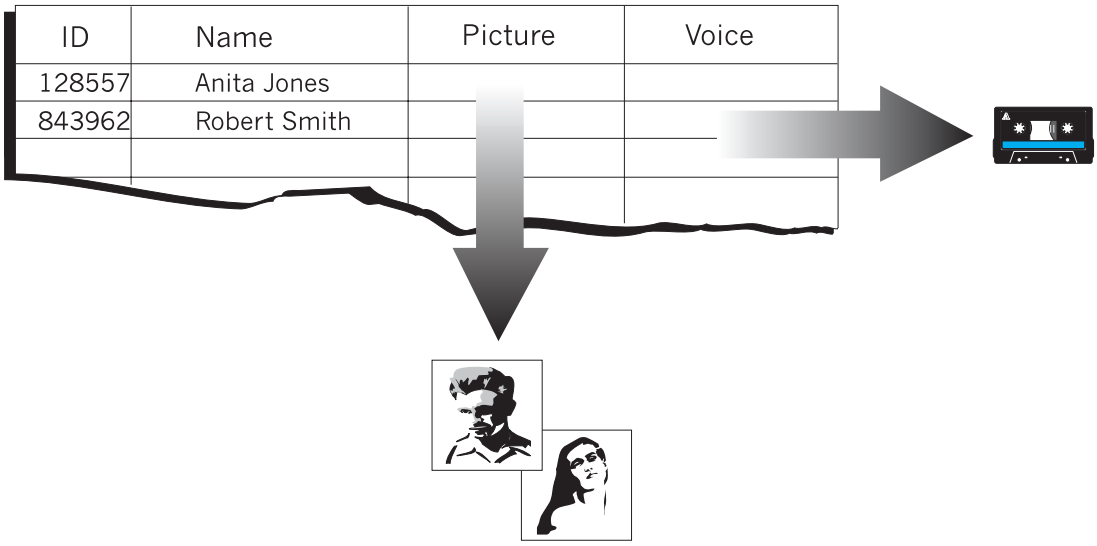


图 14. 添加了 audio 列的 Employee 表

系统管理员通过添加新列改变表，并启用数据库、表和列，以供 Audio Extender 使用。

然后，人事部的用户将数据插入表中、选择和显示表中的数据、更新表中的数据，并从表中删除数据。

启动 Extender 服务

Extender 使用服务器中的服务作为其操作的一部分。若这些服务未作为服务器的正常“启动”操作的功能可用，则系统管理员启动它们。

系统管理员的工作: 系统管理员以 Extender 实例所有者身份注册到 AIX 服务器上。然后，系统管理员在服务器上发出以下命令:

```
DMBSTART
```

发生的情况: 对服务器上的 Extender 实例启动 Extender 服务。DMBSTART 命令还会启动 DB2 实例（若它尚未启动的话）。

准备数据库

系统管理员创建并启用该人事数据库，以供 Image Extender 使用。

系统管理员的工作: 系统管理员使用下列 SQL 语句来在 DB2 AIX 版中创建人事数据库:

```
CREATE DATABASE personnl           /*name of the database*/
       ON /persdb                  /*name of the database directory*/
       WITH "Personnel database"   /*comment*/
```

系统管理员连接数据库并启用它，以供 Image Extender 使用。系统管理员使用 db2ext 命令行处理器来发出下列命令:

```
CONNECT TO personnl
ENABLE DATABASE FOR DB2IMAGE
```

发生的情况: 作为对 ENABLE DATABASE 命令的响应，Image Extender:

- 为图像对象创建名为 DB2IMAGE 的用户定义类型。
- 为图像对象创建管理支持表。
- 为图像对象创建用户定义函数。UDF 列示在表 4 中。

表 4. Image Extender 创建的用户定义函数

UDF 名	描述
Comment	获取或更新用户注释
Content	获取或更新图像的内容
DB2Image	存储图像的内容
Filename	获取包含图像的文件的名称
Format	获取图像格式（例如，GIF）
Height	获取图像的高度，以像素计
Importer	获取图像的导入者的用户标识
ImportTime	获取导入图像时的时间戳记
NumColors	获取图像中使用的颜色数
QbScoreFromName	获取图像的类似性得分（使用命名查询）
QbScoreFromStr	获取图像的类似性得分（使用查询字符串）
QbScoreTBFromName	获取图像列的类似性得分表（使用命名查询）
QbScoreTBFromStr	获取图像列的类似性得分表（使用查询字符串）
Replace	更新图像的内容和用户注释
Size	获取图像的大小，以字节计
Thumbnail	获取缩略图大小的图像版本
Updater	获取图像的更新者的用户标识
UpdateTime	获取更新图像时的时间戳记
Width	获取图像的宽度，以像素计

准备表

DBA 创建 Employee 表，并启用它和 picture 列，以供 Image Extender 使用。

DBA 的工作: 为方便起见，DBA 通过使用以下 SQL 语句来在当前函数路径中添加 mmdbsys 模式:

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
```

这允许指定 UDT 和 UDF 名，而不必在它们前面加上 mmdbsys 模式名作为前缀。（mmdbsys 模式不必是函数路径中的第一个模式。）参见第 45 页的『UDF 和 UDT 名』以获取有关 UDT 和 UDF 名的更多信息。

DBA 创建 employee 表。DBA 使用 DB2 命令行处理器来发出以下 SQL 语句:

```
CREATE TABLE   Employee           /*name of the table*/
       (id      CHAR(6)           /*Employee identification*/
       name    VARCHAR(40)        /*Employee name*/
       picture DB2IMAGE)          /*Employee picture*/
```

然后，DBA 使用 db2ext 命令行处理器来发出下列命令:

```
ENABLE TABLE employee FOR DB2IMAGE
ENABLE COLUMN Employee picture FOR DB2IMAGE
```

发生的情况: 作为对 ENABLE TABLE 命令的响应，Image Extender:

- 标识 employee 表，以供使用。
- 创建存放启用的列中的图像对象的属性信息的管理支持表。

作为对 ENABLE COLUMN 命令的响应，Image Extender:

- 标识 picture 列以供使用。
- 创建触发器。作为对 Employee 表上的插入、更新和删除操作的响应，这些触发器更新各种管理支持表。

改变表

DBA 将 audio 列添加至 Employee 表，并启用它，以供 Audio Extender 使用。

DBA 的工作: DBA 使用 db2ext 命令行处理器来启用人事数据库，以供 Audio Extender 使用:

```
ENABLE DATABASE FOR DB2AUDIO
```

然后，DBA 发出以下 SQL 语句来改变 employee 表。DBA 使用 DB2 命令行处理器来发出 SQL 语句。

```
ALTER TABLE employee              /*name of the table*/
      ADD voice DB2AUDIO          /*employee audio recording*/
```

DBA 使用 db2ext 命令行处理器来启用 Employee 表和 voice 列，以供 Audio Extender 使用:

```
ENABLE TABLE employee FOR DB2AUDIO
ENABLE COLUMN Employee voice FOR DB2AUDIO
```

发生的情况: 作为对 ENABLE DATABASE 命令的响应，Audio Extender:

- 为音频对象创建名为 DB2AUDIO 的用户定义类型。
- 为音频对象创建管理支持表。
- 为音频对象创建用户定义函数。UDF 列示在 表 5中。

表 5. Audio Extender 创建的用户定义函数

UDF 名	描述
AlignValue	获取音频的字节 / 样本值
BitsPerSample	获取用来表示音频的位数
BytesPerSec	获取音频的平均每秒字节数
Comment	获取或更新用户注释
Content	获取或更新音频的内容
DB2Audio	存储音频的内容
Duration	获取音频的播放时间
Filename	获取包含音频的文件的名称
FindInstrument	获取音频中记录特定乐器的声道号
FindTrackName	获取录音中命名声道的声道号

表 5. Audio Extender 创建的用户定义函数 (续)

UDF 名	描述
Format	获取音频格式
GetInstruments	获取音频中记录的乐器的名称
GetTrackNames	获取音频中的声道名
Importer	获取音频的导入者的用户标识
ImportTime	获取导入音频时的时间戳记
NumAudioTracks	获取音频中录制的声道数
NumChannels	获取声道数
Replace	更新录音的内容和用户注释
SamplingRate	获取音频的采样速率
Size	获取音频的大小，以字节计
TicksPerQNote	获取录音时使用的每四分音符时钟滴答数
TicksPerSec	获取录音时使用的每秒时钟滴答数
Updater	获取音频的更新者的用户标识
UpdateTime	获取更新音频时的时间戳记

作为对 ENABLE TABLE 的响应，Audio Extender:

- 标识 employee 表，以供使用。
- 创建存放启用的列中的音频对象的属性信息的管理支持表。

作为对 ENABLE COLUMN 命令的响应，Audio Extender:

- 标识 voice 列以供使用。
- 创建触发器。作为对 Employee 表上的插入、更新和删除操作的响应，这些触发器更新各种管理支持表。

将数据插入表中

用户为 Anita Jones 将一条记录插入 Employee 表中。该记录包括 Anita 的身份证号码（128557）、姓名、照片和录音。源图像和音频内容在服务器上的文件中。图像以 BLOB 形式存储在表中；音频的内容仍在服务器文件中（表条目引用服务器文件）。

用户的工作：用户使用包括语句的应用程序来在 employee 表中插入记录，这些语句显示在第 58 页的图 15 中。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvInt_Stor;
long hvExt_Stor;
   EXEC SQL END DECLARE SECTION;

hvInt_Stor = MMDB_STORAGE_TYPE_INTERNAL;
hvExt_Stor = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                           /*Image Extender UDF*/
           CURRENT SERVER,                  /*database server name in*/
                                            /CURRENT SERVER register*/
           '/Employee/images/ajones.bmp'    /*image source file*/
           'ASIS',                          /*keep the image format*/
           :hvInt_Stor,                     /*store image in DB as BLOB*/
           'Anita''s picture'),             /*comment*/
        DB2AUDIO(                           /*Audio Extender UDF*/
           CURRENT SERVER,                  /*database server name in*/
                                            /*CURRENT SERVER register*/
           '/Employee/Sounds/ajones.wav',   /*audio source file*/
           'WAVE',                          /* audio format */
           :hvExt_Stor,                     /*retain content in server file*/
           'Anita''s voice')                /*comment*/
      );
```

图 15. 将数据插入表

发生的情况 作为对 INSERT 语句中的 DB2Image UDF 的响应，Image Extender:

- 从源图像文件头读图像的属性，如其高度、宽度和颜色数。
- 创建图像的唯一句柄，并在管理支持表中创建记录:
  - 图像的句柄
  - 时间戳记
  - 以字节计的图像大小
  - 注释 “Anita 的照片”
  - 图像的内容

  图像源是名为 ajones.bmp 的服务器文件。将该文件的内容以 BLOB 形式插入管理支持表记录中。存储的图像的格式与源图像相同；不进行格式转换。
- 在管理支持表中存储记录。记录包含特定于图像的属性，如图像中的颜色数，以及缩略图大小的图像版本。

作为对 INSERT 语句中的 DB2Audio UDF 的响应，Audio Extender:

- 从音频文件头中读音频的属性，如声道和声道数。
- 创建音频的唯一句柄
- 在管理支持表中存储记录。记录包含:
  - 音频的句柄
  - 时间戳记
  - 以字节计的音频大小
  - 注释 “Anita 的话音”

  音频内容在名为 ajones.wav 的服务器文件中；管理支持表记录引用此文件。

- 在另一管理支持表中存储记录。该记录包含音频特定属性，如音频的采样速率。

触发器将图像和音频属性数据插入到各种管理支持表中。

选择表中的数据

用户检索关于如何在 Employee 表中存储 Robert Smith 的图像和录音的信息。

用户的工作: 用户使用包括 SQL 语句的应用程序获取信息，这些 SQL 语句显示在图 16 中。

```
EXEC SQL BEGIN DECLARE SECTION;
char[255]  hvImg_Time;
char[255]  hvAud_Time;
   EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT IMPORTTIME(PICTURE),              /*when image was stored*/
       IMPORTTIME(VOICE)                          /*when audio was stored*/
       INTO :hvImg_Time, :hvAud_Time
  FROM EMPLOYEE
       WHERE NAME='Robert Smith';
```

图 16. 选择表中的数据

发生的情况: 作为对 PICTURE 列的 ImportTime UDF 的响应，Image Extender 返回包含存储图像的日期和时间的时间戳记。作为对 VOICE 列的 ImportTime UDF 的响应，Audio Extender 返回包含存储录音的日期和时间的时间戳记。

显示和播放对象

用户显示 Robert Smith 的图像并播放 Robert Smith 的录音。图像以 BLOB 形式存储在 Employee 表中；录音的内容存储在服务器文件中。

用户的工作: 用户使用包括 SQL 语句的应用程序显示图像和播放录音，这些 SQL 语句显示在第 60 页的图 17 中。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvImg_hdl [251];
char hvAud_hdl [251];
   EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT PICTURE,                                  /*Get image handle*/
                VOICE                                     /*Get audio handle*/
       INTO :hvImg_hdl, :hvAud_hdl
  FROM EMPLOYEE
       WHERE NAME='Robert Smith';

rc=DBiBrowse(
        NULL,                                             /*Use default image browser*/
        MMDB_PLAY_HANDLE,                                 /*Use handle*/
        hvImg_hdl,                                        /*Image handle*/
        MMDB_PLAY_NO_WAIT);                                /*Run browser independently*/

rc=DBaPlay(
        NULL,                                             /*Use default audio player*/
        MMDB_PLAY_HANDLE,                                 /*Use handle*/
        hvAud_hdl,                                        /*Audio handle*/
        MMDB_PLAY_WAIT);                                  /*Wait for player to end*/
                                                          /*before continuing*/
```

图 17. 显示和播放对象

发生的情况: DB2 检索 Robert Smith 的图像和录音的句柄。然后，作为对 DBiBrowse API 的响应，Image Extender 获取与检索到的图像句柄相关联的图像内容。Image Extender 从数据库检索图像内容，并将其放到临时客户机文件中，以供图像浏览器显示。NULL 参数指示将使用用户系统的缺省图像浏览器。浏览器将独立于调用程序运行，这意味着调用程序在继续之前，不会等待图像浏览器完成。

作为对 DBaPlay API 的响应，Audio Extender 获取与检索到的音频句柄相关联的音频的文件名，并将该文件名传送至音频播放程序。NULL 参数指示将使用用户系统的缺省音频播放程序。调用程序在继续之前，将等待用户结束音频播放程序。

更新表中的数据

Anita Jones 用最新的照片替换 Employee 表中她的照片。新照片的内容在服务器文件中。

用户的工作: 用户使用包括 SQL 语句的应用程序替换 employee 表中的照片，这些 SQL 语句显示在第 61 页的图 18 中。

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvComment [16385];
     long hvStorageType;
   EXEC SQL END DECLARE SECTION;

strcpy(hvComment, "Picture taken at Anita's promotion");
hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
       SET PICTURE=REPLACE(
              PICTURE,                      /*image handle*/
              '/myimages/newone.bmp',       /*source image content*/
              'BMP',                        /*source format*/
              :hvStorageType,               /*store image in table as BLOB*/
              :hvComment)                   /*replace comment*/
       WHERE NAME='Anita Jones';
```

图 18. 更新表中的数据

发生的情况: 作为对 UPDATE 语句中的 Replace UDF 的响应，Image Extender 读取新图像的属性。Image Extender 使用新图像的属性来更新存储在管理支持表中的旧图像属性。图像源在名为 newone.bmp 的服务器文件中。将文件的内容以 BLOB 形式插入管理支持表记录中，并替换旧图像的 BLOB 内容。

触发器替换各种管理支持表中的图像属性数据。

删除表中的数据

用户从 Employee 表中删除 Anita Jones 的记录。

用户的工作: 用户使用包括下列 SQL 语句的应用程序从 employee 表中删除记录:

```
DELETE FROM EMPLOYEE
     WHERE NAME='Anita Jones';
```

发生的情况: 触发器删除各种管理支持表中 Anita Jones 的条目。

删除数据

第 2 部分 管理图像、音频和视频数据

© Copyright IBM Corp. 1998, 2003

第 5 章 管理概述

本章提供创建使用 DB2 Extender 的应用程序时涉及的管理任务的概述。

DB2 Extender 提供了两种方法来执行大多数管理任务:

- 管理应用程序编程接口（API）。可以在 C 语言程序中包括 DB2 Extender API。参见第 217 页的第 14 章，『应用程序编程接口』以获取关于这些 API 的参考信息。
- 管理命令。可将管理命令提交给 db2ext 命令行处理器。这些命令不在 DB2 命令行上运行。参见第 411 页的第 15 章，『客户机的管理命令』以获取有关输入管理命令的指示信息和其它参考信息。

可以使用 DB2 Extender 执行的管理任务

有五种类别的管理任务:

- 管理 Extender 服务。DB2 Extender 在 DB2 的最上层运行它们自己的服务器。在应用程序可使用 Extender 数据之前，系统管理员启动 Extender 服务，且用户与存放 Extender 数据的数据库连接。
- 准备 Extender 数据的数据对象。准备数据库、表和列以通过启用它们来存放 Extender 数据。启用数据对象时，Extender 创建并维护管理支持表（又称为元数据表）来管理 Extender 数据。
- **仅限于 EEE。**在分区环境中再分发 Extender 数据。在分区数据库中添加或删除分区时，可再分发数据以利用新节点配置。
- 跟踪数据对象和媒体文件。调试使用 DB2 Extender 的应用程序时，了解对 Extender 数据启用了哪些数据对象是很有用的。了解用户表与外部媒体文件之间的相关性也是很有用的。
- 清除管理支持表。使用 DB2 Extender 时，管理支持表中最后可能会累积过时的条目。删除过时的元数据可改进性能并收回存储空间。

第 66 页的表 6 列示了所有涉及管理 Extender 数据的任务。该表指定提供哪些工具来执行每一任务，以及在何处查找更多信息。

在 **Extender API** 列中，x 表示每个 API 语句的第三个字符。视您正在使用的 Extender 的不同，此字符会有所变化:

字符	Extender
a	音频
i	图像
v	视频

例如，启用图像数据表的 API 是 DB**i**EnableTable，启用音频表的 API 是 DB**a**EnableTable，启用视频表的 API 是 DB**v**EnableTable。Extender API 列中的值否表示没有 Extender API 用于该任务。Extender 命令列中的值否 表示没有 Extender 命令用于该任务。

© Copyright IBM Corp. 1998, 2003

QBIC 要求附加的管理：若计划使用 Image Extender 的“按图像内容查询”（QBIC）能力，则需要执行附加的管理任务，如创建 QBIC 目录。关于这些任务的信息，参见第 125 页的第 12 章，『按内容查询图像』。

表 6. DB2 Extender 的管理任务和设施

任务	Extender API	Extender 命令	参见
管理 Extender 服务			
启动 Extender 服务	否	DMBSTART	第 3 页
获取 Extender 服务的状态	否	DMBSTAT	第 5 页
停止 Extender 服务	否	DMBSTOP	第 4 页
连接数据库	否	CONNECT	第 3 页
启动数据库的 Extender 服务	否	START SERVER	第 4 页
获取数据库的 Extender 服务的状态	否	GET SERVER STATUS	第 5 页
停止数据库的 Extender 服务	否	STOP SERVER	第 4 页
创建和管理 Extender 实例	否	DMBICRT, DMBILIST, DMBIDROP, DMBIMIGR	第 5 页
准备多媒体数据的数据对象			
启用数据库	DBxEnableDatabase	ENABLE DATABASE	第 69 页
禁用数据库	DBxDisableDatabase	DISABLE DATABASE	第 73 页
启用表	DBxEnableTable	ENABLE TABLE	第 71 页
禁用表	DBxDisableTable	DISABLE TABLE	第 73 页
启用列	DBxEnableColumn	ENABLE COLUMN	第 72 页
禁用列	DBxDisableColumn	DISABLE COLUMN	第 73 页
在分区环境中再分发 Extender 数据（仅限于 EEE）			
根据新节点组配置再分发 Extender 数据	DMBRedistribute	REDISTRIBUTE NODEGROUP	第 17 页
跟踪数据对象和媒体文件			
找出数据库是否已启用	DBxIsDatabaseEnabled	GET Extender STATUS	第 75 页
找出表是否已启用	DBxIsTableEnabled	GET Extender STATUS	第 75 页
找出列是否已启用	DBxIsColumnEnabled	GET Extender STATUS	第 75 页
查找引用其限定符是当前用户标识的表中的文件的表条目	DBxIsFileReferenced	否	第 76 页
查找引用特定限定符或数据库中所有表的文件的表条目	DBxAdminIsFileReferenced	否	第 76 页
查找其限定符是当前用户标识的表中的表条目引用的文件	DBxGetReferencedFiles	GET REFERENCED FILES	第 76 页
查找特定限定符的所有表或数据库中的所有表中的表条目引用的文件	DBxAdminGetReferencedFiles	GET REFERENCED FILES	第 76 页
查找其限定符是当前用户标识的所有表中的表条目引用的不可存取的文件	DBxGetInaccessibleFiles	GET INACCESSIBLE FILES	第 77 页

表 6. DB2 Extender 的管理任务和设施 (续)

任务	Extender API	Extender 命令	参见
查找特定限定符的所有表或数据库中的所有表中的表条目引用的不可存取的文件	DBxAdminGetInaccessibleFiles	GET INACCESSIBLE FILES	第 77 页
清除管理支持（元数据）表			
清除特定用户表或其限定符是当前用户标识的所有用户表的元数据表	DBxReorgMetadata	REORG	第 7 页
清除带特定限定符的所有用户表或数据库中的所有用户表的元数据表	DBxAdminReorgMetadata	REORG	第 7 页

管理任务的顺序: 以下列表是首次使用 Extender 时执行的管理任务的有序摘要。使用 DB2 命令或语句来执行某些任务。用 DB2 Extender 来执行另一些任务。此顺序假设 DB2 系统正在运行。

必需的任务:

1. 启动 Extender 服务。
2. 创建数据库（通过使用 DB2）。
3. 连接数据库数据库服务器。
4. 启用数据库。
5. 创建表和列（通过使用 DB2）。
6. 启用数据库中的表。
7. 启用表中的列。

可选的任务:

1. 跟踪数据对象和媒体文件。
2. 设置函数路径（使用 DB2）。
3. 清除管理支持表。

示例: 下面五章中的大多数示例假设系统管理员（SYSADM）或数据库管理员（DBA）正在执行任务。很少任务不需要 DBA 或 SYSADM 权限。

这些示例假设 DBA 已在当前函数路径中添加 MMDBSYS 模式。这允许 DBA 指定 UDT 名，而这些名称不必带有 MMDBSYS 模式名作为前缀。有关 UDT 名的更多信息，参见第 45 页的『UDF 和 UDT 名』。

本节中的许多 API 示例是以随 Extender 一起提供的样本应用程序代码为基础的。样本代码在客户机上的 SAMPLES 子目录中。

第 6 章 准备 Extender 数据的数据对象

准备数据库、表和列以通过启用它们来存放 Extender 数据。首先启用数据库。然后启用数据库中的表。最后，启用表中的列。

当不再需要数据对象中的 Extender 数据时，可禁用这些对象。

可通过在 C 语言程序中使用 API 或从 db2ext 命令行启用和禁用这些对象。本章中，提供了每种方法的示例。

启用数据库

使用 DBxEnableDatabase API（其中，x 为 a 时表示音频、为 i 时表示图像，为 v 时表示视频）或使用 ENABLE DATABASE 命令来对 DB2 extender 启用数据库。

启用数据库时，Extender:

- 为数据对象创建名为 DB2xxxxx 的用户定义类型（UDT），其中 xxxxx 是 Image、Audio 或 Video。使用该 UDT 来在用户表中定义存放该类型的对象的句柄的列。
- 创建数据库的管理支持表（也称为元数据表）。这些表不是用户表（用户在其中存储商业数据的表）。Extender 使用它们来管理 Extender 数据。不要手工编辑它们。
- 创建与 Extender 相关联的用户定义函数（UDF）。第 159 页的『用户定义函数』中列示了 UDF。

启用数据库时，还必须指定表空间来存放数据库的管理支持表（及其索引）。指定的一个或多个表空间可以是空值，在此情况下，使用缺省表空间。

您需要 DBA 权限才可启用数据库。

仅限于 EEE: 当启用分区环境中 Extender 的数据库时，应在包括分区数据库系统中的所有节点的节点组中定义指定的表空间。并且，指定的表空间应与用户表位于同一节点组中。

示例

在下列示例中，使用缺省表空间来启用数据库，以存放图像数据。

使用 API: 第 70 页的图 19 中的代码在启用现有的数据库之前，先连接它。此示例是使用 DB2 调用层接口编写的。它包括一些设置和错误检查代码。完整的样本程序在 SAMPLES 子目录中的 ENABLE.C 文件中。

© Copyright IBM Corp. 1998, 2003

```
/*---- Set-up -----------------------------------------------*/
#include <stdio.h>
#include <stlib.h>
#include <string.h>
#include "dmbimage.h"       /* image Extender function prototypes (DBi ) */
#include "utility.h"        /* utility functions */

#define MMDB_ERROR_MSG_TEXT_LEN         1000
#define SERVER_IS_DB2390  (strcmp(dbms, "DB2" ) ==0 || strcmp(dmbs, "DSN06010" ) ==0 )
int
main(int argc, char *argv[] ) {
        SQLHENV henv = SQL_NULL_HENV;
        SQLHDBC hdbc = SQL_NULL_HDBC;
        SQLHSTMT hstmt = SQL_NULL_HSTMT;
SQLCHAR uid[18+1];
SQLCHAR pwd[30+1];
        SQLCHAR dbname[SQL_MAX_DSN_LENGTH+1];
        SQLCHAR buffer[500];
        SQL SMALLINT dbms_sz = 0;
        char    dbms[20];

        SQLRETURN rc = SQL_SUCCESS;
        SQLINTEGER sqlcode = 0;
        char errorMsgText[MMDB_ERROR_MSG_TEXT_LEN+1];
        char *program = "enable;
        char *step;
```

图 19. 启用数据库的样本代码 (1/3)

```
/*---- Prompt for database name, userid, and password ----*/
       if (argc > 5) ||  (argc >=2 && strcmp(argv[1],"?")== 0))
          {
            printf("Syntax for enable - enabling a DB2 UDB database: \n"
                   "   enable database_name userid password\n");
            exit(0);
          }

        if (argc == 4) {
          strcpy((char *)dbname, argv[1]);
          strcpy((char *)uid   , argv[2]);
          strcpy((char *)pwd   , argv[3]);
          }
        else {
          printf("Enter database name:\n");
          gets((char *) dbName);
          printf("Enter userid:\n");
          gets((char *) uid);
          printf("Enter password:\n");
          gets((char *) pwd);
          }
/*----- connect to the database ------------------------------------------------*/
         rc = cliInitialize(&henv, &hdbc, dbname, uid, pwd);
         cliCheckError(henv, hdbc, SQL_NULL_HSTMT, rc);
         if (rc < 0) goto SERROR;

/*----- find out if application is connected to DB2/UDB or DB2/390?------------*/
         rc = SQLGetInfo(hdbc, SQL_DBMS_NAME, (SQLPOINTER) &dbms,
                         sizeof(dbms), &dbms_sz);
         cliCheckError(henv, hdbc, SQL_NULL_HSTMT, rc);
         if (rc < 0) goto SERROR;
```

图 19. 启用数据库的样本代码 (2/3)

```
/****** enable server for image extender ************************************/
         if (!SERVER_IS_DB2390)
           {
            printf("%s: Enabling database......\n", program);
           }
         printf("%s: This may take a few minutes, please wait......\n", program);

         if (!SERVER_IS_DB2390)
           {
            step="DBiEnableDatabase with NULL tablespace"
            rc=DBiEnableDatabase(NULL);
           }
         if (rc < 0) {
            printf ("%s: %s failed!\n", program, step);
            printMsg(rc);
            DBiGetError(&sqlcode, errorMsgText);
            if (sqlcode)
              printf("sqlcode=%i, ",sqlcode);
         }else if (rc > 0) {
            printf("%s: %s, warning detected.\n", program, step);
            printMsg(rc);
            DBiGetError(&sqlcode, errorMsgText);
            printf("warning MsgText=%s\n", errorMsgText);
         } else
            printf("%s: %s OK\n", program, step);
/****** end of enable server *************************************************/
```

图 19. 启用数据库的样本代码 (3/3)

使用 db2ext 命令行: 在此示例中，已连接数据库。

```
enable database for db2image
```

启用表

使用 DBxEnableTable API（其中，x 为 a 时表示音频，为 i 时表示图像，为 v 时表示视频）或 ENABLE TABLE 命令来对 DB2 Extender 启用表。

启用用户表时，还必须指定表空间来存放随它一起的管理支持表（及其索引）。指定的一个或多个表空间可以是空值，在此情况下，使用缺省表空间。

仅限于 EEE: 当在分区环境中对 Extender 启用数据库时，应在包括分区数据库系统中的所有节点的节点组中定义指定的表空间。并且，指定的表空间必须与用户表位于同一节点组中。

仅限于 EEE: 不能将 DB2 Extender 列用作分区数据库环境中的分区键列。

您需要“控制”或“改变”用户表的权限。在启用数据库中的表之前，必须先启用该数据库。

在下列示例中，使用缺省表空间来启用表，以存放图像数据。数据库已启用。

使用 API: 在第 72 页的图 20中，在启用表之前，代码创建表并落实更改。该示例包括一些错误检查代码。完整的样本程序在 SAMPLES 子目录中的 ENABLE.C 文件中。

```
        SQLCHAR szCreate_DB2UDB[]="CREATE TABLE %s(%s mmdbsys.DB2Image,
    %s mmdbsys.DB2Video, %s mmdbsys.DB2Audio, artist varchar(25), title varchar(25)
        stock_no char(11 ) , tw char(10 ) , price char(10 ) ) ";

        SQLRETURN rc = SQL_SUCCESS;
        SQLINTEGER sqlcode = 0;
        char errorMsgText[MMDB_ERROR_MSG_TEXT_LEN+1];
        char tableName[8+18+1] = "sobay_catalog";
        char audioColumn[18+1] = "music";
        char imageColumn[18+1] = "covers";
        char videoColumn[18+1] = "video";
        char *program = "enable";
        char *step;

/*-----create table ------------------------------------------------*/
       printf("%s: Creating table ......\n", program);
       if (!SERVER_IS_DB2390)
         sprintf((char*) buffer, (char*) szCreate_DB2UDB,
                   tableName, imageColumn, videoColumn, audioColumn):

       rc = SQLAllocStmt(hdbc, &hstmt);
       cliCheckError(SQL_NULL_HENV, hdbc, SQL_NULL_HSTMT, rc);
       rc = SQLExecDirect(hstmt, buffer, SQL_NTS);
       cliCheckError(SQL_NULL_HENV, SQL_NULL_HDBC, hstmt, rc);

/*---- enable table for image extender ---------------------------------*/
     printf("%s: Enabling table......\n", program);
     step="DBiEnableTable";
     if (!SERVER_IS_DB2390)
         rc = DBiEnableTable(NULL,tableName);
           }
         if (rc < 0) {
                printf("%s: %s failed!\n", program, step);
                printMsg(rc);
                DBiGetError(&sqlcode, errorMsgText);
                if (sqlcode)
                  printf("sqlcode=%i, "sqlcode");
                printf("errorMsgText=%s\n", errorMsgText);
         } else if (rc > 0) {
                printf("%s: %s, warning detected.\n", program, step);
                printMsg(rc);
                DBiGetError(&sqlcode, errorMsgText);
                printf("warningMsgText=%s\n", errorMsgText);
         } else
                printf("%s: %s OK\n", program, step)
/*---- end of enable table ---------------------------------------------*/
```

图 20. 启用表的样本代码

使用 db2ext 命令行: 在此示例中，表已存在，并且已启用数据库。

```
enable table employee for db2image
```

启用列

使用 DBxEnableColumn API（其中，x 为 a 时表示音频，为 i 时表示图像，为 v 时表示视频）或 ENABLE COLUMN 命令来对 DB2 Extender 启用列。发出 API 或命令时，您指定适当的表和列。

启用列时，Extender 将信息添加至属于用户表的管理支持表。您需要对该列所在的用户表的“控制”或“改变”权限。在启用列之前，数据库和表必须都启用。

在下列示例中，启用 EMPLOYEE 表中的 PICTURE 列来存放图像数据。数据库和表已启用。

使用 API：此示例包括一些错误检查代码。完整的样本程序在 SAMPLES 子目录中的 ENABLE.C 文件中。

```
        char imageColumn[18+1] = "covers";

/*---- enable column for image extender ----*/
     printf("%s: Enabling columns......\n", program);
     step="DBiEnableColumn";
     rc = DBiEnableColumn(tableName, imageColumn);
     if (rc < 0) {
            printf("%s: %s failed!\n", program, step);
            printMsg(rc);
           DBiGetError(&sqlcode, errorMsgText);
           if (sqlcode)
              printf("sqlcode=%i, ", sqlcode);
           printf("errorMsgText=%s\n", errorMsgText)

    } else if (rc > 0) {
           printf("%s: %s, warning detected.\n", program, step);
           printMsg(rc);
           DBiGetError(&sqlcode, errorMsgText);
           printf("warningMsgText=%s\n", errorMsgText);
    } else
           printf("%s: %s OK\n", program, step);
/*---- enable column for image extender ----*/
```

图 21. 启用列的样本代码

使用 db2ext 命令行：在此示例中，列已存在，数据库和表已被启用。

```
enable column employee picture for db2image
```

禁用数据对象

若从数据库、表或列中除去 Extender 数据，则不再需要启用它。您有两种方法来禁用数据对象：DISABLE 命令和 API。有关 Extender 命令的更多信息，参见第 411 页的第 15 章，『客户机的管理命令』。有关 Extender API 的更多信息，参见第 217 页的第 14 章，『应用程序编程接口』。

在删除包含 Extender 数据的表或数据库之前，禁用它，并停止该数据库的服务器。

禁用

第 7 章 跟踪数据对象和媒体文件

当创建和调试使用 DB2 Extender 的应用程序时，了解对 Extender 数据启用了哪些数据对象是很有用的。例如，若能确定图像数据启用了某一表，应用程序可成功地将图像文件存储在该表中。

了解用户表和外部媒体文件之间的关系（例如，哪些表引用某个文件或哪些文件由特定表引用）也是很有用的。发现表是否引用不再存在于系统上的文件也很有用。

您需要适当的特权：您需要具有对表的存取权才能跟踪该表中的数据。若要执行全面的跟踪操作，如查找数据库中所有用户表中哪些条目引用文件，您需要 SYSADM 权限、DBADM 权限，或对搜索的所有用户表和关联管理支持表中启用的列的 SELECT 特权。若您并非对所有表都具有存取权，则 Extender 将仅对您可以存取的那些表返回跟踪信息。它们还将返回一个代码，指示您对某些必需的表不具有存取权。

检查数据对象的状态

您可以检查是否启用了数据库、表和列来存放 Extender 数据。以下示例确定是否对 Image Extender 启用当前数据库。已连接数据库。完整的样本程序在 SAMPLES 子目录中的 API.C 文件中。

使用 API: 在 图 22中的样本代码包括一些错误检查代码。

```
/*---- Query the database using DBiIsDatabaseEnabled API. ----------*/
step="DBiIsDatabaseEnabled API";
rc = DBiIsDatabaseEnabled(&status);
if (rc < 0) {
     printf("%s: %s FAILED!\n", argv[0], step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
     fail = TRUE;
} else if (rc > 0) {
     printf("%s: %s, warning detected.\n", argv[0], step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else {
     if (status == 1) {
         printf("%s: \"%s\" database is enabled for Image Extender\n",
                argv[0], dbName);
         printf("%s: %s PASSED\n\n", argv[0], step);
     } else if (status == 0) {
         printf("%s: \"%s\" database is not enabled for Image Extender\n",
                argv[0], dbName);
         printf("%s: %s PASSED\n\n", argv[0], step);
     } else
         printf("%s: %s FAILED, invalid status!\n", argv[0], step);
}
```

图 22. 检查数据库是否已启用的样本代码

使用 db2ext 命令行:

```
get Extender status
```

© Copyright IBM Corp. 1998, 2003

检查启用情况

检查用户表和列的状态类似于检查数据库的状态。使用 DBxIsTableEnabled 和 DBxIsColumnEnabled API，或 GET EXTENDER STATUS 命令。

查找引用文件的表条目

您可以检查用户表中的哪些条目引用外部媒体文件。使用 DBxAdminIsFileReferenced API 来检查当前数据库中所有或者部分用户表的哪些条目引用外部媒体文件。使用 DBxIsFileReferenced API 来检查特定用户表中的哪些条目引用外部媒体文件。

使用 API: 图 23 中的样本代码返回引用文件的次数以及引用它的位置。它包括一些错误检查代码。完整的样本程序在 SAMPLES 子目录中的 API.C 文件中。

```
/*---- Query the database using DBiAdminIsFileReferenced API. ------*/
step="DBiAdminIsFileReferenced API";
rc = DBiAdminIsFileReferenced((char*) uid, filename, &count, &filelist);
if (rc < 0) {
   printf("%s: %s FAILED!\n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else if (rc > 0) {
   printf("%s: %s, warning detected.\n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsg Text);
} else {
   if (count == 0)
          printf("%s: \"%s\" file is not referenced\n",
                 program, filename);
     else {
          printf("%s: \"%s\" file is referenced %d times\n",
                 program, filename);
          for (i=0; i < count; i++)
             {
                 /* filename is NULL for any IsFileReferenced APIs */

                  printf ("filename = %s\n", filelist[i].filename);
   printf ("\tqualifier = %s\n", filelist[i].tqualifier);
   printf ("\ttable = %s\n", filelist[i].tname);
   printf ("\thandle = %s\n", filelist[i].handle);
   printf ("\tcolumn = %s\n", filelist[i].column);
                  if (filelist[i].filename)
                          free (filelist[i].filename);
             }
          }
          if (filelist)
                free (filelist);
          printf("%s: %s PASSED\n\n", argv[0], step);
}
```

图 23. 用于检查用户表是否引用某个文件的样本代码

查找表条目引用的文件

使用 DBxAdminGetReferencedFiles API 或 GET REFERENCED FILES 命令来列示当前数据库中的所有用户表或这些表的一个子集引用的外部媒体文件。使用 DBxGetReferencedFiles API 或 GET REFERENCED FILES 命令来列示特定表中引用的外部媒体文件。

使用 API: 图 24 中的样本代码返回它找到的文件数和文件的列表。完整的样本程序在 SAMPLES 子目录中的 API.C 文件中。

```
/*---- Query the database using DBiAdminGetReferencedFiles API. ----------*/
step="DBiAdminGetReferencedFilesAPI"
rc = DBiAdminGetReferencedFiles((char*) uid, &count, &filelist);
if (rc < 0) {
   printf("%s: %s FAILED!\n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
   printf{"sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else if (rc > 0) {
   printf("%s: %s, warning detected.\n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
   printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else {
   if (count == 0)
     printf("%s: no referenced files\n", program);
  else {
     printf("%s: %d referenced files\n", program, count);
     for (i=0; i < count; i++)
       {
   printf ("filename = %s\n", filelist[i].filename);
   printf ("\tqualifier = %s\n", filelist[i].tqualifier);
   printf ("\ttable = %s\n", filelist[i].tname);
   printf ("\thandle = %s\n", filelist[i].handle);
   printf ("\tcolumn = %s\n", filelist[i].column);
       if (filelist[i].filename)
                free (filelist[i].filename);
     }
   }
   if (filelist)
       free (filelist);
   printf("%s: %s PASSED\n\n", argv[0], step);
}
```

图 24. 获取引用文件的列表的样本代码

使用 db2ext 命令行:

```
get referenced files user anitas for db2image
```

检查媒体文件是否存在

假设某人从系统中删除了某个媒体文件，但未更新引用它的用户表。您可能想列示用户表引用的所有不可存取的媒体文件。

使用 DBxAdminGetInaccessibleFiles API 或 GET INACCESSIBLE FILES 命令来列示当前数据库中的所有用户表或这些表的一个子集所引用的不可存取的媒体文件。使用 DBxGetInaccessibleFiles API 或 GET INACCESSIBLE FILES 命令来列示特定表引用的不可存取的媒体文件。

检查不可存取的媒体

第 8 章 授予和取消对管理支持表的特权

用户发出 UDF 来对用户表选择、插入、更新或删除图像、音频和视频对象。要执行请求的操作，UDF 必须能够存取存放对象属性信息的管理支持表，必要时还必须能够在该表中进行插入、更新和删除操作。对于用户表的所有者，Extender 自动将处理请求的操作所需的存取权给予 UDF。然而，必须将对管理支持表的选择特权授予需要从用户表中选择对象的除表所有者之外的用户。

另外，对用户表中的图像对象执行 QBIC 操作的用户必须对组成那些对象的 QBIC 目录的管理支持表具有适当的特权。例如，对图像列发出 QBIC 查询的用户必须对该图像列的 QBIC 目录表具有 SELECT 特权。更改 QBIC 目录的用户应对关联的 QBIC 目录表具有 SELECT、INSERT、UPDATE 和 DELETE 特权。

用户表的所有者或数据库的 DBA（具有 GRANT 特权）可以使用 DB2 Extender 命令 GRANT 来授予对管理支持表的特权。发出 GRANT 命令时，您指定:

- 必需的特权，例如 SELECT 或 UPDATE。
- Extender 的名称: DB2IMAGE、DB2AUDIO 或 DB2VIDEO。还可以对全部三种 Extender 指定 ALL。
- 用户表的名称。
- 用户的标识。可以将可选关键字 USER 放在用户标识前面。还可以对所有用户指定 PUBLIC。

若指定 SELECT，则将对与用户表相关联的命名 extender 的管理支持表的 SELECT 特权授予指定的用户。若指定 DB2IMAGE，则还授予对与用户表相关联的 QBIC 目录的管理支持表的 SELECT 特权。例如，以下命令授予对与 employee 表相关联的 Image Extender 的管理支持表的 SELECT 特权。此特权授予用户标识 ajones。此命令还将对与 employee 表相关联的 QBIC 目录的 SELECT 特权授予用户标识 ajones:

```
grant select for db2image on employee to ajones
```

以下命令授予对与 employee 表相关联的 Image Extender、Audio Extender 和 Video Extender 的管理支持表的 SELECT 特权。此特权授予所有用户。此命令还将对与 employee 表相关联的 QBIC 目录的 SELECT 特权授予所有用户:

```
grant select for all on employee to public
```

对于插入、更新或删除操作，Extender 进行检查，以确定用户是否对用户表具有所需的 INSERT、UPDATE 或 DELETE 特权。若用户具有必需的特权，则 Extender 允许 UDF 按需要存取管理支持表。

要授予对 “QBIC 目录” 表的 INSERT、UPDATE 和 DELETE 特权，在 GRANT 命令中指定 UPDATE 和 DB2Image。例如，以下命令将对与 employee 表相关联的 “QBIC 目录” 表的 INSERT、UPDATE 和 DELETE 特权授予用户标识 ajones:

```
grant update for db2image on employee to user ajones
```

当用户不再适于存取用户表中的对象时，用户表的所有者或数据库的 DBA（具有 GRANT 特权）可以取消用户对管理支持表的 SELECT 特权。这还包括组成 QBIC 目录的管理支持表。使用 DB2 Extender 命令 REVOKE 来取消对管理支持表和 QBIC 目录表的特权。REVOKE 命令的格式类似于 GRANT 命令。例如，以下命令取消对与

© Copyright IBM Corp. 1998, 2003

employee 表相关联的 Image extender 的管理支持表的 SELECT 特权。用户标识 ajones 的特权被取消。此命令还取消对与 employee 表相关联的 “QBIC 目录” 表的 SELECT 特权:

```
revoke select for db2image on employee from ajones
```

您还可以取消对组成 QBIC 目录的管理支持表的 INSERT、UPDATE 和 DELETE 特权。在 REVOKE 命令上使用 UPDATE 参数。例如，以下命令取消对与 employee 表相关联的 “QBIC 目录” 表的 INSERT、UPDATE 和 DELETE 特权。用户标识 ajones 的特权被取消。

```
revoke update for db2image on employee from ajones
```

在添加所有特征之后授予对 QBIC 目录的特权: 所授予的对组成 QBIC 目录的管理支持表的特权包括对 QBIC 特征表的特权，但仅仅是对已添加至目录的特征的特权。若在授予对目录的特权之后将特征添加至目录，则将必须再次授予对目录的特权。因此，只应该在创建 QBIC 目录并已添加所有特征之后才授予对该目录的特权。

第 3 部分 图像、音频和视频数据的编程

© Copyright IBM Corp. 1998, 2003

第 9 章 编程概述

本章概述了 DB2 Extender 的编程。它提供在您开始 Extender 的编程之前所需的信息，并提供一个样本应用程序，它演示了如何对 Extender 进行编码。

使用 Extender UDF 和 API

DB2 Extender 提供了用户定义函数来存储、存取和处理数据库中的图像、音频和视频数据。通过请求 SQL 内部函数的同样方式，在应用程序中使用 SQL 语句来编码对这些 UDF 的请求。与内部函数类似，UDF 是在数据库服务器中运行的。

C 应用程序中的下列 SQL 语句请求一个名为 DB2Image 的 Image Extender UDF 以便将图像存储在数据库表中；源图像的内容位于服务器文件中:

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*/
           CURRENT SERVER,                 /*database */
           '/Employee/images/ajones.bmp', /*image content*/
           'ASIS',                          /*keep the image format*/
           :hvStorageType,                 /*store image in DB as BLOB*
          'Anita''s picture')              /*comment*/
       );
```

您使用 Extender 应用程序编程接口来显示图像并播放音频或视频对象。通过使用 C 语言中的客户机函数调用来编码这些 API。这些函数在数据库客户机工作站中运行。

下列 C 语句包括名为 DBiBrowse 的 API。此 API 检索图像句柄的数据并启动浏览器来显示图像。

```
EXEC SQL BEGIN DECLARE SECTION;
char hvImg_hdl [251];
EXEC SQL END DECLARE SECTION

EXEC SQL SELECT PICTURE INTO :hvImg_hdl
      WHERE NAME='Robert Smith';

rc=DBiBrowse(
             "ib %s",                /*image browser*/
             MMDB_PLAY_HANDLE,       /*use image handle*/
             hvImg_hdl,              /*image handle*/
             MMDB_PLAY_NO_WAIT);     /*run browser independently*/
```

UDF 必须在该实例的用户标识下运行: DB2 Extender UDF 必须在与 DB2 Extender 实例相同的用户标识下运行。另外，若要创建 DB2 Extender 实例或使用现有的 DB2 Extender 实例，UDF 必须在与 DB2 实例相同的用户标识下运行。

必须正确地配置 DB2: 必须正确地配置 DB2 才能确保 DB2 Extender 正确操作，特别是确保 DB2 Extender UDF 正确操作。尤其是，必须正确地设置 APP_CTL_HEAP_SZ 数据库配置参数。

© Copyright IBM Corp. 1998, 2003

可使用 Extender UDF 和 API 来执行的任务

表 7 列示了可使用 Extender UDF 和 API 来执行的任务，并显示各任务的描述位置。

表 7. 可使用 DB2 Extender API 来执行的任务

任务	参见
存储图像、音频或视频对象	页 95
检索图像、音频或视频对象	页 104
检索和使用图像、音频和视频属性	页 108
检索与图像、音频或视频相关联的注释	页 110
更新图像、音频或视频对象	页 110
显示图像对象	页 121
显示缩略图大小的图像或视频帧	页 123
播放音频或视频对象	页 124
按内容查询图像	页 125
检测视频画面切换	页 17

Extender 示例的样本表

在本章的整个章节中，您将看到很多使用 DB2 Extender 的编程示例。这些示例都假设您已创建一个名为 EMPLOYEE 并包含人员信息的数据库表。该表包括雇员的标识和姓名的列。根据 extender 的不同，该表还包括雇员照片、语音祝辞和视频剪辑的列。

第 85 页的图 25 说明了 Employee 表的结构，并显示了用于创建该表的 SQL 语句。

```
CREATE TABLE  employee(
    id          CHAR(6),
    name        VARCHAR(40),
    picture     DB2Image,

    sound       DB2Audio,

    video       DB2Video
    );
```

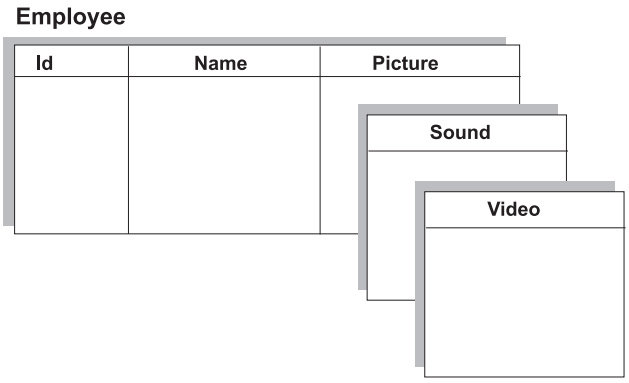


图 25. DB2 Extender 编程示例中使用的表

在开始 DB2 Extender 的编程之前

在开发使用 DB2 Extender 的程序之前，您应熟悉 *DB2 Application Development Guide* 中描述的 DB2 应用程序开发过程和编程技术。开发使用 DB2 Extender 的程序的流程基本上与开发传统 DB2 应用程序的流程相同。

由于 Extender 定义了新的数据结构和函数，因此应用程序代码将与传统 DB2 应用程序不同。例如，第 86 页的图 26 显示了一个用 C 语言编码的应用程序，它使用 Image Extender 来识别存储在数据库表中的 GIF 图像。在识别出图像之后，该程序调用图像浏览器加以显示。

正如示例中所示，使用 DB2 Extender 的应用程序需要执行下列功能:

1 包括 Extender 定义。示例中的 `dmbimage.h` 文件是 Image Extender 的包含（头）文件。该包含文件为 Extender 定义常量、变量和函数原型。

2 定义包含 UDF 的输入或输出或 API 调用的输入所必需的主变量。在该示例中，`hvFormat`、`hvSize`、`hvWidth`、 `hvHeight` 和 `hvComment` 是用于包含 Image Extender UDF 检索到的数据的主变量。主变量 `hvImg_hdl` 用于包含被指定为 Image Extender API 调用的输入的图像句柄。

3 指定必需的 UDF 请求。在示例中，`SIZE`、`WIDTH`、`HEIGHT`、`COMMENT` 和 `FORMAT` 都是 Image Extender UDF。

4 指定必需的 API 调用。在该示例中，`DBiBrowse` 是对某个本地 C 函数的 API 调用，该函数显示从表中检索到它的句柄的图像。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlcodes.h>
#include <dmbimage.h>  1

int count=0;

long
main(int argc,char *argv[])
{
EXEC SQL BEGIN DECLARE SECTION;  2
         char hvImg_hdl[251];                  /* image handle */
         char hvDBName[19];                    /* database name */
         char hvName[40];                      /* employee name */
         char hvFormat[9];                     /* image format */
         long hvSize;                          /* image size */
         long hvWidth;                         /* image width */
         long hvHeight;                        /* image height */
         struct  {        short len;
                  char data[32700]
         } hvComment;                          /* comment about the image */
 EXEC SQL END DECLARE SECTION;

/*  Connect to database  */
strcpy(hvDBName, argv[1]);                     /* copy the database name */

EXEC SQL CONNECT TO :hvDBName IN SHARE MODE;
/*
 * Set current function path*/
EXEC SQL SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH;
```

图 26. 使用 DB2 Extender 的应用程序 (1/2)

```
/*
 * Select (query) using Image Extender UDF
 *
 * The SQL statement below finds all images in GIF format.
 */
EXEC SQL DECLARE c1 CURSOR FOR
         SELECT PICTURE, NAME,                3
                SIZE(PICTURE), WIDTH(PICTURE),
                HEIGHT(PICTURE), COMMENT(PICTURE)
  FROM EMPLOYEE
         WHERE PICTURE IS NOT NULL AND
               FORMAT(PICTURE) LIKE 'GIF%'
FOR FETCH ONLY;

EXEC SQL OPEN c1;
for (;;) {
         EXEC SQL FETCH c1 INTO :hvImg_hdl, :hvName, :hvSize,
                                :hvWidth, :hvHeight, :hvComment;         if (SQLCODE != 0)
                   break;

         printf("\nRecord %d:\n", ++count);
         printf("Employee name = '%s'\n", hvName);
         printf("image size = %d bytes, width=%d, height=%d\n",
                                              hvSize, hvWidth, hvHeight);
         hvComment.data[Comment.len]='\0';         printf("comment len = %d\n", hvComment.len);
 printf("comment = %s\n", hvComment.data);
/*
* The API call below displays the images
*/
 4  rc=DBiBrowse ("ib %s",MMDB_PLAY_HANDLE,hvImg_hdl,
                          MMDB_PLAY_WAIT);
}

EXEC SQL CLOSE c1;

/* end of program */
```

图 26. 使用 DB2 Extender 的应用程序 (2/2)

包括 Extender 定义

对于使用的每个 Extender，在应用程序中都需要一个包含（头）文件。每个包含文件定义 Extender 所使用的常量、变量和函数原型。包含文件的名称是:

包含文件	Extender
dmbimage.h	图像
dmbqbapi.h	图像（按图像内容查询）
dmbaudio.h	音频
dmbvideo.h	视频
dmbshot.h	视频（画面切换检测）

用 #include 伪指令将包含文件引入 C 程序中。例如，以下伪指令将引入 Image Extender 的包含文件:

```
#include <dmbimage.h>
```

开始之前

指定 UDF 和 UDT 名称

DB2 Extender UDF 的全名是 mmdbsys.function-name。DB2 Extender UDT 的全名是 mmdbsys.type-name，其中 mmdbsys 是函数或单值类型的模式名。例如，Content UDF 的全名是 mmdbsys.Content；由 Image Extender 创建的 DB2Image 数据类型的全名是 mmdbsys.DB2Image。可省略 mmdbsys 模式名，前提是先前已将当前函数路径设置为 mmdbsys，例如:

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH

SET CURRENT PATH = mmdbsys, CURRENT PATH
```

传送大对象

可用各种方式在应用程序与 DB2 数据库之间传送大对象，如图像、音频剪辑和视频剪辑。使用的方法取决于是将对象传送至文件或内存缓冲区还是从文件或内存缓冲区传送对象。使用的方法还取决于文件是在客户机中还是在数据库服务器中。

若是在表和服务器文件之间传送对象

当在数据库表和服务器文件之间传送对象时，在适当的 Extender UDF 请求中指定文件路径。因为 Extender UDF 和文件都在服务器上，所以 Extender 将能够找到该文件。例如，在以下 SQL 语句中，将其内容在服务器文件中的图像存储在数据库表中:

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2Image(
         CURRENT SERVER,
          '/Employee/images/ajones.bmp',
                  'ASIS',
         :hvStorageType,
          'Anita''s picture')
       );
```

若将对象传送至或传送自客户机缓冲区

Extender 不能 直接存取内存缓冲区。如果要将对象传送至或传送自客户机上的缓冲区，不能使用指定缓冲区位置这种方法，而需要另一种方法。将对象传送至或传送自缓冲区的一种方法是通过主变量。这是在应用程序与 DB2 数据库之间传送对象的常用方法。

通过对传统的字符和数字对象定义和使用主变量的方法，以对大对象定义和使用主变量。在 DECLARE 节中声明主变量，对它们赋值以便传送，或存取传送给它们的值。

当声明用于图像、音频或视频数据的主变量时，指定数据类型 BLOB。当使用 UDF 来存储、检索或更新对象时，在 UDF 请求中指定适当的主变量作为自变量. 对 SQL 语句中指定的其它主变量使用相同的格式。

例如，下列 SQL 语句声明并使用名为 hvaudio 的主变量来将音频剪辑传送至数据库:

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) hvaudio;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
```

```
'Anita Jones',
DB2Audio(
   CURRENT SERVER,
    :hvaudio,
   'WAVE',
    CAST(NULL as LONG VARCHAR),          'Anita''s voice')
 );
```

使用 LOB 定位器

诸如音频和视频剪辑之类的大对象可能非常大，使用主变量可能不是最有效率的处理方法。**LOB 定位器**可能是在应用程序中处理 LOB 的更好的方法。

LOB 定位器是存储在主变量中的小（4 个字节）值，您的程序可使用它来引用 DB2 数据库中大得多的 LOB。通过使用 LOB 定位器，您的程序可处理 LOB，就象该 LOB 存储在常规主变量中一样。不同之处在于：无需在数据库服务器与客户机上的应用程序之间传送 LOB。例如，当您选择数据库表中的 LOB 时，LOB 保留在服务器上，而 LOB 定位器将它移至客户机。

在 DECLARE 节中声明 LOB 定位器，并按使用主变量的方式使用它。当声明用于图像、音频或视频数据的 LOB 定位器时，指定数据类型 BLOB_LOCATOR。例如，下列 SQL 语句声明和使用名为 video_loc 的 LOB 定位器，以从数据库表中检索视频剪辑：

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR video_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(VIDEO)
     INTO :video_loc
     FROM EMPLOYEE
     WHERE NAME='Anita Jones';
```

UDF 使用 LOB 定位器：存储、检索和更新图像、音频和视频对象的 DB2 Extender UDF 使用 LOB 定位器。DB2 Extender V1 中的这些 UDF 不使用 LOB 定位器，因为这一点，所以无法处理大于 2 MB 的对象。此限制强制用户分段传送大于 2 MB 的对象。因为这些 UDF 现在使用 LOB 定位器，所以消除了 2 MB 的限制。

若将对象传送至或传送自客户机文件

使用文件引用变量来将对象传入或传出客户机上的文件。如果使用文件引用变量，您就不必在应用程序中为大对象分配缓冲区空间。当将文件引用变量与 UDF 配合使用时，DB2 在文件和 UDF 之间直接传送 BLOB 内容。

在 DECLARE 节中声明文件引用变量，并按使用主变量的方式使用它。在声明用于图像、音频或视频数据的文件引用变量时，指定数据类型 BLOB_FILE。但是，与包含对象的内容的主变量不同，文件引用变量包含文件的名称。文件的大小不能超过为 UDF 定义的 BLOB 大小。

对于如何将文件引用变量用于输入和输出，您有各种选择。通过在您的程序中的文件引用变量结构中设置 FILE_OPTIONS 字段来选择所要的选项。可从下列选项中进行选择：

用于输入的选项：

SQL_FILE_READ。可打开、读和关闭此文件。文件中数据的长度（以字节计）是在打开文件时确定的。文件引用变量结构的 data_length 字段存放文件的长度（以字节计）。

用于输出的选项:

SQL_FILE_CREATE。此选项创建新文件（如果该文件不存在的话）。若文件已存在，则返回错误消息。文件引用变量结构的 data_length 字段存放文件的长度（以字节计）。

SQL_FILE_OVERWRITE。此选项创建新文件（如果该文件不存在的话）。若文件已存在，则新数据将覆盖文件中的数据。文件引用变量结构的 data_length 字段存放文件的长度（以字节计）。

SQL_FILE_APPEND。此选项将输出追加到文件后（如果该文件已存在的话）。如果文件不存在，此选项将创建新文件。文件引用变量结构的 data_length 字段存放添加至文件的数据的长度（以字节计），而不是文件的总长度。

例如，下列语句声明名为 Img_file 的文件引用变量，并使用它将内容在客户机文件中的图像存储到数据库表中。注意 FILE_OPTIONS 字段中的 SQL_FILE_READ 赋值:

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Img_file;
  EXEC SQL END DECLARE SECTION;

strcpy (Img_file.name,"/Employee/images/ajones.bmp");
Img_file.name_length=strlen(Img_file.name);
Img_file.file_options=SQL_FILE_READ;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2Image(
         CURRENT SERVER,
          :Img_file,
                'ASIS',
          CAST(NULL as LONG VARCHAR),          'Anita''s picture')
       );
```

在传送对象时指定文件名

DB2 Extender 提供了存储、检索或更新对象时如何指定文件名的灵活性。

虽然可对存储、检索和更新操作指定全限定文件名（即，后跟文件名的完整路径），但最好指定相对文件名。在 AIX、HP-UX 和 Solaris 中，相对文件名是任何不以斜杠开始的文件名。在 Windows 中，相对文件名是任何不以后跟冒号和反斜杠的驱动器盘符开始的文件名。

若指定相对文件名，则 Extender 将使用各种客户机和服务器环境变量中的目录规范作为搜索路径来解析文件名。全路径名由前导部分（通常与安装点相关）和结尾路径名（它唯一标识所需的文件）组成。结尾路径名在 UDF 中指定。环境变量提供一个前导路径名列表，尝试解析相对文件名时，将搜索这些路径名。有关 DB2 Extender 用来解析文件名的环境变量的信息，参见第 475 页的附录 A, 『设置 DB2 Extender 的环境变量』。

Extender 还在适当的时候转换文件名格式。当将文件名传送至服务器时，它被转换为适合于服务器的操作系统的格式。

处理返回码

程序中的所有嵌入式 SQL 语句或 DB2 CLI 调用，包括请求 DB2 Extender UDF 的语句或调用，都会生成一些代码，以指示嵌入式 SQL 语句或 DB2 CLI 调用是否成功

运行其它 DB2 Extender API（如管理 API）也返回指示成功与否的代码。程序应检查和响应嵌入式 SQL 语句、CLI 调用和 API 所返回的代码

有关处理这些返回码的信息，参见第 447 页的第 16 章，『诊断信息』。

在 Extender API 不能成功完成其工作单元的情况下，执行回滚操作。API 还返回错误代码。执行回滚操作，使数据库可以返回到其先前一致点。参考第 217 页的第 14 章，『应用程序编程接口』以了解详细信息。

Unicode 支持

留意有关 Image Extender、Audio Extender 和 Video Extender 的 Unicode 支持的下列几点:

- 可以是 Unicode 字符串的参数只有下列 UDF 中的注释字段:
  - mmdbsys.db2image( ) 导入图像
  - mmdbsys.db2audio( ) 导入音频
  - mmdbsys.db2video( ) 导入视频
  - mmdbsys.replace( ) 替换图像、音频或视频
  - mmdbsys.comment( ) 注释更新
- 如果您计划存取 Unicode 数据库，则必须使用设置为支持 Unicode 的 DB2 Extender 实例。Unicode 实例将只处理 Unicode 数据库。

  为了让 Extender 实例支持 Unicode，请在调用 DMBSTART 之前将环境变量 DB2CODEPAGE 设置为 1208。

第 10 章 存储、检索和更新对象

本章描述如何使用 DB2 Extender 的用户定义函数来存储、检索和更新图像、音频或视频。

图像、音频和视频格式

表 8 列示在存储、检索或更新图像、音频和视频对象时可使用的格式。可让 Image Extender 在存储、检索或更新图像时转换它的格式（仅适用于图像对象）。（在存储、检索或更新时，不能转换音频和视频对象的格式。）

表中“读”和“写”列指示可读的格式和在写时可以转换的格式。当表中“读”列中的条目是 x 时，表示在存储、检索或更新时可使用对应的对象格式。当“写”列中的条目是 x 时，表示在存储、检索或更新时，可将对象（仅限于图像）转换为相对应的格式。例如，在存储、检索或更新时，可将 BMP 格式的图像转换为 GIF 格式。可将 JPG 格式的图像转换为 TIF 格式。但不能将 TIF 格式的图像转换为 JPG 格式。

虽然在表中是以大写列示的，但存储、检索或更新请求中的格式规范是不区分大小写的。例如，规范 GIF、gif 和 Gif 是等价的。

表 8. DB2 Extender 可处理的格式

格式	描述	读	写
	图像格式		
_IM	PS/2 视听连接（AVC）	x	
BMP	OS/2 - Microsoft Windows 位图[2]	x	x
EPS	封装 PostScript		x
EP2	封装级别 2 PostScript		x
GIF	Compuserve GIF89a（包括动画 GIF [3]）和 87	x	x
IMG	IOCA 图像	x	x
IPS	Brooktrout FAX 卡片文件	x	x
JPG	JPEG[4]（JFIF 格式）	x	
PCX	PC 绘画文件（仅灰度）	x	x
PGM	可移植灰度图（来自 PBMPLUS）	x	x
PS	PostScript		x
PSC	压缩 PostScript 图像		x
PS2	PostScript 级别 2（彩色）		x
TIF	所有 TIFF 5.0 格式	x	x
YUV	YUV 的数字视频	x	x
	音频格式		
AIF 或 AIFF	音频交换文件格式	x	
AIFFC	压缩的音频交换文件格式	x	
AU	Sun 音频文件格式	x	
MIDI	乐器数字化接口	x	

© Copyright IBM Corp. 1998, 2003

格式

表 8. DB2 Extender 可处理的格式 (续)

格式	描述	读	写
MPG1 或 MPEG1	活动照片专家组 1	x	
WAV 或 WAVE	声波	x	
	视频格式		
AVI	交错音频 / 视频	x	
MPG1 或 MPEG1	活动照片编码专家组 1	x	
MPG2 或 MPEG2	活动照片编码专家组 2	x	
QT	Quicktime（AVI）	x	

图像转换选项

表 9 列示在存储、检索或更新图像时可对其指定的（格式转换之外的）转换选项。Image Extender 将您的规范应用于目标图像；不更改源图像。

每个转换选项都是以一对参数 / 值的形式指定的。该表中列示了每个参数允许的值。

表 9. 图像转换选项

参数	描述	值
-b	用来表示每一图像样本的位数	1 或 8 位
-s[5]	比例因子	任何大于 0 的十进制值。比例因子指定转换后的图像与原始图像的大小比例。例如，比例因子 0.5 将图像转换为原始大小的一半。比例因子 2.0 将图像转换为原始大小的两倍。
-p	光度学（图像反转）。此选项根据指定的值更改图像的解释。它不更改图像本身。此选项仅适用于黑白或灰度图像，不适用于 GIF 格式的图像。	0 = 表示黑色 1 = 表示白色
-n	光度学（图像反转）。此选项通过将黑色反转为白色，以及将白色反转为黑色来更改图像。此选项仅适用于黑白或灰度图像。	无
-r[5]	旋转	0 = 0 度（不旋转）1 = 90 度（反时针方向） 2 = 90 读（顺时针方向） 3 = 180 度
-x[5]	以像素计的宽度	像素数
-y[5]	以像素计的高度	像素数

2. Windows V2、Windows V3 和 Windows NT BMP 格式支持读。

3. DB2 Image Extender 仅存储动画 GIF 文件中第一幅图像的属性信息。

4. 支持使用部分基于“独立 JPEG 组”的工作的软件。

表 9. 图像转换选项 (续)

参数	描述	值
-c	压缩类型	0 = IBM MMR 1 = CCITT 第 3 组 1-D 2 = CCITT 第 3 组 2-D (k=2) 3 = CCITT 第 3 组 2-D (k=4) 4 = CCITT 第 4 组 6 = TIFF 第 2 类 10 = 不压缩 14 = LZW 15 = TIFF 压缩位数 25 = JBIG

存储图像、音频或视频对象

在 SQL INSERT 语句中使用 DB2Image、DB2Audio 或 DB2Video UDF，以在数据库中存储图像、音频或视频对象。

可以存储来源于缓冲区、客户机中的文件或服务器文件的对象。对于任何这些来源，都可以将对象以 BLOB 形式存储在数据库表中，或存储在数据库服务器上的文件中。

当请求 UDF 时，需要指定:

- 当前连接的数据库服务器的名称；这包含在 CURRENT SERVER 专用寄存器中。
- 对象内容的源；在客户机缓冲区、客户机文件或服务器文件中。
- 是想将内容以 BLOB 的形式存储在数据库表中，还是存储在文件服务器上。
- 源的格式。
- 要与对象存储在一起的注释（若不想存储注释，则指定空值或空字符串）。

即使 Image Extender、Audio Extender 和 Video Extender 不识别对象的格式，它们也允许存储该对象。在未识别格式的情况下，您需要指定对象的属性。当存储具有用户提供的属性的图像或视频时，您还可以存储缩略图。缩略图是表示图像或视频的微型图像。

仅对于图像，可选择在存储时转换图像的格式。若请求格式转换，则需要同时指定图像的源和目标格式。在格式转换请求中，还可指定对图像的进一步更改，如进行裁剪或旋转。通过指定转换选项来指示这些更改。

落实存储操作: 在将图像、音频或视频对象存储在数据库中之后，落实工作单元。这将释放 Extender 挂起的锁，从而可以对存储的对象执行更新操作。

DB2Image、DB2Audio 和 DB2Video UDF 格式

DB2Image、DB2Audio 和 DB2Video UDF 是重载的，即它们的格式随使用 UDF 的方法不同而有所不同。每个 UDF 都具有下列格式（格式中显示的 xxxxx 可以是 Image、Audio 或 Video）:

格式 1: 从客户机缓冲区或客户机文件存储对象:

5. 若对交错 GIF 图像指定此选项，还应指定压缩类型 LZW。

```
DB2xxxxx(
     CURRENT SERVER,       /* database name name in CURRENT SERVER REGISTER */
     content,              /* object content */
     format,               /* source format */
     target_file,          /* target file name for storage in file server */
                           /* or NULL for storage in table as BLOB */
     comment               /* user comment */
  );
```

格式 2: 从服务器文件存储对象:

```
DB2xxxxx(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     source_file,          /* source file name */
     format,               /* source format */
     stortype,             /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                           /* in file server*/
                           /* MMDB_STORAGE_TYPE_INTERNAL=store */
                           /* as a BLOB*/
     comment               /* user comment */
  );
```

格式 3: 从客户机缓冲区或客户机文件存储具有用户提供的属性的对象:

```
DB2xxxxx(
     CURRENT SERVER,        /* database name in CURRENT SERVER REGISTER */
     content,               /* object content */
     target_file,           /* target file name for storage in file server */
                            /* or NULL for storage in table as BLOB */
     comment,               /* user comment */
     attrs,                 /* user-supplied attributes */
     thumbnail              /* thumbnail (image and video only) */
  );
```

格式 4: 存储服务器文件中具有用户提供的属性的对象:

```
DB2xxxxx(
     CURRENT SERVER,           /* database name in CURRENT SERVER REGISTER */
     source_file,              /* source file name */
     stortype,                 /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                               /* in file server*/
                               /* MMDB_STORAGE_TYPE_INTERNAL=store */
                               /* as a BLOB*/
     comment,                  /* user comment */
     attrs,                    /* user-supplied attributes */
     thumbnail                 /* thumbnail (image and video only) */
  );
```

DB2Image UDF 包括下列附加格式:

格式 5: 从客户机缓冲区或客户机文件存储图像，并进行格式转换:

```
DB2Image(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     content,              /* object content */
     source_format,        /* source format */
     target_format,        /* target format */
     target_file,          /* target file name for storage in file server */
                           /* or NULL for storage in table as BLOB */
     comment               /* user comment */
  );
```

格式 6: 从服务器文件存储图像，并进行格式转换:

```
DB2Image(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     source_file,          /* server file name */
```

```
    source_format,        /* source format */
    target_format,        /* target format */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

格式 7: 从客户机缓冲区或客户机文件存储图像，并进行格式转换和其它更改:

```
DB2Image(
    CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
    content,              /* object content */
    source_format,        /* source format */
    target_format,        /* target format */
    conversion_options,   /* Conversion options */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

格式 8: 从服务器文件存储图像，并进行格式转换和其它更改:

```
DB2Image(
    CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
    source_file,          /* server file name */
    source_format,        /* source format */
    target_format,        /* target format */
    conversion_options    /* conversion options */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

例如，C 应用程序中的下列语句将包括图像的行插入到 Employee 表中。源图像在名为 ajones.bmp 的服务器文件中。图像以 BLOB 形式存储在 Employee 表中。（这与上面的格式 2 相对应。）

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*/
          CURRENT SERVER,                  /*database*/
          '/employee/images/ajones.bmp',   /*source file */
          'ASIS',                          /*keep the image format*/
          :hvStorageType                   /*store image in DB as BLOB*/
          'Anita''s picture')              /*comment */
       );
```

C 应用程序中的下列语句象前一示例那样将同一行存储在 Employee 表中。但是此处，在存储时将图像由 BMP 转换为 GIF 格式。（这与上面的格式 6 相对应。）

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*
          CURRENT SERVER,                  /*database*/
          '/employee/images/ajones.bmp',   /*source file */
```

```
          'ASIS',                          /*source image format*/
          'GIF',                           /*target image format*/
          'Anita''s picture')              /*comment*/
      );
```

当存储图像、音频或视频对象时，Extender 计算诸如图像中使用的颜色数、音频播放时间或视频压缩格式之类的属性。若存储了具有不可识别的格式的对象，则需要提供这些属性以作为 UDF 的输入。Extender 将这些属性与其它属性（如关于对象的注释和存储该对象的用户的标识）一起存储在数据库中。您在以后可以在查询中使用这些属性。

存储驻留在客户机上的对象

通过使用主变量或文件引用变量，将图像、音频或视频对象的内容从客户机缓冲区或客户机文件传送至服务器。

若对象在客户机文件中，则使用文件引用变量来传送它的内容，以便将它存储在服务器中。例如，C 应用程序中的下列语句定义名为 Audio_file 的文件引用变量，并使用它来传送内容位于客户机文件中的音频剪辑。该音频剪辑存储在服务器上的数据库表中。注意，文件引用变量的 file_option 字段为输入设置成 SQL_FILE_READ。另外还需注意，文件引用变量被用作 DB2Audio UDF 的内容自变量。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Audio_file;
EXEC SQL END DECLARE SECTION;

strcpy (Audio_file.name, "/employee/sounds/ajones.wav");
Audio_file.name_length= strlen(Audio_file.name);
Audio_file.file_options= SQL_FILE_READ;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
          :Audio_file,          /* file reference variable */
          'WAVE',
          CAST(NULL as LONG VARCHAR),

          'Anita''s voice')
      );
```

若对象在客户机缓冲区中，则使用定义为 BLOB 或 BLOB_LOCATOR 的主变量来传送其内容，以便存储在服务器中。在下列 C 应用程序语句中，使用名为 Video_loc 的主变量来传送视频剪辑的内容，以便存储在服务器中。该视频剪辑以 BLOB 的形式存储在数据库表中。注意，该主变量被用作 DB2Video UDF 的内容自变量。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR Video_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
        DB2VIDEO(
         CURRENT SERVER,
          :Video_loc,         /* host variable */
         'MPEG1',
          '',
          'Anita''s video')
       );
```

确保有足够的 UDF 内存: 当存储其内容在客户机缓冲区中的对象时，需要确保将“数据库管理器配置”中的 UDF_MEM_SZ 参数设置为 4 MB 或更大。可通过使用 DB2 命令 UPDATE DATABASE MANAGER CONFIGURATION 来更新 UDF_MEM_SZ 参数。有关 UPDATE DATABASE MANAGER 命令的更多信息，参见 *DB2 Command Reference*。

存储驻留在服务器上的对象

当要存储的图像、音频或视频位于服务器文件中时，指定其路径作为 UDF 的内容自变量。例如，C 应用程序中的以下语句将包括图像的行存储到数据库中。图像内容位于服务器上的文件中。存储的图像保留在服务器文件中，将从数据库指向它。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp', /*source in server file */
          'BMP',
          :hvStorageType,
          'Anita''s picture')
       );
```

指定正确的路径: 当存储来源于服务器文件的对象时，可指定文件的全限定名或相对名称。若指定相对名称，则需要确保 DB2 服务器中的适当环境变量包括正确的文件路径。有关设置这些环境变量的信息，参见第 475 页的附录 A, 『设置 DB2 Extender 的环境变量』。

指定数据库或文件存储器

可以 BLOB 形式将图像、音频或视频对象存储在数据库中，或存储在服务器文件中。若将对象存储在服务器文件中，则数据库指向该文件。

若从客户机缓冲区或客户机文件存储对象，则指示 BLOB 或服务器文件存储器作为在 target_file 参数中指定的内容的结果。若指定文件名，它指示要将对象存储在服务器文件中。若指定空值或空字符串，则它指示要将对象以 BLOB 形式存储在数据库表中。target_file 参数的数据类型是 LONG VARCHAR。若指定空值，切记将其强制转换为 LONG VARCHAR 数据类型。

例如，C 应用程序中的下列语句将包括图像的行存储到数据库表中。图像源在客户机缓冲区中。图像存储在服务器文件中。数据库表指向服务器文件:

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR Img_buf
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          :Img_buf,
```

```
          'ASIS',
          '/employee/images/ajones.bmp',    /* store image in server file */
          'Anita''s picture')
       );
```

若存储位于服务器文件中的对象，则指定常量 MMDB_STORAGE_TYPE_INTERNAL，以便将该对象以 BLOB 的形式存储到数据库表中。若要存储对象，且要使其内容保留在服务器文件中，则指定常量 MMDB_STORAGE_TYPE_EXTERNAL。MMDB_STORAGE_TYPE_INTERNAL 的值为整数 1。MMDB_STORAGE_TYPE_EXTERNAL 的值为整数 0。

例如，在以下 C 应用程序中，将音频剪辑存储在服务器文件中。源音频内容已经在服务器文件中。存储操作将文件名放在数据库中，从而可通过 SQL 语句来存取该文件。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
          '/employee/sounds/ajones.wav',
          'WAVE',
          :hvStorageType,             /* store audio in server file */
          'Anita''s voice')
       );
```

标识存储格式

当存储对象时，需要标识其格式。第 93 页的表 8 列示了可指定的格式。Extender 将把图像、音频或视频对象存储成与源相同的格式。可选择让 Image Extender 转换所存储图像的格式（仅对于图像对象）。若选择转换图像格式，则需要指定源图像的格式和目标图像的格式。目标图像就是所存储的图像。

标识不进行转换的存储格式

当您存储对象时不进行格式转换时，指定源图像、音频或视频对象的格式。例如，C 应用程序中的以下语句将位图（BMP）图像存储到数据库表中。源的内容位于服务器文件中。目标图像的格式将与源的格式相同。

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /*image in BMP format */
          '',
          'Anita''s picture')
       );
```

还可指定空值或空字符串作为格式，或者也可以指定字符串 ASIS（仅针对 Image Extender）。然后，Extender 将通过检查源来确定格式。

将 NULL 或 ASIS 用作可识别的格式: 仅当 Extender 可识别该格式时，也就是说，如果它是 第 93 页的表 8 中对 Extender 列示的格式中的一种时，才指定空值、空字符串或 ASIS。否则，Extender 将无法存储对象。

标识进行格式转换的存储的格式和转换选项

若在存储图像时进行格式转换，同时指定源和目标图像的格式。第 93 页的表 8 列示了允许的格式转换。

另外，还可指定转换选项，其标识要应用于所存储图像的其它更改（如旋转或压缩）。通过参数和相关联的值指定每个转换选项。第 94 页的表 9 中列示了参数和允许的值。可通过指定多对参数 / 值来请求对存储的图像进行多处更改。

在以下示例中，当存储在数据库表中时，将把内容在服务器文件中的位图（BMP）图像转换为 GIF 格式。

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2IMAGE(
         CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /* source format */
          'GIF',                          /* target format */
          '',
          'Anita''s picture')
       );
```

在以下示例中，当存储在数据库表中时，将把前一示例中的图像转换为 GIF 格式。另外，在存储时，将把该图像修剪成宽度为 110 像素，高度为 150 像素，并使用 LZW 压缩来进行压缩。

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2IMAGE(
         CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /* source format */
          'GIF',                          /* target format */
          '-x 110 -y 150 -c 14',          /* conversion options */
          '/Employee/images/ajones.gif',
          'Anita''s picture')
       );
```

存储具有用户提供的属性的对象

当您存储图像、音频或视频对象时，将不限制您只使用 Extender 理解的格式。您可以指定您自己的格式。因为 Extender 不理解该格式，所以必须指定源对象的属性。在属性结构中指定属性值。必须将属性结构存储在 UDF 中 LONG VARCHAR FOR BIT DATA 变量的数据字段中。

服务器上的 UDF 代码总是期望数据处于“大尾数法格式”。大尾数法格式是大多数 UNIX 平台使用的格式。若正在存储处于“小尾数法格式”的对象，则需要准备用户提供的属性数据，以使服务器上的 UDF 代码可以正确地处理它。小尾数法格式是 Intel® 和其它微处理器平台中通常使用的格式。（即使不以小尾数法格式存储对象，准备用户提供的属性数据也是一个好主意。）使用 DBiPrepareAttrs API 来准备图像对象的属性。使用 DBaPrepareAttrs API 来准备音频对象的属性。使用 DBvPrepareAttrs API 来准备视频对象的属性。

例如，C 应用程序中的下列语句将包括图像的行存储在数据库表中。服务器文件中的源图像具有用户定义的格式，高度为 640 像素，宽度为 480 像素。注意，在存储图像之前属性已准备好。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
       }hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattr->format,"FormatI");
pimgattr->width=640;
pimgattr->height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

DBiPrepareAttrs(pimgattr);

DBEXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          :hvStorageType,
          'Anita''s picture',
          :hvImgattrs,                  /* user-specified attributes */
          CAST(NULL as LONG VARCHAR)
       );
```

C 应用程序中的以下语句将包括音频剪辑的行存储在数据库表中。服务器文件中的源音频剪辑具有用户定义的格式，采样速率为 44.1 k，并有两个录制声道。音频剪辑不是 MIDI，因而对声道名和乐器指定空字符串。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct (
       short len;
       char data[600];
}hvAudattr;
EXEC SQL END DECLARE SECTION;

MMDBAudioAttrs          *paudiattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

paudioattr=(MMDBAudioAttrs *) hvAudattr.data;
strcpy(paudioAttr->cFormat,"FormatA");
paudioAttr->ulSamplingRate=44100;
paudioAttr->usNumChannels=2;
hvAudattrs.len=sizeof(MMDBAudioAttrs);

DBaPrepareAttrs(paudioAttr);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
               DB2AUDIO(
         CURRENT SERVER,
               '/Employee/Sounds/ajones.aud',
```

```
        :hvStorageType,
         'Anita''s voice',
         :hvAudattr)                    /* user-specified attributes */
      );
```

存储缩略图（仅适用于图像和视频）

当存储使用您自己的格式的图像时，还可存储**缩略图**，即图像的缩微图大小的版本。您控制缩略图的大小和格式。当存储使用 Image Extender 可识别的格式的图像时，它为该对象自动生成并存储一个缩略图。Image Extender 创建使用 GIF 格式、大小为 112 x 84 平方像素的缩略图。

当存储使用您自己的格式的视频对象时，还可存储将视频对象符号化的缩略图。当存储使用 Video Extender 可识别的格式的视频对象时，它为该对象自动存储一个类属缩略图。Video Extender 创建使用 GIF 格式、大小为 108 x 78 平方像素的缩略图。

若在存储具有用户提供的属性的图像或视频对象时不想存储缩略图，则指定空值或空字符串来代替缩略图。

可以在程序中生成缩略图，但 Extender 并不提供生成缩略图的 API。在程序中创建缩略图的结构，并在 UDF 中指定缩略图结构。

C 应用程序中的下列语句将包括视频剪辑的行存储在数据库表中。其内容在服务器文件中的源视频剪辑具有用户定义格式。视频内容将保留在服务器中，将从表中指向它。还会存储代表性视频帧的缩略图。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
struct {
       short len;
       char data[4000];
       }hvVidattrs;
struct {
       short len;
       char data[10000];
       }hvThumbnail;
EXEC SQL END DECLARE SECTION;

MMDBVideoAttrs         *pvideoAttr;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

pvideoAttr=(MMDBVideoAttrs *)hvVidattrs.data;
strcpy(pvideoAttr->cFormat,"Formatv");
hvVidattrs.len=sizeof(MMDBVideoAttrs);

/* Generate thumbnail and assign data in video structure */

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
        DB2VIDEO(
         CURRENT SERVER,
          '/Employee/Videos/ajones.vid',
         :hvStorageType,
          'Anita''s video',
          :hvVidattrs,
          :hvThumbnail)                     /* Thumbnail*/
       );
```

存储注释

通过在 UDF 请求中指定注释来将注释与图像、音频或视频对象存储在一起。注释是自由格式的文本，数据类型为 LONG VARCHAR，可长达 32,700 个字节。若存储对象时不想存储注释，则指定空值或空字符串来代替注释。若指定空值，切记将其强制转换为 LONG VARCHAR 数据类型。

例如，C 应用程序中的下列语句将注释与视频剪辑存储在一起。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          '/employee/videos/ajones.mpg',
          'MPEG1',
          :hvStorageType,
          'Anita''s video')               /* comment */
       );
```

C 应用程序中的下列语句存储不带注释的图像。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'GIF',
          :hvStorageType,
          Cast(NULL as LONG VARCHAR)            /* no comment */
       );
```

检索图像、音频或视频对象

通过在 SQL SELECT 语句中使用 Content UDF，在数据库中检索图像、音频或视频对象。可以将对象检索至客户机缓冲区、客户机文件或服务器文件。

用于检索的 Content UDF 格式

Content UDF 是重载的，即它随使用此 UDF 的方法的不同而具有不同的格式。格式如下:

格式 1: 将对象检索至客户机缓冲区或客户机文件:

```
Content(
     handle,                    /* object handle */
  );
```

格式 2: 将对象段检索至客户机缓冲区或客户机文件:

```
Content(
     handle,                      /* object handle */
     offset,                      /* offset where retrieval begins */
     size                         /* number of bytes to retrieve */
  );
```

格式 3: 将对象检索至服务器文件:

```
Content(
     handle,                      /* object handle */
     target_file,           /* server file name */
     overwrite                    /* 0=Do not overwrite target file if it exists */
                            /* 1=Overwrite target file */

  );
```

另外，Content UDF 还包括下列仅针对图像对象的格式:

格式 4: 将图像检索至客户机缓冲区或文件，并进行格式转换:

```
Content(
     handle,                      /* object handle */
     target format                /* target format */
       );
```

格式 5: 将对象检索至服务器文件，并进行格式转换:

```
Content(
     handle,                      /* object handle */
     target_file,           /* server file name */
     overwrite,             /* 0=Do not overwrite target file if it exists */
                            /* 1=Overwrite target file */
     target format                /* target format */

  );
```

格式 6: 将对象检索至客户机缓冲区或文件，并进行格式转换和其它更改:

```
Content(
     handle,                      /* object handle */
     target format,              /* target format */
     conversion_options     /* conversion options */
  );
```

格式 7: 将对象检索至服务器文件，并进行格式转换和其它更改:

```
Content(
     handle,                      /* object handle */
     target_file,           /* server file name */
     overwrite,             /* 0=Do not overwrite target file if it exists */
                            /* 1=Overwrite target file */
     target format,              /* target format */
     conversion_options     /* conversion options */
  );
```

例如，以下语句将从 Employee 表中检索到的图像放到服务器上的文件中。（这与格式 3 相对应。）

```
EXEC SQL SELECT CONTENT(                /* retrieval UDF */
       PICTURE,                         /* image handle */
       '/employee/images/ajones.bmp',   /* target file */
       1)                               /* overwrite target file */
FROM EMPLOYEE
WHERE NAME = 'Anita Jones';
```

C 应用程序中的下列语句将从 Employee 表中检索到的图像放到服务器上的文件中。在检索图像时，将进行格式转换。（这与格式 5 相对应。）

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvImg_fname[255];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(                    /* retrieval UDF */
                  PICTURE,                            /* image handle */
       '/employee/images/ajones.bmp',       /* target file */
       1,                                   /* overwrite target file */
        'GIF')                                /* target format */
     INTO :hvImg_fname
FROM EMPLOYEE
WHERE NAME = 'Anita Jones';
```

将对象检索至客户机

可使用 Content UDF 来将图像、音频或视频对象检索至客户机缓冲区或客户机文件，且不进行格式转换。另外，还可以选择让 Image Extender 在检索图像时转换其格式。

将对象检索至客户机，不进行格式转换

使用 LOB 定位器来将图像、音频或视频对象检索至客户机缓冲区，或检索 LOB。使用文件引用变量来将图像、音频或视频对象检索至客户机文件。

当对象的内容以 BLOB 形式存储在数据库表中时，使用主变量将图像、音频或视频对象检索至客户机缓冲区，或使用文件引用变量将其检索至客户机文件是合适的。若内容位于服务器文件中，则将内容从客户机文件复制至客户机文件可能更有效率。

指定对象的句柄。您还可选择指定从字节 1 开始的偏移（检索的起始位置），以及要检索的字节数。

C 应用程序中的下列语句使用名为 audio_loc 的 LOB 定位器来将音频剪辑检索到客户机缓冲区中。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR audio_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        SOUND)                                /* audio handle */
     INTO :audio_loc
  FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

确保有足够的 UDF 内存：当将对象检索至客户机缓冲区时，需要确保将“数据库管理器配置”中的 UDF_MEM_SZ 参数设置为 4 MB 或更大。可用 DB2 命令 UPDATE DATABASE MANAGER CONFIGURATION 来更新 UDF_MEM_SZ 参数。有关 UPDATE DATABASE MANAGER 命令的信息，参见出版物 *DB2 Command Reference*。

将图像检索至客户机，并进行转换

使用 LOB 定位器来将存储的图像检索至客户机缓冲区，并进行格式转换，或使用 LOB 定位器来检索 LOB。使用文件引用变量来将存储的图像检索至客户机文件，并进行格式转换。

当图像的内容以 BLOB 形式存储在数据库表中时，使用主变量将图像检索至客户机缓冲区，或使用文件引用变量将其检索至客户机文件是合适的。若内容位于服务器文件中，则将内容从客户机文件复制至客户机文件可能更有效率。

若检索图像时进行格式转换，则需要指定其目标格式，即转换后的格式。第 93 页的表 8 标识了允许的格式转换。还可指定转换选项，它们标识要应用于检索到的图像的其它更改（如旋转或按比例缩放）。第 94 页的表 9 列示了可指定的转换选项。

例如，C 应用程序中的下列语句将图像检索至客户机文件。源图像处于位图格式，它以 BLOB 形式存储在数据库表中。将检索到的图像转换为 GIF，并按比例放大为其原始大小的 3 倍。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Img_file;
EXEC SQL END DECLARE SECTION;

strcpy (Img_file.name, "/Employee/images/ajones.gif");
Img_file.name_length= strlen(Img_file.name);
Img_file.file_options= SQL_FILE_CREATE;

EXEC SQL SELECT CONTENT(
        PICTURE,                              /* image handle */
        'GIF',                                /* target format */
        '-s 3.0')                             /* conversion options */
     INTO :Img_file,
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

将对象检索至服务器文件

可使用 Content UDF 来将图像、音频或视频对象检索至服务器文件，且不进行格式转换。另外，还可使用 Content UDF 来将图像检索至服务器文件，并进行格式转换。

当将图像、音频或视频对象检索至服务器上的文件，且不进行转换时，指定对象的句柄、目标文件名和覆盖指示符。覆盖指示符告诉 Extender 在目标文件已存在于服务器上时，是否用检索到的数据覆盖目标文件。若目标文件不存在，则 Extender 在服务器上创建目标文件。

若指定的覆盖指示符值为 1，Extender 将用检索到的数据覆盖目标文件。若指定的覆盖指示符值为 0，Extender 不覆盖目标文件，因而也就不检索数据。

若要检索的对象以 BLOB 形式存储在数据库表中，则覆盖指示符被忽略。无论为覆盖指示符指定了什么内容，都将创建或覆盖目标文件。

当您将对象检索至服务器文件时，它将返回服务器文件的名称。例如，C 应用程序中的以下语句将视频检索至服务器上的文件。服务器文件的文件名存储在主变量 hvVid_fname 中。

```
EXEC SQL BEGIN DECLARE SECTION;
struct{
       short len;
       char data[250];
      }hvVid_fname[;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        VIDEO,                                /* video handle */
        '/employee/videos/ajones.mpg',        /* server file */
        1)                                    /* overwrite target file */
     INTO :hvVid_fname;
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

检索

如果将对象以 BLOB 的形式存储在数据库表中，则使用 Content UDF 来将对象检索至服务器文件且不进行转换是合适的。若对象存储在服务器文件中，则将源文件的内容复制至目标文件可能更有效率。

当将图像检索至服务器文件，并进行格式转换时，指定图像句柄、目标文件名、覆盖目标指示符和目标格式。第 93 页的表 8 标识了允许的格式转换。还可选择对目标格式指定空值或空白字符串，或指定字符串 ASIS。在这种情况下，检索到的图像的格式将与源的格式相同。

例如，C 应用程序中的下列语句将图像检索至服务器上的文件。源图像使用位图格式，并以 BLOB 形式存储在数据库表中。将把检索到的图像转换为 GIF 格式。服务器文件的文件名存储在主变量 hvImg_fname 中。

```
EXEC SQL BEGIN DECLARE SECTION;
  struct{
        short len;
        char [400];
        }hvImg_fname[;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        PICTURE,                              /* image handle */
        '/employee/images/ajones.gif',        /* target file */
        1,                                    /* overwrite target file */
        'GIF')                                /* target format */
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

服务器文件必须是可存取的：当将对象检索至服务器文件时，必须指定目标文件的全限定名称。此外，必须确保将 DB2IMAGEEXPORT、DB2AUDIOEXPORT 和 DB2VIDEOEXPORT 环境变量设置为能正确解析不完整的文件名规范。

检索和使用属性

当您把图像、音频或视频对象存储在数据库中时，Extender 还在数据库中存储对象的属性。更新对象时，Extender 将更新存储在数据库中的对象属性。可在查询中使用这些属性。

Extender 对其管理的每一属性都创建了 UDF。结果是，可在 SQL 语句中指定 UDF 来存取并使用对象属性。表 10 列示了 Extender 管理的属性及其 UDF。它还指示了每个属性的对象类型。某些属性，如对象的格式和文件名，对所有对象类型都是通用的。这些属性与图像、音频和视频对象相关联。其它属性，如采样速率或压缩类型，则是专门针对某些对象类型（如音频和视频）的。

表 10. DB2 Extender 管理的属性. 可通过各自的 UDF 来存取每个属性。

属性	UDF	图像	音频	视频
在其中存储对象的服务器文件的名称	Filename	x	x	x
存储对象的人员的用户标识	Importer	x	x	x
存储对象的日期和时间	ImportTime	x	x	x
以字节计的对象大小	Size	x	x	x
上次更新对象的人员的用户标识	Updater	x	x	x
上次更新对象的日期和时间	UpdateTime	x	x	x
对象的格式（如 GIF 或 MPEG1）	Format	x	x	x

表 10. DB2 Extender 管理的属性 (续). 可通过各自的 UDF 来存取每个属性。

属性	UDF	图像	音频	视频
关于对象的注释	Comment	x	x	x
对象的高度（以像素计）	Height	x		x
对象的宽度（以像素计）	Width	x		x
对象中的颜色数	NumColors	x		
对象的缩略图大小的图像	Thumbnail	x		x
音频中或视频的声道中每次采样返回的字节数	AlignValue		x	x
用来表示每个样本的位数	BitsPerSample		x	x
录制声道数	NumChannels		x	x
持续时间（以秒计）	Duration		x	x
采样速率（采样数／秒）	SamplingRate		x	x
每秒传送时间的平均字节	BytesPerSec		x	
乐器的声道数	FindInstrument		x	
命名声道的声道号	FindTrackName		x	
录制乐器的名称	GetInstruments		x	
录制乐器的声道号和名称	GetTrackNames		x	
音频的每秒时钟滴答数	TicksPerSec		x	
音频的每四分音符时钟滴答数	TicksPerQNote		x	
长宽比	AspectRatio			x
视频压缩格式（如 MPEG1）	CompressType			x
吞吐量的帧频	FrameRate			x
最大吞吐量（以每秒字节数计）	MaxBytesPerSec			x
声道数	NumAudioTracks		x	x
帧数	NumFrames			x
视频声道数	NumVideoTracks			x

可在 SQL 语句 SELECT 子句表达式或 WHERE 子句搜索条件中使用属性 UDF。在请求 UDF 时，指定数据库表中包含对象句柄的列的名称。

例如，以下语句在 SQL SELECT 语句的 SELECT 子句中使用 Updater UDF，来检索上次更新 Employee 表中的图像的人员的用户标识:

```
EXEC SQL BEGIN DECLARE SECTION;
char hvUpdatr[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT UPDATER(PICTURE)
     INTO :hvUpdatr
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

以下语句在 SELECT 语句的 SELECT 语句中使用 Filename UDF，并在 WHERE 子句中使用 NumAudioTracks UDF 来查找存储在 Employee 表中并带有声道的视频:

```
EXEC SQL BEGIN DECLARE SECTION;
char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;
```

```
EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMAUDIOTRACKS(VIDEO)>0;
```

检索注释

使用 Comment UDF 来检索与图像、音频或视频对象存储在一起的注释。当检索对象的注释时，指定数据库表中包含对象句柄的列。例如，下列语句检索与 employee 表中的音频剪辑存储在一起的注释。

```
EXEC SQL BEGIN DECLARE SECTION;
struct {
       short len;
       char data[32700];
       }hvComment
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT COMMENT(SOUND)
     INTO :hvComment
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

还可将 Comment UDF 用作 SQL 查询的 WHERE 子句中的谓词。例如，以下语句检索 Employee 表中被标记为 “touched up” 的所有图像的文件名。

```
EXEC SQL BEGIN DECLARE SECTION;
struct {
       short len;
       char data[250];
       }hvImg_fname
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE COMMENT(PICTURE)
           LIKE '%touch%up';
```

更新图像、音频或视频对象

在 SQL UPDATE 语句中使用 Content UDF 来更新数据库表中的图像、音频或视频对象。在 SQL UPDATE 语句中使用 Replace UDF 来更新数据库表中的图像、音频或视频，并更新与对象相关联的注释。在任一情况下，Extender 都会更新与对象相关联的属性。

可更新以 BLOB 形式存储在数据库表中的对象，或存储在服务器文件中（并从数据库指向它）的对象。更新的源可以在缓冲区、客户机文件或服务器文件中。

第 93 页的表 8 列示了可更新的图像、音频和视频对象的格式。不过，您还可以更新其格式没有被 Extender 识别的对象。在这种情况下，用户在存储对象时指定该对象的属性。更新具有用户指定属性的对象时，您需要指定该对象的已更新的属性。在 SQL UPDATE 语句中使用 ContentA UDF 来更新数据库中具有用户提供的属性的图像、音频或视频对象。在 SQL UPDATE 语句中使用 ReplaceA UDF 来更新数据库表中具有用户提供的属性的图像、音频或视频，并更新与对象相关联的注释。更新具有用户指定属性的对象时，您需要指定对象的属性、其格式，以及指定其压缩格式（仅对于视频对象）。

还可更新存储的图像或视频的缩略图。

落实更新操作: 在更新数据库中的图像、音频或视频对象之后，落实工作单元。这将释放 Extender 挂起的锁，从而可以对存储的对象执行后续的更新操作。

用于更新的 Content UDF 格式

Content UDF 是重载的，即它随使用此 UDF 的方法的不同而具有不同的格式。格式如下:

格式 1: 从客户机缓冲区或客户机文件更新对象:

```
Content(
     handle,                      /* object handle */
     content,                     /* object content */
     source_format,               /* source format */
     target_file                 /* target file name for storage in file */
                                 /* server or NULL for storage in table as BLOB */
  );
```

格式 2: 从服务器文件更新对象:

```
Content(
     handle,                      /* object handle */
     source_file,                 /* server file name */
     source_format,               /* source format */
     stortype                    /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                  /* in file server */
                                  /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
  );
```

格式 3: 从客户机缓冲区或客户机文件更新具有用户提供的属性的对象:

```
Content(
     handle,                      /* object handle */
     content,                     /* object content */
     target_file,                 /* target file name for storage in file server */
                                  /* or NULL for storage in table as BLOB */
     attrs,                       /* user-supplied attributes */
     thumbnail                    /* thumbnail (image and video only) */
  );
```

格式 4: 从服务器文件更新具有用户提供的属性的对象:

```
Content(
     handle,                        /* object handle */
     source_file,               /* source file name */
     stortype,                      /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                    /* in file server*/
                                /* MMDB_STORAGE_TYPE_INTERNAL=store */
                                /* as a BLOB*/
     attrs,                         /* user-supplied attributes */
     thumbnail                      /* thumbnail (image and video only) */
  );
```

Content UDF 有下列附加格式（仅针对图像对象）:

格式 5: 从客户机缓冲区或客户机文件更新图像，并进行格式转换:

```
Content(
     handle,                      /* object handle */
     content,                     /* object content */
     source format,               /* source format */
```

```
     target format,              /* target format */
     target_file                 /* target file name for storage in file server */
                                 /* or NULL for storage in table as BLOB */
  );
```

格式 6: 从服务器文件更新对象，并进行格式转换:

```
Content(
     handle,                     /* object handle */
     source_file,                /* server file name */
     source format,              /* source format */
     target format,              /* target format */
     target_file                 /* target file name for storage in file server */
                                 /* or NULL for storage in table as BLOB */
  );
```

格式 7: 从客户机缓冲区或客户机文件更新图像，并进行格式转换和其它更改:

```
Content(
     handle,                     /* object handle */
     content,                    /* object content */
     source format,              /* source format */
     target format,              /* target format */
     conversion_options,         /* conversion options */
     target_file                 /* target file name for storage in file server */
                                 /* or NULL for storage in table as BLOB */
  );
```

格式 8: 从服务器文件更新对象，并进行格式转换和其它更改:

```
Content(
     handle,                     /* object handle */
     source_file,                /* server file name */
     source format,              /* source format */
     target format,              /* target format */
     conversion_options,         /* conversion options */
     target_file                 /* target file name for storage in file server */
                                 /* or NULL for storage in table as BLOB */
  );
```

例如，C 应用程序中的下列语句更新 Employee 表中的图像。用于更新的源内容在名为 ajones.bmp 的服务器文件中。已更新的图像以 BOLB 形式存储在 employee 表中。（这与上面的格式 2 相对应。）

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
               PICTURE,                      /*image handle*/
                  '/employee/newimg/ajones.bmp',   /*source file */
            'ASIS',                         /*keep the image format*/
                  '');                            /*store image in DB as BLOB*/
       WHERE NAME='Anita Jones';
```

C 应用程序中的下列语句将要更新的图像与前例中的相同。但在此处，更新时将把该图像由 BMP 格式转换为 GIF 格式。（这与上面的格式 6 相对应。）

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
               PICTURE,                      /*image handle*/
                  '/employee/newimg/ajones.bmp',   /*source file */
                  'BMP',                           /*source format*/
                  'GIF',                           /*target format*/
                  '');                             /*store image in DB as BLOB*/
       WHERE NAME='Anita Jones';
```

用于更新的 Replace UDF 格式

Replace UDF 是重载的，这表示视使用此 UDF 的方法的不同，它有不同的格式。格式如下:

格式 1: 从客户机缓冲区或客户机文件更新对象，并更新其注释:

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_file,               /* target file name for storage in file */
     comment                    /* user comment */
  );
```

格式 2: 从服务器文件更新对象，并更新其注释:

```
Replace(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source_format,             /* source format */
     stortype,                  /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server*/
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                    /* user comment */
  );
```

格式 3: 从客户机缓冲区或客户机文件替换具有用户提供的属性的对象，并更新其注释:

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     target_file,               /* target file name for storage in file */
                                /* or NULL for storage in table as BLOB */
     comment,                   /* user comment */
     attrs,                     /* user-supplied attributes */
     thumbnail                  /* thumbnail */
  );
```

格式 4: 存储服务器文件中具有用户提供的属性的对象:

```
Replace(
     handle,                    /* object handle */
     source_file,               /* server file name */
     stortype,                  /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server*/
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment,                   /* user comment */
     attrs,                     /* user-supplied attributes */
     thumbnail                  /* thumbnail */
  );
```

Replace UDF 有下列附加格式（仅针对图像对象）:

格式 5: 从客户机缓冲区或客户机文件更新图像，进行格式转换，并更新其注释:

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_format,             /* target format */
     target_file,               /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
     comment                    /* user comment */
  );
```

格式 6: 从服务器文件更新对象，进行格式转换，并更新其注释:

```
Replace(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source_format,             /* source format */
     target_format,             /* target format */
     target_file,               /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server */
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                    /* user comment */
  );
```

格式 7: 从客户机缓冲区或客户机文件更新图像，进行格式转换和其它更改，并更新其注释:

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_format,             /* target format */
     conversion_options,        /* conversion options */
     target_file,               /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
     comment                    /* user comment */
  );
```

格式 8: 从服务器文件更新对象，进行格式转换和其它更改，并更新其注释:

```
Replace(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source_format,             /* source format */
     target_format,             /* target format */
     conversion_options,        /* conversion options */
     target_file,               /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server */
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                    /* user comment */
  );
```

例如，C 应用程序中的下列语句更新 Employee 表中的音频剪辑，并更新与之相关联的注释。用于更新的源内容在名为 ajones.wav 的服务器文件中。更新后的音频剪辑以 BLOB 形式存储在 Employee 表中，不进行格式转换（Audio Extender 不支持格式转换）。（这与上面的格式 2 相对应。）

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=REPLACE(
                SOUND,                          /*audio handle*/
                '/employee/newaud/ajones.wav', /*source file */
                'WAV',                          /*keep the audio format*/
                :hvStorageType,                 /*store audio in DB as BLOB*/
                'Anita''s new greeting')        /*user comment*/
   WHERE NAME= 'Anita Jones';
```

在以下示例中，将更新图像及其相关联的注释。用于更新的源内容位于服务器文件中。更新后的图像以 BLOB 形式存储在 Employee 表中，并在更新时由 BMP 格式转化为 GIF 格式。（这与上面的格式 6 相对应。）

```
EXEC SQL UPDATE EMPLOYEE
       SET PICTURE=REPLACE(
              PICTURE,                        /*image handle*/
                 '/employee/newimg/ajones.bmp',   /*source file */
                 'BMP',                           /*source format*/
                 'GIF',                           /*target format*/
                 ''                               /*store image in DB as BLOB*/
                   'Anita''s new picture')
   WHERE NAME='Anita Jones';                      /* user comment */
```

从客户机更新对象

使用主变量或文件引用变量来从客户机缓冲区或客户机文件更新图像、音频或视频对象。

若用于更新的源位于客户机文件中，则使用文件引用变量来传送其内容。例如，C 应用程序中的下列语句定义名为 Audio_file 的文件引用变量，并使用它来更新以 BLOB 形式存储在数据库表中的音频剪辑。用于更新的源位于客户机文件中。注意，文件引用变量的 file_options 字段被设置为 SQL_FILE_READ，即用于输入。另外还需注意，该文件引用变量被用作 Content UDF 的内容自变量。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Audio_file;
EXEC SQL END DECLARE SECTION;

strcpy (Audio_file.name, "/employee/newsound/ajones.wav");
Audio_file.name_length= strlen(Audio_file.name);
Audio_file.file_options= SQL_FILE_READ;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=CONTENT(
                 SOUND,
                 :Audio_file                 /*file reference variable*/
                 'WAVE',                     /*keep the image format*/
                 CAST(NULL as LONG VARCHAR))

       WHERE NAME='Anita Jones';
```

若对象位于客户机缓冲区中，则使用主变量来传送其内容以进行更新。在以下的 C 应用程序示例中，使用了名为 Video_seg 的主变量来传送视频剪辑的内容，以进行更新。还更新了与视频剪辑相关联的注释。该视频剪辑以 BLOB 的形式存储在数据库表中。注意，主变量被用作 Replace UDF 的内容自变量。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) Video_seg
EXEC SQL END DECLARE SECTION;

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                 VIDEO,
                 :Video_seg                  /*host variable*/
                 'MPEG1',
                 CAST(NULL as LONG VARCHAR),

                 'Anita''s new video')
       WHERE NAME='Anita Jones';
```

确保有足够的 UDF 内存：当更新其内容在客户机缓冲区中的对象时，需要确保将“数据库管理器配置”中的 UDF_MEM_SZ 参数设置为 4 MB 或更大。可用 DB2 命令 UPDATE DATABASE MANAGER CONFIGURATION 来更新 UDF_MEM_SZ 参数。

从服务器更新对象

当用于图像、音频或视频对象更新的源内容位于服务器文件中时，指定文件路径作为 UDF 的内容自变量。例如，C 应用程序中的下列语句将更新数据库中的图像。图像内容在服务器文件中。数据库指向该服务器文件。用于更新的源也在服务器文件中。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,                              /* image handle */
                '/employee/newimg/ajones.bmp',        /* source file */
                'ASIS',
                :hvStorageType)
   WHERE NAME='Anita Jones';
```

指定正确的路径：当更新其源在服务器文件中的对象时，可指定该文件的全限定名或相对名称。若指定相对名称，则需要确保 DB2 服务器中的适当环境变量包括正确的文件路径。有关设置这些环境变量的信息，参见第 475 页的附录 A，『设置 DB2 Extender 的环境变量』。

指定用于更新的数据库或文件存储器

可更新以 BLOB 形式存储在数据库表中，或存储在服务器文件中（并从数据库中指向它）的图像、音频或视频对象。

若从客户机缓冲区或客户机文件更新对象，则指示 BLOB 或服务器文件存储器作为在 filename 参数中指定的内容的结果。若指定文件名，它指示要更新其内容在服务器文件中的对象。若指定空文件名，它指示要更新以 BLOB 形式存储在数据库表中的对象。

例如，C 应用程序中的下列语句更新其内容在服务器文件中的图像。更新源仅位于客户机缓冲区中。还会更新图像注释。

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) Img_buf
EXEC SQL END DECLARE SECTION;

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=REPLACE(
                  PICTURE,
                  :Img_buf,
                  'ASIS',
                  '/Employee/newimg/ajones.bmp',      /*update image in*/
                                                      /*server file*/
                  'Anita''s new picture')
       WHERE NAME='Anita Jones';
```

若从服务器文件更新对象，则指定 MMDB_STORAGE_TYPE_INTERNAL 来更新以 BLOB 形式存储在数据库表中的对象。若要更新其内容在服务器文件中的对象，指定 MMDB_STORAGE_TYPE_EXTERNAL。

例如，在以下 C 应用程序中将更新音频剪辑。音频剪辑的内容位于服务器文件中。用于更新的源也在服务器文件中。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;
```

```
hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=CONTENT(
                SOUND,
                '/Employee/newimg/ajones.wav',
          'WAVE',
                :hvStorageType)        /*update audio in server file*/
   WHERE NAME='Anita Jones';
```

标识更新格式

在更新对象时，需要标识其格式。Extender 将把图像、音频或视频对象存储成与源相同的格式。可选择让 Image Extender 转换更新后的图像的格式（仅针对图像对象）。若要转换图像格式，则需要指定更新源的格式和目标图像的格式。目标图像就是所存储的已更新图像。

标识不进行转换的更新的格式

当更新对象而不进行格式转换时，指定源图像、音频或视频对象的格式。例如，C 应用程序中的以下语句更新其内容在服务器文件中的位图（BMP）图像。将不转换更新后的图像的格式。

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/Employee/newimg/ajones.bmp',
                'BMP',                          /*image format*/
                '')
       WHERE NAME='Anita Jones';
```

还可指定空值或空字符串作为格式，或者也可以指定字符串 ASIS（仅针对 Image Extender）。然后，Extender 将通过检查源来确定格式。

将 NULL 或 ASIS 用作可识别的格式: 仅当 Extender 可识别该格式时，也就是说，如果它是第 93 页的表 8 中对 Extender 列示的格式中的一种时，才指定空值、空字符串或 ASIS。否则，Extender 不能更新对象。

标识进行格式转换的更新的格式和转换选项

若在更新图像时进行格式转换，同时指定源和目标图像的格式。第 93 页的表 8 列示了允许的格式转换。

另外，还可指定转换选项，其标识要应用于更新后的图像的附加更改（如旋转或压缩）。通过参数和相关联的值指定每个转换选项。第 94 页的表 9 中列示了参数和允许的值。可通过指定多对参数 / 值来请求对更新的图像进行多处更改。

在以下示例中，更新其内容在服务器文件中的图像。更新的源使用位图（BMP）格式。在更新时，将把格式由 BMP 转换为 GIF。

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/Employee/newimg/ajones.bmp',
                'BMP',                                  /*source format*/
                'GIF',                                  /*target format*/
                '')
       WHERE NAME='Anita Jones';
```

在以下示例中，在更新时，将把同一图像转化为 GIF 格式。另外，在更新时还会把图像顺时针旋转 90 度。

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/Employee/newimg/ajones.bmp',
                'BMP',                                  /*source format*/
                'GIF',                                  /*target format*/
                '-r 1',                                  /* conversion options */
                '')
       WHERE NAME='Anita Jones';
```

更新具有用户提供的属性的对象

当更新具有用户提供的属性的图像、音频或视频对象时，必须指定更新内容的属性。在属性结构中指定属性值。必须将属性结构存储在 UDF 中 LONG VARCHAR FOR BIT DATA 变量的数据字段中。

服务器上的 UDF 代码总是期望数据处于“大尾数法格式”。大尾数法格式是大多数 UNIX 平台使用的格式。若正在存储处于“小尾数法格式”的对象，则需要准备用户提供的属性数据，以使服务器上的 UDF 代码可以正确地处理它。小尾数法格式是 Intel 和其它微处理器平台中通常使用的格式。（即使不以小尾数法格式存储对象，准备用户提供的属性数据也是一个好主意。）使用 DBiPrepareAttrs API 来准备图像对象的属性。使用 DBaPrepareAttrs API 来准备音频对象的属性。使用 DBvPrepareAttrs API 来准备视频对象的属性。

例如，C 应用程序中的下列语句更新其内容在服务器文件中的图像。该图像具有用户定义的格式，高度为 640 像素，宽度为 480 像素。注意，在更新图像之前属性已准备好。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
struct {
       short len;
       char data[400];
       }hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattr→Format,"FormatI");
pimgattr→width=640;
pimgattr→height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

DBiPrepareAttrs(pimgattr);

EXEC SQL UPDATE EMPLOYEE
       SET PICTURE=REPLACE(
                PICTURE,
                '/Employee/newimg/ajones.bmp',
         :hvStorageType,
                'Anita''s new picture',
                :ImgAttrs,                      /*user-supplied attributes*/
          CAST(NULL as LONG VARCHAR))
       WHERE NAME='Anita Jones';
```

更新缩略图（仅适用于图像和视频）

使用 Thumbnail UDF 来更新为图像或视频对象存储的缩略图（如果没有与存储的图像或视频相关的缩略图，则添加一个）。当您使用 Thumbnail UDF 时，指定正在更新其缩略图的对象的句柄，并指定更新后的（或新建的）缩略图的内容。

在程序中生成缩略图 — Extender 未提供生成缩略图的 API。您控制更新缩略图的大小和格式。在程序中创建缩略图的结构，并在 UDF 中指定缩略图结构。

例如，C 应用程序中的下列语句更新与存储的视频剪辑相关联的缩略图。

```
EXEC SQL BEGIN DECLARE SECTION;
     struct {
       short len;
       char data[10000];
     }hvThumbnail;
EXEC SQL END DECLARE SECTION;

/*Create thumbnail and store in hvThumbnail*/

EXEC SQL UPDATE Employee
   SET picture=Thumbnail(
                picture,
                :hvThumbnail)
       WHERE name='Anita Jones';
```

当更新具有用户提供的属性的图像或视频对象时，也可以更新缩略图。事实上，当您更新具有用户提供的属性的图像或视频时，必须指定缩略图作为输入。若更新对象时不想更新缩略图，则指定空值或空字符串来代替缩略图规范。

C 应用程序中的下列语句更新具有用户提供的属性的视频剪辑，并更新与该视频相关联的缩略图。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
   struct {
       short len;
       char data[400];
   }hvVidattrs;
   struct {
       short len;
       char data[10000];
   }hvThumbnail;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

MMDBVideoAttrs         *pvideoAttr;
pvideoAttr=(MMDBVideoAttrs *)hvVidattrs.data;
strcpy(pvideoAttr->cformat,"Formatv");
hvVidattrs.len=sizeof(MMDBVideoAttrs);

/* Update video content and thumbnail */

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                '/Employee/newvid/ajones.mpg',
         :hvStorageType,
                'Anita''s new video',
                :VidAttrs,
                :hvThumbnail)              /*thumbnail*/
       WHERE NAME='Anita Jones';
```

更新注释

可更新注释本身，也可在更新与该注释相关联的对象时更新它。

使用 Comment UDF 来更新注释本身。指定要更新的注释的内容以及包含对象句柄的表列。使用主变量将内容传送至服务器。例如，下列语句说明名为 hvRemarks 的主变量，并使用它来更新对存储的视频剪辑的现有注释。

```
EXEC SQL BEGIN DECLARE SECTION;
  struct {
         short len;
         char data [40];
  }hvRemarks;
EXEC SQL END DECLARE SECTION;

/* Get the old comment */

EXEC SQL SELECT COMMENT(VIDEO)
     INTO :hvRemarks
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
/* Append to old comment */

hvRemarks.data[Remarks.len]='\0';
hvRemarks.len=strlen(hvRemarks.data);
strcat (hvRemarks.data, "Updated video");
EXEC SQL UPDATE EMPLOYEE
     SET VIDEO=COMMENT(VIDEO, :hvRemarks)
     WHERE NAME = 'Anita Jones';
```

通过使用 Replace UDF，在您更新与注释相关联的对象时更新注释。例如，下列语句更新存储在服务器文件中的视频剪辑及其相关注释。

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                '/Employee/newvid/ajones.mpg',
         'MPEG1',
         :hvStorageType,
                'Anita''s new video')        /*updated comment*/
       WHERE NAME='Anita Jones';
```

第 11 章 显示或播放图像、音频或视频对象

本章描述如何使用 DB2 Extender 应用程序编程接口来显示或播放存储在数据库中的图像、音频或视频对象。

使用显示或播放 API

可使用 Extender API 来显示存储在数据库中的图像或视频帧。可显示缩略图大小格式或实际大小格式的图像或视频帧。还可使用 Extender API 来播放存储在数据库中的音频或视频对象。

使用下列 API 来显示或播放对象:

使用此 API	以
DBiBrowse	显示图像或视频帧
DBaPlay	播放音频剪辑
DBvPlay	播放视频剪辑或显示视频帧

请求任何这些 API 时，需要指定:

- 显示或播放程序的名称
- 要显示或播放的对象是以 BLOB 形式存储在数据库表中，还是存储在表中所指向的文件中
- 源文件的名称，或存储在数据库表中的句柄的名称
- 在继续之前是让应用程序等待用户关闭显示还是播放程序

标识显示或播放程序

指定要使用的图像浏览器、音频播放程序或视频播放程序的名称。在名称后面加上 %s。Extender 将用存放对象内容的文件替换 %s。

还可指定空值，而不是命名特定的显示或播放程序。在这种情况下，Extender 启动缺省图像浏览器、音频播放程序或视频播放程序，这些程序是在 DB2IMAGEBROWSER、DB2AUDIOPLAYER 或 DB2VIDEOPLAYER 环境变量中命名的。有关 DB2 Extender 如何使用环境变量的信息，参见第 475 页的附录 A，『设置 DB2 Extender 的环境变量』。

例如，C 应用程序中的以下语句启动 DB2AUDIOPLAYER 环境变量中标识的缺省音频播放程序:

```
rc = DBaPlay(
     NULL,                              /* use default audio player */
     MMDB_PLAY_FILE,
     "/Employee/Sounds/ajones.wav",
     MMDB_PLAY_NO_WAIT
     );
```

环境变量必须命名一个程序: 若（通过指定一个空值）请求缺省的显示或播放程序，确保适当的环境变量指定了显示或播放程序。若未指定程序，则 API 将返回错误代码。

© Copyright IBM Corp. 1998, 2003

指定 BLOB 或文件内容

可显示或播放以 BLOB 形式存储在数据库表中的对象，也可播放其内容存储在文件中（并从数据库表中指向）的对象。若以 BLOB 形式存储对象，则指定 MMDB_PLAY_HANDLE。若对象内容存储在文件中，则指定 MMDB_PLAY_FILE。MMDB_PLAY_HANDLE 和 MMDB_PLAY_FILE 是由 Extender 定义的常量。

例如，C 应用程序中的以下语句播放其内容在文件中的视频:

```
rc = DBvPlay(
     "explore %s",
     MMDB_PLAY_FILE,                    /* content in file */
     "/Employee/Videos/ajones.mpg",
     MMDB_PLAY_NO_WAIT
     );
```

显示和播放程序通常接受来自文件的输入。若指定 MMDB_PLAY_FILE，则 Extender 将使用环境变量中的值来解析文件的相对文件名和路径。然后，Extender 启动浏览程序并将文件名传送给它。若指定 MMDB_PLAY_HANDLE，则 Extender 从句柄中抽取文件名（倘若文件名非空的话）。若句柄中的文件名为空，则表示对象以 BLOB 形式存储。Extender 将在客户机中创建一个临时文件，并将对象的内容从数据库表复制至该客户机文件。然后，Extender 将启动程序，并将存放内容的文件（或临时文件）的名称传送给它。

例如，C 应用程序中的下列语句获取以 BLOB 形式存储的图像的句柄，并使用该句柄来显示图像:

```
EXEC SQL BEGIN DECLARE SECTION;
char hvImg_hdl [251];
EXEC SQL END DECLARE SECTION;

rc = DBiBrowse (
     "ib %s",
     MMDB_PLAY_HANDLE,                  /* content is BLOB */
     hvImg_hdl,
     MMDB_PLAY_NO_WAIT
     );
```

该内容必须是可存取的: 确保显示或播放程序可存取对象内容。若内容在服务器文件中，但程序需要客户机上的内容，则将该文件复制或客户机文件或使用 Content UDF。若内容是以 BLOB 形式存储的，则 Extender 将自动将其检索至客户机。

指定等待指示符

指定在应用程序继续之前（即，在 DBiBrowse、DBaPlay 或 DBvPlay API 返回代码之前），是让应用程序等待用户结束显示还是播放程序。若要让应用程序等待，则指定 MMDB_PLAY_WAIT。若不想让应用程序等待，则指定 MMDB_PLAY_NO_WAIT。MMDB_PLAY_WAIT 和 MMDB_PLAY_NO_WAIT 是由 Extender 定义的常量。

若指定 MMDB_PLAY_WAIT，则显示或播放程序将在应用程序所在的线程或进程中运行。若指定 MMDB_PLAY_NO_WAIT，则显示或播放程序将在其自己的线程或进程中运行，独立于您的应用程序。

例如，作为以下语句的结果，在应用程序继续之前，应用程序将等待用户关闭图像浏览器:

```
rc = DBiBrowse (
     "explore %s",
     MMDB_PLAY_FILE,
     "/Employee/images/ajones.bmp",
     MMDB_PLAY_WAIT                    /* wait for browser to close */
     );
```

若指定 DBxPlay 和 MMDB_PLAY_NO_WAIT，则务必小心：发出 DBaPlay 或 DBvPlay 时，若下列任何一项为真，则 Extender 将创建一个临时文件:

- 对象以 BLOB 形式存储
- 不能使用环境变量中的值解析相对文件名
- 不能在客户机上存取该文件

该临时文件是在 TMP 环境变量指定的目录中创建的。若指定 MMDB_PLAY_WAIT，Extender 将在播放对象之后删除该临时文件。但是，若指定 MMDB_PLAY_NO_WAIT，将不删除该临时文件您将必须自己删除该临时文件。

显示缩略图大小的图像或视频帧

缩略图是存储的图像或视频帧的微型图格式。当将图像存储在数据库中时，Image Extender 将图像的缩略图存储在属性表中。当将视频存储在数据库中时，Video Extender 将表示视频对象的类属缩略图存储在属性表中。

在缺省情况下，Image Extender 自动创建的图像缩略图的大小约为 112 x 84 像素。Video Extender 插入的类属视频缩略图的大小是 108 x 78 像素。图像缩略图和类属视频缩略图都以 GIF 格式存储。根据图像或视频帧中数据的密度，这对应于大概 4.5 KB 到 5 KB 的数据。若存储或更新具有用户提供的属性的图像或视频，可指定您选择的大小和格式的缩略图。

在 SQL SELECT 语句中使用 Thumbnail UDF 来从数据库检索缩略图。使用文件引用变量来将缩略图传送至文件。指定 UDF 时，需要指定数据库表中包含图像或视频句柄的列的名称。然后使用 DBiBrowse API 来显示图像或视频帧缩略图。

例如，下列语句检索缩略图图像，然后显示它:

```
long rc, outCount;
char Thumbnail_filename[254];
FILE *file_handle;

EXEC SQL BEGIN DECLARE SECTION;
         struct  {
         short len
       char data[10000];
         }Thumbnail_buffer;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT THUMBNAIL(PICTURE)
     INTO :Thumbnail_buffer
  FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
strcpy (Thumbnail_filename,"/tmp/ajones.tmb");
file_handle=fopen(Thumbnail_filename,"wb+");
outCount=fwrite(Thumbnail_buffer.data, 1, Thumbnail_buffer.len, file_handle);
fclose(file_handle);
rc = DBiBrowse (
               NULL,                        /* use the default display program */
```

```
MMDB_PLAY_FILE,             /* thumbnail image in file */
Thumbnail_filename,         /* thumbnail image content */
MMDB_PLAY_WAIT);            /* wait for user to finish */
```

显示实际大小的图像或视频帧

使用 DBiBrowse API 来显示存储在数据库表中的图像。参见第 121 页的『使用显示或播放 API』以获取有关使用此 API 的详细信息。

使用 DBvGetNextFrame API 或 DBvSeekFrame API 来获取实际大小的视频帧。帧以 YUV 格式存储在缓冲区中，且可使用 DBvFrameDatato24BitRGB API 将其转换为 RGB 格式。将头文件附加至已转换的帧（例如，附加 BMP 文件类型头文件），并将头文件和帧数据写至文件。然后使用 DBiBrowse API 来显示文件的内容。参见第 17 页的『检测视频画面切换』以获取关于使用 DBvGetNextFrame、DBvSeekNextFrame 和 DBvFrameDatato24BitRGB API 的详细信息，并获取关于显示视频帧的进一步的信息。

播放音频或视频

使用 DBaPlay API 来播放存储在数据库表中的音频。使用 DBvPlay API 来播放存储在数据库表中的视频。参见第 121 页的『使用显示或播放 API』以获取有关使用这些 API 的详细信息。

第 12 章 按内容查询图像

图 27 显示了一个允许用户在数据库中搜索图像的应用程序，它使用可视示例作为搜索标准，即一个有突出颜色或纹理图案的图像。借助于这样的应用程序，用户可提供图像作为搜索的输入。然后，应用程序将源图像的颜色或纹理与存储的图像的颜色或纹理相对照，并返回颜色或纹理与输入最接近的图像。

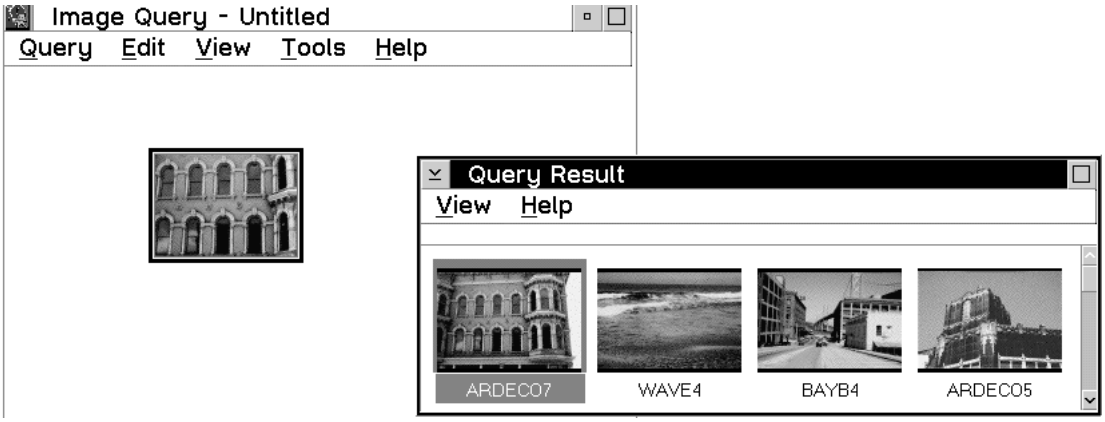


图 27. 按图像内容查询. 使用可视示例的颜色或纹理来搜索存储在数据库表中的图像。

这种按可视特征来查询图像的能力称为**按图像内容查询（QBIC）** [6]。本章描述如何使用随 Image Extender 一起提供的 API 和 UDF，来构建与刚才描述的应用程序类似的应用程序。它还描述如何使用随 Image Extender 一起提供的命令和 API 来执行 QBIC 管理任务。

如何按图像内容查询

要按图像内容查询:

1. 为这些图像创建一个 QBIC 目录。
2. 编目这些图像。这表示将图像的条目添加至目录中，并存储图像特征值。

   参见第 48 页的『QBIC 目录』以获取 QBIC 目录和图像特征的描述。
3. 构建一个查询。该查询标识要用作搜索标准的特征、它们的值及其权重（即，对每一特征的重视程度）。可以在称为查询字符串的字符串中指定这些查询属性。或者，也可以创建一个查询对象并将这些属性与查询对象相关联。之后，便可以保存该查询字符串并重新使用它。
4. 运行该查询。当运行查询时，指定查询字符串作为输入，或标识要查询的查询对象。在任一情况下，都要标识要搜索的图像。在任一情况下，都可从 DB2 命令行或从程序中提交查询。

作为响应，Image Extender 计算查询的特征值。它将该值与存储在 QBIC 目录中的目标图像的特征值作比较。然后 Image Extender 计算得分，得分指示每一目标图像的特征值与源图像的相应值有多类似。

6. Image Extender 包括加州大学伯克利分校及其程序编写者开发的软件。

© Copyright IBM Corp. 1998, 2003

可让 Image Extender 返回其特征值最接近于源的图像。还可以让 Image Extender 返回一个或多个图像的得分。

管理 QBIC 目录

在可以按内容查询图像之前，必须将图像编目到 QBIC 目录中。QBIC 目录存放关于图像的可视特征的数据。

为用户表中您希望可按内容查询的图像的每一列，创建一个 QBIC 目录。对于用户表中图像的每一列，不能有多个 QBIC 目录，且多列不能共享同一 QBIC 目录。

在您创建 QBIC 目录时，您标识要让 Image Extender 存储其数据的特征。还指示是否要让 Image Extender 自动编目图像。自动编目表示在将图像存储到用户表中，Image Extender 将自动在目录中创建图像的条目。若不自动编目图像，则您必须手工编目它。这表示您明确地要求 Image Extender 在目录中创建图像的条目。

在创建 QBIC 目录之后，可以:

- 打开该目录，以便对它执行后续操作
- 将自动编目的设置更改为手工编目，或将手工更改为自动
- 将特征添加至目录，这标识要让 Image Extender 存储其数据的特征
- 从目录中除去特征
- 检索关于目录的信息，如与该目录相关联的用户表和列的名称，或目录中存储其数据的特征
- 将图像手工编目到目录中
- 撤消编目图像（即，从目录中除去图像的条目）
- 重新编目图像
- 再分发该目录（即，在分区数据库系统中添加或删除节点）
- 关闭目录
- 删除目录

可以使用 Image Extender 提供的 API 来执行这些任务，包括创建 QBIC 目录。还可以使用 db2ext 命令行处理器来执行其中许多任务。

创建 QBIC 目录

使用 QbCreateCatalog API 或 CREATE QBIC CATALOG 命令来创建 QBIC 目录。要创建目录，您必须是将要编目的图像所属的用户表的所有者。另外，您必须对将包含目录的数据库具有 CREATE TABLE 权限。在创建图像列中的图像的 QBIC 目录之前，必须对 Image Extender 启用用户表和该列。

当创建 QBIC 目录时，您:

- 命名包含要编目的图像的用户表和列。
- 指示是否将自动编目图像。自动编目表示在将图像存储在用户表中之后，Image Extender 将编目该图像。Extender 定期进行检查，查看是否有正等待编目的图像。可通过设置环境变量 DB2CATALOGDELAY 的值来指定一个检查周期（以秒计）。此值可以设置为从 1 秒到一个极大的值的范围内。缺省值是 60 秒。

手工编目表示您明确请求 Image Extender 编目图像。（参见第 131 页的『手工编目图像』 以获取有关如何人工编目图像的信息。）

必须启用用户表和列: 在为图像列中的图像创建 QBIC 目录之前，必须对 Image Extender 启用用户表和该列。（有关如何对 Image Extender 启用用户表和列的信息，参见第 69 页的第 6 章，『准备 Extender 数据的数据对象』。）

使用 API: 使用 QbCreateCatalog API 时，您通过指定自动编目值来指示自动或手工编目。值 1 指示自动编目；值 0 指示手工编目。

例如，下列语句为 employee 表的 picture 列中的图像创建一个 QBIC 目录。当将图像存储在 employee 表中时，将自动进行编目:

```
SQLINTEGER autoCatalog=1;                          /* automatic cataloging */

rc=QbCreateCatalog(
                 "employee",                      /* user table */
                 "picture",                       /* image column */
                  autoCatalog);                    /* auto catalog setting */
```

使用命令行: 发出 CREATE QBIC CATALOG 命令时，可通过指定 ON 指示进行自动编目。指定 OFF 则指示进行手工编目。OFF 是缺省值。

例如，以下命令创建与 API 示例中相同的 QBIC 目录:

```
CREATE QBIC CATALOG employee picture on
```

备份 QBIC 目录: Image Extender 将 QBIC 目录存储在文件中。应定期地备份这些文件，以备恢复目录之用。在 AIX、HP-UX 或 Sun Solaris 服务器中，这些文件位于 /home/*instance_owner*/dmb/qbic 目录中，其中，*instance_owner* 是实例所有者的用户标识。在 Windows 服务器中，这些文件位于 *destination*\instance*instance_name*\qbic 目录中，其中 *destination* 是在其中安装 Image Extender 的目录，而 *instance_name* 是 Extender 实例的名称。

打开 QBIC 目录

需要打开 QBIC 目录才可执行更改目录的后续操作。例如，在将特征添加至目录之前，需要打开 QBIC 目录。

要打开 QBIC 目录，使用 QbOpenCatalog API 调用或 OPEN QBIC CATALOG 命令。打开 QBIC 目录时，您:

- 命名该目录的用户表和图像列。
- 指定打开目录的方式（当使用命令 OPEN QBIC CATALOG 时，这是隐式的）。可以打开目录，执行读取目录的操作，如按内容搜索图像。也可以打开目录，执行更新目录的操作，如添加特征。您必须对用户表具有 SELECT 权限才可为读操作打开目录。您必须对用户表具有 UPDATE 权限才可为更新操作打开目录。

目录已打开怎么办? 若在另一会话中已为更新操作打开目录，则不能为更新操作打开该目录。当您打开 QBIC 目录时，Image Extender 关闭当前会话中已打开的任何 QBIC 目录。

使用 API: 使用 QbOpenCatalog API 时，明确地指定要打开目录的方式。指定:

- API 参数 qbiRead，以打开目录，执行读操作。
- API 参数 qbiUpdate，以打开目录，执行更新操作。

QbiRead 和 QbiUpdate 是 QBIC 的包含（头）文件 dmbqbapi.h 中定义的常量。

还需要指向目录句柄。目录句柄具有 QBIC 特有的数据类型 QbCatalogHandle。此数据类型也是在 dmbqbapi.h 中定义的。Image Extender 返回句柄值作为 API 的输出。

例如，以下 API 调用打开 QBIC 目录以执行读操作:

```
SQLINTEGER mode;
QbCatalogHandle  *CatHdl;

mode=qbiRead;                                   /* open catalog for */
                                                /* read operations */
rc=QbOpenCatalog(
                 "employee",                    /* user table */
                 "picture",                     /* image column */
                 mode,                          /* open catalog mode */
                 &CatHdl);                      /* catalog handle */
```

使用命令行: 当您发出 OPEN QBIC CATALOG 命令时，Image Extender 尝试为更新操作打开目录。若另一会话中当前已为更新操作打开该目录，则 Image Extender 为读操作打开该目录。

例如，以下语句打开 QBIC 目录: Image Extender 尝试为更新操作打开它:

```
OPEN QBIC CATALOG employee picture
```

当与 QBIC 相关的活动完成时，关闭目录: 当您打开 QBIC 目录时，Image Extender 对其分配资源，如内存。当与 QBIC 相关的活动完成时，关闭目录。这将释放分配的资源。

更改自动编目设置

使用 QbSetAutoCatalog API 或 SET QBIC AUTOCATALOG 命令从自动编目更改为手工编目，或从手工更改为自动。在更改目录设置之前，必须为更新操作打开 QBIC 目录。

更改不可回溯: 当您更改自动编目设置时，该更改仅应用于在更改之后添加至用户表列的图像。已存储在用户表列中的图像不受影响。例如，若将设置由手工编目更改为自动编目，将只对在更改之后添加至用户表列的图像进行自动编目。若要编目已在表列中的图像，则需要进行手工编目。（参见第 131 页的『手工编目图像』 以获取有关如何人工编目图像的信息。）

使用 API: 在使用 QbSetAutoCatalog API 时，指定 QBIC 目录的句柄（此句柄是在您用 QbOpenCatalog API 打开目录时返回的）。另外还指定自动编目值 1（表示自动编目）或值 0（表示手工编目）。

在以下示例中，对与 employee 表的 picture 列中的图像相关联的 QBIC 目录指定了手工编目。注: 首先为更新目录的操作打开 QBIC 目录。

```
SQLINTEGER mode;
SQLINTEGER autoCatalog=0;                       /* manual cataloging */

QbCatalogHandle  *CatHdl;

mode=qbiUpdate;                                 /* open catalog for */
                                                /* update */
/* Open a QBIC catalog */
rc=QbOpenCatalog(
                 "employee",                    /* user table */
```

```
                "picture",                      /* image column */
                mode,                           /* open catalog mode */
                &CatHdl);                       /* catalog handle */

/* Change the auto catalog setting */
rc=QbSetAutoCatalog(
               CatHdl,                             /* catalog handle */
                 autoCatalog);                 /* auto catalog flag */
```

使用命令行: 发出 SET QBIC AUTOCATALOG 命令时，可指定 ON 指示进行自动编目。指定 OFF 则指示进行手工编目。此命令对当前打开的目录进行操作。

例如，以下命令对当前打开的 QBIC 目录将自动编目设置为 off:

```
SET QBIC AUTOCATALOG off
```

将特征添加至 QBIC 目录

使用 QbAddFeature API 或 ADD QBIC FEATURE 命令将特征添加至 QBIC 目录。必须将至少一个特征添加至 QBIC 目录中，才可以在该目录中编目图像。在可添加特征之前，必须为更新操作打开 QBIC 目录。

将特征添加至目录时，指定要添加的特征的名称（特征名列示在表 11 中）。

表 11. QBIC 特征名

特征名	描述
QbColorFeatureClass	平均颜色
QbColorHistogramFeatureClass	直方图颜色
QbDrawFeatureClass	位置颜色
QbTextureFeatureClass	纹理

可能必须重新编目图像: 当您将特征添加至 QBIC 目录时，Image Extender 将不会自动存储关于已编目图像的新特征的数据，即使打开了自动编目也是如此。要包括关于已编目图像的新特征的数据，需要重新编目这些图像（参见第 133 页的『重新编目图像』）。

使用 API: 在使用 QbAddFeature API 时，除指定特征名外，还需指定 QBIC 目录的句柄。注意，使用了常量 qbiMaxFeatureName 来表示特征名的长度。该常量在 QBIC 的包含（头）文件 dmbqbapi.h 中定义，值为 50。

在以下示例中，使用了 QbAddFeature API 来将直方图颜色特征添加至 QBIC 目录:

```
char  featureName[qbiMaxFeatureName];

QbCatalogHandle   CatHdl;

strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbAddFeature(
               CatHdl,                          /* catalog handle */
               featureName);                    /* feature name */
```

使用命令行: ADD QBIC FEATURE 命令对当前打开的目录进行操作。在以下示例中，使用了该命令将位置颜色特征添加至当前打开的目录中:

```
ADD QBIC FEATURE QbDrawFeatureClass
```

从 QBIC 目录中除去特征

使用 QbRemoveFeature API 或 REMOVE QBIC FEATURE 命令从 QBIC 目录中除去特征。Image Extender 删除该特征的目录表。结果是，在您编目图像时将不存储该特征的数据。在除去特征之前，必须为更新操作打开 QBIC 目录。

当从目录中除去特征时，指定要除去的特征的名称。

使用 API: 在使用 QbRemoveFeature API 时，除指定特定名外，还需要指定 QBIC 目录的句柄。

在以下示例中，使用了 QbRemoveFeature API 从 QBIC 目录中除去直方图颜色特征:

```
char  featureName[qbiMaxFeatureName];

QbCatalogHandle   CatHdl;

strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbRemoveFeature(
                CatHdl,                          /* catalog handle */
                featureName);                    /* feature name */
```

使用命令行: REMOVE QBIC FEATURE 命令对当前打开的目录进行操作。在以下示例中，使用了该命令从当前打开的 QBIC 目录中除去位置颜色特征:

```
REMOVE QBIC FEATURE QbDrawFeatureClass
```

检索关于 QBIC 目录的信息

可检索关于 QBIC 目录的下列信息:

- 与该目录相关联的用户表和图像列的名称。
- 目录中存储了其数据的特征的数目及其名称。
- 当图像存储在用户表中时，Image Extender 是否进行自动编目。

使用 QbGetCatalogInfo API 来检索用户表和列名、特征数和自动编目设置。使用 QbListFeatures API 来检索特征名。或使用 GET QBIC CATALOG INFO 命令来检索所有信息。

在可检索信息之前，必须打开 QBIC 目录。

使用 API: 在使用 QbGetCatalogInfo API 时，需要指定 QBIC 目录的句柄。还需要指向 Image Extender 返回的目录信息所在的结构。目录信息结构在 QBIC 的包含（头）文件 dmbqapi.h 中定义，如下所示:

```
typedef struct{
          char       tableName[qbiMaxTableName+1]    /* user table */
          char       columnName[qbiMaxColumnName+1]  /* image column */
          SQLINTEGER featureCount;                   /* number of features */
          SQLINTEGER autoCatalog;                    /* auto catalog flag */
} QbCatalogInfo;
```

在发出 QbListFeatures API 调用时，需要分配缓冲区来存放返回的特征名。存储在缓冲区中的特征名由空白字符分隔。还需要指定目录句柄，以及存放返回的特征名的缓冲区的大小。要估计所需的缓冲区大小，可使用 QbGetCatalogInfo API 返回的特征计数，并将该计数乘以最长的特征名的长度。可使用常量 qbiMaxFeatureName 作为最长特征名的大小。

以下示例中的 API 调用检索关于 QBIC 目录的信息。注意如何使用 QbGetCatalogInfo API 返回的特征数和 qbiMaxFeatureName 常量来计算 QbListFeatures API 的缓冲区大小:

```
long  bufSize;
long  count;
char  *featureNames;

QbCatalogHandle   CatHdl;
QbCatalogInfo    catInfo;

/* Get user table name, image column name, feature count, */
/* and auto catalog setting */

rc=QbGetCatalogInfo(
                CatHdl,                              /* catalog handle */
                &catInfo);                           /* catalog info. structure */

/* List feature names */

bufSize=catInfo.featureCount*qbiMaxFeatureName;
featureNames=malloc(bufSize);

rc=QbListFeatures(
                CatHdl,                              /* catalog handle */
                bufSize                             /* size of buffer */
                count,                               /* feature count */
                featureNames);                       /* buffer for feature names */
```

使用命令行: GET QBIC CATALOG INFO 对当前打开的目录进行操作。在以下示例中，使用了此命令来检索关于当前打开的 QBIC 目录的信息:

```
GET QBIC CATALOG INFO
```

手工编目图像

在创建目录时，您应指示当图像存储在用户表中时，是否要让 Image Extender 自动编目该图像。若不自动编目图像，则当图像存储在用户表中之后，您必须进行手工编目。可以手工编目单个图像，也可以编目整个图像列。

手工编目单个图像

使用 QbCatalogImage API 来手工编目单个图像。因为没办法在命令行上标识个别图像，所以不能通过命令来编目单个图像。在使用该 API 时，指定目录句柄和图像句柄（可从用户表中检索图像句柄）。在手工编目图像之前，必须打开 QBIC 目录。

例如，下列语句从用户表中检索图像句柄，然后编目图像:

```
/* Retrieve the image handle */

EXEC SQL BEGIN DECLARE SECTION;
char Img_hdl[251];
EXEC SQL END DECLARE SECTION;

QbCatalogHandle   CatHdl;

EXEC SQL SELECT PICTURE INTO :Img_hdl
  FROM EMPLOYEE
  WHERE NAME='Anita Jones';

/* Catalog the image*/
```

```
rc=QbCatalogImage(
                CatHdl,                              /* catalog handle */
                Img_hdl);                            /* image handle */
```

手工编目图像列

使用 QbCatalogColumn API 或 CATALOG QBIC COLUMN 命令来手工编目图像列。Image Extender 仅编目列中在上次编目该列后新插入、更新或删除的图像。Image Extender 将那些图像的所有特征都编目到目录中。在手工编目图像列之前，必须为更新操作打开 QBIC 目录。

使用 API: 在使用 QbCatalogColumn API 时，指定目录句柄。Image Extender 使用与指定目录相关联的用户表列中的图像。

例如，以下 API 调用编目与指定目录相关联的用户表列中未编目的图像。将把图像的所有特征都编目到该目录中:

```
QbCatalogHandle   CatHdl;

rc=QbCatalogColumn(
                 CatHdl);                             /* catalog handle */
```

使用命令行: 使用 CATALOG QBIC COLUMN 命令来手工编目图像列。还可使用该命令来重新编目图像（参见第 133 页的『重新编目图像』）。指定参数 FOR 和 NEW。（FOR 和 NEW 是缺省参数。）

在以下示例中，使用了该命令来编目与当前打开的目录相关联的表列中未编目的图像。将把图像的所有特征都编目到该目录中:

```
CATALOG QBIC COLUMN FOR NEW
```

撤消编目图像

撤消编目图像表示从 QBIC 目录中除去图像的条目。使用 QbUncatalogImage API 来撤消编目图像。因为没办法在命令行上标识个别图像，所以不能通过命令来撤消编目图像。在使用该 API 时，指定目录句柄和图像句柄（可从用户表中检索图像句柄）。在撤消编目图像之前，必须为更新操作打开 QBIC 目录。

例如，下列语句从用户表中检索图像句柄，然后撤消编目图像:

```
/* Retrieve the image handle */

EXEC SQL BEGIN DECLARE SECTION;
char Img_hdl[251];
EXEC SQL END DECLARE SECTION;

QbCatalogHandle   CatHdl;

EXEC SQL SELECT PICTURE INTO :Img_hdl
  FROM EMPLOYEE
  WHERE NAME='Anita Jones';

/* Uncatalog the image */

rc=QbUncatalogImage(
                CatHdl,                              /* catalog handle */
                Img_hdl);                            /* image handle */
```

重新编目图像

当编目图像时，Image Extender 会分析对 QBIC 目录标识的图像的特征，并将那些特征值存储在目录中。当您把特征添加至 QBIC 目录时，Image Extender 不会自动分析已编目图像的新特征。要将新特征的值添加至目录，需要重新编目所有图像。

使用 QbReCatalogColumn API 或 CATALOG QBIC COLUMN 命令来重新编目 QBIC 目录中的图像。Image Extender 除去当前在目录中的所有特征数据。然后，它分析图像的所有特征，包括任何新特征，并编目图像。在重新编目图像之前，必须打开 QBIC 目录。

使用 API：使用 QbReCatalogColumn API 时，指定目录句柄。

在以下示例中，将重新分析 QBIC 目录中的图像:

```
QbCatalogHandle   CatHdl;

rc=QbReCatalogColumn(
                 CatHdl);                              /* catalog handle */
```

使用命令行：使用 CATALOG QBIC COLUMN 命令来重新编目图像。此命令对当前打开的目录进行操作。还可以使用命令来手工编目图像（参见第 131 页的『手工编目图像』）。

在发出此命令时，指定参数 FOR 和 ALL。这告诉 Image Extender 您要重新编目所有图像。

在以下示例中，将重新编目当前打开的 QBIC 目录中的已编目图像:

```
CATALOG QBIC COLUMN FOR ALL
```

再分发 QBIC 目录（仅限于 EEE）

当在节点组中添加或除去节点时，使用 DMBRedistribute API 或 REDISTRIBUTE NODEGROUP 命令来再分发 QBIC 特征数据。此命令将 QBIC 特征数据放在对应的用户数据所在的节点上。

若再分发进程返回一个错误，可重新运行该命令，并根据命令响应提供的指示信息指定或不指定 CONTINUE 参数。此选项指导系统从停止的地方继续，而不是从头开始。在运行 DB2 的 REDISTRIBUTE 命令之后，如果是首次运行 REDISTRIBUTE NODEGROUP 命令，不应使用 CONTINUE 参数。

要维护数据完整性，一次再分发一个节点组。在启动另一再分发之前，等待一个节点组完成再分发。

使用 API：以下示例显示如何在称为 groupone 的节点组中再分发 QBIC 特征数据:

```
                #include <dmbrdst.h>

        rc = DMBRedistribute(groupone,"continue");
```

使用命令行：以下示例显示如何使用 REDISTRIBUTE NODEGROUP 命令并指定 CONTINUE 参数来再分发称为 my_nodegroup 的节点的数据:

```
redistribute nodegroup my_nodegroup continue
```

关闭 QBIC 目录

使用 QbCloseCatalog API 或 CLOSE QBIC CATALOG 命令来关闭 QBIC 目录。在关闭目录之前，必须先打开它。

使用 API: 在发出 QbCloseCatalog API 调用时，指定目录句柄。例如:

```
QbCatalogHandle   CatHdl;

rc=QbCloseCatalog(
                 CatHdl);                          /* catalog handle */
```

使用命令行: CLOSE QBIC CATALOG 命令对当前打开的目录进行操作。在以下示例中，使用此命令来关闭当前打开的 QBIC 目录:

```
CLOSE QBIC CATALOG
```

删除 QBIC 目录

删除 QBIC 目录将删除目录表中的所有特征数据。结果是，再也不能对相关联的图像进行“按内容查询”。要删除 QBIC 目录，必须对与该目录相关联的表具有 ALTER 或 CONTROL 权限。在删除目录之前，必须先打开它。

使用 QbDeleteCatalog API 或 DELETE QBIC CATALOG 命令来删除 QBIC 目录。在删除 QBIC 目录时，命名与该目录相关联的用户表和列。

使用 API: 在以下示例中，将使用 QbDeleteCatalog API 来删除 QBIC 目录:

```
rc=QbDeleteCatalog(
                 "employee",                    /* user table */
                  "picture");                    /* image column */
```

使用命令行: DELETE QBIC CATALOG 命令对当前打开的目录进行操作。在以下示例中，使用此命令来删除当前打开的 QBIC 目录:

```
DELETE QBIC CATALOG employee picture
```

QBIC 目录样本程序

第 135 页的图 28 显示了用 C 语言编写的，用于创建 QBIC 目录的程序的一部分。该程序还将一列图像编目到 QBIC 目录中。可在 SAMPLES 子目录中的 QBCATDMO.C 文件中找到完整的程序。在运行整个程序之前，必须运行 ENABLE 和 POPULATE 样本程序（也可在 SAMPLES 子目录中找到）。有关样本程序的更多信息，参见第 481 页的附录 B, 『样本程序和媒体文件』。

注意程序中的下列几点:

1. 包括 dmbqbapi 头文件。
2. 连接到数据库。
3. 创建目录。在自动编目关闭的情况下创建目录。
4. 为更新操作打开目录。
5. 将平均颜色特征添加至该目录。
6. 编目图像列。
7. 关闭目录。

```
#include <sql.h>
#include <sqlcli.h>
#include <sqlcli1.h>
#include <dmbqbqpi.h> 1
#include <stdio.h>

/*****************************************************************************/
/*  Define the function prototypes                                           */
/*****************************************************************************/

void printError(SQLHSTMT hstmt);
void createCatalog();
void openCatalog();
void closeCatalog();
void addFeature();
void catalogImageColumn();

QbCatalogHandle cHdl = 0;

static  SQLHENV    henv;
static  SQLHDBC    hdbc;
static  SQLHSTMT   hstmt;
static  SQLRETURN  rc;
char tableName[] = "sobay_catalog";
char columnName[] = "covers";

SQLCHAR uid[18+1];
SQLCHAR pwd[30+1];
SQLCHAR dbnName[SQL_MAX_DSN_LENGTH+1];

void main ()
{
/*---- prompt for database name, userid, and password ----*/
  printf("Enter database name:\n");
  gets((char *) dbName);
  printf("Enter userid:\n");
  gets((char *) pwd);
/* set up the SQL CLI environment */
 SQlAllocEnv(&henv);
 SQLAllocConnect(henv, &hdbc);
 rc = SQLConnect(hdbc,dbname,SQL_NTS,uid,SQL_NTS,pwd,SQL_NTS); 2
 if (rc != SQL_SUCCESS)
 {
  printError(SQL_NULL_HSTMT);
  exit(1);
 }
```

图 28. QBIC 目录样本程序 (1/4)

```
 createCatalog();
   openCatalog();
   addFeature();
 getCatalogInfo();
 listFeatures();
 catalogImageColumn();
 closeCatalog();

 SQLDisconnect(hdbc);
 SQLFreeConnect(hdbc);
 SQLFreeEnv(henv);
}
/*********************************************************************/
void createCatalog()
{
 SQLINTEGER autoCatalog = 0;
 SQLINTEGER retLen;
   SQLINTEGER errCode = 0;
   char errMsg[500];

   QbCreateCatalog(  3
         (char *) tableName,
         (char *) columnName,
         autoCatalog,
         0
        );

    DBiGetError(&errCode, errMsg);
    if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
}
/*********************************************************************/
void openCatalog()
{
   SQLINTEGER errCode = 0;
   char errMsg[500];
 SQLINTEGER mode = qbiUpdate;

   QbOpenCatalog(  4
         (char *) tableName,
         (char *) columnName,
         mode,
         &cHdl
        );

    DBiGetError(&errCode, errMsg);
    if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
}
```

图 28. QBIC 目录样本程序 (2/4)

```
/*********************************************************************/
void addFeature()
{
 SQLINTEGER errCode=0;
 char errMsg[5
 if(cHdl) /* if we have an open catalog, else do nothing */
 {
   char featureName*lbrk.] = "QbColorFeatureClass"; 5
     QbaddFeature(
         cHdl,
         featureName
        );

    DBiGetError(&errCode, errMsg);
    if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
 }
else
 {
  exit(1);
 }
}
/*********************************************************************/
void catalogImageColumn()
{
   SQLINTEGER errCode = 0;
   char errMsg[500];

 if(cHdl) /* if we have an open catalog, else do nothing */
 {
  SQLRETURN rc;
    QbCatalogColumn( 6
         cHdl,
        );

    DBiGetError(&errCode, errMsg);
    if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);

else
 {
  exit(1);
 }
}
```

图 28. QBIC 目录样本程序 (3/4)

```
/*********************************************************************/
void closeCatalog()
{
 if(cHdl) /* if we have an open catalog, else do nothing */
 {
    QbCloseCatalog( 7
         cHdl,
        );
 }
}
/*********************************************************************/
```

图 28. QBIC 目录样本程序 (4/4)

构建查询

在按内容查询图像时，需标识查询的输入以及一组目标编目图像。查询的输入指定要在查询中使用的特征的名称、特定值和特征权重（即，对每个特征的重视程度）。

有两种方法来提供此输入:

- 在查询中指定查询字符串。查询字符串是为查询指定特征、特征值和特征权重的字符串。
- 创建查询对象，并在查询中引用它。查询对象指定特征和特征权重。它还标识每一特征的数据源。数据源提供了每一特征的值。

指定查询字符串

可使用查询字符串来标识要查询的特征、特征值和特征权重。**查询字符串**是格式为 *feature_name value* 的字符串，其中 *feature_name* 是 QBIC 特征名，而 *value* 是与特征相关联的值。

可在一个查询中指定多个特征。然后，对每个特征指定一对特征名 - 值，如『特征值』中所述。各对之间用子句 AND 分隔。当在一个查询中指定多个特征时，还可对一个或多个特征指定权重，如第 139 页的『特征权重』中所述。这样，查询字符串就具有格式 *feature_name value weight*，其中 *weight* 是对该特征指定的权重。

Image Extender 提供了使用查询字符串的一个 API（QbQueryStringSearch）和两个 UDF（QbScoreFromStr 和 QbScoreTBFromStr）。在发出查询时，使用适当的 API 或 UDF，并将查询字符串指定为输入参数。（有关详细信息，参见第 145 页的『按图像内容发出查询』。）

特征值

在查询字符串中指定查询中各个特征的特征值。

当在 DB2 命令内传送查询时，必须遵循查询的某些命名约定才能使查询正确工作。必须将包含空格或右尖括号（>）的文件名括在双引号中；其它文件名可以括在双引号中，也可以不括在双引号中。如果使用引号来括住文件名，则必须在每个引号前面放一个转义字符（\）。若不在 DB2 命令内传送查询，则无需将转义字符与引号包括在一起。

在以下示例中，DB2 命令中传送了查询字符串:

```
db2 "select image_id from table
(mmdbsys.QbScoreTBFromStr
('texture file=<server,patterns/ptrn07.gif>',
'fabric',
'swatch_img',
10))
as T1"
```

第 139 页的表 12 列示可为每个特征指定的值：在每个特征名下面的字符串是可替代它的缩写格式。

表 12. 可在查询字符串中指定的特征值

特征名	值
averageColor，average 或 QbColorFeatureClass	color=<*Rvalue*，*Gvalue*，*Bvalue*> 每个颜色值都是 0 至 255 的整数，标识图像的红色值（*Rvalue*）、绿色值（*Gvalue*）和蓝色值（*Bvalue*）。 file=<*file_location*，*filename*> 对于服务器文件，*file_location* 是服务器。*filename* 是以适合于文件所驻留的系统的格式指定的完整文件路径，或是相对文件名。DB2 Extender 使用环境变量解析相对文件名（参见第 475 页的『如何使用环境变量来解析文件名』）。
histogram，histogramcolor 或 QbColorHistogramFeatureClass	histogram=<(*hist_value*，*Rvalue*，*Gvalue*，*Bvalue*>)，... 每个直方图颜色值都在子句中指定，子句标识该颜色在直方图中的百分比（1 至 100）（*hist_value*），以及该颜色的红色值（*Rvalue*）、绿色值（*Gvalue*）和蓝色值（*Bvalue*）。 file=<*file_location*，*filename*> 对于服务器文件，*file_location* 是服务器。*filename* 是以适合于文件所驻留的系统的格式指定的完整文件路径，或是相对文件名。DB2 Extender 使用环境变量解析相对文件名。
draw，positional 或 QbDrawFeatureClass	file=<*file_location*，*filename*> handle=<*image_handle*> 对于服务器文件，*file_location* 是服务器。*filename* 是以适合于文件所驻留的系统的格式指定的完整文件路径，或是相对文件名。DB2 Extender 使用环境变量解析相对文件名。
纹理或 QbTextureFeatureClass	file=<*file_location*，*filename*> handle=<*image_handle*> 对于服务器文件，*file_location* 是服务器。*filename* 是以适合于文件所驻留的系统的格式指定的完整文件路径，或是相对文件名。DB2 Extender 使用环境变量解析相对文件名。

特征权重

若在查询字符串中指定多个特征，则还可对其中一个或多个特征指定权重。特征权重指示，当 Image Extender 计算类似性得分并返回 “按图像内容查询” 的结果时，它对特征的重视程度。对特征指定的权重越高，在查询中对该特征越重视。权重是大于 0.0 的实数，如 2.5 或 10.0。若不在查询字符串中指定权重，则 Image Extender 将使用特征的缺省权重。若该特征是查询字符串中指定的唯一特征，则指定权重并无意义。（该特征将总是具有查询中的全部权重。）

特征的权重是相对于查询中指定的其它特征而言的。例如，假设在查询字符串中指定平均颜色和纹理特征，并对平均颜色指定权重值 2.0。这告诉 Image Extender 对平均颜色值的重视程度是纹理值的两倍。

示例

以下查询字符串指定红色平均颜色:

```
averageColor color=<255, 0, 0>
```

以下查询字符串指定由 10% 红色、50% 绿色和 40% 蓝色组成的直方图:

```
histogram histogram=<(10, 255, 0, 0), (50, 0, 255, 0),
                                       (40, 0, 0, 255)>
```

以下查询字符串指定平均颜色值和纹理值。纹理值是由服务器文件中的图像提供的。纹理的权重是平均颜色的两倍:

```
averageColor color=<30, 200, 25> and
  texture file=<server, "\patterns\pattern7.gif"> weight=2.0
```

使用查询对象

可使用查询对象来标识查询的特征、特征值和特征权重。创建查询对象并向其添加特征。然后指定每个特征的数据源。数据源提供特征的值。例如，数据源可能是文件中的图像。若平均颜色是相关的特征，则图像的平均颜色与查询对象相关联。若将多个特征添加至查询对象中，可对其中一个或多个特征指定权重。

Image Extender 提供了三个使用查询对象的 API（QbQuerySearch、QbQueryStringSearch 和 QbQueryNameSearch）和两个使用查询对象的 UDF（QbScoreFromName 和 QbScoreTBFromName）。在发出查询时，使用适当的 API 或 UDF，并指定查询对象作为输入参数。（有关详细信息，参见第 145 页的『按图像内容发出查询』。）

创建查询对象

使用 QbQueryCreate API 来创建查询对象。作为响应，Image Extender 将返回查询对象的句柄。句柄具有 QBIC 特有的数据类型 QbQueryHandle，该类型是在 QBIC 的包含（头）文件 dmbqbapi.h 中定义的。

使用 API 时，需要指向查询对象句柄。还需要在对查询对象执行其它操作（如添加特征）的 API 中指定该句柄。

例如，以下 API 调用创建查询对象:

```
QbQueryHandle  qHandle;

rc=QbQueryCreate(
              &qHandle);                          /* query object handle */
```

将特征添加至查询对象

通过将特征添加至查询对象来标识要让 Image Extender 查询的图像特征。

使用 QbQueryAddFeature API 来将特征添加至查询对象。在使用 API 时，指定查询对象句柄。同时还命名特征。仅可在该 API 中指定一个特征。必须对要添加至查询对象的每个特征分别发出 API 调用。

在以下示例中，使用 QbQueryAddFeature API 来将平均颜色特征添加至查询对象:

```
char  featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;

rc=QbQueryAddFeature(
               qHandle,                        /* query object handle */
               "QbColorFeatureClass");           /* feature name */
```

指定查询对象中特征的数据源

使用 QbQuerySetFeatureData API 来指定查询对象中特征的数据源。数据源可以是:

- 用户表列中已编目或未编目的图像
- 客户工作站上的图像文件
- 客户工作站上缓冲区中的图像

另外，可明确地指定平均颜色或直方图特征的数据。例如，可指定平均颜色的红色、绿色和蓝色值。

使用 API 时:

- 指定查询对象句柄
- 命名特征。
- 指向 QbImageSource 结构（参见页 141 以获取详细信息）。

使用数据源结构: 使用了各种结构来提供查询对象的数据源信息。这些结构是:

- QbImageSource
- QbColor
- QbHistogramColor

QbImageSource: QbImageSource 结构标识查询对象中特征的源的类型。该结构是在 QBIC 的包含（头）文件 dmbqbapi.h 中定义的，如下所示:

```
typedef struct{
          SQLINTEGER    type;
          union {
                char              imageHandle[MMDB_BASE_HANDLE_LEN+1];
                QbImageFile       clientFile;
                QbImageBuffer     buffer;
                QbSampleSource    reserved;
                QbColor           averageColor;
                QbHistogramColor  histogramColor[qbiHistogramCount];
          };
} QbImageSource;
```

QbImageSource 结构中的类型字段指示源的类型。可按如下方式设置字段中的值:

值	含义
qbiSource_ImageHandle	源位于用户表列中
qbiSource_ClientFile	源在客户机工作站文件中
qbiSource_Buffer	源在客户机工作站缓冲区中
qbiSource_ServerFile	源在服务器文件中
qbiSource_AverageColor	源是平均颜色规范
qbiSource_HistogramColor	源是直方图颜色规范

这些设置仅对适当的特征有效。例如，qbiSource_AverageColor 仅对平均颜色特征有效。

如果将类型字段设置为 qbiSource_ServerFile，则将 clientFile 用于服务器上的文件的名称和类型。

根据源类型，Image Extender 还检查您指定的其它信息。如表 13 中所示。

表 13. Image Extender 在 QbImageSource 中检查什么内容

源	Image Extender 检查什么内容	在何处指定
用户表	图像句柄	QbImageSource 的图像句柄字段
文件	文件名 文件格式	QbImageSource 的客户机文件字段
缓冲区	文件名	QbImageBuffer（参见页 142 以了解关于使用此结构的详细信息）
平均颜色规范	红色、绿色和蓝色颜色值	QbColor 结构 （参见页 142 以获取关于使用此结构的详细信息）
直方图颜色规范	颜色值和百分比	QbHistogramColor 结构 （参见页 142 以获取关于使用此结构的详细信息）

QbImageBuffer: 当数据源在缓冲区中时，使用 QbImageBuffer 结构来指定图像的格式、长度和内容。该结构是在 QBIC 的包含（头）文件 dmbqbapi.h 中定义的，如下所示:

```
typedef struct{
          char          format[qbiImageFormatLength+1];
          SQLINTEGER    length;
          char*         image;
} QbImageBuffer;
```

QbColor: 当数据源是平均颜色规范时，使用 QbColor 结构来指定平均颜色的红色、绿色和蓝色值。该结构是在 QBIC 的包含（头）文件 dmbqbapi.h 中定义的，如下所示:

```
typedef struct{
          SQLUSMALLINT    red;          /*0 off - 65535 (fully on) */
          SQLUSMALLINT    green;        /*0 off - 65535 (fully on) */
          SQLUSMALLINT    blue;         /*0 off - 65535 (fully on) */
} QbColor;
```

在 QbColor 中设置一些值以指示将作为平均值计算中因子的红色、绿色和蓝色像素的数量。这些值的范围是 0 至 65535。值 0 表示忽略该项。

QbHistogramColor: 使用 QbHistogramColor 结构来指定直方图颜色规范中的每种颜色组成部分。对直方图颜色的完整规范包含在 QbHistogramColor 结构的一个数组中。每个结构都包含颜色值和百分比。颜色值由红色、绿色和蓝色像素值组成。百分比指定在目标图像中需要多少百分比的这种颜色。

该结构是在 QBIC 的包含（头）文件 dmbqbapi.h 中定义的，如下所示:

```
typedef struct{
          QbColor         color;
          SQLUSMALLINT    percentage;   /*0 - 100 */
} QbHistogramColor;
```

在 QbColor 中设置一些值以指示颜色的红色、绿色和蓝色像素量。这些值的范围是 0 至 65535。设置百分比以指示在目标图像中需要多少百分比的指定颜色。值的范围是 1 至 100。直方图颜色中颜色组成部分的百分比之和必须小于或等于 100。

示例: 以下示例中的 API 在查询对象中指定直方图颜色特征的数据源。数据源是客户机工作站中的文件。

```
char           featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;
QbImageSource  imgSource;

imgSource.type=qbiSource_ClientFile;
strcpy(imgSource.clientFile.fileName,"/tmp/image.gif");
strcpy(imgSource.clientFile.format,"GIF");

rc=QbQuerySetFeatureData(
                qHandle,                              /* query object handle */
                "QbColorHistogramFeatureClass",       /* feature name */
                &imgSource);                          /* feature data source */
```

在以下示例中，数据源是红色的平均颜色规范:

```
char                 featureName[qbiMaxFeatureName];
QbColor              avgColor;
QbImageSource        imgSource;

imgSource.type=qbSource_AverageColor;
avgColor.red=255;
avgColor.green=0;
avgColor.blue=0;
strcpy(featureName,"QbColorFeatureClass");

rc=QbQuerySetFeatureData(
                qHandle,                             /* query object handle */
                featureName,                         /* feature name */
                &imgSource);                         /* feature data source */
```

在查询对象中设置特征的权重

若已将多个特征添加至查询对象，则可在查询中指定对一个或多个特征给出的权重。使用 QbQuerySetFeatureWeight API 来指定特征的权重。特征权重指示，当 Image Extender 计算类似性得分并返回“按图像内容查询”的结果时，它对特征的重视程度。对特征指定的权重越高，对查询对象中该特征也越重视。

虽然每次发出 QbQuerySetFeatureWeight API 时仅可对一个特征指定权重，但可以在查询对象中指定一个或多个特征的权重。若不在查询对象中对特征指定权重，则 Image Extender 将使用特征的缺省权重。若查询对象中只有一个特征，则对该特征指定权重并无意义。（该特征将总是具有查询对象中的全部权重。）

使用 API 时:

- 指定查询对象句柄。
- 指定特征名。
- 指向特征权重。可将权重设置为大于 0 的实数，例如 2.5 或 10.0。指定的值越高，对该特征也越重视。这些设置更改查询对象中先前对该特征设置的任何权重。

在以下示例中，查询对象包含平均颜色特征和至少一个其它特征。使用 QbQuerySetFeatureWeight API 在查询对象中指定平均颜色特征的权重:

```
char                 featureName[qbiMaxFeatureName];
double               weight;
QbQueryObjectHandle  qoHandle;

strcpy(featureName,"QbColorFeatureClass");
weight=5.00;

rc=QbQuerySetFeatureWeight(
```

```
qoHandle,                          /* query object handle */
featureName,                       /* feature name */
&weight);                          /* feature weight */
```

保存和重用查询字符串

除非保存查询对象，否则它们是瞬时的。它们仅在单个数据库连接期间存在。您可以保存来自一个查询的查询字符串以便在程序中再次使用它，即使当前数据库连接已删除，也可以使用它。另外，也可以跨程序调用使用它。

Image Extender 提供了 QbQueryGetString API，它从一个查询对象返回查询字符串。之后，您可以使用该查询字符串作为 QbQueryStringSearch API 的输入，或作为其它按内容查询中的 QbScoreFromStr 和 QbScoreTBFromStr UDF 的输入（参见第 145 页的『按图像内容发出查询』）。

查询字符串是在您使用下列各项构建查询时构建的:

- QbQueryCreate
- QbQueryAddFeature
- QbQuerySetFeatureData
- QbQuerySetFeatureWeight
- QbQueryRemoveFeature

在构建查询之后，您可以调用 QbQueryGetString 来获得该字符串。您可以在该程序内的调用中使用此查询字符串，也可以将其保存至文件，以便在应用程序的后续调用和其它数据库连接中使用。在完成使用 QbQueryGetString 所返回的查询字符串之后，必须显式地释放空间。

在以下示例中，将使用 QbQueryGetString 来从查询对象中检索查询字符串:

```
SQLRETURN rc;
char* qryString;
QbQueryHandle  qHandle;

.....           /* Here you create and use the query */

rc = QbQueryGetString(qHandle, &qryString);
if ( rc == 0) {
   ...         /* Use the query string as input here  */
   free((void *)qryString);
   qryString=(char *)0;
}
```

限制: 当使用客户机文件来指定特征的数据源时，查询字符串并不反映特征数据。

检索关于查询对象的信息

可确定有哪些特征（若有的话）已添加至查询对象。还可确定特征的当前权重。

使用此 API	检索
QbQueryGetFeatureCount	查询对象中特征的数目
QbQueryListFeatures	查询对象中特征的名称

在发出 QbQueryGetFeatureCount API 时，指定查询对象句柄。还需要指向计数器。Image Extender 在计数器中返回特征数。

在以下示例中，使用 QbQueryGetFeatureCount API 来确定查询对象中的特征数:

```
SQLINTEGER     count;
QbQueryHandle  qHandle;

rc=QbQueryGetFeatureCount(
                qHandle,                        /* query object handle */
                &count);                            /* feature count */
```

在发出 QbQueryListFeatures API 调用时，需要分配缓冲区来存放返回的特征名。还需要指定目录句柄，以及存放返回的特征名的缓冲区的大小。

在以下示例中，将使用 QbQueryListFeatures API 来检索查询对象中每个特征的名称:

```
SQLINTEGER     retCount,bufSize;
char*          featureName;
QbQueryHandle  qHandle;

bufSize=qbiMaxFeatureName;
featureName=(char*)malloc(bufSize);

rc=QbQueryListFeatures(
                qHandle,                        /* query object handle */
                bufSize                             /* size of buffer */
                &retCount,                          /* feature count */
                featureName);                       /* buffer for feature names */
```

从查询对象中除去特征

使用 QbQueryRemoveFeature API 从查询对象中除去特征。在使用此 API 时，指定查询对象句柄并命名特征。

在以下示例中，将使用 QbQueryRemoveFeature API 从查询对象中除去直方图颜色特征:

```
char           featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;

strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbQueryRemoveFeature(
                qHandle,                            /* query object handle */
                featureName);                       /* feature name */
```

删除查询对象

使用 QbQueryDelete API 来删除未命名的查询对象。Image Extender 从当前连接的数据库中删除查询。

当使用 QbQueryDelete API 时，指定查询对象句柄。

在以下示例中，将使用 QbQueryDelete API 来删除查询对象:

```
QbQueryHandle  qHandle;

rc=QbQueryDelete(
           qHandle);                                /* query object handle */
```

如果已使用了有名查询，则使用 QbQueryNameDelete API 来删除查询对象。

按图像内容发出查询

在编目图像之后，可按内容查询其中一个或多个图像。在按内容查询图像时，标识查询的输入和一组目标编目图像。可在查询字符串（参见第 138 页的『指定查询字符串』）或查询对象（参见第 140 页的『使用查询对象』）中指定输入。

发出 QBIC 查询

若使用查询字符串，则可从 DB2 命令行或在程序内提交查询。如果使用查询对象，则从程序内通过引用其句柄提交查询。

Image Extender 将查询中指定的特征值与目标图像的特征值作比较，并计算每一图像的得分。得分指示目标图像的特征值与查询中指定的特征值有多类似。

可检索特征值最类似于查询的图像。还可查询单个已编目图像并获取其得分，或获取表列中所有已编目图像的得分。

查询图像

Image Extender 提供了三个 API 来查询表列中的已编目图像。这些 API 的区别仅在于：是要求将查询字符串作为输入，还是要求将查询对象作为输入：

API	输入
QbQueryStringSearch	查询字符串
QbQuerySearch	查询对象句柄
QbQueryNameSearch	查询对象名

在所有这三个 API 中，还：

- 命名包含要搜索的图像的表和列。必须将图像编目到 QBIC 目录中。
- 指定要返回的最大结果数。
- 指向指定查询作用域的结构。将指针设置为 0、空值或空白字符串。这将指定搜索表列中的所有已编目图像。
- 指定常量 qbiArray 以指示将结果存储在数组中。qbiArray 常量是在 QBIC 的包含（头）文件 dmbqbapi.h 中定义的。

还要指向要包含搜索结果的输出结构的数组。作为响应，Image Extender 在这些结构中返回其特征值最类似于查询特征值的目标图像的句柄。它还返回每一图像的得分，该得分指示图像的特征值与查询有多类似。该结构是在 QBIC 的包含（头）文件 dmbqbapi.h 中定义的，如下所示：

```
typedef struct{
          char       imageHandle[MMDB_BASE_HANDLE_LEN+1];
          SQLDOUBLE  SCORE
} QbResult;
```

必须分配大小足以存放指定的最大结果数的数组，并在 API 中指向该数组。还必须指向计数器；Image Extender 将该计数器的值设置为它返回的结果数。

在以下示例中，使用 QbQueryStringSearch API，以便按内容查询表列中已编目的图像。注意指向查询作用域的指针被设置为零值。

```
QbResult       returns[MaxQueryReturns];
SQLINTEGER     maxResults=qbiMaxQueryReturns;
SQLINTEGER     count;
QbQueryHandle  qHandle;
QbResult       results[qbiMaxQueryReturns];

rc=QbQueryStringSearch(
          "QbColorFeatureClass color=<255, 0, 0>" /*query string */
                "employee",                       /* user table */
                "picture",                        /* image column */
                maxResults,                       /* maximum number of results */
                0,                                /* query scope pointer * /
```

```
                qbiArray,                       /* store results in an array */
                &count,                         /* count of returned images */
                results);                       /* array of returned results */
```

下面是使用 QbQuerySearch API 的请求。注意查询对象句柄被指定为输入。

```
QbResult       returns[MaxQueryReturns];
SQLINTEGER     maxResults=qbiMaxQueryReturns;
SQLINTEGER     count;
QbQueryHandle  qHandle;
QbResult       results[qbiMaxQueryReturns];

rc=QbQuerySearch(
                 qHandle,                       / query object handle */
                "employee",                     /* user table */
                "picture",                      /* image column */
                maxResults,                     /* maximum number of results */
                0,                              /* query scope pointer * /
                qbiArray,                       /* store results in an array */
                &count,                         /* count of returned images */
                results);                       /* array of returned results */
```

检索图像得分

Image Extender 提供了四个检索得分的 UDF，可在 SQL 语句中用来检索表列中已编目图像的得分。得分是双精度浮点值，从 0.0 到接近无穷大的非常大的值。得分越低，图像的特征值与查询中指定的特征值越匹配。得分 0.0 表示图像完全匹配。

这些 UDF 是:

- QbScorefromStr
- QbScoreTBfromStr
- QbScoreFromName
- QbScoreTBFromName

建议：使用 QbScoreFromStr UDF 来获取单个已编目图像的得分。使用 QbScoreTBFromStr UDF 来获取表列中多个已编目图像的得分。

检索单个图像的得分

使用 QbScoreFromStr UDF 来获取表列中单个已编目图像的得分。指定查询字符串作为 QbScoreFromStr UDF 的输入。如果使用 QbScoreFromName UDF，则指定查询对象的名称作为 QbScoreFromName UDF 的输入。对于任一个 UDF，还需指定包含目标图像的表列的名称。

在以下查询中，使用 QbScoreFromStr UDF 来查找表列中其平均颜色得分非常接近于红色的已编目图像。

```
SELECT name, description
decimal (QbScoreFromStr(swatch_img,
            'QbColorFeatureClass color=<255, 0, 0>'),   /* query string *
                 10, 5) AS score
FROM fabric                                             /* table column */
ORDER BY score
```

检索多个图像的得分

使用 QbScoreTBFromStr UDF 来获取表列中多个已编目图像的得分。如果是有名查询，则可以使用 QbScoreTBFromName UDF。这两个 UDF 都返回一个有两列的表，这两列

分别用于图像句柄和图像得分；表中的行是按得分升序排列的。结果表中句柄列的名称是 IMAGE_ID；得分列的名称是 SCORE。

将查询字符串指定为对 QBScoreTBFromStr UDF 的输入。指定查询对象名作为 QbScoreTBFromName UDF 的输入。对于任一个 UDF，还需指定包含目标图像的表和列的名称。还可指定要在结果表中返回的最大行数。若不指定最大结果数，则 UDF 将为目标表列中的每一已编目图像返回一行。

在以下查询中，将使用 QbScoreTBFromStr UDF 在表列中查找 10 个其纹理与服务器文件中的纹理最接近的已编目图像。

```
SELECT name, description
 FROM fabric
WHERE CAST (swatch_img as varchar(250)) IN
 SELECT CAST (image_id as varchar(25)) FROM TABLE
 (QbScoreTBFromStr
 (QbTextureFeatureClass file=<server,"patterns/ptrn07.gif">'   /*query string */

           'fabric',                                    /* table */
     'swatch_img',                                          /* table column */
     10))                                          /* maximum number of results */
  AS T1));
```

QBIC 查询样本程序

第 149 页的图 29 显示了用 C 语言编写的、用于构建并运行 QBIC 查询的程序的一部分。图中的代码按平均颜色查询图像。它提示用户输入颜色或图像文件的名称。用户还可以将查询返回的图像用作后续查询的示例图像。该程序将命名颜色或图像的颜色用作平均颜色来查询图像列。

可在 SAMPLES 子目录中的 QBICDEMO.C 文件中找到完整的程序。可使用完整的程序按直方图颜色或位置颜色来查询图像，以及按平均颜色查询图像。要运行完整的程序，必须运行 ENABLE、POPULATE 和 QBCATDMO 样本程序（也在 SAMPLES 子目录中）。有关样本程序的更多信息，参见第 481 页的附录 B，『样本程序和媒体文件』。

注意第 149 页的图 29 中的下列几点:

1 包括 dmbqbapi 头文件。

2 提示用户输入数据库信息。

3 连接到数据库。

4 创建查询对象。

5 将特征添加至查询对象。

6 提示用户提供输入类型（颜色名、图像文件或先前检索到的图像）。

7 指定特征的数据源。数据源是对平均颜色的明确规范。

8 发出查询。Image Extender 搜索整个图像列。它还指定 10 作为要返回的最大图像数。

9 显示返回的图像集中的下一图像。有关显示图像的进一步信息，参见第 124 页的『显示实际大小的图像或视频帧』。

10 删除查询对象。

SAMPLES 子目录包括另一个用于演示如何构建并使用 QBIC 查询的程序。程序 QbicQry.java 显示了如何以图形方式指定 QBIC 查询的搜索标准。例如，该程序提供

了一个颜色选择器以选择平均颜色。该程序将选择转换为查询字符串。

```
#include <sql.h>
#include <sqlcli.h>
#include <sqlcli1.h>
#include <dmbqbqpi.h> 1
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <color.h>
#include <ctype.h>

#define   MaxQueryReturns        10

#define   MaxDatabaseNameLength  SQL_SH_IDENT
#define   MaxUserIdLength        SQL_SH_IDENT
#define   MaxPasswordLength      SQL_SH_IDENT
#define   MaxTableNameLength     SQL_LG_IDENT
#define   MaxColumnNameLength    SQL_LG_IDENT

static  char       databaseName[MaxDatabaseNameLength+1];
static  char       userid[MaxUserIdLength+1];
static  char       password[MaxPasswordLength+1];

static  char       tableName[MaxTableNameLength+1];
static  char       columnName[MaxColumnNameLength+1];

static  char       line[4000];

static  QbResult   results[MaxQueryReturns];
static  long       currentImage   = -1;
static  long       imageCount  = 0;

static  char*      tableAndColumn;

static  QbQueryHandle averageHandle = 0;
static  QbQueryHandle histogramHandle = 0;
static  QbQueryHandle drawHandle = 0;
static  QbQueryHandle lastHandle = 0;

static  SQLHENV    henv;
static  SQLHDBC    hdbc;
static  SQLHSTMT   hstmt;
static  SQLRETURN  rc;

static  char*    listQueries =
 "SELECT NAME,DESCRIPTION FROM MMDBSYS.QBICQUERIES ORDER BY NAME";

static  char*    menu[] = {
```

图 29. QBIC 查询样本程序 (1/6)

```
/*
 12345678901234567890123456789012345678901234567890123456789012345678901234567890 */
"",
"+-------------------------------------------------------------------------------+",
"| AVERAGE COLOR colorname                                                       |",
"| AVERAGE FILE filename format                                                  |",
"| AVERAGE LAST                                                                  |",
"| Press Enter to display the next image in the series                           |",
"+-------------------------------------------------------------------------------+",
"",
0
};

static  char*       help[] = {
"",
"AVERAGE     Execute an average color query",
"  COLOR       Specifies the color to query for",
"  FILE        Specifies the file to compute the average color from",
"  LAST        Specifies the last displayed image be used to compute the color",
"  NEXT        Displays the next image from the current query or nothing if",
"                all of the image have been displayed."
"",
">>pause<<",
0
};
/*****************************************************************************/
/* doNext()                                                                  */
/*****************************************************************************/
static void doNext(void)
{
 int ret;
   if (currentImage < imageCount)
      currentImage++;
   if (currentImage < imageCount)
      ret = DBiBrowse("/usr/local/bin/xv %s", MMDB_PLAY_HANDLE,
            results[currentImage].imageHandle, MMDB_PLAY_NO_WAIT); 9
}
```

图 29. QBIC 查询样本程序 (2/6)

```
/*****************************************************************************/
/* doAverage()                                                               */
/*****************************************************************************/
static void doAverage(void)
{
   QbQueryHandle qohandle = 0; QbImageSource is; char* type;
   char* arg1; char* arg2;

   type = nextWord(0);
   if (abbrev(type, "color", 1)) {
      is.type = qbiSource_AverageColor;
      arg1 = nextWord(0);
      if (arg1 == 0) {
         printf("AVERAGE COLOR command requires a colorname argument.\n");
      return;
      }
      if (getColor(arg1, &is.averageColor) == 0) {
         printf("The colorname entered was not recognized.\n");
      return;
      }
   }
   else if (abbrev(type, "file", 1)) {
      is.type = qbiSource_ClientFile;
      arg1 = nextWord(0);
      if (arg1 == 0) {
         printf("AVERAGE FILE command requires a filename argument.\n");
      return;
      }
      arg2 = nextWord(0);
      if (arg2 == 0) {
         printf("AVERAGE FILE command requires a file format argument.\n");
      return;
      }
      strcpy(is.clientFile.fileName, arg1);
      strcpy(is.clientFile.format, arg2);
   }
   else if (abbrev(type, "last", 1)) {
      is.type = qbiSource_ImageHandle;
      if (0 <= currentImage &&; currentImage < imageCount)
         strcpy(is.imageHandle, results[currentImage]imageHandle);
      else {
         printf("No last image for AVERAGE LAST command\n");
         return;
      }
   }
```

图 29. QBIC 查询样本程序 (3/6)

```
   else {
      printf("AVERAGE command only supports COLOR, FILE, and LAST types.\n");
      return;
   }

   _QbQuerySetFeatureData(averageHandle, "QbColorFeatureClass", &is); 7
   _QbQuerySearch(averageHandle, tableAndColumn, "IMAGE",
         MaxQueryReturns, 0, 0, &imageCount, results); 8
   lastHandle = averageHandle;

   currentImage = -1;
}
/*****************************************************************************/
/* commandLoop()                                                             */
/*****************************************************************************/
void commandLoop(void)
{
   int   done = 0;

   while (!done) { 6
      displayText(menu);
      printf("%d", currentImage + 1);
      if (0 <= currentImage &&; currentImage < imageCount)
            printf("  %8.6f", results[currentImage].score);
      printf("> ");
   gets(line);
      done = processCommand(line);
   }
}
```

图 29. QBIC 查询样本程序 (4/6)

```
/*****************************************************************************/
/* main()                                                                    */
/*****************************************************************************/
void main(void)
{
   char*    inst;
   int      i;

   printf("\n\n");
   printf("Please enter: database_name [user_id] [password] "\n"); 2
   gets(line);
   if (copyWord(line, databaseName, sizeof(databaseName)) == 0)
 exit (0);
   copyWord(0, userid, sizeof(userid));
   copyWord(0, password, sizeof(password));
   printf("\n");

   if (SQLAllocEnv(&henv) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLAllocConnect(henv, &hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLConnect(hdbc, 3
                  (SQLCHAR*)databaseName,
                  SQL_NTS,
                  (SQLCHAR*)userid,
                  SQL_NTS,
                  (SQLCHAR*)password,
                  SQL_NTS) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);

   printf("Initializing . . .\n");
```

图 29. QBIC 查询样本程序 (5/6)

```
   inst = getenv("DB2INSTANCE");
   if (inst != 0 &&; strcmp(inst, "keeseyt") == 0)
      tableAndColumn = "KEESEY.TEST";
else
      tableAndColumn = "QBICDEMO.TEST";

   _QbQueryCreate(&averageHandle); 4
   _QbQueryAddFeature(averageHandle, "QbColorFeatureClass");
   _QbQueryCreate(&histogramHandle);
   _QbQueryAddFeature(histogramHandle, "QbColorHistogramFeatureClass");
   _QbQueryCreate(&drawHandle);
   _QbQueryAddFeature(drawHandle, "QbDrawFeatureClass"); 5

   commandLoop();

   _QbQueryDelete(drawHandle);
   _QbQueryDelete(histogramHandle); 10
   _QbQueryDelete(averageHandle);

   if (SQLDisconnect(hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLFreeConnect(hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLFreeEnv(henv) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
}
```

图 29. QBIC 查询样本程序 (6/6)

发出 QBIC 查询

第 4 部分 参考

© Copyright IBM Corp. 1998, 2003

第 13 章 用户定义类型和用户定义函数

本章给出 DB2 Extender 创建的 UDT 和 UDF 的参考信息。

模式

Extender 将 MMDBSYS 模式用于与它们的对象相关的所有对象，包括 UDT 和 UDF。

用户定义类型

表 14 列示并描述了 DB2 Extender 创建的用户定义类型。它还列示了每个 UDT 的 DB2 源数据类型。

表 14. DB2 Extender 创建的用户定义类型

UDT	源数据类型	描述
DB2IMAGE	VARCHAR(250)	图像句柄。变长字符串，它包含存取图像对象所需的信息。图像句柄存储在允许 Image Extender 使用的用户表列中。
DB2AUDIO	VARCHAR(250)	音频句柄。变长字符串，它包含存取音频对象所需的信息。音频句柄存储在允许 Audio Extender 使用的用户表列中。
DB2VIDEO	VARCHAR(250)	视频句柄。变长字符串，它包含存取视频对象所需的信息。视频句柄存储在允许 Video Extender 使用的用户表列中。

用户定义函数

本节给出 DB2 Extender 的参考信息。UDF 以字母次序列示。

对个 UDF 给出下列信息:

- 提供 UDF 的 Extender
- 简要描述
- UDF 的包含（头）文件
- UDF 的 SQL 语法
- UDF 参数的描述，包括数据类型
- UDF 返回的值，包括其数据类型
- 使用示例

表 15 列示了 UDF 并标识了提供每个 UDF 的 Extender。该表还指向可找到更多关于每个 UDF 的信息的地方。可在嵌入式 SQL 语句或 DB2 CLI 调用中编码此表中的 UDF。

表 15. DB2 Extender UDF

UDF	描述	图像	音频	视频	参见页
AlignValue	返回 WAVE 音频中，或视频的声道中每个采样的字节数。		x	x	162

© Copyright IBM Corp. 1998, 2003

表 15. DB2 Extender UDF (续)

UDF	描述	图像	音频	视频	参见页
AspectRatio	返回 MPEG1 和 MPEG2 视频的第一个声道的长宽比。			x	163
BitsPerSample	返回用来表示音频中 WAVE 或 AIFF 音频的，或视频中的声道的每个采样的数据位数。		x	x	164
BytesPerSec	返回 WAVE 音频的数据传送速率，以平均每秒字节数计。		x		165
Comment	返回或更新与图像、音频或视频存储在一起的注释。	x	x	x	166
CompressType	返回视频的压缩格式，如 MPEG-1。			x	168
Content	从数据库中检索或更新图像、音频或视频的内容。	x	x	x	169
DB2Audio	将音频的内容存储在数据库表中。		x		173
DB2Image	将图像的内容存储在数据库表中。	x			176
DB2Video	将视频的内容存储在数据库表中。			x	180
Duration	返回 WAVE 或 AIFF 音频，或视频的持续时间（即，以秒计的播放时间）。		x	x	183
Filename	返回包含图像、音频或视频的内容的服务器文件名。	x	x	x	184
FindInstrument	返回 MIDI 音频中指定的乐器的首次出现的声道号。		x		185
FindTrackName	返回 MIDI 音频中指定的命名声道的名称。		x		186
Format	返回图像、音频或视频的格式。	x	x	x	187
FrameRate	返回视频的吞吐量，以帧频计。			x	188
GetInstruments	返回 MIDI 音频中所有乐器的乐器名。		x		189
GetTrackNames	返回 MIDI 音频中所有声道的名称。		x		190
Height	返回图像或视频的高度（以像素计）。	x		x	191
Importer	返回将图像、音频或视频存储在数据库表中的人的用户标识。	x	x	x	192
ImportTime	返回指示何时将图像、音频或视频存储在数据库表中的时间戳记。	x	x	x	193
MaxBytesPerSec	返回视频的最大吞吐量，以每秒字节数计。			x	194
NumAudioTracks	返回视频或 MIDI 音频中的声道数。		x	x	195
NumChannels	返回 WAVE 或 AIFF 音频中，或视频中录制的声道的数。		x	x	196
NumColors	返回图像中的色彩数。	x			197
NumFrames	返回视频中的帧数。			x	198
NumVideoTracks	返回视频中的视频声道数。			x	199
QbScoreFromName	返回图像的得分（使用命名查询对象）。（替换 QbScore。）	x			200
QbScoreFromStr	返回图像的得分（使用查询字符串）。	x			201

表 15. DB2 Extender UDF (续)

UDF	描述	图像	音频	视频	参见页
QbScoreTBFromName	从图像列返回得分表（使用命名查询对象）。	x			202
QbScoreTBFromStr	从图像列返回得分表（使用查询字符串）。	x			204
Replace	更新存储在数据库中的图像、音频或视频的内容，并更新其注释。	x	x	x	205
SamplingRate	返回 WAVE 或 AIFF 音频的，或视频中声道的采样速率，以每秒采样数计。		x	x	208
Size	返回图像、音频或视频的大小，以字节计。	x	x	x	209
Thumbnail	返回或更新存储在数据库中的图像或视频帧的缩略图大小版本。	x		x	210
TicksPerQNote	返回录制的 MIDI 音频的时钟速率，以每四分音符的滴答数计。		x		212
TicksPerSec	返回录制的 MIDI 音频的时钟速率，以每秒滴答数计。		x		213
Updater	返回上次更新数据库表中的图像、音频或视频的人的用户标识。	x	x	x	214
UpdateTime	返回指示上次更新数据库表中的图像、音频或视频时的时间戳记。	x	x	x	215
Width	返回图像或视频帧的宽度，以像素计。	x		x	216

AlignValue

图像	音频	视频
	X	X

返回 WAVE 音频中的，或视频的声道中的每个采样的字节数 WAVE 音频可使用 1 字节／采样（8 位单声道，称为“字节对齐”）、 2 字节／采样（8 位立体声或 16 位单声道，称为“字对齐”）或 4 字节／采样（16 位立体声，称为“双字对齐”）来保存其数据。

包含文件

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2AUDIO 或 DB2VIDEO）
包含音频的句柄的列名或主变量。

返回值（数据类型）

WAVE 音频的，或视频中的声道的字节／采样值（SMALLINT）。值可以是:

1 字节对齐

2 字对齐

4 双字对齐

空值 其它格式的音频

例

获取存储在 employee 表的 sound 列中的所有（字对齐的）音频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
   EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
      INTO :hvAud_fname
  FROM EMPLOYEE
      WHERE ALIGNVALUE(SOUND) = 2;
```

查找视频中声道的字节／采样值；视频存储在 Employee 表的 Anita Jones 的 Video 列中:

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvAlign_val;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT ALIGNVALUE(VIDEO)
      INTO :hvAlign_val
      FROM EMPLOYEE
      WHERE NAME='Anita Jones';
```

AspectRatio

图像	音频	视频
		X

返回 MPEG 视频的第一个声道的长宽比。

包含文件

dmbvideo.h

语法

参数（数据类型）

handle（**DB2VIDEO**）

包含视频的句柄的列名或主变量。

返回值（数据类型）

MPEG 视频的第一个声道的长宽比，或者为空值，表示其它格式的视频（SMALLINT）

例

获取 employee 表的 video 列中存储的 Robert Smith 的视频的长宽比:

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvAsp_ratio;
   EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT ASPECTRATIO(VIDEO)
      INTO :hvAsp_ratio
  FROM EMPLOYEE
     WHERE NAME='Robert Smith';
```

BitsPerSample

图像	音频	视频
	X	X

返回用来表示音频中 WAVE 或 AIFF 音频的，或视频中的声道的每个采样的数据位数。

包含文件

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2AUDIO 或 DB2VIDEO）
包含音频或视频的句柄的列名或主变量。

返回值（数据类型）

用来表示视频的，或 WAVE 或 AIFF 音频的每个采样的数据位数（SMALLINT）。对于其它格式的音频，返回空值

例

获取存储在 Employee 表的 Sound 列中，其采样位数等于 8 的所有 WAVE 音频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
   EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
      INTO :hvAud_fname
  FROM EMPLOYEE
      WHERE FORMAT(SOUND)='WAVE'
      AND BITSPERSAMPLE(SOUND) = 8;
```

BytesPerSec

图像	音频	视频
	X	

返回 WAVE 音频的数据传送速率，以平均每秒字节数计。

包含文件

dmbaudio.h

语法

参数（数据类型）

handle（**DB2AUDIO**）

包含音频的句柄的列名或主变量。

返回值（数据类型）

数据传送速率（INTEGER）。对于其它格式的音频，返回空值。

例

获取存储在 Employee 表的 Sound 列中的，其传送速率（以平均每秒字节数计）小于 44100 的所有音频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE BYTESPERSEC(SOUND) < 44100;
```

Comment

图像	音频	视频
X	X	X

返回或更新与图像、音频或视频存储在一起的注释。

包含文件

图像（**image**）
dmbimage.h

音频（**audio**）
dmbaudio.h

视频（**video**） dmbvideo.h

语法

检索注释

语法

更新注释

参数（数据类型）

handle（**DB2IMAGE**、**DB2AUDIO** 或 **DB2VIDEO**）
包含图像、音频或视频的句柄的列名或主变量。

new_comment （**LONG VARCHAR**）
用于更新的新注释。空值或空字符串删除现有注释。

返回值（数据类型）

对于更新，是图像、音频或视频的句柄（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）。
对于检索，是注释（LONG VARCHAR）。

例

从 Employee 表的 Picture 列中获取相关注释中的字 “confidential” 的所有图像的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[255;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE COMMENT(PICTURE)
        LIKE '%confidential%';
```

更新与 employee 表的 video 列中 Anita Jones 的视频剪辑相关联的注释:

```
EXEC SQL BEGIN DECLARE SECTION;
struct{
      short len;
```

```
      char data[4000];
      }hvRemarks;
EXEC SQL END DECLARE SECTION;

/* Get the old comment */

EXEC SQL SELECT COMMENT(VIDEO)
     INTO :hvRemarks
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';

/* Update the comment */

hvRemarks.data[hvRemarks.len]='\0';
strcat (hvRemarks.data, "Updated video");
hvRemarks.len=strlen(hvRemarks.data);

EXEC SQL UPDATE EMPLOYEE
     SET VIDEO=COMMENT(VIDEO, :hvRemarks)
     WHERE NAME = 'Anita Jones';
```

CompressType

图像	音频	视频
		X

返回视频的压缩格式，如 MPEG-1。

包含文件

dmbvideo.h

语法

参数（数据类型）

handle（DB2VIDEO）

包含视频的句柄的列名或主变量

返回值（数据类型）

视频的压缩格式（VARCHAR(8)）

例

获取存储在 employee 表的 video 列中的压缩格式为 MPEG-1 的所有视频的名称:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE COMPRESSTYPE(VIDEO) = 'MPEG1';
```

Content

图像	音频	视频
X	X	X

从数据库中检索或更新图像、音频或视频的内容。可以将内容检索至客户机缓冲区、客户机文件或服务器文件。

包含文件

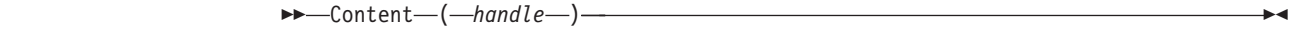
图像（**image**）
dmbimage.h

音频（**audio**）
dmbaudio.h

视频（**video**） dmbvideo.h

语法

将内容检索至缓冲区或客户机文件



语法

将内容段检索至缓冲区或客户机文件

语法

将内容检索至服务器文件

语法

将内容检索至缓冲区或客户机文件，并进行格式转换 — 仅适用于图像

语法

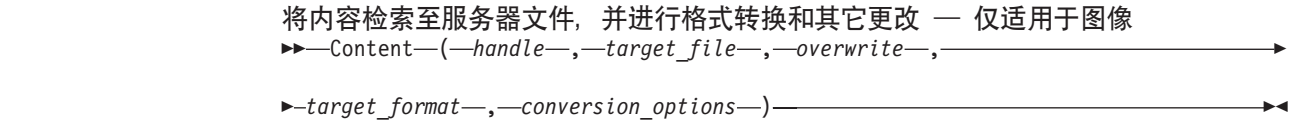
将内容检索至服务器文件，并进行格式转换 — 仅适用于图像

语法

将内容检索至缓冲区或客户机文件，并进行格式转换和其它更改 — 仅适用于图像

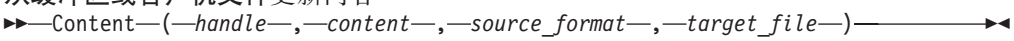
语法

将内容检索至服务器文件，并进行格式转换和其它更改 — 仅适用于图像



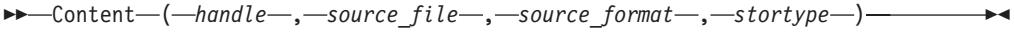
语法

从缓冲区或客户机文件更新内容



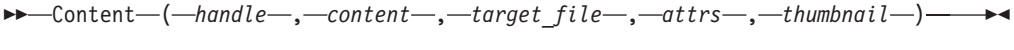
语法

从服务器文件更新内容



语法

从缓冲区或客户机文件更新具有用户提供的属性的内容

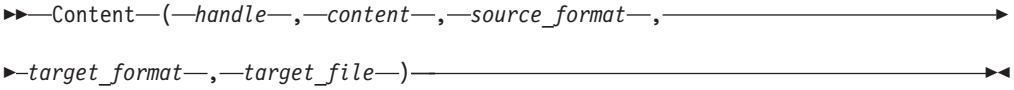


语法

从服务器文件更新具有用户提供的属性的内容

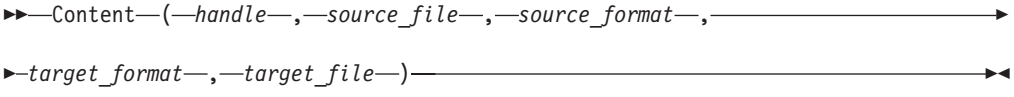
语法

从缓冲区或客户机文件更新内容，并进行格式转换 — 仅适用于图像



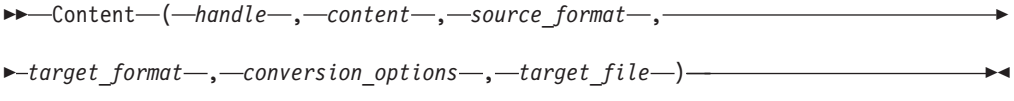
语法

从服务器文件更新内容，并进行格式转换 — 仅适用于图像



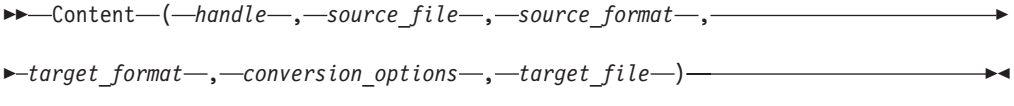
语法

从缓冲区或客户机文件更新内容，并进行格式转换和其它更改 — 仅适用于图像



语法

从服务器文件更新内容，并进行格式转换和附加的更改 — 仅适用于图像



参数（数据类型）

handle（**DB2IMAGE**、**DB2AUDIO** 或 **DB2VIDEO**）
包含图像、音频或视频的句柄的列名或主变量。

offset（**INTEGER**）
要检索的图像、音频或视频的起始偏移（从 1 开始）。

size（INTEGER）

要检索的图像、音频或视频的字节数。

source_file （LONG VARCHAR）

包含图像、音频或视频的更新内容的文件的名称。

target_file （LONG VARCHAR）

对于检索，是要将图像、音频或视频检索到其中的文件的名称。对于更新，是包含要更新的图像、音频或视频的文件的名称。

stortype（INTEGER）

指示将在何处存储已更新的图像、音频或视频的值。常量 MMDB_STORAGE_TYPE_INTERNAL（值 = 1）指示将把更新过的对象以 BLOB 形式存储在数据库中。常量 MMDB_STORAGE_TYPE_EXTERNAL（值 = 0）指示将把更新过的对象存储在服务器文件中。

overwrite（INTEGER）

指示是否覆盖目标文件（若它已存在的话）的值。值可以是 0 或 1。值 0 表示将不覆盖目标文件（实际上，将不发生检索）。值 1 表示若目标文件已存在，则覆盖它。

target_format（VARCHAR(8)）

检索或更新之后图像的格式。将适当地转换源图像的格式。对于将图像检索至服务器文件，若 target_file 与 source_file 相同，则目标格式必须与源格式相同。对于 MPG1 格式，可指定 MPG1、mpg1、MPEG1 或 mpeg1。对于 MPG2 格式，可指定 MPG2、mpg2、MPEG2 或 mpeg2。

conversion_options（VARCHAR(100)）

指定检索或更新图像时，要对其应用的更改，如旋转和压缩。参见第 94 页的表 9 以获取受支持的转换选项。

content（BLOB(2G) AS LOCATOR）

包含图像、音频或视频的更新内容的主变量的名称。主变量的类型可以是 BLOB、BLOB_FILE 或 BLOB_LOCATOR。DB2 使内容的数据类型转换为 BLOB_LOCATOR，并将 LOB 定位器传送至 Content UDF。

source_format（VARCHAR(8)）

图像、音频或视频的更新源的格式。可指定空值或空字符串，或仅对于图像，可指定字符串 ASIS；在这三种情况下，Extender 将试图自动确定格式。对于 MPG1 格式，可指定 MPG1、mpg1、MPEG1 或 mpeg1。对于 MPG2 格式，可指定 MPG2、mpg2、MPEG2 或 mpeg2。

attrs（LONG VARCHAR FOR BIT）

图像、音频或视频的属性

thumbnail（LONG VARCHAR FOR BIT DATA）

图像或视频帧的缩略图（仅适用于图像和视频）

返回值（数据类型）

检索到的图像、音频或视频的内容（若检索至缓冲区），（BLOB(2G) AS LOCATOR）。若检索至文件，VARCHAR(254)。

对于更新，是要更新的图像、音频或视频的句柄（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）。

例

将 employee 表的 picture 列中存储的 Anita Jones 的图像检索到服务器文件中:

```
struct{
      short len;
      char data[250];
      }hvImg_fname;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT (PICTURE,
       '/employee/images/ajones.bmp',1)
      INTO :hvImg_fname
      FROM EMPLOYEE
      WHERE NAME='Anita Jones';
```

将 Employee 表的 Sound 列中存储的 Robert Smith 的 1-MB 的音频剪辑检索到客户机缓冲区中:

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS BLOB_LOCATOR audio_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT (SOUND, 1, 1000000)
      INTO :audio_loc
      FROM EMPLOYEE
      WHERE NAME='Robert Smith';
```

更新 Employee 表的 picture 列中 Anita Jones 的图像; 将图像的格式由 BMP 转换为 GIF, 并将图像缩小为原始大小的 50 %:

```
EXEC SQL UPDATE EMPLOYEE
  SET picture = CONTENT(PICTURE,
       '/Employee/newimg/ajones.bmp',
          'BMP',
          'GIF',
        '-s 0.5',
       '');
       WHERE NAME='Anita Jones';
```

DB2Audio

图像	音频	视频
	X	

将音频的内容存储在数据库表中。音频源可以在客户机缓冲区、客户机文件或服务器文件中。可将音频以 BLOB 形式存储在数据库表中，或存储在服务器文件中（由数据库表引用）。音频源可处于受支持的格式，在这种情况下，DB2Audio Extender 标识其存储属性，也可处于不受支持的格式，在这种情况下，必须在 UDF 中指定属性。

包含文件

dmbaudio.h

语法

从缓冲区或客户机文件存储内容

语法

从服务器文件存储内容

语法

从缓冲区或客户机文件存储具有用户提供的属性的内容

语法

从服务器文件存储具有用户提供的属性的内容

参数（数据类型）

dbname（VARCHAR(18)）

当前连接的数据库的名称，由 CURRENT SERVER 专用寄存器指示。

content（BLOB(2G) AS LOCATOR）

包含音频的内容的主变量。主变量的类型可以是 BLOB、BLOB_FILE 或 BLOB_LOCATOR。DB2 将内容的数据类型提升为 BLOB_LOCATOR，并将 LOB 定位器传送至 DB2Audio UDF。

format（VARCHAR(8)）

源音频的格式。可指定空值或空字符串，在这种情况下，Audio Extender 将试图自动确定源格式。将把音频存储成与其源的格式相同。参见第 93 页的表 8 以获取受支持的音频格式。

target_file （LONG VARCHAR）

目标服务器文件的名称（对于存储至服务器文件），或是空值或空字符串（对于以 BLOB 形式存储到数据库表中）。目标文件可以是全限定名。若未限定名称，则使用服务器上的 DB2AUDIOSTORE 和 DB2MMSTORE 环境变量来找出该文件。

source_file （LONG VARCHAR）

源服务器文件的名称。源文件名可以是全限定名或非限定名；但不能是空值或空字符串。若未限定名称，则使用服务器上的 DB2AUDIOPATH 和 DB2MMPATH 环境变量来找出该文件。

stortype（INTEGER）

指示将在何处存储音频的值。常量 MMDB_STORAGE_TYPE_INTERNAL（值 = 1）指示将把音频以 BLOB 形式存储在数据库中；常量 MMDB_STORAGE_TYPE_EXTERNAL（值 = 0）指示将把音频内容存储在（数据库中指向的）服务器文件中。

comment （LONG VARCHAR）

要与音频存储在一起的注释。

attrs（LONG VARCHAR FOR BIT DATA）

音频的属性。

返回值（数据类型）

音频的句柄（DB2AUDIO）

例

将包括 Anita Jones 的音频剪辑的记录插入到 Employee 表中。音频源在客户机缓冲区中。将音频剪辑以 BLOB 形式存储在表中:

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS BLOB (5M) aud_seg;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2AUDIO(
         CURRENT SERVER,
         :aud_seg,
         'WAVE',
         CAST(NULL as LONG VARCHAR),

         'Anita''s voice'));
```

将包括 Robert Smith 的音频剪辑的记录插入到 Employee 表中。音频源在服务器文件中。Employee 表记录将指向该文件。

```
EXEC SQL BEGIN DECLARE SECTION;
     long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2AUDIO(
         CURRENT SERVER,
         '/employee/sounds/rsmith.wav',
         'WAV',
         :hvStorageType,
         'Robert''s voice'));
```

将包括 Anita Jones 的音频剪辑的记录插入到 Employee 表中。以 BLOB 形式存储音频剪辑。服务器文件中的源音频剪辑具有用户定义的格式，采样速率为 44.1 KHz，并有两个录制声道。

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[600];
}hvAudattr;
EXEC SQL END DECLARE SECTION;

MMDBAudioAttrs          *paudiattr;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;

paudioattr=(MMDBAudioAttrs *) hvAudattr.data;
strcpy(paudioAttr->cFormat,"cFormatA");
paudioAttr->ulSamplingRate=44100;
paudioAttr->usNumChannels=2;
hvAudattrs.len=sizeof(MMDBAudioAttrs);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
               DB2AUDIO(
        CURRENT SERVER,
               '/Employee/Sounds/ajones.aud',
        :hvStorageType,
               'Anita"s voice',
               :hvAudattr)
       );
```

DB2Image

图像	音频	视频
X		

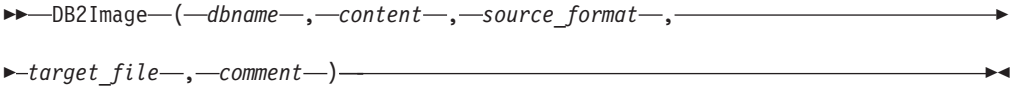
将图像的内容存储在数据库表中。图像源可以在客户机缓冲区、客户机文件或服务器文件中。可将图像以 BLOB 形式存储在数据库表中，或存储在服务器文件中（由数据库表引用）。图像源可以处于受支持的格式，在这种情况下，DB2Image Extender 标识其存储属性，也可处于不受支持的格式，在这种情况下，必须在 UDF 中指定属性。

包含文件

dmbimage.h

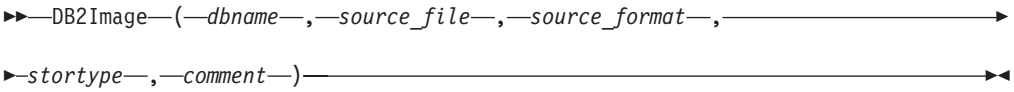
语法

从缓冲区或客户机文件存储内容



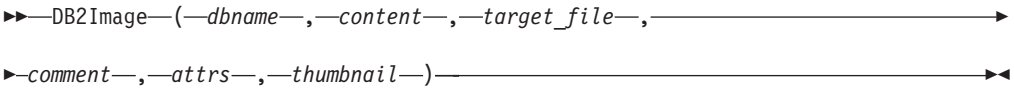
语法

从服务器文件存储内容



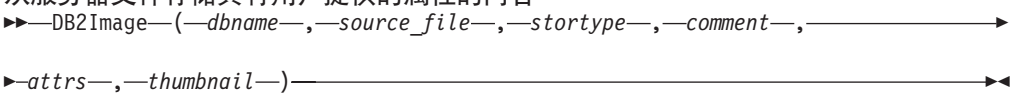
语法

从缓冲区或客户机文件存储具有用户提供的属性的内容



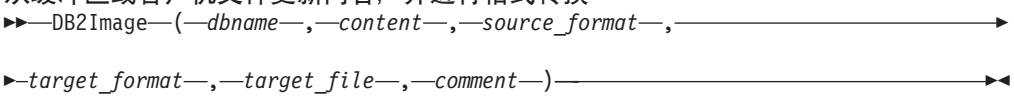
语法

从服务器文件存储具有用户提供的属性的内容



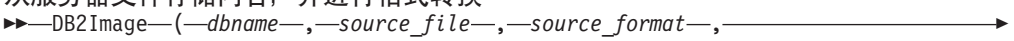
语法

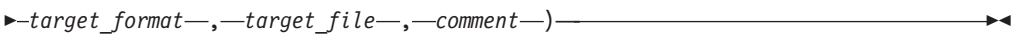
从缓冲区或客户机文件更新内容，并进行格式转换



语法

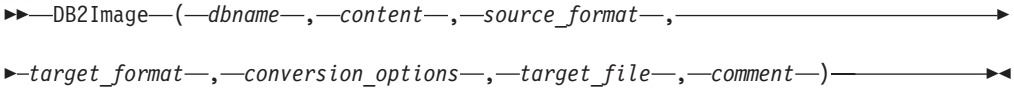
从服务器文件存储内容，并进行格式转换





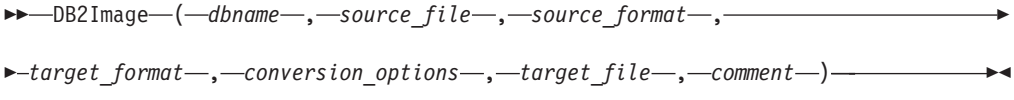
语法

从缓冲区或客户机文件存储内容，并进行格式转换和其它更改



语法

从服务器文件存储内容，并进行格式转换和其它更改



参数（数据类型）

dbname（VARCHAR(18)）

当前连接的数据库的名称，由 CURRENT SERVER 专用寄存器指示。

content（BLOB(2G) AS LOCATOR）

包含图像的内容的主变量。主变量的类型可以是 BLOB、BLOB_FILE 或 BLOB_LOCATOR。DB2 将内容的数据类型转换为 BLOB_LOCATOR，并将 LOB 定位器传送至 DB2Image UDF。

source_format（VARCHAR(8)）

源图像的格式。可指定空值、空字符串或字符串 ASIS；在任何这三种情况下，Image Extender 都将试图自动确定源格式。将把图像存储成与它的源的格式相同。参见第 93 页的表 8 以获取受支持的图像格式。

target_format（VARCHAR(8)）

存储之后图像的格式。将适当地转换源图像的格式。

target_file （LONG VARCHAR）

目标服务器文件的名称（对于存储至服务器文件），或是空值或空字符串（对于以 BLOB 形式存储到数据库表中）。目标文件名可以是全限定名。若未限定名称，则使用服务器上的 DB2IMAGESTORE 和 DB2MMSTORE 环境变量来找出该文件。若存储图像时进行了格式转换，则需要在 DB2IMAGEPATH 和 DB2MMPATH 环境变量中指定目标文件的路径。

source_file （LONG VARCHAR）

源服务器文件的名称。源文件名可以是全限定名或非限定名；但不能是空值或空字符串。若未限定名称，则使用服务器上的 DB2IMAGEPATH 和 DB2MMPATH 环境变量来找出该文件。

stortype（INTEGER）

指示将在何处存储图像的值。常量 MMDB_STORAGE_TYPE_INTERNAL（值 = 1）指示将把图像以 BLOB 形式存储在数据库中；常量 MMDB_STORAGE_TYPE_EXTERNAL（值 = 0）指示将把图像内容存储在（数据库中指向的）服务器文件中。

comment （LONG VARCHAR）

要与图像存储在一起的注释。

attrs（LONG VARCHAR FOR BIT DATA)
图像的属性。

thumbnail（LONG VARCHAR FOR BIT DATA）
图像的缩略图。

conversion_options（VARCHAR(100)）
指定存储图像时，要对其应用的更改，如旋转和压缩。参见第 94 页的表 9 以获取受支持的转换选项。

返回值（数据类型）

图像的句柄（DB2IMAGE）

例

将包括 Anita Jones 的图像的记录插入到 Employee 表中。图像源在客户机缓冲区中。将图像以 BLOB 形式存储在表中:

```
EXEC SQL BEGIN DECLARE SECTION
     SQL TYPE IS BLOB (2M) hvImg
   EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2IMAGE(
         CURRENT SERVER,
         :hvImg,
         'ASIS',
         CAST(NULL as LONG VARCHAR),

         'Anita''s picture'));
```

将包括 Robert Smith 的图像的记录插入到 Employee 表中。图像源在服务器文件中。Employee 表记录将指向该文件。存储时，将图像的格式由 BMP 转换为 GIF。并将图像修剪成宽度为 110 像素，高度为 150 像素，并使用 LZW 类型压缩来压缩图像:

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2IMAGE(
         CURRENT SERVER,
         '/employee/pictures/rsmith.bmp',
          'BMP',
          'GIF',
         '-x 110 -y 150 -c 14',
          '',
         'Robert"s picture'));
```

将包括 Robert Smith 的图像的记录插入到 Employee 表中。服务器文件中的源图像具有用户定义的格式，高度为 640 像素，宽度为 480 像素。以 BLOB 形式存储图像:

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
}hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;
```

```
pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattrt->cFormat,"FormatI");
pimgattr->width=640;
pimgattr->height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2IMAGE(
         CURRENT SERVER,
          '/Employee/images/ajones.bmp',
         :hvStorageType,
          'Anita''s picture',
          :hvImgattrs,
          CAST(NULL as LONG VARCHAR))
       );
```

DB2Video

图像	音频	视频
		X

将视频的内容存储在数据库表中。视频源可以在客户机缓冲区、客户机文件或服务器文件中。可将视频以 BLOB 形式存储在数据库表中，或存储在服务器文件中（由数据库表引用）。视频源可处于受支持的格式，在这种情况下，DB2Video Extender 标识其存储属性，也可处于不受支持的格式，在这种情况下，必须在 UDF 中指定属性。

包含文件

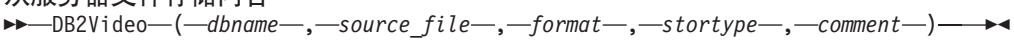
dmbvideo.h

语法

从缓冲区或客户机文件存储内容

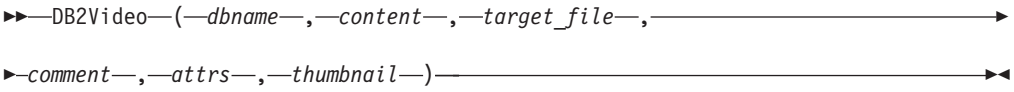
语法

从服务器文件存储内容



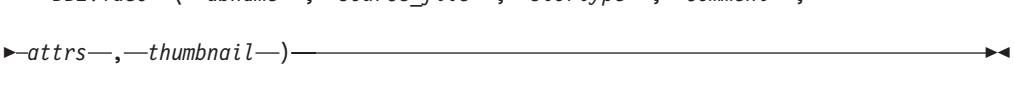
语法

从缓冲区或客户机文件存储具有用户提供的属性的内容



语法

从服务器文件存储具有用户提供的属性的内容



参数（数据类型）

dbname（VARCHAR(18)）

当前连接的数据库的名称，由 CURRENT SERVER 专用寄存器指示。

content（BLOB(2G) AS LOCATOR）

包含视频的内容的主变量。主变量的数据类型可以是 BLOB、BLOB_FILE 或 BLOB_LOCATOR。DB2 将内容转换为 BLOB_LOCATOR，并将 LOB 定位器传送至 DB2Video UDF。若内容在客户机缓冲区中，则缓冲区必须包含内容的至少前 640 KB 才能确保读整个视频头。

format（VARCHAR(8)）

源视频的格式。若指定了空值或空字符串，则 Video Extender 将尝试自动确定源格式。将把视频存储成与它的源的格式相同。参见第 93 页的表 8 以了解受支持的视频格式。对于 MPG1 格式，可指定 MPG1、mpg1、MPEG1 或 mpeg1。对于 MPG2 格式，可指定 MPG2、mpg2、MPEG2 或 mpeg2。

target_file （LONG VARCHAR）

目标服务器文件的名称（对于存储至服务器文件），或是空值或空字符串（对于以 BLOB 形式存储到数据库表中）。服务器文件只能是全限定名。若未限定文件名，则使用服务器上的 DB2VIDEOSTORE 和 DB2MMSTORE 环境变量来找出该文件。

source_file （LONG VARCHAR）

源服务器文件的名称。名称可以是全限定名或非限定名；但不能是空值或空字符串。若未限定名称，则使用服务器上的 DB2VIDEOPATH 和 DB2MMPATH 环境变量来找出该文件。

stortype（INTEGER）

指示将在何处存储视频的值。常量 MMDB_STORAGE_TYPE_INTERNAL（值 = 1）指示将把视频以 BLOB 形式存储在数据库中；常量 MMDB_STORAGE_TYPE_EXTERNAL（值 = 0）指示将把视频内容存储在（数据库中指向的）服务器文件中。

comment （LONG VARCHAR）

要与视频存储在一起的注释。

attrs（LONG VARCHAR FOR BIT DATA）

视频的属性。

thumbnail（LONG VARCHAR FOR BIT DATA）

表示视频的缩略图像。

返回值（数据类型）

视频的句柄（DB2VIDEO）

例

将包括 Anita Jones 的视频剪辑的记录插入到 Employee 表中。视频源在客户机缓冲区中。将视频剪辑以 BLOB 形式存储在表中:

```
EXEC SQL BEGIN DECLARE SECTION
     SQL TYPE IS BLOB (8M) vid;
   EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
        DB2VIDEO(
         CURRENT SERVER,
         :vid,
         'MPEG1',
         CAST(NULL as LONG VARCHAR),         'Anita''s video'));
```

将包括 Robert Smith 的视频剪辑的记录插入到 Employee 表中。视频源在服务器文件中。Employee 表记录将指向该文件:

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
        DB2VIDEO(
         CURRENT SERVER,
```

```
          '/Employee/Videos/rsmith.mpg',
          'MPEG1',
          :hvStorageType,
          'Robert''s video'));
```

将包括视频剪辑的记录插入到数据库表中。服务器文件中的源视频剪辑具有用户定义的格式。将视频内容存放在服务器文件中（数据库表记录将指向该文件）。还存储表示该视频的缩略图:

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
}hvVidattrs;
struct {
       short len;
       char data[10000];
}hvThumbnail;
EXEC SQL END DECLARE SECTION;

MMDBVideoAttrs         *pvideoAttr;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

pvideoAttr=(MMDBVideoAttrs *)hvVidattrs.data;
strcpy(pvideoAttr→cFormat,"Formatv");
pvideoAttr.len=sizeof(MMDBVideoAttrs);
:
:
/* Generate thumbnail and assign data   */
/* in video structure                   */
:
:
EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
        DB2VIDEO(
         CURRENT SERVER,
          '/Employee/Videos/ajones.vid',
         :hvStorageType,
          'Anita''s video',
          :hvVidattrs,
                :hvThumbnail)
       );
```

Duration

图像	音频	视频
	X	X

返回 WAVE 或 AIFF 音频，或视频的持续时间（即，以秒计的播放时间）。

包含文件

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2AUDIO 或 DB2VIDEO）
包含音频或视频的句柄的列名或主变量。

返回值（数据类型）

视频的以秒计的持续时间，或 WAVE、AIFF 或用户定义格式音频的以秒计的持续时间（INTEGER）。对于其它格式的音频，返回空值。

例

显示存储在 employee 表的 video 列中的所有视频的持续时间:

```
EXEC SQL BEGIN DECLARE SECTION;
long hvDur_vid;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT DURATION(VIDEO)
     INTO :hvDur_vid
     FROM EMPLOYEE;
```

Filename

图像	音频	视频
X	X	X

返回包含图像、音频或视频的内容的服务器文件的名称（若对象内容存储在数据库表中指向的文件中）。若将图像、音频或视频以 BLOB 形式存储在数据库表中，则返回空值。

包含文件

图像（image）
dmbimage.h

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）
包含图像、音频或视频的句柄的列名或主变量。

返回值（数据类型）

若对象内容在服务器文件中，则是服务器文件的文件名 （VARCHAR(250)）；若以 BLOB 形式存储对象，则是空值。

例

显示 Employee 表中的 Robert Smith 项的视频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NAME='Robert Smith';
```

FindInstrument

图像	音频	视频
	X	

返回 MIDI 音频中指定的乐器的首次出现的声道号。

包含文件

dmbaudio.h

语法

参数（数据类型）

handle（**DB2AUDIO**）

包含音频的句柄的列名或主变量。

instrument（**VARCHAR(255)**）

要搜索的乐器的名称。Audio Extender 将寻找其名称与提供的名称完全匹配的乐器。

返回值（数据类型）

包含指定乐器名的首次出现的声道号（SMALLINT）；若找不到指定的乐器名，则返回值 -1。对于其它格式的音频，返回 NULL。

例

在存储在 Employee 表的 Sound 列中的 Robert Smith 的录音中查找 PIANO 的首次出现:

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvInstr;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FINDINSTRUMENT(SOUND, 'PIANO')
      INTO :hvInstr
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

FindTrackName

图像	音频	视频
	X	

返回 MIDI 音频中指定的命名声道的名称。

包含文件

dmbaudio.h

语法

参数（数据类型）

handle（DB2AUDIO）

包含音频的句柄的列名或主变量。

trackname（VARCHAR(255)）

要搜索的声道的名称。Audio Extender 将寻找其名称与提供的名称完全匹配的声道。

返回值（数据类型）

指定的乐器名的命名声道号（SMALLINT）。若找不到指定的名称的声道，则返回值 -1。对于其它格式的音频，返回空值。

例

确定 Robert Smith 的 MIDI 录音中是否有名为 WELCOME 的声道。录音存储在 employee 表的 sound 列中:

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvTrack;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FINDTRACKNAME(SOUND,
          'WELCOME')
      INTO :hvTrack
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

Format

图像	音频	视频
X	X	X

返回图像、音频或视频的格式。

包含文件

图像（image）
dmbimage.h

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）
包含图像、音频或视频的句柄的列名或主变量。

返回值（数据类型）

图像、音频或视频的格式（VARCHAR(8)）。参见第 93 页的表 8 以获取受支持的图像、音频和视频格式。

例

获取其存储在 Employee 表的 Picture 列中的图像处于 GIF 格式的所有雇员的姓名:

```
EXEC SQL BEGIN DECLARE SECTION;
char hvName[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NAME
     INTO :hvName
     FROM EMPLOYEE
     WHERE FORMAT(PICTURE)='GIF';
```

FrameRate

图像	音频	视频
		X

返回视频的吞吐量，以帧频计。

包含文件

dmbvideo.h

语法

参数（数据类型）

handle（DB2VIDEO）

包含视频的句柄的列名或主变量。

返回值（数据类型）

视频的帧速率（SMALLINT）。若吞吐量速率可变，则返回空值。

例

获取 employee 表的 video 列中存储的 Anita Jones 的视频的帧速率:

```
EXEC SQL BEGIN DECLARE SECTION;
short hvFm_rate;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FRAMERATE (VIDEO)
     FROM EMPLOYEE
      INTO :hvFm_rate
     WHERE NAME='Anita Jones';
```

GetInstruments

图像	音频	视频
	X	

返回 MIDI 音频中所有乐器的乐器名。

包含文件

dmbaudio.h

语法

参数（数据类型）

handle（**DB2AUDIO**）

包含音频的句柄的列名或主变量。

返回值（数据类型）

MIDI 音频中所有乐器的乐器名（VARCHAR(1536)）。这些值是以声道号次序返回的（例如，PIANO; TRUMPET; BASS）。结果分成 *n* 个字段，其中 *n* 是 MIDI 音频的声道号。若声道没有相关联的乐器，则其字段是空白。对于非 MIDI 的音频格式，返回空值。

例

在 Robert Smith 的录音中查找所有乐器（即，声道号和乐器名）。录音存储在 employee 表的 sound 列中:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_Instr[1536];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT GETINSTRUMENTS(SOUND)
      INTO :hvAud_Instr
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

GetTrackNames

图像	音频	视频
	X	

返回 MIDI 音频中所有声道的名称。

包含文件

dmbaudio.h

语法

参数（数据类型）

handle（DB2AUDIO）

包含音频的句柄的列名或主变量。

返回值（数据类型）

MIDI 音频中所有声道的名称（VARCHAR(1536)）。这些值是以声道号次序返回的（例如，PIANO TUNE; TRUMPET FANFARE）。结果分成 *n* 个字段，其中 *n* 是 MIDI 音频的声道号。若声道没有名称，则其字段是空白。对于非 MIDI 的音频格式，返回空值。

例

获取存储在 Employee 表的 Sound 列中的 Robert Smith 的 MIDI 录音中的所有声道号和声道名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvTracks[1536];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT GETTRACKNAMES(SOUND)
      INTO :hvTracks
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

Height

图像	音频	视频
X		X

返回图像或视频的高度，以像素计。

包含文件

图像（image）
dmbimage.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE 或 DB2VIDEO）
包含图像或视频的句柄的列名或主变量。

返回值（数据类型）

以像素计的高度（INTEGER）

例

获取 Employee 表的 Picture 列中所有短于 500 像素的图像的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE HEIGHT(PICTURE)<500;
```

Importer

图像	音频	视频
X	X	X

返回将图像、音频或视频存储在数据库表中的人的用户标识。

包含文件

图像（image）
dmbimage.h

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）
包含图像、音频或视频的句柄的列名或主变量。

返回值（数据类型）

导入者的用户标识（CHAR(8)）

例

获取用户标识 rsmith 存储在 Employee 表的 Sound 中的音频的所有文件的名称:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE IMPORTER(SOUND)='rsmith';
```

ImportTime

图像	音频	视频
X	X	X

返回指示何时将图像、音频或视频存储在数据库表中的时间戳记。

包含文件

图像（image）
dmbimage.h

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）
包含图像、音频或视频的句柄的列名或主变量。

返回值（数据类型）

存储图像、音频或视频时的时间戳记（TIMESTAMP）

例

获取在 Employee 表的 Picture 列中存储时间超过 1 年的图像的所有文件的名称:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE(CURRENT TIMESTAMP -
        IMPORTTIME(PICTURE))>365;
```

MaxBytesPerSec

图像	音频	视频
		X

返回视频的最大吞吐量，以每秒字节数计。

包含文件

dmbvideo.h

语法

参数（数据类型）

handle（DB2VIDEO）

包含视频的句柄的列名或主变量。

返回值（数据类型）

视频的吞吐量（INTEGER）。若吞吐量速率可变，则返回空值。

例

获取 employee 表的 video 列中存储的 Anita Jones 的视频的最大吞吐量:

```
EXEC SQL BEGIN DECLARE SECTION;
long hvMax_BytesPS;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT MAXBYTESPERSEC(VIDEO)
     INTO :hvMax_BytesPS
     FROM EMPLOYEE
     WHERE NAME='Anita Jones';
```

NumAudioTracks

图像	音频	视频
	X	X

返回视频或 MIDI 音频中的声道数。

包含文件

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2AUDIO 或 DB2VIDEO）
包含音频或视频的句柄的列名或主变量。

返回值（数据类型）

视频或 MIDI 音频中的声道数（SMALLINT）。对于其它格式的音频，返回空值。

例

获取 Employee 表的 Video 列中未包含任何声道的任何文件的名称:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMAUDIOTRACKS(VIDEO) = 0;
```

NumChannels

图像	音频	视频
	X	X

返回 WAVE 或 AIFF 音频中，或视频中录制的声道的数。

包含文件

音频（audio）

dmbaudio.h

视频（video）

dmbvideo.h

语法

参数（数据类型）

handle（DB2AUDIO 或 DB2VIDEO）

包含音频或视频的句柄的列名或主变量。

返回值（数据类型）

视频，或 WAVE 或 AIFF 音频中录制的声道数（SMALLINT）。对于其它格式的音频，返回空值。

例

获取 Employee 的 Sound 中以立体声（即双声道）录制的所有音频文件的名称:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE NUMCHANNELS(SOUND) = 2;
```

NumColors

图像	音频	视频
X		

返回图像中的色彩数。

包含文件

dmbimage.h

语法

参数（数据类型）

handle（DB2IMAGE）

包含图像的句柄的列名或主变量。

返回值（数据类型）

图像中的色彩数（INTEGER）

例

获取 Employee 表的 Picture 列中少于 16 色的图像文件的名称:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE NUMCOLORS(PICTURE) < 16;
```

NumFrames

图像	音频	视频
		X

返回视频中的帧数。

包含文件

dmbvideo.h

语法

参数（数据类型）

handle（DB2VIDEO）

包含视频的句柄的列名或主变量。

返回值（数据类型）

视频中的帧数（INTEGER）。若吞吐量速率可变，则返回空值。

例

获取 employee 表的 video 列中存储的 Robert Smith 的视频中的帧数:

```
EXEC SQL BEGIN DECLARE SECTION;
long hvNum_Frames;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NUMFRAMES (VIDEO)
     INTO :hvNum_Frames
     FROM EMPLOYEE
     WHERE NAME='Robert Smith';
```

NumVideoTracks

图像	音频	视频
		X

返回视频中的视频声道数。

包含文件

dmbvideo.h

语法

参数（数据类型）

handle（**DB2VIDEO**）

包含视频的句柄的列名或主变量。

返回值（数据类型）

视频声道数（SMALLINT）

例

获取 Employee 表的 Video 列中有多个视频声道的所有视频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMVIDEOTRACKS(VIDEO) > 1;
```

QbScoreFromName

图像	音频	视频
X		

返回图像的得分，它是表示图像的特征与查询对象的特征的匹配程度的数字。使用与图像句柄所属的列相关的 QBIC 目录来计算图像的得分。得分越低，图像的特征就与指定的查询对象的特征越匹配。（QbScoreFromName 替换 QbScore，但仍接受 QbScore。）

注:

1. **仅限于 EEE**: QbScoreFromName 在分区数据库环境中不受支持。作为代替，在使用 QbQueryGetString API 获取查询字符串之后，是使用 QbScoreFromStr UDF。
2. 在将来的发行版中，非分区数据库环境也将不支持 QbScoreFromName。要重新使用查询，应使用 QbQueryGetString API 来获取查询字符串并保存该字符串，以供应用程序以后使用。

包含文件

无

语法

语法

不支持版本

参数（数据类型）

imgHandle (DB2Image)

图像的句柄。

queryName（varchar(18)）

查询对象的名称。

返回值（数据类型）

图像的得分（DOUBLE）。得分可以是从 0.0 到接近无穷大的非常大的值。得分越低，目标图像的特征值就与查询中指定的特征值越匹配。得分 0.0 表示精确匹配。得分为空值表示未编目该图像；当未编目该图像时，此 UDF 的不支持版本返回得分 -1。

例

查找表列中平均色彩非常接近于红色的已编目图像:

```
EXEC SQL BEGIN DECLARE SECTION;
char Img_fnd[100];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NAME
  INTO :Img_fnd
  FROM FABRIC
  WHERE (QBSCOREFROMNAME(SWATCH_IMG,
         'fshavgcol'))<0.1;
```

QbScoreFromStr

图像	音频	视频
X		

返回图像的得分，它是表示图像的特征与查询字符串的特征有多匹配的数字。使用与图像句柄所属的列相关的 QBIC 目录来计算图像的得分。得分越低，图像的特征就与查询字符串的特征越匹配。

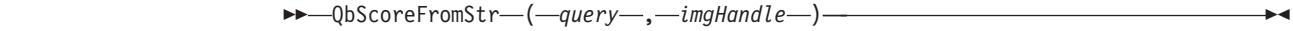
包含文件

无

语法

语法

不支持版本



参数（数据类型）

imgHandle（DB2Image）

图像的句柄。

query（VARCHAR(1024)）

查询字符串。

返回值（数据类型）

图像的得分（DOUBLE）。得分可以是从 0.0 到接近无穷大的非常大的值。得分越低，目标图像的特征值就与查询中指定的特征值越匹配。得分 0.0 表示精确匹配。得分为空值表示未编目该图像；当未编目该图像时，此 UDF 的不支持版本返回得分 -1。

例

查找表列中平均色彩非常接近于红色的已编目图像:

```
SELECT name
 FROM fabric
WHERE (QbScoreFromStr(Swatch_Img,
       'QbColorFeatureClass color=<255, 0, 0>'))<0.1
```

QbScoreTBFromName

图像	音频	视频
X		

返回图像列的得分表。每个得分都是表示图像的特征与查询对象的特征有多匹配的数字。使用与图像句柄所属的指定表和列相关的 QBIC 目录来计算图像的得分。任何图像的得分越低，该图像的特征与查询对象的特征越匹配。

注:

1. **仅限于 EEE**: QbScoreTBFromName 在分区数据库环境中不受支持。作为代替，在使用 QbQueryGetString API 获取查询字符串之后，是使用 QbScoreFromStr UDF。
2. 将来，非分区数据库环境也将不支持 QbScoreTBFromName。要重新使用查询，应使用 QbQueryGetString API 来获取查询字符串并保存该字符串，以供应用程序以后使用。

包含文件

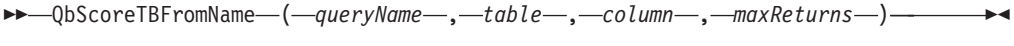
无

语法

返回列中所有已编目图像的得分

语法

返回列中特定数目的已编目图像的得分



参数（数据类型）

queryName（VARCHAR(18)）
查询对象的名称。

table（CHAR (18)）
包含图像列的表的限定表。如果表模式与用来启动 DB2 Extender 服务的用户标识相同，则可以使用非限定表名。

column（CHAR(18)）
图像列的名称。

maxReturns（INTEGER）
要返回的结果表的最大句柄数。若未指定值，则返回的最大句柄数是 100。

返回值（数据类型）

列中图像的图像句柄和得分的表。结果表有两个列: IMAGE_ID（DB2Image），它包含图像句柄，以及 SCORE（DOUBLE），它包含得分。结果表是按得分的升序排列的。得分可以是从 0.0 到接近无穷大的非常大的值。得分越低，目标图像的特征值就与查询中指定的特征值越匹配。得分 0.0 表示精确匹配。得分 -1 表示未编目该图像。

例

将表列中图像的纹理与查询对象中指定的纹理作比较; 返回图像句柄及其得分:

```
SELECT name, description
INTO :hvName, :hvDesc
FROM fabric
WHERE CAST (swatch_img as varchar(250)) IN
 (SELECT CAST (image_id as varchar(250)) FROM TABLE
 (QbScoreTBFromName
     'fstxtr',
     'clothes.fabric',
     'swatch_img'))
 AS T1));
```

QbScoreTBFromStr

图像	音频	视频
X		

返回图像列中的得分表。每个得分都是表示图像的特征与查询字符串中指定的特征有多匹配的数字。使用与图像句柄所属的表和列相关的 QBIC 目录来计算图像的得分。图像的得分越低，该图像的特征与查询字符串的特征越匹配。

包含文件

无

语法

返回列中所有已编目图像的得分

语法

返回列中特定数目的已编目图像的得分

参数（数据类型）

query（VARCHAR(1024)）

查询字符串。

table（CHAR (18)）

包含图像列的表的限定表。如果表模式与用来启动 DB2 Extender 服务的用户标识相同，则可以使用非限定表名。

column（CHAR(18)）

要查询的图像列。

maxReturns（INTEGER）

要返回的结果表的最大句柄数。若未指定值，则返回的最大图像句柄数是 100。

返回值（数据类型）

列中图像的图像句柄和得分的表。结果表有两个列: IMAGE_ID（DB2Image），它包含图像句柄，以及 SCORE（DOUBLE），它包含得分。结果表是按得分的升序排列的。得分可以是从 0.0 到接近无穷大的非常大的值。得分越低，目标图像的特征值就与查询中指定的特征值越匹配。得分 0.0 表示精确匹配。得分 -1 表示未编目该图像。

例

在表列中查找 10 个其纹理最接近于服务器文件中的图像的纹理的已编目图像:

```
SELECT name, description
 FROM fabric
WHERE CAST (swatch_img as varchar(250)) IN
 (SELECT CAST (image_id as varchar(250)) FROM TABLE
 (QbScoreTBFromStr
 (QbTextureFeatureClass file=<server,"patterns/ptrn07.gif">'
      'clothes.fabric',
      'swatch_img',
      10))
  AS T1));
```

Replace

图像	音频	视频
X	X	X

更新存储在数据库中的图像、音频或视频的内容，并更新其注释。

包含文件

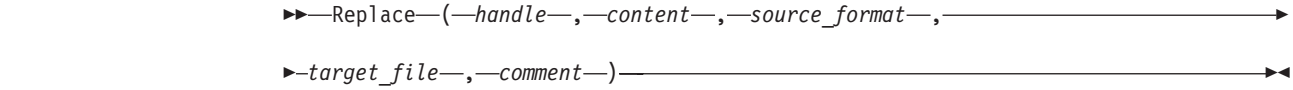
图像（**image**）
dmbimage.h

音频（**audio**）
dmbaudio.h

视频（**video**） dmbvideo.h

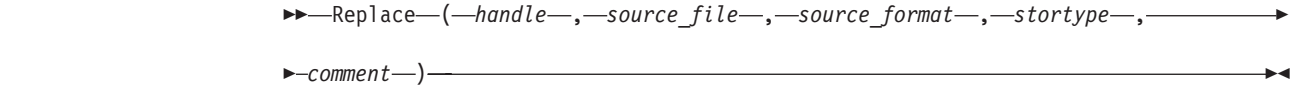
语法

从缓冲区或客户机文件更新内容，并更新注释



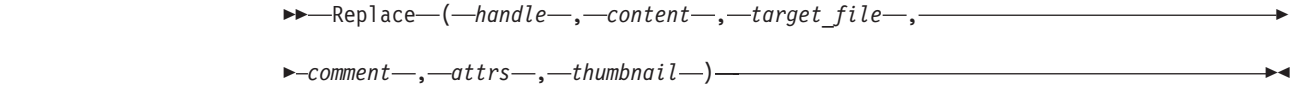
语法

从服务器文件更新内容，并更新注释



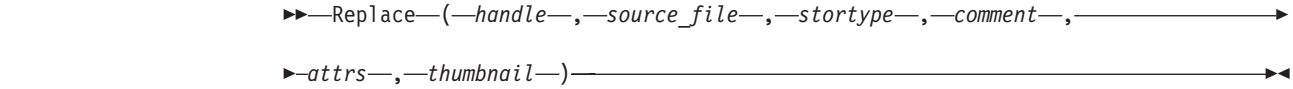
包含文件

从缓冲区或客户机文件更新具有用户提供的属性的内容，并更新注释



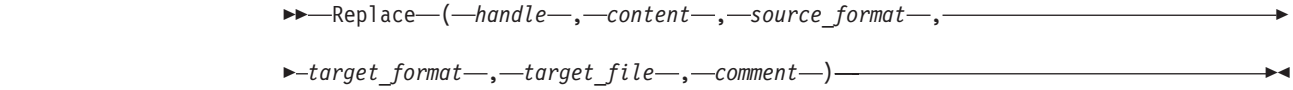
包含文件

从服务器文件更新具有用户提供的属性的内容，并更新注释



语法

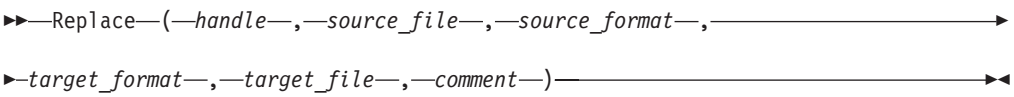
从缓冲区或客户机文件更新内容，进行格式转换，并更新注释 — 仅适用于图像



语法

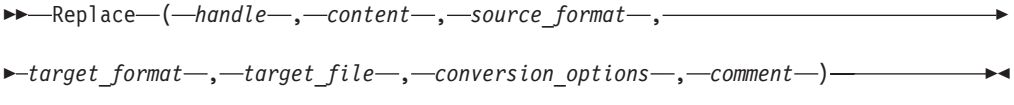
从服务器文件更新内容，进行格式转换，并更新注释 — 仅适用于图像

Replace



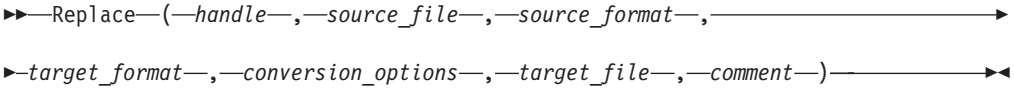
语法

从缓冲区或客户机文件更新内容，进行格式转换和其它更改，并更新注释 — 仅适用于图像



语法

从服务器文件更新内容，并进行格式转换、附加更改和更新注释 — 仅适用于图像



参数（数据类型）

handle（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）
包含图像、音频或视频的句柄的列名或主变量。

source_file （LONG VARCHAR）
包含图像、音频或视频的更新内容的文件名。

target_file （LONG VARCHAR）
包含要更新的图像、音频或视频的内容的文件名。

create_target（INTEGER）
指示当源内容在服务器文件中时，是否将创建目标文件的值。值可以是 0 或 1。值 0 表示将不创建目标文件（实际上，将不发生检索）。值 1 表示将创建目标文件（若目标文件已存在，此值的作用是用来覆盖该文件）。若源内容是 BLOB，则总是创建目标文件（若该文件已存在，将覆盖它）。

target_format（VARCHAR(8)）
检索之后图像的格式。将适当地转换源图像的格式。若更新内容时进行了格式转换，则需要在 DB2IMAGEPATH 和 DB2MMPATH 环境变量中指定目标文件的路径。对于 MPG1 格式，可指定 MPG1、mpg1、MPEG1 或 mpeg1。对于 MPG2 格式，可指定 MPG2、mpg2、MPEG2 或 mpeg2。

content（BLOB(2G) AS LOCATOR）
包含图像、音频或视频的更新内容的主变量的名称。主变量的类型可以是 BLOB、BLOB_FILE 或 BLOB_LOCATOR。DB2 将数据类型转换为 BLOB_LOCATOR，并将 LOB 定位器传送至 Replace UDF。

source_format（VARCHAR(8)）
图像、音频或视频的更新源的格式。可指定空值或空字符串，或仅对于图像，可指定字符串 ASIS；在这三种情况下，Extender 试图自动确定格式。对于 MPG1 格式，可指定 MPG1、mpg1、MPEG1 或 mpeg1。对于 MPG2 格式，可指定 MPG2、mpg2、MPEG2 或 mpeg2。

comment （LONG VARCHAR）
注释。

attrs（LONG VARCHAR FOR BIT DATA）
图像、音频或视频的属性

thumbnail（LONG VARCHAR FOR BIT DATA）
图像或视频帧的缩略图（仅适用于图像和视频）

conversion_options（VARCHAR(100)）
指定更新图像时，要对其应用的更改，如旋转和压缩。参见第 94 页的表 9 以获取受支持的转换选项。

返回值（数据类型）

要更新的图像、音频或视频的句柄（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）。

例

更新 Employee 表的 Picture 列中 Anita Jones 的图像，将图像的格式由 BMP 转换为 GIF，并更新注释:

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;
EXEC SQL UPDATE EMPLOYEE
  SET PICTURE = REPLACE(PICTURE,
        '/employee/newimg/ajones.bmp',
        'BMP',
        'GIF',
        :hvStorageType,
        'Anita''s new picture')
  WHERE NAME='Anita Jones';
```

SamplingRate

图像	音频	视频
	X	X

返回 WAVE 或 AIFF 音频的，或视频中声道的采样速率，以每秒采样数计。

包含文件

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2AUDIO 或 DB2VIDEO）
包含音频或视频的句柄的列名或主变量。

返回值（数据类型）

视频、WAVE 或 AIFF 音频的采样速率（INTEGER）。对于其它格式的音频，返回空值。

例

获取 Employee 表的 Sound 列中其采样速率是 44.1 KHz 的所有音频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
      INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE SAMPLINGRATE(SOUND) = 44100;
```

Size

图像	音频	视频
X	X	X

返回图像、音频或视频的大小，以字节计。

包含文件

图像（image）
dmbimage.h

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）
包含图像、音频或视频的句柄的列名或主变量。

返回值（数据类型）

图像、音频或视频的大小，以字节计（INTEGER）。

例

获取 Employee 表的 Picture 列中其大小大于 310 KB 的所有图像的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE SIZE(PICTURE) > 310000;
```

Thumbnail

图像	音频	视频
X		X

返回或更新存储在数据库中的图像或视频帧的缩略图大小版本。

包含文件

图像（**image**）
dmbimage.h

视频（**video**） dmbvideo.h

语法

检索缩略图

语法

更新缩略图

参数（数据类型）

handle（**DB2IMAGE 或 DB2VIDEO**）
包含图像或视频的句柄的列名或主变量。

new_thumbnail （**LONG VARCHAR FOR BIT DATA**）
缩略图的更新源内容

返回值（数据类型）

对于检索，为检索到的缩略图的内容 （LONG VARCHAR FOR BIT DATA），对于更新，为图像或视频的句柄（DB2IMAGE 或 DB2VIDEO）。

例

获取存储在 Employee 表中的 Anita Jones 的图像的缩略图:

```
EXEC SQL BEGIN DECLARE SECTION;
 struct{
       short len;
       char data [32000];
         }hvThumbnail;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT THUMBNAIL(PICTURE)
     INTO :hvThumbnail
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

更新与 employee 表中 Anita Jones 的视频相关联的缩略图:

```
EXEC SQL BEGIN DECLARE SECTION;
     struct {
       short len;
       char data[10000];
     }hvThumbnail;
EXEC SQL END DECLARE SECTION;
```

```
/*  Create thumbnail and    */
/*  store in hvThumbnail    */

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=THUMBNAIL(
                VIDEO,
               :hvThumbnail)
       WHERE NAME='Anita Jones';
```

TicksPerQNote

图像	音频	视频
	X	

返回录制的 MIDI 音频的时钟速率，以每四分音符的滴答数计。

包含文件

dmbaudio.h

语法

参数（数据类型）

handle（**DB2AUDIO**）

包含音频的句柄的列名或主变量。

返回值（数据类型）

MIDI 音频的每四分音符的时钟滴答数（SMALLINT）。对于其它格式的音频，返回空值。

例

获取 Employee 表的 Sound 列中以高于每四分音符 200 时钟滴答数的速度录制的所有 MIDI 音频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE FORMAT(SOUND)='MIDI'
      AND TICKSPERQNOTE(SOUND)>200;
```

TicksPerSec

图像	音频	视频
	X	

返回录制的 MIDI 音频的时钟速率，以每秒滴答数计。

包含文件

dmbaudio.h

语法

参数（数据类型）

handle（**DB2AUDIO**）

包含音频的句柄的列名或主变量。

返回值（数据类型）

MIDI 音频的每秒的时钟滴答数（SMALLINT）。对于其它格式的音频，返回空值。

例

获取 Employee 表的 Sound 列中以低于每秒 50 时钟滴答数的速度录制的所有 MIDI 音频的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE FORMAT(SOUND)='MIDI'
      AND TICKSPERSEC(SOUND)<50;
```

Updater

图像	音频	视频
X	X	X

返回上次更新数据库表中的图像、音频或视频的人的用户标识。

包含文件

图像（**image**）
dmbimage.h

音频（**audio**）
dmbaudio.h

视频（**video**） dmbvideo.h

语法

参数（数据类型）

handle（**DB2IMAGE**、**DB2AUDIO** 或 **DB2VIDEO**）
包含图像、音频或视频的句柄的列名或主变量。

返回值（数据类型）

上次更新图像、音频或视频的人的用户标识（CHAR(8)）

例

获取上次更新存储在 Employee 表的 Video 列中的 Robert Smith 的视频的人的用户标识:

```
EXEC SQL BEGIN DECLARE SECTION;
char hvUpdater[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT UPDATER(VIDEO)
     INTO :hvUpdater
     FROM EMPLOYEE
     WHERE NAME='rsmith';
```

UpdateTime

图像	音频	视频
X	X	X

返回指示上次更新数据库表中的图像、音频或视频时的时间戳记。

包含文件

图像（image）
dmbimage.h

音频（audio）
dmbaudio.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE、DB2AUDIO 或 DB2VIDEO）
包含图像、音频或视频的句柄的列名或主变量。

返回值（数据类型）

上次更新图像、音频或视频的时间戳记（TIMESTAMP）

例

获取 Employee 表的 Picture 列中前两天内更新的图像的文件的名称:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE(CURRENT TIMESTAMP -
       UPDATETIME(PICTURE))< 2;
```

Width

图像	音频	视频
X		X

返回图像或视频帧的宽度，以像素计。

包含文件

图像（image）
dmbimage.h

视频（video） dmbvideo.h

语法

参数（数据类型）

handle（DB2IMAGE 或 DB2VIDEO）
包含图像或视频的句柄的列名或主变量。

返回值（数据类型）

宽度，以像素计（INTEGER）

例

获取 Employee 表的 Picture 列中窄于 300 个像素的所有图像的文件名:

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE WIDTH(PICTURE)<300;
```

第 14 章 应用程序编程接口

本章提供关于 DB2 Extender 管理 API 的参考信息。API 按照字母次序列出。

为每个 API 提供了下列信息:

- 提供该 API 的 Extender
- 简要描述
- 使用此 API 所需的权限
- API 的库文件
- API 的包含（头）文件
- API 调用的 C 语法
- API 参数的描述
- API 所返回的值
- 使用示例

© Copyright IBM Corp. 1998, 2003

DBaAdminGetInaccessibleFiles

图像	音频	视频
	X	

返回用户表的音频列中引用的不可存取的文件的名称。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
在表中引用的不可存取的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

SQL_ERROR 或其它 SQL 返回码
从 DB2 返回了错误。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

例

列示用户标识 rsmith 拥有的表的音频列中引用的所有不可存取的文件:

```
#include <dmbaudio.h>
long idx;

rc = DBaAdminGetInaccessibleFiles("rsmith",
       &count, &filelist);
```

DBaAdminGetReferencedFiles

图像	音频	视频
	X	

返回用户表的音频列中引用的文件名。若某个文件不可存取（例如，不能使用环境变量规范解析其文件名），则文件名前面有一个星号。此 API 不使用 FILEREF 数据结构的 FILENAME 字段，因此将其设置为 NULL。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
表中引用的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

例

列示 ajones 拥有的表中音频列中引用的所有文件:

```
#include <dmbaudio.h>
long idx;

rc = DBaAdminGetReferencedFiles("ajones",
       &count, &fileList);
```

DBaAdminIsFileReferenced

图像	音频	视频
	X	

返回用户表中引用指定文件的音频列项的列表。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

fileName（输入）
引用的文件的名称。

count（输出）
输出列表中的条目数。

tableList（输出）
引用指定文件的表条目的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

例

列示当前数据库中所有表的音频列中引用文件 /audios/asmith.wav 的条目:

```
#include <dmbaudio.h>
long idx;

rc = DBaAdminIsFileReferenced(NULL,
       "/audios/asmith.wav",
       &count, &tableList);
```

DBaAdminReorgMetadata

图像	音频	视频
	X	

“清除”与音频相关的元数据表，例如:

- 收回音频元数据表中不再使用的空间
- 删除音频元数据表中对不再存在的音频文件的引用

在调用此 API 之前，应用程序必须与数据库相连。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaAdminReorgMetadata(
      char *qualifier
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定了用户标识，则清除带指定限定符的所有表。若指定了空值，则清除当前数据库中的所有表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

例

清除用户标识 rsmith 拥有的表中音频列的元数据表:

```
#include <dmbaudio.h>

rc = DBaAdminReorgMetadata("rsmith");
```

DBaDisableColumn

图像	音频	视频
	X	

对音频（DB2Audio 数据）禁用列，使其不能存放音频数据。将列条目的内容设置为 NULL，并删除与此列相关联的元数据。还将删除 Audio Extender 为此列定义的所有触发器。可将新行插入到包含被禁用列的表中，新行可包括用类型 DB2Audio 定义的数据，但（管理支持表中）没有与新行相关联的元数据。在调用此 API 之前，应用程序必须与数据库相连。建议在调用此 API 之后发出 SQL COMMIT 语句。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaDisableColumn(
      char *tableName,
      char *colName,
      );
```

参数

tableName（输入）
包含音频列的表的名称。

colName（输入）
音频列的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对音频（DB2Audio 数据）禁用 Employee 表中的 Sound 列:

```
#include <dmbaudio.h>

rc = DBaDisableColumn("employee", "sound");
```

DBaDisableDatabase

图像	音频	视频
	X	

对音频（DB2Audio 数据）禁用数据库，使其不能存放音频数据。还禁用该数据库中对 DB2Audio 定义的所有表。将删除 Audio Extender 为该数据库定义的元数据和 UDF。可将新行插入到该数据库中用类型 DB2Audio 定义的表中，但（管理支持表中）没有与新行相关联的元数据。建议在调用此 API 之后发出 SQL COMMIT 语句。

授权

DBADM、SYSADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaDisableDatabase(
      );
```

参数

DBaDisableDatabase 没有参数。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对音频（DB2Audio 数据）禁用当前数据库:

```
#include <dmbaudio.h>

rc = DBaDisableDatabase();
```

DBaDisableTable

图像	音频	视频
	X	

对音频（DB2Audio 数据）禁用表，使其不能存放音频数据。还禁用表中对 DB2Audio 定义的所有列。将删除 Audio Extender 对表定义的某些元数据。可将新行插入到用类型 DB2Audio 定义的表中，但（管理支持表中）没有与新行相关联的元数据。在调用此 API 之前，应用程序必须与数据库相连。建议在调用此 API 之后发出 SQL COMMIT 语句。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaDisableTable(
      char *tableName
      );
```

参数

tableName（输入）
包含音频列的表的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对音频（DB2Audio 数据）禁用 Employee 表:

```
#include <dmbaudio.h>

rc = DBaDisableTable("Employee");
```

DBaEnableColumn

图像	音频	视频
	X	

对音频（DB2Audio 数据）启用列。此 API 定义并管理此列与元数据表之间的关系。在调用此 API 之前，应用程序必须与数据库相连。建议在调用此 API 之后发出 SQL COMMIT 语句。

授权

控制、改变、SYSADM 或 DBADM

还需要对 API 参数中指定的表空间和缓冲池的使用特权。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaEnableColumn(
      char *tableName,
      char *colName,
      );
```

参数

tableName（输入）
包含音频列的表的名称。

colName（输入）
音频列的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
列已启用。

MMDB_RC_WRONG_SIGNATURE
所指定列的数据类型不正确。期望用户定义数据类型 MMDBSYS.DB2AUDIO。

MMDB_RC_COLUMN_DOESNOT_EXIST
该列未在指定表中定义。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NOT_ENABLED
未启用数据库或表。

例

对音频（DB2Audio 数据）启用 Employee 表中的 Sound 列:

```
#include <dmbaudio.h>

rc = DBaEnableColumn("employee", "sound");
```

DBaEnableDatabase

图像	音频	视频
	X	

对音频（DB2Audio 数据）启用数据库。对于每个数据库，调用一次此 API。它向数据库管理器定义 DB2 用户定义的类型 DB2Audio。它还创建处理 DB2Audio 数据的所有 UDF。建议在调用此 API 之后发出 SQL COMMIT 语句。

授权

DBADM、SYSADM、SYSCTRL

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaEnableDatabase(
      char *tableSpace
      );
```

参数

tableSpace（输入）

表空间的名称，表空间是在其中存储管理表的一组容器。表空间规范由如下三个部分组成：*datats*、*indexts 和 longts*，其中，*datats* 是在其中创建元数据表的表空间；*indexts* 是在其中创建元数据表上的索引的表空间；*longts* 是在其中存储元数据表中的长型列（诸如包含 LONG VARCHAR 和 LOB 数据类型的列）的值的表空间。若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

仅限于 EEE：在对 Extender 启用数据库时指定的表空间，应在包括分区数据库系统中所有节点的节点组上定义。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
数据库已启用。

MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER
连接的服务器不支持此命令。

MMDB_WARN_NOT_ALL_NODES
指定的表空间不包括 Extender 的所有节点。（仅限于 EEE）

MMDB_RC_NOT_SAME_NODEGROUP
指定的表空间不在同一节点组中。（仅限于 EEE）

例

在表空间 MYTS 中对音频（DB2Audio 数据）启用当前数据库: 对索引和长表空间使用缺省值:

```
#include <dmbaudio.h>

rc = DBaEnableDatabase("myts,,");
```

对音频（DB2Audio 数据）启用当前数据库; 使用缺省表空间:

```
#include <dmbaudio.h>

rc = DBaEnableDatabase(NULL);
```

DBaEnableTable

图像	音频	视频
	X	

对音频（DB2Audio 数据）启用表。对于每个表，调用一次此 API。它创建元数据表来存储和管理表中音频列的属性。要避免出现锁定的可能性，应用程序应在调用此 API 之前落实事务。在调用此 API 之前，应用程序必须与数据库相连。建议在调用此 API 之后发出 SQL COMMIT 语句。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaEnableTable(
      char *tableSpace,
      char *tableName
      );
```

参数

tableSpace（输入）

表空间的名称，表空间是在其中存储管理表的一组容器。表空间规范由如下三个部分组成：*datats*、*indexts 和 longts*，其中，*datats* 是在其中创建元数据表的表空间；*indexts* 是在其中创建元数据表上的索引的表空间；*longts* 是在其中存储元数据表中的长型列（诸如包含 LONG VARCHAR 和 LOB 数据类型的列）的值的表空间。若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

仅限于 EEE：指定的表空间应与用户表在同一节点组中。

tableName（输入）

将包含音频列的表的名称。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
表已启用。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_TABLE_DOESNOT_EXIST
表不存在。

MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP
指定的表空间与用户表不在同一节点组中。（**仅限于 EEE**）

例

在表空间中对音频（DB2Audio 数据）启用 Employee 表。使用索引和长型表空间的缺省值:

```
#include <dmbaudio.h>

rc = DBaEnableTable("myts,,",
       "employee");
```

对音频（DB2Audio 数据）启用 Employee 表。使用缺省表空间:

```
#include <dmbaudio.h>

rc = DBaEnableTable(NULL,
       "employee");
```

DBaGetError

图像	音频	视频
	X	

返回上一错误的描述。在任何其它 API 返回错误代码之后，调用此 API。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

参数

sqlcode（输出）
类属 SQL 错误代码。

errorMsgText（输出）
SQL 错误消息文本。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

例

获取上一错误，在 errCode 中存储 SQL 错误代码，并在 errMsg 中存储消息文本:

```
#include <dmbaudio.h>

rc = DBaGetError(&errCode, &errMsg);
```

DBaGetInaccessibleFiles

图像	音频	视频
	X	

返回用户表的音频列中引用的不可存取的文件的名称。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

对搜索的所有用户表和相关联的管理支持表中启用的音频列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对不可存取文件的引用。若指定了空值，则搜索带指定限定符的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
在表中引用的不可存取的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 employee 表的音频列中引用的所有不可存取的文件:

```
long idx;
#include <dmbaudio.h>

rc = DBaGetInaccessibleFiles("Employee",
       &count, &filelist);
```

DBaGetReferencedFiles

图像	音频	视频
	X	

返回用户表的音频列中引用的文件名。若某个文件不可存取（例如，不能使用环境变量规范解析其文件名），则文件名前面有一个星号。此 API 不使用 FILEREF 数据结构的 FILENAME 字段，因此将其设置为 NULL。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构。

授权

对搜索的所有用户表和相关联的管理支持表中启用的音频列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对文件的引用。若指定了空值，则搜索当前用户标识数据库所拥有的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
表中引用的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 employee 表的音频列中引用的所有文件:

```
#include <dmbaudio.h>
long idx;

rc = DBaGetReferencedFiles("employee",
       &count, &filelist);
```

DBaIsColumnEnabled

图像	音频	视频
	X	

确定是否已对音频（DB2Audio 数据）启用列。在调用此 API 之前，应用程序必须与数据库相连。

授权

SYSADM、DBADM、表所有者或对用户表的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaIsColumnEnabled(
      char *tableName,
      char *colName,
short  *status
      );
```

参数

tableName（输入）
限定的或非限定的表名。

colName（输入）
列的名称。

status（输出）
指示列是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

确定是否已对音频启用 Employee 表中的 Sound 列:

```
#include <dmbaudio.h>

rc = DBaIsColumnEnabled("employee",
       "sound", &status);
```

DBaIsDatabaseEnabled

图像	音频	视频
	X	

确定是否已对音频（DB2Audio 数据）启用数据库。在调用此 API 之前，应用程序必须与数据库相连。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaIsDatabaseEnabled(
short  *status
      );
```

参数

status（输出）

指示数据库是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED

应用程序没有与数据库的有效连接。

例

确定是否已对音频启用 personnl 数据库:

DBaIsDatabaseEnabled

```
#include <dmbaudio.h>

rc = DBaIsDatabaseEnabled(&status);
```

DBaIsFileReferenced

图像	音频	视频
	X	

返回引用指定文件的表条目的列表。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

对搜索的所有用户表和相关联的管理支持表中启用的音频列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对指定文件的引用。若指定了空值，则搜索当前用户标识所拥有的所有表。

fileName（输入）
引用的文件的名称。

count（输出）
输出列表中的条目数。

tableList（输出）
引用指定文件的表条目的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 employee 表的音频列中引用文件 /audios/ajones.wav 的条目:

```
#include <dmbaudio.h>
long idx;

rc = DBaIsFileReferenced(NULL,
       "/audios/ajones.wav",
       &count, &tableList);
```

DBaIsTableEnabled

图像	音频	视频
	X	

确定是否已对音频（DB2Audio 数据）启用表。在调用此 API 之前，应用程序必须与数据库相连。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaIsTableEnabled(
      char *tableName,
short  *status
      );
```

参数

tableName（输入）

表名。

status（输出）

指示表是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

确定是否已对音频（DB2Audio 数据）启用 Employee 表:

```
#include <dmbaudio.h>

rc = DBaIsTableEnabled("employee", &status);
```

DBaPlay

图像	音频	视频
	X	

在客户机上打开音频播放器，并播放音频剪辑。可将音频剪辑存储在音频列或外部文件中:

- 若将音频剪辑存储在外部文件中，则可将文件的名称或音频句柄传送至此 API。此 API 使用客户机环境变量 DB2AUDIOPATH 来解析文件位置。该文件必须可以从客户机工作站存取。
- 若将音频剪辑存储在列中，则必须将音频句柄传送至此 API。应用程序必须与数据库相连，且必须对其中存储音频剪辑的表具有读存取权。

若音频存储在列中，则 Extender 创建一个临时文件，并将对象的内容从该列复制到该文件中。若音频存储在外部文件中，且不能使用环境变量中的值解析其相对文件名，或不能在客户机上存取该文件，则 Extender 也可能会创建一个临时文件。该临时文件是在 DB2AUDIOTEMP 环境变量中指定的目录中创建的。然后， Extender 播放临时文件中的音频。

授权

若播放在列中的音频剪辑，则选择用户表上的权限。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

播放存储在列中的音频

```
long DBaPlay(
      char *playerName,
      MMDB_PLAY_HANDLE,
      DB2Audio *audioHandle,
      waitFlag
      );
```

语法

播放存储为文件的音频

DBaPlay

```
long DBaPlay(
      char *playerName,
     MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

参数

playerName（输入）
音频播放器的名称。若设置为 NULL，则使用 DB2AUDIOPLAYER 环境变量指定的缺省音频播放器。

MMDB_PLAY_HANDLE（输入）
指示将音频存储成 BLOB 的常量。

MMDB_PLAY_FILE（输入）
一个常量，它指示将音频存储成可从客户机存取的文件。

audioHandle（输入）
音频的句柄。播放在列中的音频剪辑时，必须传送此参数。如果音频句柄表示外部文件，则使用客户机环境变量 DB2VIDEOPATH 来解析文件位置。

fileName（输入）
包含音频的文件的名称。

waitFlag（输入）
一个常量，它指示程序在继续之前是否等待用户关闭播放器。MMDB_PLAY_WAIT 在应用程序所在的线程中运行播放器。MMDB_PLAY_NO_WAIT 在另一线程中运行播放器。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

播放 audioHandle 标识的音频。在应用程序所在的线程中运行缺省播放器:

```
#include <dmbaudio.h>

rc = DBaPlay(NULL, MMDB_PLAY_HANDLE,
       audioHandle, MMDB_PLAY_WAIT);
```

DBaPrepareAttrs

图像	音频	视频
	X	

准备用户提供的音频属性。当存储或更新具有用户提供的属性的音频对象时，使用此 API。在服务器上运行的 UDF 代码总是期望数据处于“大尾数法”格式，此格式是大多数 UNIX 平台使用的格式。若以“小尾数法”格式存储或更新音频对象，即，从非 UNIX 客户机存储或更新音频对象，则在进行存储或更新请求之前，必须使用 DBaPrepare API。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudio.lib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
void DBaPrepareAttrs(
      MMDBAudioAttrs *audAttr
      );
```

参数

audAttr（输入）
用户提供的音频属性。

例

准备用户提供的音频属性:

```
#include <dmbaudio.h>

DBaPrepareAttrs(&imgattr);
```

DBaReorgMetadata

图像	音频	视频
	X	

“清除”与音频相关的元数据表，例如:

- 收回音频元数据表中不再使用的空间
- 删除音频元数据表中对不再存在的音频文件的引用

在调用此 API 之前，应用程序必须与数据库相连。

授权

改变、控制、SYSADM、SYSCTRL、SYSMAINT 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbaudiolib	libdmbaudio.a（AIX） libdmbaudio.sl（HP-UX） libdmbaudio.so（Solaris）

包含文件

dmbaudio.h

语法

```
long DBaReorgMetadata(
      char *tableName
      );
```

参数

tableName（输入）

限定的、非限定的或空的表名。若指定表名，则对与指定的用户表相关联的音频元数据表执行清除。若指定了空值，则清除当前用户标识拥有的所有表中的音频列的元数据表。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED

应用程序没有与数据库的有效连接。

例

清除 Employee 表中音频列的元数据表:

```
#include <dmbaudio.h>

rc = DBaReorgMetadata("employee");
```

DBiAdminGetInaccessibleFiles

图像	音频	视频
X		

返回用户表的图像列中引用的不可存取的文件的名称。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

count (out)
输出列表中的条目数。

filelist（输出）
在表中引用的不可存取的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示用户标识 rjones 拥有的表的图像列中引用的所有不可存取的文件:

```
#include <dmbimage.h>
long idx;

rc = DBiAdminGetInaccessibleFiles
       ("rjones", &count, &filelist);
```

DBiAdminGetReferencedFiles

图像	音频	视频
X		

返回用户表的图像列中引用的文件的名称。若某个文件不可存取（例如，不能使用环境变量规范解析其文件名），则文件名前面有一个星号。此 API 不使用 FILEREF 数据结构的 FILENAME 字段，因此将其设置为 NULL。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
表中引用的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 ajones 拥有的表的图像列中引用的所有文件:

```
#include <dmbimage.h>
long idx;

rc = DBiAdminGetReferencedFiles("ajones",
       &count, &filelist);
```

DBiAdminIsFileReferenced

图像	音频	视频
X		

返回用户表中引用指定文件的图像列条目的列表。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

fileName（输入）
引用的文件的名称。

count（输出）
输出列表中的条目数。

tableList（输出）
引用指定文件的表条目的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示当前数据库中所有表的图像列中引用文件 /images/asmith.bmp 的条目:

```
#include <dmbimage.h>
long idx;

rc = DBiAdminIsFileReferenced(NULL,
       "/images/asmith.bmp",
       &count, &tableList);
```

DBiAdminReorgMetadata

图像	音频	视频
X		

“清除”与图像相关的元数据表:

- 收回图像元数据表中不再使用的空间
- 删除图像元数据表中对不再存在的图像文件的引用

在调用此 API 之前，应用程序必须与数据库相连。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiAdminReorgMetadata(
      char *qualifier
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定了用户标识，则清除带指定限定符的所有表。若指定了空值，则清除当前数据库中的所有表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

例

清除用户标识 rsmith 拥有的表中图像列的元数据表:

```
#include <dmbimage.h>

rc = DBiAdminReorgMetadata("rsmith");
```

DBiBrowse

图像	音频	视频
X		

在客户机上打开图像浏览器并显示图像。可将图像存储在图像列或外部文件中:

- 若将图像存储在外部文件中，则可将文件的名称或图像句柄传送至此 API。此 API 使用客户机环境变量 DB2IMAGEPATH 来解析文件位置。该文件必须可以从客户工作站进行存取。
- 若将图像存储在列中，则必须将图像句柄传送至此 API。应用程序必须与数据库相连，且必须对其中存储图像的表具有读存取权。

若浏览器不能直接存取图像，则 Extender 将在 DB2IMAGETEMP 环境变量中指定的目录中创建一个临时文件。然后，Extender 显示临时文件中的图像。

授权

若浏览列中的图像，则选择用户表上的权限。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

浏览存储在列中的图像

```
long DBiBrowse(
      char *browserName,
      MMDB_PLAY_HANDLE,
      DB2Image *imageHandle,
      waitFlag
      );
```

语法

浏览存储为文件的图像

```
long DBiBrowse(
      char *browserName,
     MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

参数

browserName（输入）
图像浏览器的名称。若设置为 NULL，则使用 DB2IMAGEBROWSER 环境变量指定的缺省图像浏览器。

MMDB_PLAY_HANDLE（输入）
一个常量，指示将图像存储成 BLOB。

MMDB_PLAY_FILE（输入）
一个常量，指示将图像存储成可从客户机存取的文件。

imageHandle（输入）
图像的句柄。在您浏览列中的图像时，必须传送此参数。若图像句柄代表外部文件，则使用客户机环境变量 DB2IMAGEPATH 来解析文件位置。

fileName（输入）
包含图像的文件的名称。

waitFlag（输入）
一个常量，指示程序在继续之前是否等待用户关闭浏览器。MMDB_PLAY_WAIT 在应用程序所在的线程中运行浏览器。MMDB_PLAY_NO_WAIT 在另一线程中运行浏览器。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

显示由 imageHandle 标识的图像。在应用程序所在的线程中运行缺省浏览器:

```
#include <dmbimage.h>

rc = DBiBrowse(NULL, MMDB_PLAY_HANDLE,
       imageHandle, MMDB_PLAY_WAIT);
```

DBiDisableColumn

图像	音频	视频
X		

对图像（DB2Image 数据）禁用列，使其不能存放图像数据。将列条目的内容设置为 NULL，并删除与此列相关联的元数据。还将删除与此列相关联的 QBIC 目录。还删除 Image Extender 为此列定义的所有触发器。可将新行插入到包含禁用列的表中，新行可包括用类型 DB2Audio 定义的数据，但（管理支持表中）没有与新行相关联的元数据。在调用此 API 之前，应用程序必须与数据库相连。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiDisableColumn(
      char *tableName,
      char *colName,
      );
```

参数

tableName（输入）
包含图像列的表的名称。

colName（输入）
图像列的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对图像（DB2Image 数据）禁用 Employee 表中的 picture 列:

```
#include <dmbimage.h>

rc = DBiDisableColumn("employee",
       "picture");
```

DBiDisableDatabase

图像	音频	视频
X		

对图像（DB2Image 数据）禁用数据库，使其不能存放图像数据。还禁用对 DB2Image 定义的数据库中的所有表。将删除 Image Extender 对数据库定义的元数据和 UDF。可将新行插入到该数据库中用类型 DB2Image 定义的表中，但（管理支持表中）没有与新行相关联的元数据。

授权

DBADM、SYSADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiDisableDatabase(
      );
```

参数

DBiDisableDatabase 没有参数。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对图像（DB2Image 数据）禁用当前数据库:

```
#include <dmbimage.h>

rc = DBiDisableDatabase();
```

DBiDisableTable

图像	音频	视频
X		

对图像（DB2Image 数据）禁用表，使其不能存放图像数据。还禁用对 DB2Image 定义的表中的所有列。将删除 Image Extender 对表定义的某些元数据。还删除与表中的图像列相关联的所有 QBIC 目录。可将新行插入到用类型 DB2Image 定义的表中，但（管理支持表中）没有与新行相关联的元数据。在调用此 API 之前，应用程序必须与数据库相连。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiDisableTable(
      char *tableName
      );
```

参数

tableName（输入）
包含图像列的表的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对图像（DB2Image 数据）禁用 Employee 表:

DBiDisableTable

```
#include <dmbimage.h>

rc = DBiDisableTable("Employee");
```

DBiEnableColumn

图像	音频	视频
X		

对图像（DB2Image 数据）启用列。此 API 定义并管理此列与元数据表之间的关系。在调用此 API 之前，应用程序必须与数据库相连，且必须落实用户表。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiEnableColumn(
      char *tableName,
      char *colName,
      );
```

参数

tableName（输入）
包含图像列的表的名称。

colName（输入）
图像列的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
列已启用。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_WRONG_SIGNATURE
所指定列的数据类型不正确。期望用户定义类型 MMDBSYS.DB2IMAGE。

MMDB_RC_COLUMN_DOESNOT_EXIST
该列未在指定表中定义。

MMDB_RC_NOT_ENABLED
未启用数据库或表。

例

对图像启用 Employee 表中的 picture 列:

```
#include <dmbimage.h>

rc = DBiEnableColumn("employee",
       "picture");
```

DBiEnableDatabase

图像	音频	视频
X		

对图像（DB2Image 数据）启用数据库。对于每个数据库，调用一次此 API。它向数据库管理器定义 DB2 用户定义类型 DB2Image。它还创建处理 DB2Image 数据的所有 UDF。

授权

DBADM、SYSADM、SYSCTRL

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiEnableDatabase(
      char *tableSpace
      );
```

参数

tableSpace（输入）

表空间的名称，表空间是在其中存储管理表的一组容器。表空间规范由如下三个部分组成：*datats*、*indexts 和 longts*，其中，*datats* 是在其中创建元数据表的表空间；*indexts* 是在其中创建元数据表上的索引的表空间；*longts* 是在其中存储元数据表中的长型列（诸如包含 LONG VARCHAR 和 LOB 数据类型的列）的值的表空间。若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

仅限于 EEE：在对 Extender 启用数据库时指定的表空间，应在包括分区数据库系统中所有节点的节点组上定义。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
数据库已启用。

MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER
连接的服务器不支持此命令。

MMDB_WARN_NOT_ALL_NODES
指定的表空间不包括 Extender 的所有节点。（**仅限于 EEE**）

MMDB_RC_NOT_SAME_NODEGROUP
指定的表空间不在同一节点组中。（**仅限于 EEE**）

例

对名为 MYTS 的表空间中的图像（DB2Image 数据）启用当前数据库。对索引和长表空间使用缺省值:

```
#include <dmbimage.h>

rc = DBiEnableDatabase("myts,,");
```

对图像（DB2Image 数据）启用当前数据库。使用缺省表空间:

```
#include <dmbimage.h>

rc = DBiEnableDatabase(NULL);
```

DBiEnableTable

图像	音频	视频
X		

对图像（DB2Image 数据）启用表。对于每个表，调用一次此 API。它创建元数据表来存储和管理一个表中图像列的属性。要避免出现锁定的可能性，应用程序应在调用此 API 之前落实事务。在调用此 API 之前，应用程序必须与数据库相连。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiEnableTable(
      char *tableSpace,
      char *tableName
      );
```

参数

tableSpace（输入）

表空间的名称，表空间是在其中存储管理表的一组容器。表空间规范由如下三个部分组成：*datats*、*indexts 和 longts*，其中，*datats* 是在其中创建元数据表的表空间；*indexts* 是在其中创建元数据表上的索引的表空间；*longts* 是在其中存储元数据表中的长型列（诸如包含 LONG VARCHAR 和 LOB 数据类型的列）的值的表空间。若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

仅限于 EEE：指定的表空间应与用户表在同一节点组中。

tableName（输入）

将包含图像列的表的名称。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
表已启用。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_TABLE_DOESNOT_EXIST
表不存在。

MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP
指定的表空间与用户表不在同一节点组中。**（仅限于 EEE）**

例

对表空间 MYTS 中的图像（DB2Image 数据）启用 employee 表。使用索引和长型表空间的缺省值:

```
#include <dmbimage.h>

rc = DBiEnableTable("myts,,",
       "Employee");
```

对图像（DB2Image 数据）启用 employee 表。使用缺省表空间:

```
#include <dmbimage.h>

rc = DBiEnableTable(NULL,
       "Employee");
```

DBiGetError

图像	音频	视频
X		

返回上一错误的描述。在任何其它 API 返回错误代码之后，调用此 API。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

参数

sqlcode（输出）
类属 SQL 错误代码。

errorMsgText（输出）
SQL 错误消息文本。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

例

获取上一错误，在 errCode 中存储 SQL 错误代码，并在 errMsg 中存储消息文本:

```
#include <dmbimage.h>

rc = DBiGetError(&errCode, &errMsg);
```

DBiGetInaccessibleFiles

图像	音频	视频
X		

返回用户表的图像列中引用的不可存取的文件的名称。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

对搜索的所有用户表和相关联的管理支持表中启用的图像列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对不可存取文件的引用。若指定了空值，则搜索带指定限定符的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
在表中引用的不可存取的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 employee 表中的图像列中引用的所有不可存取的文件:

```
#include <dmbimage.h>
long idx;

rc = DBiGetInaccessibleFiles("Employee",
       &count, &filelist);
```

DBiGetReferencedFiles

图像	音频	视频
X		

返回用户表的图像列中引用的文件的名称。若某个文件不可存取（例如，不能使用环境变量规范解析其文件名），则文件名前面有一个星号。此 API 不使用 FILEREF 数据结构的 FILENAME 字段，因此将其设置为 NULL。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构。

授权

对搜索的所有用户表和相关联的管理支持表中启用的图像列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对文件的引用。若指定了空值，则搜索当前用户标识所拥有的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
表中引用的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 employee 表中的图像列中引用的所有文件:

```
#include <dmbimage.h>
long idx;

rc = DBiGetReferencedFiles("employee",
       &count, &filelist);
```

DBIsColumnEnabled

图像	音频	视频
X		

确定是否已对图像（DB2Image 数据）启用列。在调用此 API 之前，应用程序必须与数据库相连。

授权

SYSADM、DBADM、表所有者或对用户表的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiIsColumnEnabled(
      char *tableName,
      char *colName,
short  *status
      );
```

参数

tableName（输入）
限定的或非限定的表名。

colName（输入）
列的名称。

status（输出）
指示列是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
列已启用。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

确定是否已对图像启用 Employee 表中的 picture 列:

```
#include <dmbimage.h>

rc = DBiIsColumnEnabled("employee",
       "picture", &status);
```

DBiIsDatabaseEnabled

图像	音频	视频
X		

确定是否已对图像（DB2Image 数据）启用数据库。在调用此 API 之前，应用程序必须与数据库相连。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiIsDatabaseEnabled(
short  *status
      );
```

参数

status（输出）

指示数据库是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED

应用程序没有与数据库的有效连接。

例

确定是否已对图像启用 personnl 数据库:

```
#include <dmbimage.h>

rc = DBiIsDatabaseEnabled(&status);
```

DBiIsFileReferenced

图像	音频	视频
X		

返回图像列中引用指定文件的表条目的列表。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

对搜索的所有用户表和相关联的管理支持表中启用的图像列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对指定文件的引用。若指定了空值，则搜索当前用户标识所拥有的所有表。

fileName（输入）
引用的文件的名称。

count（输出）
输出列表中的条目数

tableList（输出）
引用指定文件的表条目的列表

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 employee 表的图像列中引用文件 /images/ajones.bmp 的条目:

```
#include <dmbimage.h>
long idx;

rc = DBiIsFileReferenced(NULL,
       "/images/ajones.bmp",
       &count, &tableList);
```

DBIsTableEnabled

图像	音频	视频
X		

确定是否已对图像（DB2Image 数据）启用表。在调用此 API 之前，应用程序必须与数据库相连。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiIsTableEnabled(
      char *tableName,
short  *status
      );
```

参数

tableName（输入）
表名。

status（输出）
指示表是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

确定是否已对图像启用 Employee 表:

```
#include <dmbimage.h>

rc = DBiIsTableEnabled("Employee",
       &status);
```

DBiPrepareAttrs

图像	音频	视频
X		

准备用户提供的图像属性。当存储或更新具有用户提供的属性的图像对象时，使用此 API。在服务器上运行的 UDF 代码总是期望数据处于“大尾数法”格式，此格式是大多数 UNIX 平台使用的格式。若以“小尾数法”格式存储或更新图像对象，即，从非 UNIX 客户机存储或更新图像对象，则在进行存储或更新请求之前，必须使用 DBiPrepare API。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
void DBiPrepareAttrs(
      MMDBImageAttrs *imgAttr
      );
```

参数

imgAttr（输入）
用户提供的图像属性。

例

准备用户提供的图像属性:

```
#include <dmbimage.h>

DBiPrepareAttrs(&imgattr);
```

DBiReorgMetadata

图像	音频	视频
X		

“清除”与图像相关的元数据表，例如:

- 收回图像元数据表中不再使用的空间
- 删除图像元数据表中对不再存在的图像文件的引用

在调用此 API 之前，应用程序必须与数据库相连。

授权

改变、控制、SYSADM、SYSCTRL、SYSMAINT 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbimage.lib	libdmbimage.a（AIX） libdmbimage.sl（HP-UX） libdmbimage.so（Solaris）

包含文件

dmbimage.h

语法

```
long DBiReorgMetadata(
      char *tableName
      );
```

参数

tableName（输入）

限定的、非限定的或空的表名。若指定表名，则对与指定的用户表相关联的图像元数据表执行清除。若指定了空值，则清除当前用户标识拥有的所有表中的图像列的元数据表。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED

应用程序没有与数据库的有效连接。

例

清除 Employee 表中图像列的元数据表:

```
#include <dmbimage.h>

rc = DBiReorgMetadata("employee");
```

DBvAdminGetInaccessibleFiles

图像	音频	视频
		X

返回用户表的视频列中引用的不可存取的文件的名称。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。（输入）若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
在表中引用的不可存取的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示用户标识 rsmith 拥有的表的视频列中引用的所有不可存取的文件:

```
#include <dmbvideo.h>
long idx;

rc = DBvAdminGetInaccessibleFiles
       ("rsmith", &count,
       &filelist);
```

DBvAdminGetReferencedFiles

图像	音频	视频
		X

返回用户表的视频列中引用的文件的名称。若某个文件不可存取（例如，不能使用环境变量规范解析其文件名），则文件名前面有一个星号。此 API 不使用 FILEREF 数据结构的 FILENAME 字段，因此将其设置为 NULL。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
表中引用的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 ajones 拥有的表的视频列中引用的所有文件:

```
#include <dmbvideo.h>
long idx;

rc = DBvAdminGetReferencedFiles
       ("ajones", &count,
       &filelist);
```

DBvAdminIsFileReferenced

图像	音频	视频
		X

返回用户表中引用指定文件的视频列条目的列表。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定用户标识，则搜索带指定限定符的所有表。若指定空值，则搜索当前数据库中的所有表。

fileName（输入）
引用的文件的名称。

count（输出）
输出列表中的条目数。

tableList（输出）
引用指定文件的表条目的列表。

DBvAdminIsFileReferenced

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示当前数据库中所有表的视频列中引用文件 /videos/asmith.mpg 的条目:

```
#include <dmbvideo.h>
long idx;

rc = DBvAdminIsFileReferenced(NULL,
       "/videos/asmith.mpg",
       &count, &tableList);
```

DBvAdminReorgMetadata

图像	音频	视频
		X

“清除”与视频相关的元数据表，例如:

- 收回视频元数据表中不再使用的空间
- 删除视频元数据表中对不再存在的视频文件的引用

在调用此 API 之前，应用程序必须与数据库相连。

授权

SYSADM、SYSCTRL 或 SYSMAINT

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvAdminReorgMetadata(
      char *qualifier
      );
```

参数

qualifier（输入）
有效用户标识或空值。若指定了用户标识，则清除带指定限定符的所有表。若指定了空值，则清除当前数据库中的所有表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_NO_AUTH
用户没有调用此 API 的正确权限。

例

清除用户标识 rsmith 拥有的表中视频列的元数据表:

```
#include <dmbvideo.h>

rc = DBvAdminReorgMetadata("rsmith");
```

DBvBuildStoryboardFile

图像	音频	视频
		X

创建一个镜头目录文件，并在该文件中构建关于视频中所有镜头的条目。源视频可以在数据库或文件中。对于每一镜头，此 API 存储镜头号、起始帧号、结束帧号和至少一个代表性帧的信息。DBvStoryboardCtrl 数据结构中的值确定对一个镜头标识多少代表性帧。对于长度在 DBvStoryboardCtrl 中的阈值之下的镜头，此 API 标识一个代表性的帧。对于长度在 DBvStoryboardCtrl 中的低阈值和高阈值之间的镜头，此 API 标识两个代表性的帧。对于长度在 DBvStoryboardCtrl 中的阈值之上的镜头，此 API 标识三个代表性的帧。代表性的帧信息包括其帧号和包含帧内容的文件的名称。此信息可用来显示故事板，即，视频的可视摘要。

授权

插入，控制

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvBuildStoryboardFile(
      char *fileName,
      DBvIOType *video,
      DBvShotControl *shotCtrl,
      DBvStoryBoardCtrl *sbCtrl
            );
```

参数

catalogName（输入）
指向镜头目录文件的名称的指针。

video（输入）
指向视频结构的指针。

shotCtrl（输入）
指向镜头控制结构的指针

sbCtrl（输入）
指向故事板控制结构的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

例

创建一个名为 hotshots 的镜头目录文件，并用视频中的所有镜头数据填充它:

```
#include <dmbshot.h>


rc = DBvBuildStoryboardFile("hotshots",
       video, &shotCtrl, &sbCtrl);
```

DBvBuildStoryboardTable

图像	音频	视频
		X

在镜头目录中构建视频中所有镜头的条目。源视频可以在数据库或文件中。镜头目录位于数据库中。对于每一镜头，此 API 存储源视频的句柄或文件信息。它还存储镜头号、起始帧号、结束帧号和最少一个代表性帧的信息。DBvStoryboardCtrl 数据结构中的值确定对一个镜头标识多少代表性帧。对于长度在 DBvStoryboardCtrl 中的阈值之下的镜头，此 API 标识一个代表性帧。对于长度在 DBvStoryboardCtrl 中的低阈值和高阈值之间的镜头，此 API 标识两个代表性帧。对于长度在 DBvStoryboardCtrl 中的阈值之上的镜头，此 API 标识三个代表性帧。代表性帧信息包括其帧号和帧数据。存储在镜头目录中的代表性帧信息可用来显示故事板，即，视频的可视摘要。

在调用此 API 之前，应用程序必须与数据库相连。

授权

插入，控制

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvBuildStoryboardTable(
      char *catalogName,
      DBvIOType *video,
      DBvShotControl *shotCtrl,
      DBvStoryBoardCtrl *sbCtrl,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
指向镜头目录的名称的指针。

video（输入）
指向视频结构的指针。

shotCtrl（输入）
指向镜头控制结构的指针

sbCtrl（输入）
指向故事板控制结构的指针。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

在名为 hotshots 的镜头目录中创建视频的条目:

```
#include <dmbshot.h>


rc = DBvBuildStoryboardTable("hotshots",
       video, &shotCtrl, &sbCtrl, hdbc);
```

DBvClose

图像	音频	视频
		X

关闭为检测画面切换而打开的视频文件。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbmpeg.lib	libdmbmpeg.a（AIX） libdmbmpeg.sl（HP-UX） libdmbmpeg.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvClose(
      DB2vIOType *video
      );
```

参数

video（输入）
指向视频结构的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_CANNOT_CLOSE
未能关闭视频文件

例

关闭先前为检测视频画面切换而打开的视频文件:

```
#include <dmbshot.h>

rc = DBvClose(video);
```

DBvCreateIndex

图像	音频	视频
		X

创建存储在文件中的视频的索引。Video Extender 使用该索引来存取视频中的镜头和帧。索引存储在源视频文件所在目录中的平面文件中。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbmpeg.lib	libdmbmpeg.a（AIX） libdmbmpeg.sl（HP-UX） libdmbmpeg.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvCreateIndex(
      char *fileName
      );
```

参数

fileName（输入）
指向视频文件名的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_OPEN_VIDEO
未能打开视频文件以进行处理。

MMDB_RC_INDEX_FAIL
未能构建索引。

例

为文件 \Videos\ajones.mpg 中的视频创建索引:

```
#include <dmbshot.h>

rc = DBvCreateIndex("\videos\ajones.mpg");
```

DBvCreateIndexFromVideo

图像	音频	视频
		X

为视频创建索引。首先，必须为检测镜头而打开该视频。Video Extender 使用该索引来存取视频中的镜头和帧。索引存储在平面文件中。文件名存储在 DBvIOType 数据结构中。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvCreateIndexFromVideo(
      DBvIOType *video
      );
```

参数

video（更新）
指向视频结构的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_OPEN_VIDEO
未能打开视频文件以进行处理。

MMDB_RC_INDEX_FAIL
未能构建索引。

例

为视频创建索引:

```
#include <dmbshot.h>

rc = DBvCreateIndexFromVideo(video);
```

DBvCreateShotCatalog

图像	音频	视频
		X

创建镜头目录，它是一组表，包含关于镜头的信息，如帧号。

应用程序必须与对 db2video 和 db2image 都启用的数据库相连。

授权

创建、SYSADM、DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvCreateShotCatalog(
      char *catalogName,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
要创建的镜头目录的名称。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

创建名为 hotshots 的镜头目录:

```
#include <dmbshot.h>

rc = DBvCreateShotCatalog("hotshots", hdbc);
```

DBvDeleteShot

图像	音频	视频
		X

从目录中删除镜头。

授权

插入，控制

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvDeleteShot(
      char *catalogName,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
目录名。

shotHandle（输入）
镜头句柄。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_ACCESS
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

例

通过使用镜头的句柄从 hotshots 目录中删除镜头:

```
#include <dmbshot.h>

rc = DBvDeleteShot("hotshots", shot,
       hdbc);
```

DBvDeleteShotCatalog

图像	音频	视频
		X

删除镜头目录。

授权

控制、SYSADM、DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvDeleteShotCatalog(
      char *catalogName,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
要删除的镜头目录的名称。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_ACCESS
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

删除镜头目录 hotshots:

例

```
#include <dmbshot.h>


rc = DBvDeleteShotCatalog("hotshots",
       hdbc);
```

DBvDetectShot

图像	音频	视频
		X

搜索视频文件中的下一镜头。若检测到镜头，则记录在检测到的镜头中的第一个帧的帧号和帧数据。必须检查 shotDetected 标志才能确定是否已检测到镜头。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvDetectShot(
      DBvIOType *video,
      unsigned long *start_frame,
      char *shotDetected,
      DBvShotControl *shotCtrl,
      DBvShotType *shot,
            );
```

参数

video（更新）
指向视频结构的指针。

start_frame（输入／输出）
用作搜索起始点的帧号。返回时，此参数被更新为开始寻找下一镜头的位置。

shotDetected（输出）
镜头检测标志: 1= 检测到帧，0= 未检测到帧。

shotCtrl（输入）
指向镜头控制数据的指针。

shot（输出）
指向检测到的镜头和镜头数据的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_EOF
到达文件尾。

MMDB_NO_INDEX
视频索引不存在。

例

在视频文件中从帧 1 开始搜索下一镜头:

```
#include <dmbshot.h>


long start_frame=1;

rc = DBvDetectShot(video, start_frame&Detected,
       &shotCtrl, &shot);
```

DBvDisableColumn

图像	音频	视频
		X

对视频（DB2Video 数据）禁用列，使其不能存放视频数据。将列条目的内容设置为 NULL，并删除与此列相关联的元数据。还删除 Video Extender 对此列定义的所有触发器。可将新行插入到包含禁用列的表中，新行可包括用类型 DB2Video 定义的数据，但（管理支持表中）没有与新行相关联的元数据。在调用此 API 之前，应用程序必须与数据库相连。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvDisableColumn(
      char *tableName,
      char *colName,
      );
```

参数

tableName（输入）
包含视频列的表的名称。

colName（输入）
视频列的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对视频（DB2Video 数据）禁用 Employee 表中的 tv_ads 列:

```
#include <dmbvideo.h>

rc = DBvDisableColumn("employee",
       "tv_ads");
```

DBvDisableDatabase

图像	音频	视频
		X

对视频（DB2Video 数据）禁用数据库，使其不能存放视频数据。还禁用该数据库中对 DB2Video 定义的所有表。将删除 Video Extender 对该数据库定义的元数据和 UDF。可将新行插入到该数据库中用类型 DB2Video 定义的表中，但（管理支持表中）没有与新行相关联的元数据。

授权

DBADM、SYSADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvDisableDatabase(
      );
```

参数

DBvDisableDatabase 没有参数。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对视频（DB2Video 数据）禁用当前数据库:

```
#include <dmbvideo.h>

rc = DBvDisableDatabase();
```

DBvDisableTable

图像	音频	视频
		X

对视频（DB2Video 数据）禁用表，使其不能存放视频数据。还禁用该表中对 DB2Video 定义的所有列。将删除 Video Extender 对表定义的某些元数据。可将新行插入到用类型 DB2Video 定义的表中，但（管理支持表中）没有与新行相关联的元数据。在调用此 API 之前，应用程序必须与数据库相连。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvDisableTable(
      char *tableName
      );
```

参数

tableName（输入）
包含视频列的表的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

对视频（DB2Video 数据）禁用 Employee 表:

```
#include <dmbvideo.h>

rc = DBvDisableTable("employee");
```

DBvEnableColumn

图像	音频	视频
		X

对视频（DB2Video 数据）启用列。此 API 定义并管理此列与元数据表之间的关系。在调用此 API 之前，应用程序必须与数据库相连，且必须落实用户表。

授权

控制、改变、SYSADM 或 DBADM

还需要对 API 参数中指定的表空间和缓冲池的使用特权。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvEnableColumn(
      char *tableName,
      char *colName,
      );
```

参数

tableName（输入）
包含视频列的表的名称。

colName（输入）
视频列的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
列已启用。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_WRONG_SIGNATURE
所指定列的数据类型不正确。期望用户定义的数据类型 MMDBSYS.DB2VIDEO。

MMDB_RC_COLUMN_DOESNOT_EXIST
该列未在指定表中定义。

MMDB_RC_NOT_ENABLED
未启用数据库或表。

例

对视频启用 Employee 表中的 video 列:

```
#include <dmbvideo.h>

rc = DBvEnableColumn("Employee",
       "video");
```

DBvEnableDatabase

图像	音频	视频
		X

对视频（DB2Video 数据）启用数据库。对于每个数据库，调用一次此 API。它向数据库管理器定义 DB2 用户定义的类型 DB2Video。它还创建处理 DB2Video 数据的所有 UDF。

授权

DBADM、SYSADM、SYSCTRL

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvEnableDatabase(
      char *tableSpace
      );
```

参数

tableSpace（输入）

表空间的名称，表空间是在其中存储管理表的一组容器。表空间规范由如下三个部分组成：*datats*、*indexts 和 longts*，其中，*datats* 是在其中创建元数据表的表空间；*indexts* 是在其中创建元数据表上的索引的表空间；*longts* 是在其中存储元数据表中的长型列（诸如包含 LONG VARCHAR 和 LOB 数据类型的列）的值的表空间。若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

仅限于 EEE：在对 Extender 启用数据库时指定的表空间，应在包括分区数据库系统中所有节点的节点组上定义。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
数据库已启用。

MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER
连接的服务器不支持此命令。

MMDB_WARN_NOT_ALL_NODES
指定的表空间不包括 Extender 的所有节点。（仅限于 EEE）

MMDB_RC_NOT_SAME_NODEGROUP
指定的表空间不在同一节点组中。（仅限于 EEE）

例

在名为 MYTS 的表空间中对视频（DB2Video 数据）启用当前数据库。对索引和长表空间使用缺省值:

```
#include <dmbvideo.h>

rc = DBvEnableDatabase("myts,,");
```

对视频（DB2Video 数据）启用当前数据库。使用缺省表空间:

```
#include <dmbvideo.h>

rc = DBvEnableDatabase(NULL);
```

DBvEnableTable

图像	音频	视频
		X

对视频（DB2Video 数据）启用表。对于每个表，调用一次此 API。它创建元数据表来存储和管理一个表中视频列的属性。要避免出现锁定的可能性，应用程序应在调用此 API 之前落实事务。在调用此 API 之前，应用程序必须与数据库相连。

授权

控制、改变、SYSADM 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvEnableTable(
      char *tableSpace,
      char *tableName
      );
```

参数

tableSpace（输入）

表空间的名称，表空间是在其中存储管理表的一组容器。表空间规范由如下三个部分组成：*datats*、*indexts 和 longts*，其中，*datats* 是在其中创建元数据表的表空间；*indexts* 是在其中创建元数据表上的索引的表空间；*longts* 是在其中存储元数据表中的长型列（诸如包含 LONG VARCHAR 和 LOB 数据类型的列）的值的表空间。若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

仅限于 EEE：指定的表空间应与用户表在同一节点组中。

tableName（输入）

将包含视频列的表的名称。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_WARN_ALREADY_ENABLED
表已启用。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_TABLE_DOESNOT_EXIST
表不存在。

MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP
指定的表空间与用户表不在同一节点组中。（**仅限于 EEE**）

例

对表空间 MYTS 中的视频（DB2Video 数据）启用 employee 表。使用索引和长型表空间的缺省值:

```
#include <dmbvideo.h>

rc = DBvEnableTable("myts,,",
       "Employee");
```

对视频（DB2Video 数据）启用 employee 表。使用缺省表空间:

```
#include <dmbvideo.h>

rc = DBvEnableTable(NULL,
       "Employee");
```

DBvFrameDataTo24BitRGB

图像	音频	视频
		X

将视频帧从 YUV 颜色值格式（如 MPEG）转换为 24 位 RGB 格式。在发出此 API 调用之前，用户必须分配目标缓冲区。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbmpeg.lib	libdmbmpeg.a（AIX） libdmbmpeg.sl（HP-UX） libdmbmpeg.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvFrameDataTo24BitRGB(
      unsigned char *RGB,
      DBvFrameData *fd,
      unsigned long dx,
      unsigned long dy
      );
```

参数

RGB（输出）
指向目标 RGB 缓冲区的指针。

fd（输入）
指向要转换的帧数据的指针。

dx（输入）
帧宽度

dy（输入）
帧高度

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

例

将视频帧由 MPEG 转换为 24 位 RGB:

```
#include <dmbshot.h>


rc = DBvFrameDataTo24BitRGB(RGB, &video->fd,
       video->dx, video->dy);
```

DBvGetError

图像	音频	视频
		X

返回上一错误的描述。在任何其它 API 返回错误代码之后，调用此 API。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

参数

sqlcode（输出）
类属 SQL 错误代码。

errorMsgText（输出）
SQL 错误消息文本。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

例

获取上一错误，在 errCode 中存储 SQL 错误代码，并在 errMsg 中存储消息文本:

```
#include <dmbvideo.h>

rc = DBvGetError(&errCode, &errMsg);
```

DBvGetFrame

图像	音频	视频
		X

获取视频文件中的当前帧。将在 DBvFrameData 视频结构中返回帧数据。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbmpeg.lib	libdmbmpeg.a（AIX） libdmbmpeg.sl（HP-UX） libdmbmpeg.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvGetFrame(
      DBvIOType *video
      );
```

参数

video（更新）
指向视频结构的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_EOF
到达文件尾。

例

获取视频文件中的当前帧:

```
#include <dmbshot.h>


rc = DBvGetFrame(video);
```

DBvGetInaccessibleFiles

图像	音频	视频
		X

返回用户表的视频列中引用的不可存取文件的名称。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

对搜索的所有用户表和相关联的管理支持表中启用的视频列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对不可存取文件的引用。若指定了空值，则搜索带指定限定符的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
在表中引用的不可存取文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 Employee 表的视频列中引用的所有不可存取的文件:

```
#include <dmbvideo.h>
long idx;

rc = DBvGetInaccessibleFiles("Employee",
       &count, &filelist);
```

DBvGetReferencedFiles

图像	音频	视频
		X

返回用户表的视频列中引用的文件的名称。若某个文件不可存取（例如，不能使用环境变量规范解析其文件名），则文件名前面有一个星号。此 API 不使用 FILEREF 数据结构的 FILENAME 字段，因此将其设置为 NULL。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构。

授权

对搜索的所有用户表和相关联的管理支持表中启用的视频列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对文件的引用。若指定了空值，则搜索当前用户标识所拥有的所有表。

count（输出）
输出列表中的条目数。

fileList（输出）
表中引用的文件的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 Employee 表的视频列中引用的所有文件:

```
#include <dmbvideo.h>
long idx;

rc = DBvGetReferencedFiles("employee",
       &count, &filelist);
```

DBvInitShotControl

图像	音频	视频
		X

初始化镜头控制数据结构中的值。

授权

插入，控制

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvInitShotControl(
      DBvShotControl *shotCtrl,
      );
```

参数

shotCtrl（输入）
指向镜头控制数据结构的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_ACCESS
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

初始化镜头控制数据结构中的值:

```
#include <dmbshot.h>


rc = DBvInitShotControl(shotCtrl);
```

DBvInitStoryboardCtrl

图像	音频	视频
		X

初始化故事板控制数据结构中的值。

授权

插入，控制

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvInitStoryboardCtrl(
      DBvStoryboardCtrl *sbCtrl,
      );
```

参数

shotCtrl（输入）
指向镜头控制数据结构的指针。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_ACCESS
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

初始化故事板控制数据结构中的值:

```
#include <dmbshot.h>


rc = DBvInitStoryboardCtrl(shotCtrl);
```

DBvInsertShot

图像	音频	视频
		X

将镜头插入镜头目录中。

授权

插入，控制

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvInsertShot(
      char *catalogName,
      DBvShotType *shot,
      DBvIOType *video,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
目录名。

shot（输入）
指向要插入到目录中的扩展镜头的指针。

shotHandle（输入）
镜头句柄。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_ACCESS
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

例

在称为 hotshots 的镜头目录中插入镜头:

```
rc = DBvInsertShot(
"hotshots",      /* shot catalog name */
shot,                          /* pointer to shot structure */
video,                         /* pointer to video structure */
shotHandle,                    /* pointer to shot handle */
hdbc);                         /* database connection handle */
```

DBvIsColumnEnabled

图像	音频	视频
		X

确定是否已对视频（DB2Video 数据）启用列。在调用此 API 之前，应用程序必须与数据库相连。

授权

SYSADM、DBADM、表所有者或对用户表的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvIsColumnEnabled(
      char *tableName,
      char *colName,
short  *status
      );
```

参数

tableName（输入）
限定的或非限定的表名。

colName（输入）
列的名称。

status（输出）
指示列是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

确定是否已对视频启用 Employee 表中的 video 列:

```
#include <dmbvideo.h>

rc = DBvIsColumnEnabled("employee",
       "video", &status);
```

DBvIsDatabaseEnabled

图像	音频	视频
		X

确定是否已对视频（DB2Video 数据）启用数据库。在调用此 API 之前，应用程序必须与数据库相连。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvIsDatabaseEnabled(
short  *status
      );
```

参数

status（输出）

指示数据库是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED

应用程序没有与数据库的有效连接。

例

确定是否已对视频启用 personnl 数据库:

```
#include <dmbvideo.h>

rc = DBvIsDatabaseEnabled(&status);
```

DBvIsFileReferenced

图像	音频	视频
		X

返回视频列中引用指定文件的表条目的列表。在调用此 API 之前，应用程序必须与数据库相连。

重要的是在调用此 API 之后释放它所分配的资源。特别是，必须释放 filelist 数据结构以及 filelist 中各条目的 filename 字段。

授权

对搜索的所有用户表和相关联的管理支持表中启用的视频列的 SELECT 特权

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

参数

tableName（输入）
限定的、非限定的或空的表名。若指定了表名，则搜索该表中对指定文件的引用。若指定了空值，则搜索当前用户标识所拥有的所有表。

fileName（输入）
被引用文件的名称。

count（输出）
输出列表中的条目数。

tableList（输出）
引用指定文件的表条目的列表。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_MALLOC
系统不能分配返回结果所需的内存。

例

列示 employee 表的视频列中引用文件 /videos/ajones.mpg 的条目:

```
#include <dmbvideo.h>
long idx;

rc = DBvIsFileReferenced(NULL,
       "/videos/ajones.mpg",
       &count, &tableList);
```

DBvIsIndex

图像	音频	视频
		X

检查视频索引是否存在。在调用此 API 之前，应用程序必须与数据库相连。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbmpeg.lib	libdmbmpeg.a（AIX） libdmbmpeg.sl（HP-UX） libdmbmpeg.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvIsIndex(
      char *fileName,
      short *status
      );
```

参数

fileName（输入）
被引用文件的名称。

status（输出）
指示索引是否已存在。值 1 表示索引已存在；值 0 表示索引不存在。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_ERROR
状态无效。

例

检查视频文件 \Videos\ajones.mpg 的索引是否存在:

```
#include <dmbshot.h>

rc = DBvIsIndex("\videos\ajones.mpg", &status);
```

DBvIsTableEnabled

图像	音频	视频
		X

确定是否已对视频（DB2Video 数据）启用表。在调用此 API 之前，应用程序必须与数据库相连。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvIsTableEnabled(
      char *tableName,
short  *status
      );
```

参数

tableName（输入）
表名。

status（输出）
指示表是否已启用。此参数返回一个数值。Extender 还返回一个指示状态的常量。值和常量是:

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

DBvIsTableEnabled

例

确定是否已对视频启用 Employee 表:

```
#include <dmbvideo.h>

rc = DBvIsTableEnabled("Employee",
       &status);
```

DBvMergeShots

图像	音频	视频
		X

将两个镜头合并成一个镜头。合成后的镜头使用第一个镜头的镜头句柄和起始帧。合成后的镜头使用两个镜头的结束帧中较大的那一个。将删除第二个镜头句柄所指向的行。

授权

控制、选择、删除或更新

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvMergeShots(
      char *catalogName,
      char *shotHandle1,
      char *shotHandle2,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
镜头目录的名称。

shotHandle1（输入）
第一个镜头的句柄。

shotHandle2（输入）
第二个镜头的句柄。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_CANNOT_MERGE
不能合并镜头。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

例

将带有句柄 shotHandle1 和 shotHandle2 的镜头合并到 hotshots 目录中:

```
#include <dmbshot.h>


rc = DBvMergeShots("hotshots", shotHandle1,
       shotHandle2, hdbc);
```

DBvOpenFile

图像	音频	视频
		X

为 DBvIOType 结构分配空间，并打开视频文件以进行像素存取。如果成功打开视频，它将指向第一个帧号（帧 0）。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbmpeg.lib	libdmbmpeg.a（AIX） libdmbmpeg.sl（HP-UX） libdmbmpeg.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvOpenFile(
      DBvIOType **video,
      char *fileName,
      );
```

参数

video（输出）
指向视频结构指针的指针。

fileName（输入）
要打开的视频文件的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_CANNOT_OPEN
打不开视频文件。

MMDB_RC_NO_MEMORY
没有足够内存。

MMDB_RC_NO_INDEX
没有视频随机存取索引。

例

打开视频文件 \Videos\ajones.mpg:

```
#include <dmbshot.h>


rc = DBvOpenFile(&videoa,
       "\Videos\ajones.mpg");
```

DBvOpenHandle

图像	音频	视频
		X

为 DBvIOType 结构分配空间，并打开视频句柄以进行像素存取。该结构指向第一个帧号（帧 0）。视频可以是 BLOB。将把视频复制至临时文件时，文件位于 DB2VIDEOTEMP 环境变量指定的目录中。根据随机存取索引是否存在来设置 isIdx 标志。

授权

选择

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvOpenHandle(
      DBvIOType **video,
      DB2Video *videoHandle
      SQLHDBC hdbc
      );
```

参数

video（输出）
指向视频结构的指针。

videoHandle（输入）
视频句柄。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_CANNOT_OPEN
打不开视频文件。

MMDB_RC_NO_MEMORY
没有足够内存。

MMDB_RC_NO_INDEX
没有视频随机存取索引。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_INVALID_HANDLE
视频句柄无效。

例

使用 videoa 指针打开 videoHandle:

```
#include <dmbshot.h>

rc = DBvOpenHandle(&oa, videoHandle, hdbc);
```

DBvPlay

图像	音频	视频
		X

在客户机上打开视频播放器，并播放视频。可将视频存储在视频列或外部文件中:

- 若将视频存储在外部文件中，则可将文件名或视频句柄传送至此 API。此 API 使用客户机环境变量 DB2VIDEOPATH 来解析文件位置。该文件必须可以从客户工作站进行存取。
- 若将视频存储在列中，则必须将视频句柄传送至此 API。应用程序必须与数据库相连，且必须对其中存储视频的表具有读存取权。

若视频存储在列中，则 Extender 创建一个临时文件，并将对象的内容从该列复制到该文件中。若视频存储在外部文件中，且不能使用环境变量中的值解析其相对文件名，或不能在客户机上存取该文件，则 Extender 也可能会创建一个临时文件。该临时文件是在 DB2VIDEOTEMP 环境变量中指定的目录中创建的。然后，Extender 播放临时文件中的视频。

授权

若播放在列中的视频，则选择用户表上的权限。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

播放存储在列中的视频

```
long DBvPlay(
      char *playerName,
      MMDB_PLAY_HANDLE,
      DB2Video *videoHandle,
      waitFlag
      );
```

语法

播放存储成文件的视频

```
long DBvPlay(
      char *playerName,
     MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

参数

playerName（输入）
视频播放器的名称。若设置为 NULL，则使用 DB2VIDEOPLAYER 环境变量所指定的缺省视频播放器。

MMDB_PLAY_HANDLE（输入）
指示将视频存储在列中的常量。

MMDB_PLAY_FILE（输入）
一个常量，指示将视频存储成可从客户机存取的文件。

videoHandle（输入）
视频的句柄。当播放列中的视频时，必须传送此参数。若视频句柄表示外部文件，则使用客户机环境变量 DB2VIDEOPATH 来解析文件位置。

fileName（输入）
包含视频的文件的名称。此 API 使用客户机环境变量 DB2VIDEOPATH 来解析文件位置。该文件必须可以从客户工作站进行存取。

waitFlag（输入）
一个常量，它指示程序在继续之前是否等待用户关闭播放器。MMDB_PLAY_WAIT 在应用程序所在的线程中运行播放器。MMDB_PLAY_NO_WAIT 在另一线程中运行播放器。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH
调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

例

播放由 videoHandle 标识的视频。在应用程序所在的线程中运行缺省播放器:

```
#include <dmbvideo.h>

rc = DBvPlay(NULL, MMDB_PLAY_HANDLE, videoHandle,
       MMDB_PLAY_WAIT);
```

DBvPrepareAttrs

图像	音频	视频
		X

准备用户提供的视频属性。当存储或更新具有用户提供的属性的视频对象时，使用此 API。在服务器上运行的 UDF 代码总是期望数据处于“大尾数法”格式，此格式是大多数 UNIX 平台使用的格式。若以“小尾数法”格式存储或更新视频对象，即，从非 UNIX 客户机存储或更新视频对象，则在进行存储或更新请求之前，必须使用 DBvPrepare API。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
void DBvPrepareAttrs(
      MMDBVideoAttrs *vidAttr
      );
```

参数

vidAttr（输入）
用户提供的视频属性。

例

准备用户提供的视频属性:

```
#include <dmbvideo.h>

DBvPrepareAttrs(&vidattr);
```

DBvReorgMetadata

图像	音频	视频
		X

“清除”与视频相关的元数据表，例如:

- 收回视频元数据表中不再使用的空间
- 删除视频元数据表中对不再存在的视频文件的引用

在调用此 API 之前，应用程序必须与数据库相连。

授权

改变、控制、SYSADM、SYSCTRL、SYSMAINT 或 DBADM

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbvideo.lib	libdmbvideo.a（AIX） libdmbvideo.sl（HP-UX） libdmbvideo.so（Solaris）

包含文件

dmbvideo.h

语法

```
long DBvReorgMetadata(
      char *tableName,
      );
```

参数

tableName（输入）

限定的、非限定的或空的表名。若指定了表名，则对与指定的用户表相关联的视频元数据表执行清除。若指定了空值，则清除当前用户标识拥有的所有表中的视频列。

错误代码

MMDB_SUCCESS

对 API 调用的处理成功完成。

MMDB_RC_NO_AUTH

调用者没有正确的存取权限。

MMDB_RC_NOT_CONNECTED

应用程序没有与数据库的有效连接。

例

清除 Employee 表中视频列的元数据表:

```
#include <dmbvideo.h>

rc = DBvReorgMetadata("employee");
```

DBvSetFrameNumber

图像	音频	视频
		X

将当前帧设置为指定的帧号。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbmpeg.lib	libdmbmpeg.a（AIX） libdmbmpeg.sl（HP-UX） libdmbmpeg.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvSetFrameNumber(
      DBvIOType *video
      unsigned long frameNumber
      );
```

参数

video（输入）
指向视频结构的指针。

frameNumber（输入）
请求的帧的号码。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_FRAME_NOT_FOUND
找不到请求的帧。

MMDB_NO_INDEX
视频索引不存在。

例

将当前帧设置为视频文件中的帧号 85:

```
#include <dmbshot.h>

rc = DBvSetFrameNumber(video, 85);
```

DBvSetShotComment

图像	音频	视频
		X

更新镜头内的只读注释。

授权

控制或更新

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvSetShotComment(
      char *catalogName,
      char *shotHandle,
      char *comment,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
目录名。

shotHandle（输入）
要更新的镜头的句柄。

comment（输入）
对镜头的新注释。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_CANNOT_UPDATE
此 API 不能更新镜头。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

例

更改描述目录 hotshots 中带 shotHandle 的镜头的注释:

```
#include <dmbshot.h>


rc = DBvSetShotComment("hotshot", shotHandle,
       "This is a hot shot.", hdbc);
```

DBvUpdateShot

图像	音频	视频
		X

替换目录中视频镜头的属性。除注释之外的所有属性被替换为 DBvShotType 结构中的属性。若注释指针是 NULL，则现有注释保持不变。

授权

控制、更新

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbshot.lib	libdmbshot.a（AIX） libdmbshot.sl（HP-UX） libdmbshot.so（Solaris）

包含文件

dmbshot.h

语法

```
long DBvUpdateShot(
      char *catalogName,
      DBvShotType *shot,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

参数

catalogName（输入）
目录名。

shot（输入）
指向包含镜头属性的镜头信息结构的指针。

shotHandle（输入）
镜头句柄。

hdbc（输入）
SQLConnect 中的数据库句柄。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RC_NOT_CONNECTED
应用程序没有与数据库的有效连接。

MMDB_RC_CANNOT_UPDATE
此 API 不能更新镜头。

MMDB_RC_NO_SHOT
镜头不存在。

MMDB_RC_INVALID_CATALOG
目录无效或不存在。

例

更新 hotshots 目录中镜头的属性:

```
#include <dmbshot.h>


rc = DBvUpdateShot("hotshots", shot,
       shothandle, hdbc);
```

DMBRedistribute（仅限于 EEE）

图像	音频	视频
X		

当将节点添加至节点组或从节点组中除去节点时，或当建立节点组的新分区映象时，再分发 QBIC 特征数据。

授权

必须从拥有实例的标识运行此 API。

库文件

Windows	AIX 和 Solaris
dmbrd.lib	libdmbrd.a（AIX）
	libdmbrd.so（Solaris）

包含文件

dmbrdst.h

语法

```
long DMBRedistribute (
       char  *pNodeGroupName,
       char  DataRedistOption    /* ""continue"" use CONTINUE parameter */
       );
                                 /* blank:start redistribution */
```

参数

pNodeGroupName（输入）
要再分发的节点组的名称。

错误代码

MMDB_SUCCESS
对 API 调用的处理成功完成。

MMDB_RD_NO_CONTINUE
在无 CONTINUE 参数的情况下重新提交。

MMDB_RD_CONTINUE
在有 CONTINUE 参数的情况下重新提交。

例

再分发 groupone 节点组中的 QBIC Extender 数据:

```
#include <dmbrdst.h>

rc = DMBRedistribute(groupone,"continue");
```

QbAddFeature

图像	音频	视频
X		

将特征添加至当前打开的目录。QbAddFeature 为数据库中的指定特征创建特征表。在将图像添加至用户表中的图像列之后，使用 QbReCatalogColumn，它将关于各图像的条目添加至特征表，并分析图像。

授权

改变

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbAddFeature(
      QbCatalogHandle cHdl,
      char *featureName
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

featureName（输入）
特征的名称。为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECCatalogReadOnly
目录是为只读操作打开的。

qbicECDupFeature
该特征已在目录中。

qbiECinvalidFeatureClass
指定的特征不是有效的名称格式。

例

将 QbColorFeatureClass 特征添加至句柄 CatHdl 所标识的目录中:

```
#include <dmbqbapi.h>

rc = QbAddFeature(CatHdl,
       QbColorFeatureClass);
```

QbCatalogColumn

图像	音频	视频
X		

编目用户表的图像列中未编目的图像。此 API 将关于各图像的条目添加至特征表，然后分析这些图像。当此 API 分析图像时，它创建图像数据，并将其存储在特征表中该图像的条目中。使用特征的缺省参数。必须打开该目录。

授权

插入

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbCatalogColumn(
      QbCatalogHandle cHdl
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECInvalidCatalog
指定的句柄或表列对目录无效。

qbicECCatalog Errors
编目个别图像时出错，已记录这些错误。未发生回滚。

qbicECImageNotFound
找不到或不能存取图像。

qbicECCatalogRO
目录是只读的。

qbicECSQLError
发生了 SQL 错误。

QbCatalogColumn

例

```
#include <dmbqbapi.h>

rc = QbCatalogColumn(CatHdl);
```

QbCatalogImage

图像	音频	视频
X		

编目整个图像。此 API 将关于图像的条目添加至特征表，然后分析该图像。当此 API 分析图像时，它创建图像数据，并将其存储在特征表中该图像的条目中。图像句柄所在的图像列必须与当前 QBIC 目录相关联。根据当前定义的特征类来编目图像。使用目录中特征的缺省参数。

授权

插入

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbCatalogImage(
      QbCatalogHandle cHdl,
      char *imgHandle
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

imgHandle（输入）
图像的句柄。

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECImageNotFound
找不到或不能存取图像。

qbicECCatalogRO
目录是只读的。

例

编目句柄 Img_hdl 标识的图像:

```
#include <dmbqbapi.h>

rc = QbCatalogColumn(CatHdl, Img_hdl);
```

QbCloseCatalog

图像	音频	视频
X		

关闭目录。此 API 释放打开的目录句柄和分配的资源。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbCloseCatalog(
      QbCatalogHandle cHdl
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

错误代码

qbicECIvalidHandle
目录句柄无效。

例

关闭句柄 CatHdl 所标识的目录:

```
#include <dmbqbapi.h>

rc = QbCloseCatalog(CatHdl);
```

QbCreateCatalog

图像	音频	视频
X		

在当前连接的数据库中为指定的图像列创建目录。 必须对图像数据启用该列。此 API 创建目录的名称，它用作限定符。

授权

改变

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbCreateCatalog(
     char *tableName,
     char *columnName,
     SQLINTEGER autoCatalog,
     char *reserved
     );
```

参数

tableName（输入）
包含图像列的表的名称。

columnName（输入）
正在为其创建目录的图像列的名称。

autoCatalog（输入）
指示是否将自动编目添加至图像列的图像（即，将这些图像添加至特征表并进行分析）。指定 1 以打开自动编目，或指定 0 关闭它。若不打开自动编目，则使用 QbCatalogColumn 或 QbCatalogImage API 来编目添加至图像列的图像。

reserved（输入）
当前不使用。

错误代码

qbicECSqlError
发生了 SQL 错误。

qbicECNotEnabled
未对 DB2Image 数据类型启用数据库、表或列。

qbicECDupCatalog
目录已存在。

qbicECunsupportedOption
指定了不受支持的选项。

qbicECerrorParameterTooLong
参数太长，不能处理。

qbicECqerr
发生 QBIC 错误，生成了信息。

qbicECqerrUnknown
发生内部 QBIC 错误，生成了类属错误消息。

例

为 Employee 表的 picture 列中的图像创建一个目录。打开自动编目:

```
#include <dmbqbapi.h>

rc = QbCreateCatalog("Employee",
        "picture", 1);
```

QbDeleteCatalog

图像	音频	视频
X		

从当前数据库中删除指定的目录。

授权

改变

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbDeleteCatalog(
      char *tableName,
      char *columnName
      );
```

参数

tableName（输入）
包含图像列的表的名称。

columnName（输入）
与目录相关联的图像列的名称。

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECCatalogInUse
别人正在使用该目录。

qbicECCatalogRO
目录是只读的。

qbicECSystem
发生系统错误。

qbicECSqlError
发生了 SQL 错误。

例

删除与 Employee 表中的 picture 列相关联的 QBIC 目录:

```
#include <dmbqbapi.h>

rc=QbDeleteCatalog("Employee", "picture");
```

QbGetCatalogInfo

图像	音频	视频
X		

返回包含下列信息的 QbCatalogInfo 结构:

- 用户表的名称和该目录所属的图像列。
- 目录中包括的特征数。
- 是否打开了自动编目。

授权

选择

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbGetCatalogInfo(
      QbCatalogHandle cHdl,
      QbCatalogInfo *catInfo
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

catInfo（输出）
目录信息结构。

错误代码

qbicECIvalidHandle
目录句柄无效。

例

获取关于句柄 CatHdl 所标识的目录的信息，并在名为 catInfo 的结构中返回它:

```
#include <dmbqbapi.h>

rc = QbGetCatalogInfo(CatHdl, &catInfo);
```

QbListFeatures

图像	音频	视频
X		

返回目录中当前包括的活动特征的列表。该列表被返回至您分配的缓冲区。

授权

选择

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbListFeatures(
      QbCatalogHandle cHdl,
      SQLINTEGER bufSize,
      SQLINTEGER *count,
      char *featureNames
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

bufSize（输入）
缓冲区的大小。要估计所需的缓冲区大小，可使用 QbGetCatalogInfo API 返回的特征计数，并将该计数乘以最长特征名称的长度。存储在缓冲区中的特征名由空白字符分隔。

count（输出）
返回的特征名称的数目。

featureNames（输出）
缓冲区中的特征名数组。

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECTruncateData
因为返回缓冲区太小，所以返回的数据被截断。

例

获取句柄 CatHdl 所标识的目录中活动特征的列表。在 featureNames 数组中存储信息。

首先，计算 bufSize，这是列表所需的缓冲区大小。使用 QbGetCatalogInfo API 来返回 catInfo 结构中的特征数。然后将该数目乘以常量 qbiMaxFeatureName，它是最长特征名称的大小:

```
#include <dmbqbapi.h>

rc = QbGetCatalogInfo(CatHdl, &catInfo);

bufSize =
   catInfo.featureCount*qbiMaxFeatureName;

rc = QbListFeatures(CatHdl, bufSize,
       count, featureNames);
```

QbOpenCatalog

图像	音频	视频
X		

打开特定图像列的 QBIC 目录。可以读方式或更新方式打开目录。此 API 返回打开的目录的句柄。然后，使用其它 API 中的句柄来管理和填充目录。

在用完目录之后，务必关闭它。

授权

无

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbOpenCatalog(
      char *tableName,
      char *columnName,
      SQLINTEGER mode,
      QbCatalogHandle *cHdl
      );
```

参数

tableName（输入）
包含图像列的表的名称。

columnName（输入）
图像列的名称。

mode（输入）
打开目录的方式。有效值是 qbiRead 和 qbiUpdate。

cHdl（输出）
指向目录句柄的指针。

错误代码

qbicECCatalogNotFound
找不到该目录。

qbicECCatalogInUse
别人正在使用该目录。

qbicECOpenFailed
打不开该目录。

qbicECNotEnabled
目录未启用。

qbicECNoCatalogFound
找不到目录。

qbicECSqlError
发生了 SQL 错误。

qbicECSystem
发生系统错误。

例

以读方式打开 Employee 表中 picture 列的目录:

```
#include <dmbqbapi.h>

rc=QbOpenCatalog("employee", "picture",
     qbiread, &CatHdl);
```

QbQueryAddFeature

图像	音频	视频
X		

将指定的特征添加至 “QBIC 目录”。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryAddFeature(
      QbQueryHandle qObj,
      char *featureName
      );
```

参数

qObj（输入）
该查询对象的句柄。

featureName（输入）
要添加的查询特征的名称。为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询对象。

qbiECunknownFeatureClass
指定的特征不是可识别的特征类名。

qbiECinvalidFeatureClass
指定的特征不是有效的名称格式。

QbQueryAddFeature

qbiECfeaturePresent
指定的特征已是查询对象的成员。

qbiECallocation
系统不能分配足够的内存。

例

将 QbColorFeatureClass 特征添加至 qoHandle 句柄所标识的查询对象:

```
#include <dmbqbapi.h>

rc = QbQueryAddFeature(qoHandle,
        "QbColorFeatureClass");
```

QbQueryCreate

图像	音频	视频
X		

创建查询对象，并返回句柄。可将该句柄与其它 API 配合使用来处理该查询对象。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryCreate(
      QbQueryHandle *qObj
      );
```

参数

qObj（输出）
指向查询句柄的指针。若不成功，则将此句柄设置为 0。

错误代码

qbiECallocation
系统不能分配足够的内存。

例

创建查询对象，并在 qoHandle 中返回句柄:

```
#include <dmbqbapi.h>

rc = QbQueryCreate(&qoHandle);
```

QbQueryDelete

图像	音频	视频
X		

删除未命名的查询对象。此 API 释放该查询对象使用的所有内存和添加的任何特征。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryDelete(
      QbQueryHandle qObj
      );
```

参数

qObj（输入）
查询对象的句柄。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询。

例

删除句柄 qoHandle 所标识的查询对象:

```
#include <dmbqbapi.h>
rc = QbQueryDelete(qoHandle);
```

QbQueryGetFeatureCount

图像	音频	视频
X		

返回添加至查询对象的特征的数目。为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryGetFeatureCount(
      QbQueryHandle qObj,
      SQLINTEGER* count
      );
```

参数

qObj（输入）
查询对象的句柄。

count（输出）
指向被设置为已有特征数的变量的指针。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询对象。

例

返回句柄 qoHandle 所标识的查询对象的特征数:

QbQueryGetFeatureCount

```
#include <dmbqbapi.h>

rc = QbQueryGetFeatureCount(qoHandle,
       &count);
```

QbQueryGetString

图像	音频	视频
X		

从查询返回查询字符串。您可以此查询字符串用于应用程序中 UDF 的输入，例如，在 UDF QbScoreFromStr 或 API QbQueryStringSearch 中。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryGetString(
      QbQueryHandle qObj,
      (char*)*  queryString
      );
```

参数

qObj（输入）
查询对象的句柄。

queryString (out)
指向查询对象的查询字符串的指针。

错误代码

qbiECinvalidQueryHandle
指定的查询句柄未引用有效的查询。

例

返回句柄 qrHandle 所标识的查询对象的查询字符串:

```
#include <dmbqbapi.h>

SQLRETURN  rc;
char *queryString;
QbQueryHandle qrHandle

rc = QbQueryGetString(qrHandle, &queryString);
if (rc == 0)  {
```

```
    ...                  /* use the returned queryString for input to UDFs */
    free((void*)queryString);  /* you must free queryString  */
    queryString=(char*)0;
}
```

QbQueryListFeatures

图像	音频	视频
X		

返回查询对象中的特征的当前列表。此 API 将列表返回至分配的缓冲区。为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryListFeatures(
      QbQueryHandle qObj,
      SQLINTEGER bufSize,
      SQLINTEGER* count,
      char *featureNames
      );
```

参数

qObj（输入）
查询对象的句柄。

bufSize（输入）
featureNames 缓冲区的大小。将 qbiMaxFeatureName 常量用作缓冲区大小。查询对象特征由字符串名标识。

count（输出）
返回的特征名数。

featureNames（输出）
指向查询对象的特征名数组的指针。该数组存储在您分配的缓冲区中。

错误代码

qbiECinvalidQueryHandle
指定的查询句柄未引用有效的查询。

例

返回句柄 qoHandle 所标识的查询对象中的特征数。使用 qbiMaxFeatureName 常量来确定所需的缓冲区大小。将特征名返回至 feats 缓冲区，并将特征数返回至 retCount 变量:

```
#include <dmbqbapi.h>

bufSize = qbiMaxFeatureName;

rc = QbQueryListFeatures(qoHandle, bufSize,
        &retCount, feats);
```

QbQueryNameCreate

图像	音频	视频
X		

存储并命名查询对象，以便可在 UDF 中使用它。您提供查询对象的名称，并可以提供它的描述。

注:

1. **仅限于 EEE**: QbQueryNameCreate 在分区数据库环境中不受支持。
2. 在将来的发行版中，非分区数据库环境也将不支持 QbQueryNameCreate。要保存查询，应使用 QbQueryGetString 来获取查询字符串并保存该字符串，以供应用程序以后使用。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryNameCreate(
      QbQueryHandle qObj,
      char *name,
      char *description
      );
```

参数

qObj（输入）
查询对象的句柄。

name（输入）
查询对象的名称名称最长可以是 18 个字符。

description（输入）
查询对象的简要描述，最多 250 个字符。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询。

例

为 QbQueryCreate API 所创建的查询对象提供名称和描述:

```
#include <dmbqbapi.h>

rc = QbQueryNameCreate(qHandle,
       "fshavgcol",
       "average color query, 10/15/96");
```

QbQueryNameDelete

图像	音频	视频
X		

删除查询对象。必须已使用 QbQueryNameCreate API 命名并存储该查询对象。

注:

1. **仅限于 EEE**: QbQueryNameDelete 在分区数据库环境中不受支持。
2. 在将来的发行版中，非分区数据库环境也将不支持 QbQueryNameDelete。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryNameDelete(
      char *name
      );
```

参数

name（输入）
正在删除的查询对象的名称。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询对象。

例

删除名为 fshavgcol 的查询对象:

```
#include <dmbqbapi.h>

rc = QbQueryNameDelete("fshavgcol",);
```

QbQueryNameSearch

图像	音频	视频
X		

在 QBIC 目录中搜索符合查询对象中包含的搜索标准的图像查询对象由其名称标识。结果（包括图像句柄和 QBIC 搜索得分）存储在客户机内存中的结果数组中。结果是根据其得分排序的。

注:

1. **仅限于 EEE**: QbQueryNameSearch 在分区数据库环境中不受支持。
2. 在将来的发行版中，非分区数据库环境也将不支持 QbQueryNameSearch。要保存查询，应使用 QbQueryGetString 来获取查询字符串并保存该字符串，以供应用程序以后使用。

授权

选择

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryNameSearch(
      char *qName,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

参数

qName（输入）
查询对象的名称。

tableName（输入）
包含要搜索的图像列的表的名称。

columnName（输入）
图像列的名称。必须对图像数据启用该列。

maxReturns（输入）
要返回的最大图像数。

scope（输入）（保留）
必须设置为 0（NULL）

resultType（输入）（保留）
必须设置为 qbiArray。

count（输出）
指向返回的图像数的指针。若返回零，则确保对查询对象中的所有特征编目了该图像列。

returns（输出）
指向存放返回结果的 QbResult 结构数组的指针。务必分配足够大的缓冲区来存放期望的所有结果。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询对象。

例

对 Employee 表的 picture 列中的已编目图像运行查询 FSHAVGCOL: 确保不返回 6 个以上图像:

```
#include <dmbqbapi.h>

rc = QbQueryNameSearch("fshavgcol",
       "employee", "picture",
       6, 0, qbiArray, &count, &returns);
```

QbQueryRemoveFeature

图像	音频	视频
X		

从查询对象除去查询特征，并释放任何相关联的内存。为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryRemoveFeature(
      QbQueryHandle qObj,
      char *featureName
      );
```

参数

qObj（输入）
查询对象的句柄。

featureName（输入）
要除去的特征的名称。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询对象。

qbiECinvalidFeatureClass
指定的特征不是有效的名称格式。

qbiECfeatureNotPresent
指定的特征不是查询对象的成员。

例

从句柄 qoHandle 所标识的查询对象中除去 QbColorFeatureClass 特征:

```
#include <dmbqbapi.h>

rc = QbQueryRemoveFeature(qoHandle,
        "QbColorFeatureClass");
```

QbQuerySearch

图像	音频	视频
X		

在 QBIC 目录中搜索符合查询对象中包含的搜索标准的图像。查询对象由查询对象句柄标识。结果（包括图像句柄及其 QBIC 搜索得分）存储在客户机内存中的结果数组中。它们是根据其得分排序的。

授权

选择

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQuerySearch(
      QbQueryHandle qObj,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

参数

qObj（输入）
查询对象的句柄。

tableName（输入）
包含要搜索的图像列的表的名称。

columnName（输入）
图像列的名称。必须对图像数据启用该列。

maxReturns（输入）
要返回的最大图像数。

scope（输入）（保留）
必须设置为 0（NULL）。

resultType（输入）（保留）
必须设置为 qbiArray。

count（输出）
指向返回的图像数的指针。若返回零，则确保对查询对象中的所有特征编目了该图像列。

returns（输出）
指向存放返回结果的 QbResult 结构数组的指针。务必分配足够大的缓冲区来存放期望的所有结果。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询对象。

例

查询 Employee 表的 picture 列中的已编目的图像。确保不返回 6 个以上图像:

```
#include <dmbqbapi.h>

rc = QbQuerySearch(qHandle, "Employee",
       "picture", 6, 0, qbiArray,
       &count, &returns);
```

QbQuerySetFeatureData

图像	音频	视频
X		

设置查询对象中特征的图像数据源。仅当将特征添加至查询对象之后，才可设置数据源。数据源可以是用户表、文件或工作站缓冲区中的图像。只有在非分区数据库环境中，才能使用客户机文件或工作站缓冲区作为数据源。另外，可明确地指定平均颜色或直方图特征的数据。

在使用 QbQuerySetFeatureData 设置了服务器文件中的图像数据源之后，使用 QbQueryStringSearch。QbQuerySearch 不使用服务器文件中 QbQuerySetFeatureData 设置的图像数据源。

为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQuerySetFeatureData(
      QbQueryHandle qObj,
      char *featureName,
      QbImageSource* imgSource
      );
```

参数

qObj（输入）
查询对象的句柄。

featureName（输入）
要设置的特征的名称。

imgSource（输入）

指向图像源结构的指针。若对 imgSource 指定 0（NULL），则它表示不应更改特征中的信息。有关更多信息，参见第 141 页的『使用数据源结构』。

错误代码

qbiECinvalidQueryHandle

指定的查询对象句柄未引用有效的查询对象。

qbiECunknownFeatureClass

指定的特征不是可识别的特征类名。

qbiECinvalidFeatureClass

指定的特征不是有效的名称格式。

qbiECfeatureNotPresent

指定的特征不是查询对象的成员。

qbiECfileUnreadable

找不到或不能读图像源文件。

例

在查询对象中设置直方图颜色特征的数据源。此特征的数据源是客户机工作站上的一个文件:

```
#include <dmbqbapi.h>

QbQueryHandle qoHandle;
QbImageSource imgSource;

imgSource.sourceType = qbiSource_ClientFile;
strcpy(featureName, "QbColorHistogramFeatureClass");
strcpy(imgSource.clientFile, "/tmp/image.gif");

rc = QbQuerySetFeatureData(qoHandle, featureName, &imgSource);
```

QbQuerySetFeatureWeight

图像	音频	视频
X		

设置查询对象中指定特征的权重。

授权

无。

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQuerySetFeatureWeight(
      QbQueryHandle qObj,
      sqldouble* weight
      );
```

参数

qObj（输入）
查询对象的句柄。

weight（输出）
指向要设置为特征权重的变量的指针。

错误代码

qbiECinvalidQueryHandle
指定的查询对象句柄未引用有效的查询对象。

例

设置句柄 qoHandle 所标识的查询对象中平均颜色特征的权:

```
#include <dmbqbapi.h>

weight=2.0
rc = QbQuerySetFeatureWeight(qoHandle, "QbColorFeatureClass", &weight);
```

QbQueryStringSearch

图像	音频	视频
X		

在 QBIC 目录中搜索符合查询字符串中包含的搜索标准的图像。结果（包括图像句柄及其 QBIC 搜索得分）存储在客户机内存中的结果数组中。它们是根据其得分排序的。

授权

选择

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqqry.lib	libdmbqqry.a（AIX） libdmbqqry.sl（HP-UX） libdmbqqry.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbQueryStringSearch(
      char *queryString,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

参数

queryString（输入）
查询字符串。

tableName（输入）
包含要搜索的图像列的表的名称。

columnName（输入）
图像列的名称。必须对图像数据启用该列。

maxReturns（输入）
要返回的最大图像数。

scope（输入）（保留）
必须设置为 0（NULL）。

resultType（输入）（保留）
必须设置为 qbiArray。

QbQueryStringSearch

count（输出）

指向返回的图像数的指针。若返回零，则确保对查询字符串中的所有特征编目了图像列。

returns（输出）

指向存放返回结果的 QbResult 结构数组的指针。务必分配足够大的缓冲区来存放期望的所有结果。

错误代码

qbiECinvalidQueryString

指定的查询字符串无效。

例

查询 Employee 表的 picture 列中的已编目的图像。确保不返回 6 个以上图像:

```
#include <dmbqbapi.h>

rc = QbQueryStringSearch("QbColorFeatureClass color=<255, 0, 0>"
       "Employee",
       "picture", 6, 0, qbiArray,
       &count, &returns);
```

QbReCatalogColumn

图像	音频	视频
X		

重新分析打开的 QBIC 目录中的所有现有图像中是否有新的特征。使用特征的缺省参数。当将新的特征添加至含有图像的目录时，使用此 API。

授权

更新、插入

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbReCatalogColumn (
      QbCatalogHandle cHdl
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECInvalidCatalog
指定的句柄或表列对目录无效。

qbicECCatalog Errors
编目个别图像时出错，已记录这些错误。未发生回滚。

qbicECImageNotFound
找不到或不能存取图像。

qbicECCatalogRO
目录是只读的。

qbicECSQLError
发生了 SQL 错误。

例

重新分析打开的 QBIC 目录中的所有现有图像中是否有新的特征:

```
#include <dmbqbapi.h>

rc = QbReCatalogColumn(CatHdl);
```

QbRemoveFeature

图像	音频	视频
X		

从打开的目录中删除指定的特征。

授权

改变

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbRemoveFeature(
      QbCatalogHandle cHdl,
      char *featureName
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

featureName（输入）
特征的名称。

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECCatalogReadOnly
目录是为只读操作打开的。

qbicECFeatureNotFound
特征不在目录中。

qbiECinvalidFeatureClass
指定的特征不是有效的名称格式。

例

从句柄 CatHdl 所标识的目录中除去 QbColorHistogramFeatureClass 特征:

QbRemoveFeature

```
#include <dmbqbapi.h>

rc=QbRemoveFeature(CatHdl,
     "QbColorHistogramFeatureClass");
```

QbSetAutoCatalog

图像	音频	视频
X		

自动编目导入到图像列的图像。此 API 将关于各图像的条目添加至特征表，然后分析这些图像。当此 API 分析图像时，它创建图像数据，并将其存储在特征表中该图像的条目中。

若不打开自动编目，则在将图像添加至图像列之后，再使用 QbCatalogColumn 或 QbCatalogImage API 来进行编目。

授权

改变

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbSetAutoCatalog(
      QbCatalogHandle cHdl
      SQLINTEGER autoCatalog
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

autoCatalog（输入）
指示是否将添加至图像列的图像自动添加到特征表并进行分析。指定 1 以打开自动编目，或指定 0 关闭它。

错误代码

qbicECInvalidHandle
目录句柄无效。

例

对句柄 CatHdl 所标识的目录打开自动编目:

QbSetAutoCatalog

```
#include <dmbqbapi.h>

rc=QbSetAutoCatalog(CatHdl, 1);
```

QbUncatalogImage

图像	音频	视频
X		

从目录中除去图像。图像句柄所在的图像列必须与打开的 QBIC 目录相关联。将从打开的目录中除去图像。图像属性表中相对应的行指示未编目该图像。

授权

删除

库文件

OS/2 和 Windows	AIX、HP-UX 和 Solaris
dmbqbapi.lib	libdmbqbapi.a（AIX） libdmbqbapi.sl（HP-UX） libdmbqbapi.so（Solaris）

包含文件

dmbqbapi.h

语法

```
SQLRETURN QbUncatalogImage(
      QbCatalogHandle cHdl,
      char *imgHandle
      );
```

参数

cHdl（输入）
指向目录句柄的指针。

imgHandle（输入）
图像的句柄。可从用户表检索此句柄。

错误代码

qbicECIvalidHandle
目录句柄无效。

qbicECImageNotFound
找不到或不能存取图像。

qbicECCatalogRO
目录是只读的。

例

从句柄 CatHdl 所标识的目录中除去句柄 Img_hdl 所标识的图像:

QbUncatalogImage

```
#include <dmbqbapi.h>

rc=QbUncatalogImage(CatHdl, Img_hdl);
```

第 15 章 客户机的管理命令

本章描述如何输入客户机的 DB2 Extender 管理命令。它还给出关于客户机的每一 DB2 Extender 管理命令的参考信息。

输入 DB2 Extender 管理命令

可用交互方式或命令方式将 DB2 Extender 管理命令提交给 db2ext 命令行处理器。交互方式主要以 db2ext 提示符表示。在此方式中，仅可输入 DB2 Extender 管理命令。在命令方式中，从操作系统命令提示符处输入命令；可输入 DB2 Extender 命令以及 DB2 命令和操作系统命令。

不要从 DB2 命令提示符输入 DB2 Extender 命令。

要以交互方式启动 db2ext 命令行处理器，执行下列各项:

客户机	操作
AIX、HP-UX 和 Solaris	从操作系统命令提示处输入 DB2EXT 命令。
Windows	双击 DB2 Extender 文件夹中的 “DB2EXT 命令行处理器” 图标，或从 DB2 命令窗口输入 DB2EXT 命令。

要结束交互方式，输入 quit 或 terminate 命令。quit 命令结束交互方式，但维护与 DB2 的当前连接。terminate 命令结束交互方式，删除与 DB2 的当前连接。

要以命令方式提交 DB2 extender 命令，从操作系统命令行输入它们。必须将 db2ext 放在每一 DB2 Extender 命令前面，例如:

```
db2ext enable database for db2image using mydataspace, myindxspace, mylongspace
```

获取 DB2 Extender 命令的联机帮助

要获取所有 DB2 Extender 命令的联机帮助，输入:

```
db2ext ?
```

© Copyright IBM Corp. 1998, 2003

ADD QBIC FEATURE

图像	音频	视频
X		

在当前目录中创建指定属性的属性表。Image Extender 并不自动地重新分析目录中的现有图像。

授权

改变、控制、SYSADM 或 DBADM

命令语法

命令参数

feature_name

正在添加至 QBIC 目录的特征名称。为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

例

将 QbColorFeatureClass 特征添加至当前打开的目录中:

```
add qbic feature qbcolorfeatureclass
```

用法注释

在使用此命令之前，与数据库连接。

必须打开该目录。

CATALOG QBIC COLUMN

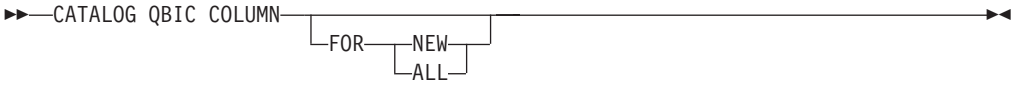
图像	音频	视频
X		

将图像编目到图像列中，并用特征数据更新当前打开的 QBIC 目录。可更新图像列中的所有图像的目录，或仅更新上次分析目录后添加至图像列的新图像的目录。

授权

插入、控制、SYSADM 或 DBADM

命令语法



命令参数

无。

例

将新图像编目到当前目录中，即，未编目的图像:

```
catalog qbic column for new
```

用法注释

指定 NEW 时，Image Extender 仅用未编目的图像更新目录。指定 ALL 时，Image Extender 分析当前目录的图像列中的每一图像。NEW 是缺省值。

在使用此命令之前，与数据库连接。

必须打开该目录。

CLOSE QBIC CATALOG

图像	音频	视频
X		

关闭 QBIC 目录。

授权

无。

命令语法

命令参数

无。

例

关闭当前目录:

```
close qbic catalog
```

用法注释

必须打开 QBIC 目录。

CONNECT

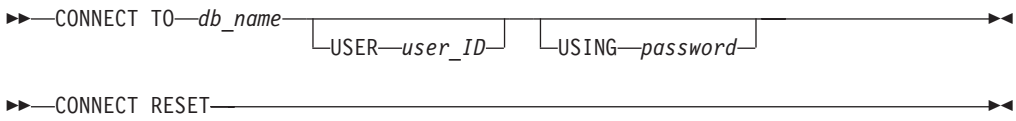
图像	音频	视频
X	X	X

与数据库连接。Extender 需要与数据库的单独连接，与 DB2 连接分开。

授权

连接

命令语法



命令参数

db_name
数据库名称。

user_ID
有权连接至数据库的用户标识。

password
该用户标识的密码。

RESET
在落实任何暂挂更改之后，断开与数据库的连接。

例

与 PERSONNL 数据库连接。用户标识是 anita，密码是 anitapas:

```
connect to personnl user anita
   using anitapas
```

用法注释

数据库是以 SHARE 方式连接的。

在运行任何其它 Extender 命令之前，运行此命令。

CREATE QBIC CATALOG

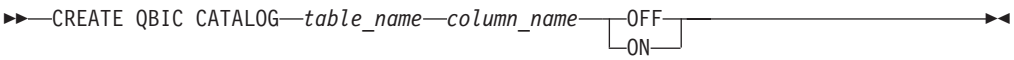
图像	音频	视频
X		

在当前数据库中创建指定的 DB2IMAGE 列的 QBIC 目录。Extender 自动生成目录名。

授权

改变、控制、SYSADM 或 DBADM

命令语法



命令参数

table_name
DB2IMAGE 启用的表的名称。

column_name
DB2IMAGE 启用的列的名称。

OFF 手工编目图像。

ON 自动编目图像。

tablespace_name
"QBIC 目录"的表空间规范和索引选项。此规范有四个部分:

- 包含特征数据的目录表的表空间的名称。必须指定表空间。此表空间应是分段表空间。
- 对于在目录表上创建的索引，这是类型 2 非分区索引的正在使用的块、空闲块、gbpcache 块或索引选项的任意组合。此规范是可选的。若不指定此部分，则将使用缺省值。
- 目录记录表的表空间的名称。此表空间可以是简单表空间或分段表空间。此规范是可选的。若不对记录表指定表空间，则使用对特征数据表指定的表空间。
- 对于在记录数据表上创建的索引，这是类型 2 非分区索引的正在使用的块、空闲块、gbpcache 块或索引选项的任意组合。此规范是可选的。若不指定此部分，则将使用缺省值。

例

创建 employee 表中 picture 列的 QBIC 目录，并将自动编目设置为 ON:

```
create qbic catalog employee picture on
```

用法注释

若指定 ON，导入该列的图像会自动编目到相关联的 QBIC 目录中。缺省值是 OFF。

在使用此命令之前，与数据库连接。

DELETE QBIC CATALOG

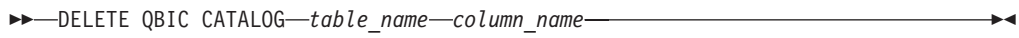
图像	音频	视频
X		

删除 QBIC 目录，包括所有 QBIC 搜索支持数据。

授权

改变、控制、SYSADM 或 DBADM

命令语法



命令参数

table_name
DB2IMAGE 启用的表的名称。

column_name
DB2IMAGE 启用的列的名称。

例

删除与 employee 表中的 picture 列相关联的目录:

```
delete qbic catalog employee picture
```

用法注释

在使用此命令之前，与数据库连接。

DISABLE COLUMN

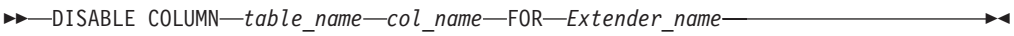
图像	音频	视频
X	X	X

禁止指定的列存储指定的介质数据。

授权

SYSADM、DBADM、控制或改变

命令语法



命令参数

table_name
当前数据库中表的名称。

col_name
要禁用的列名。

Extender_name
要禁止其使用列的 Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

例

禁用表 Employee 中的列 Photo，使其不能存放图像数据:

```
disable column employee photo for db2image
```

用法注释

在使用此命令之前，与数据库连接。

禁用列时:

- 该列不能存储指定的 Extender 的数据。这不影响是否允许或禁止该表中的其它列存储多媒体数据类型。
- 列项的内容设置为 NULL，且删除管理表中相对应的行。
- 删除与该列相关联的触发器。

DISABLE DATABASE

图像	音频	视频
X	X	X

禁止当前数据库存储介质数据。

授权

SYSADM、DBADM

命令语法

命令参数

Extender_name

要禁止其使用当前数据库的 Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

例

禁止当前数据库存放图像数据:

```
disable database for db2image
```

用法注释

在使用此命令之前，与数据库连接。

禁用数据库时，系统:

- 禁用仅对指定的 Extender 启用的所有表。
- 删除指定的 Extender 的 UDF 管理支持表。

DISABLE TABLE

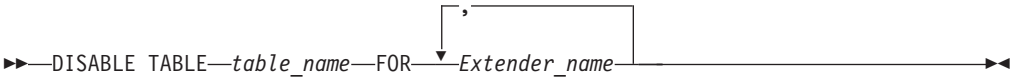
图像	音频	视频
X	X	X

禁止指定的表存储介质数据。

授权

SYSADM、DBADM、控制或改变

命令语法



命令参数

table_name
当前数据库中您要禁用的表的名称。

Extender_name
要禁止其使用该表的 Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

例

禁止表 employee 存放图像数据:

```
disable table employee for db2image
```

用法注释

在使用此命令之前连接数据库。

禁用表时，系统:

- 禁用允许指定的 Extender 使用的表中的所有列。
- 删除与表相关联的管理支持表。

DISCONNECT SERVER AT NODENUM（仅限于 EEE）

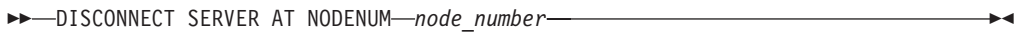
图像	音频	视频
X	X	X

从所有数据库上的指定节点断开与服务器的连接。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法



命令参数

node_number
要与服务器断开连接的节点。

例

从节点号 2 上的所有数据库断开与服务器的连接:

```
disconnect server at nodenum 2
```

用法注释

要从所有节点上所有数据库断开与服务器的连接，使用 DMBSTOP 命令。

DISCONNECT SERVER FOR DATABASE（仅限于 EEE）

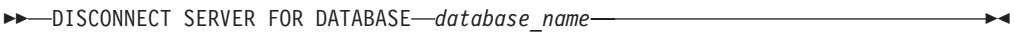
图像	音频	视频
X	X	X

从指定数据库的所有节点断开与服务器的连接。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法



命令参数

database_name
要与服务器断开连接的数据库。

例

从称为 MY_DATABASE 的数据库断开与服务器的连接:

```
disconnect server for database my_database
```

用法注释

要从所有节点上所有数据库断开与服务器的连接，使用 DMBSTOP 命令。

DISCONNECT SERVER FOR DATABASE AT NODENUM（仅限于 EEE）

图像	音频	视频
X	X	X

从指定节点上的指定数据库断开与服务器的连接。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法

命令参数

database_name
要与服务器断开连接的数据库。

node_number
要与服务器断开连接的节点。

例

从节点号 2 上称为 MY_DATABASE 的数据库断开与服务器的连接:

```
disconnect server for database my_database at nodenum 2
```

用法注释

要从所有节点上所有数据库断开与服务器的连接，使用 DMBSTOP 命令。

ENABLE COLUMN

图像	音频	视频
X	X	X

允许指定的列存储介质数据。

授权

SYSADM、DBADM、控制或改变

命令语法

命令参数

table_name
当前数据库中表的名称。

col_name
要启用的列的名称。

Extender_name
要允许其使用该表的 Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

例

允许表 employee 中的列 photo 存放图像数据:

```
enable column employee photo for db2image
```

用法注释

在使用此命令之前连接数据库。

ENABLE DATABASE

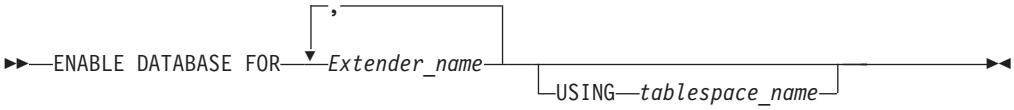
图像	音频	视频
X	X	X

允许当前数据库使用指定的表空间来存储介质数据。

授权

SYSADM、SYSCTRL、DBADM

命令语法



命令参数

Extender_name

要允许其使用当前数据库的 Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

tablespace_name

表空间的名称，它是一个在其中存储管理表的容器的集合。表空间名有如下三个部分： *datats*、*indexts*、*longts*，其中 *datats* 是在其中创建元数据的表空间的名称； *indexts* 是在其中创建元数据表上的索引的表空间的名称；而 *longts* 是在其中存储元数据表中长列（如包含 LONG VARCHAR 和 LOB 数据类型的那些列）的值的表空间的名称。若对表空间名的任何部分提供空值，则使用该部分的缺省表空间的名称。应在包括分区数据库系统中的所有节点的节点组中定义指定的表空间。

例

允许当前数据库存放图像数据:

```
enable database for db2image using mydataspace, myindxspace, mylongspace
```

用法注释

在使用此命令之前，与数据库连接。

若未指定表空间，系统将 USERSPACE1 表空间用作管理表。

ENABLE TABLE

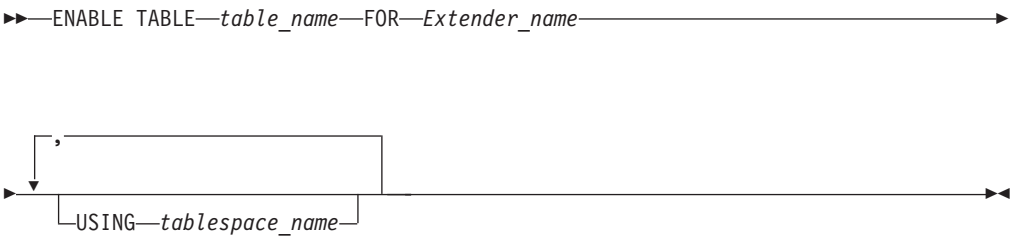
图像	音频	视频
X	X	X

允许指定的表使用指定的表空间来存储介质数据。

授权

SYSADM、DBADM、控制或改变

命令语法



命令参数

table_name
当前数据库中要启用的表的名称。

Extender_name
要允许其使用该表的 Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

tablespace_name
表空间的名称，表空间是在其中存储管理表的一组容器。表空间规范由如下三个部分组成：*datats*、*indexts 和 longts*，其中，*datats* 是在其中创建元数据表的表空间；*indexts* 是在其中创建元数据表上的索引的表空间；*longts* 是在其中存储元数据表中的长型列（诸如包含 LONG VARCHAR 和 LOB 数据类型的列）的值的表空间。若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

若对表空间规范的任何部分提供空值，则使用该部分的缺省表空间。

仅限于 EEE：指定的表空间应与用户表在同一节点组中。

例

允许表 employee 存放图像数据:

```
enable table employee for db2image
   using mydataspace, myindxspace, mylongspace
```

允许表 employee 存放图像数据。使用缺省表空间:

```
 enable table employee for db2image
```

用法注释

在使用此命令之前连接数据库。

若未指定表空间，系统将使用启用当前数据库时定义的表空间。

GET Extender STATUS

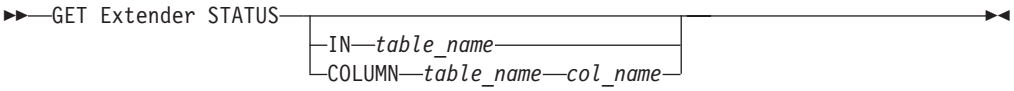
图像	音频	视频
X	X	X

显示允许其使用列、表或当前数据库的 Extender（若有的话）的名称。

授权

无

命令语法



命令参数

table_name
当前数据库中表的名称。

col_name
列的名称。

例

显示数据库中启用的 Extender 的名称:

```
get Extender status
```

显示表 employee 的状态:

```
get Extender status in employee
```

显示表 employee 中列 ADDRESS 的状态:

```
get Extender status column employee address
```

用法注释

在使用此命令之前连接数据库。

GET INACCESSIBLE FILES

图像	音频	视频
X	X	X

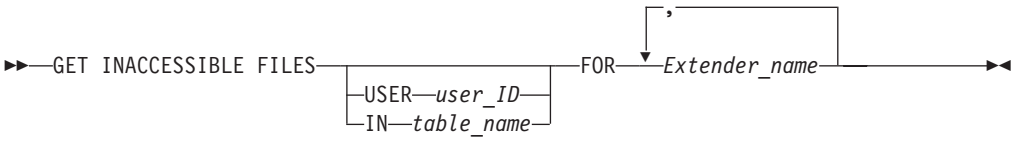
列示表、带特定限定符的表，或当前数据库中的所有表不可存取但引用了的所有媒体文件。

授权

对于当前数据库中的所有表，即，若不指定 USER 或 IN: SYSADM、SYSCTRL、SYSMAINT 或 DBADM

对于特定表（若指定 IN）或属于某限定符的表（若指定 USER）: 选择

命令语法



命令参数

user_ID
想要列示其不可存取文件的当前数据库中的表的限定符。

table_name
想要列示其不可存取文件的当前数据库中的表的名称。

Extender_name
Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

例

列示数据库中的表引用的，但不可存取的所有图像文件:

```
get inaccessible files
   for db2image
```

列示带有限定符 anita 的表中引用的，但不可存取的所有图像文件:

```
get inaccessible files
   user anita for db2image
```

列示 employee 表中的条目引用的、但不可存取的所有图像文件:

```
get inaccessible files
   in employee FOR db2image
```

用法注释

在使用此命令之前连接数据库。

若指定表，该命令列示该表的不可存取的文件。若指定限定符，该命令仅列示那些带该限定符的表的不可存取的文件。若两者都不指定，则该命令列示当前数据库中的所

GET INACCESSIBLE FILES

有表不可存取的文件。

GET QBIC CATALOG INFO

图像	音频	视频
X		

返回关于当前打开的目录的下列信息:

- 用户表的名称和与目录相关联的图像列。
- 目录中特征的名称。
- 目录中特征的数目。
- 自动分析是否打开。

授权

选择、控制、SYSADM 或 DBADM

命令语法

命令参数

无。

例

获取关于当前打开的 QBIC 目录的信息:

```
get qbic catalog info
```

用法注释

在使用此命令之前，与数据库连接。

必须打开该目录。

GET REFERENCED FILES

图像	音频	视频
X	X	X

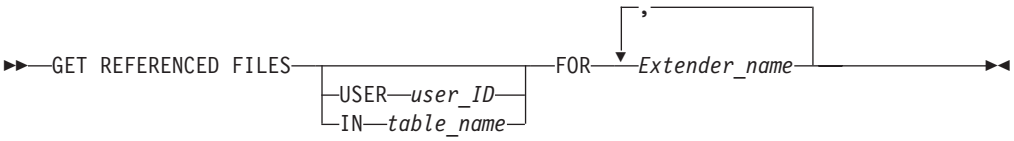
列示表、带特定限定符的表或当前数据库里所有表中的所有媒体文件及引用它们的列名。

授权

对于当前数据库中的所有表，即，若不指定 USER 或 IN: SYSADM、SYSCTRL、SYSMAINT 或 DBADM

对于特定表（若指定 IN）或属于某限定符的表（若指定 USER）: 选择

命令语法



命令参数

user_ID
想要列示其引用的文件的数据库中的表的限定符。该命令仅搜索带有该限定符的表。

table_name
当前数据库中，要列示其引用的文件的表的名称。该命令仅搜索该表。

Extender_name
Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

例

列示数据库中的所有表中的表条目引用的所有图像文件:

```
get referenced files
   for db2image
```

列示具有限定符 anita 的表中的条目引用的所有图像文件:

```
get referenced files
   user anita for db2image
```

列示 employee 表中的条目引用的所有图像文件:

```
get referenced files
   in employee for db2image
```

用法注释

在使用此命令之前连接数据库。

若不指定任何参数，则此命令搜索数据库中的所有表。

GET SERVER STATUS

图像	音频	视频
X	X	X

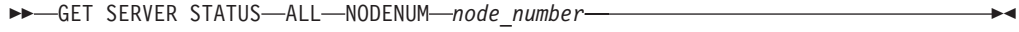
显示当前数据库或所有数据库的 Extender 服务器状态。

仅限于 EEE 若指定了节点，该命令仅显示该节点上的当前数据库或所有数据库的 Extender 服务器状态。

授权

无

命令语法



命令参数

ALL 显示所有数据库的状态。

node_number
节点的号码。该命令显示此节点的状态。（**仅限于 EEE**）

例

显示当前数据库的 Extender 服务器的状态:

```
get server status
```

显示所有数据库的 Extender 服务器的状态:

```
get server status all
```

显示所有数据库的节点号 2 的 Extender 服务器的状态:

```
get server status all nodenum 2
```

用法注释

在使用此命令之前，与数据库连接。

若未指定任何参数，该命令将显示 db2nodes.cfg 文件中列示的当前数据库的所有节点的状态。

OPEN QBIC CATALOG

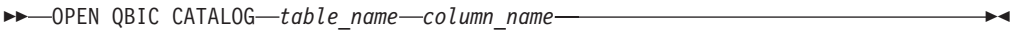
图像	音频	视频
X		

打开指定的 DB2IMAGE 列的目录。数据库将总是尝试用更新方式打开目录。若该目录已处于更新方式，将以读方式打开该目录。

授权

连接

命令语法



命令参数

table_name
DB2IMAGE 启用的表的名称。

column_name
DB2IMAGE 启用的列的名称。

例

打开 employee 表中 picture 列的 QBIC 目录:

```
open qbic catalog employee picture
```

用法注释

在使用此命令之前，与数据库连接。

此命令将导致任何打开的目录关闭。

QUIT

图像	音频	视频
X	X	X

关闭交互方式命令输入的 db2ext 命令行处理器。维护与 DB2 的连接，为此仍可以命令方式将命令提交至 db2ext 命令行处理器。

授权

无

命令语法

命令参数

无。

例

关闭交互方式的命令行接口:

```
quit
```

用法注释

QUIT 维护与数据库的连接。

RECONNECT SERVER AT NODENUM（仅限于 EEE）

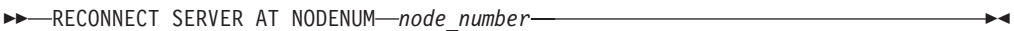
图像	音频	视频
X	X	X

将服务器与所有数据库上的指定节点重新连接。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法



命令参数

node_number
要与服务器重新连接的节点。

例

将服务器与节点号 2 上的所有数据库重新连接:

```
reconnect server at nodenum 2
```

用法注释

要从所有节点上的所有数据库重新连接服务器，使用 DMBSTART 命令。

RECONNECT SERVER FOR DATABASE（仅限于 EEE）

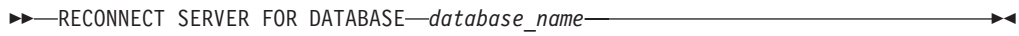
图像	音频	视频
X	X	X

将服务器与指定数据库的所有节点重新连接。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法



命令参数

database_name
要与服务器重新连接的数据库。

例

将服务器与称为 MY_DATABASE 的数据库重新连接:

```
disconnect server for database my_database
```

用法注释

要将服务器与所有节点上的所有数据库重新连接，使用 DMBSTART 命令。

RECONNECT SERVER FOR DATABASE AT NODENUM（仅限于 EEE）

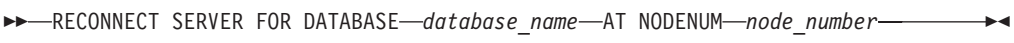
图像	音频	视频
X	X	X

将服务器与指定节点上的指定数据库重新连接。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法



命令参数

database_name
要与服务器重新连接的数据库。

node_number
要与服务器重新连接的节点。

例

将服务器与节点号 2 上称为 MY_DATABASE 的数据库重新连接:

```
reconnect server for database my_database at nodenum 2
```

用法注释

要将服务器与所有节点上的所有数据库重新连接，使用 DMBSTART 命令。

REDISTRIBUTE NODEGROUP（仅限于 EEE）

图像	音频	视频
X		

当将节点添加至节点组或从节点组中除去节点时，或当建立节点组的新分区映象时，再分发 Extender 数据。

授权

SYSADM、DBADM

命令语法

命令参数

节点组（nodegroup）
要再分发的节点组的名称。

CONTINUE
若再分发进程返回一个错误，可重新运行该命令，并根据命令响应提供的指示信息指定或不指定 CONTINUE 参数。此选项指示系统从停止的地方继续，而不是从头开始。

例

再分发称为 my_nodegroup 的节点组:

```
redistribute nodegroup my_nodegroup
```

用法注释

在使用此命令之前，与数据库连接。

在 DB2 的 REDISTRIBUTE 之后首次运行 REDISTRIBUTE NODEGROUP 命令时，不应使用 CONTINUE 参数。若是首次使用，则会记录错误，并从头开始再分发。

要维护数据完整性，必须一次再分发一个节点组。在启动另一再分发之前，等待一个节点组完成再分发。

若 REDISTRIBUTE NODEGROUP 失败，则可参考下列其中一个目录中的文件 “redist.log” 以获取详细说明:

- **Unix**: /<home-instance>/dmb/redist
- **Windows**: \\<instance_owning_machine>\DB2<instance_name>\<instance_name>\dmb\redist

REMOVE QBIC FEATURE

图像	音频	视频
X		

从打开的目录中删除指定特征的特征表。

授权

改变、控制、SYSADM、DBADM

命令语法

命令参数

feature_name

正从 QBIC 目录除去的特征的名称。为 Image Extender 提供了下列特征:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

例

从当前打开的目录除去 QbColorFeatureClass 特征:

```
remove qbic feature qbcolorfeatureclass
```

用法注释

在使用此命令之前，与数据库连接。

必须打开该目录。

REORG

图像	音频	视频
X	X	X

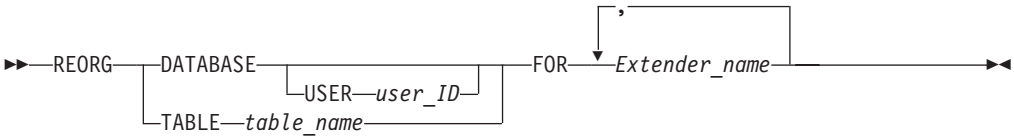
清除与特定表、带特定限定符的表，或当前数据库中的所有表相关联的管理表（管理表和属性表）。

授权

对于特定表（若运行 REORG TABLE）或带特定限定符的表（若运行 REORG DATABASE）：SYSADM、SYSCTRL、SYSMAINT、DBADM 或控制

对于数据库中的所有表（若运行 REORG DATABASE）：SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法



命令参数

user_ID
表的限定符。

table_name
当前数据库中，要清除其管理表的表名。

Extender_name
Extender 的名称。有效的 Extender 的名称是 db2image、db2audio 和 db2video。

例

重组并清除当前数据库的图像管理表:

```
reorg database for db2image
```

重组并清除所有带限定符 anita 的表中的图像管理表:

```
reorg database user anita for db2image
```

重组并清除 employee 表的图像管理表:

```
reorg table employee for db2image
```

用法注释

在使用此命令之前，与数据库连接。

SET QBIC AUTOCATALOG

图像	音频	视频
X		

当将图像导入列时，自动编目这些图像。将把图像添加至与该列相关联的 QBIC 目录。

授权

改变、控制、SYSADM、DBADM

命令语法

命令参数

无。

例

打开自动编目:

```
set qbic autocatalog on
```

用法注释

必须打开 QBIC 目录。

START SERVER（仅限于非 EEE）

图像	音频	视频
X	X	X

启动当前数据库的 Extender 服务器。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法

命令参数

无。

例

启动当前数据库的 Extender 服务器:

```
start server
```

用法注释

在使用此命令之前，与数据库连接。

STOP SERVER（仅限于非 EEE）

图像	音频	视频
X	X	X

停止当前数据库的 Extender 服务器。

授权

SYSADM、SYSCTRL、SYSMAINT 或 DBADM

命令语法

命令参数

无。

例

停止当前数据库的 Extender 服务器:

```
stop server
```

用法注释

在使用此命令之前，与数据库连接。

TERMINATE

图像	音频	视频
X	X	X

关闭 db2ext 命令行处理器，并删除与 DB2 的连接。

授权

无

命令语法

命令参数

无。

例

关闭 db2ext 命令行处理器:

```
quit
```

用法注释

TERMINATE 删除与数据库的连接。

第 16 章 诊断信息

程序中的所有嵌入式 SQL 语句和程序中的 DB2 CLI 调用，包括那些调用 DB2 Extender UDF 的语句，生成指示嵌入式 SQL 语句或 DB2 CLI 调用是否执行成功的代码。其它 DB2 Extender API（如管理 API）也返回指示成功与否的代码。程序应检查并响应这些返回码。

程序还可检索补充这些代码的信息。包括 SQLSTATE 信息和错误消息。可使用此诊断信息来排除和修正程序中的问题。

有时无法轻易地诊断出问题的来源。在这些情况下，可能需要向服务人员提供信息才可隔离和修正问题。DB2 Extender 包括一个记录 Extender 活动的跟踪设施。对于服务人员来说，跟踪信息是有价值的输入。仅应在 IBM 服务人员的指导下使用跟踪设施。

本章描述如何存取此诊断信息。它描述了:

- 如何处理 DB2 Extender UDF 返回码和 API 返回码。
- 如何控制跟踪

它还列示并描述了 Extender 可返回的 SQLSTATE 和错误消息。

处理 UDF 返回码

嵌入式 SQL 语句返回 SQLCA 结构的 SQLCODE、SQLWARN 和 SQLSTATE 字段中的代码。此结构是在 SQLCA 包含文件中定义的。（有关 SQLCA 结构和 SQLCA 包含文件的更多信息，参见 *DB2 Application Development Guide*。）

DB2 CLI 调用返回可使用 SQLError 函数进行检索的 SQLCODE 值和 SQLSTATE 值。（有关 SQLError 函数检索错误信息的更多信息，参见 *CLI Guide and Reference*。）

SQLCODE 值 0 意味着语句运行成功（可能会有警告条件）。正数 SQLCODE 值表示语句运行成功，但有警告。（嵌入式 SQL 语句在 SQLWARN 字段中返回与 0 或正数 SQLCODE 值相关的警告。） 负数 SQLCODE 值表示有错。

DB2 将信息与每一个 SQLCODE 值相关。若 DB2 Extender UDF 遇到警告或错误，它将相关联的信息传送给 DB2 以包括在 SQLCODE 消息中。

SQLSTATE 值包含补充 SQLCODE 消息的代码。参见第 448 页的『SQLSTATE 代码』以获取 DB2 Extender 返回的每个 SQLSTATE 代码的描述。

调用 DB2 Extender UDF 的嵌入式 SQL 语句和 DB2 CLI 调用可能返回对这些 UDF 是唯一的 SQLCODE 消息和 SQLSTATE 值，但 DB2 象返回其它嵌入式 SQL 语句或其它 DB2 CLI 调用的值那样返回了这些值。因此，存取这些值的方法与存取不启动 DB2 Extender UDF 的嵌入式 SQL 语句或 DB2 CLI 调用的方法相同。

参见第 448 页的『SQLSTATE 代码』以获取 Extender 可能返回的 SQLSTATE 值和相关信息的消息号。参见第 452 页的『消息』以获取关于每条信息的信息。

© Copyright IBM Corp. 1998, 2003

处理 API 返回码

每个 DB2 Extender API 调用都返回一个代码。返回码 0 指示该 API 调用处理成功。非 0 的返回码指示 API 调用处理成功，但遇到警告，或因出错而未能成功处理。

第 217 页的第 14 章，『应用程序编程接口』列示了 DB2 Extender API 可返回的每个代码的符号值，并描述了它们。

可检索关于 API 遇到的错误的附加信息。使用 DB*x*GetError API 来检索此附加信息，其中，*x* 是 a 表示 Audio Extender，i 表示 Image Extender，v 表示 Video Extender。DB*x*GetError API 返回遇到错误的上一个 DB2 Extender API 的 SQL 错误代码和相关联的信息。参见 *DB2 Messages Reference* 以了解关于 SQL 错误代码的信息。参见第 452 页的『消息』以获取关于 DB*x*GetError API 可返回的每条消息的信息。

例如，C 应用程序中的下列语句对 Audio Extender 启用一个表，然后检查错误。

```
rc=DBaEnableTable((char *)NULL, "Employee");

rc=DBaGetError(&errCode, &errMsg);
```

SQLSTATE 代码

表 16 列示并描述了 DB2 Extender 可返回的 SQLSTATE 值。每个 SQLSTATE 值的描述都包括其符号表示法。下表还列示了与每个 SQLSTATE 值相关联的消息号。参见第 452 页的『消息』以获取关于每条消息的信息。

表 16. SQLSTATE 代码和相关联的消息号

SQLSTATE	消息号。	描述
00000		MMDB_SQLSTATE_OK 成功
01H01	DMB0211W	MMDB_SQLSTATE_WARN_NO_OVERWRITE 文件覆盖未发生
38A00	DMB0101E	MMDB_SQLSTATE_AUDIO_NULL_PARM UDF 的输入参数不能是空
38A02	DMB0209E	MMDB_SQLSTATE_AUDIO_OPEN_HDR_ERROR 打开音频文件头时出错
38A03	DMB0209E	MMDB_SQLSTATE_AUDIO_BAD_WAVE_HDR 提供了无效的 Wave 文件
38V00	DMB0101E	MMDB_SQLSTATE_VIDEO_NULL_PARM UDF 的输入参数不能是空
38V02	DMB0051E	MMDB_SQLSTATE_VIDEO_OPEN_HDR_ERROR 打开视频文件头时出错
38V03	DMB0105E	MMDB_SQLSTATE_VIDEO_BAD_MPEG1_HDR 提供了无效的 mpeg1 文件
38V04	DMB0104E	MMDB_SQLSTATE_VIDEO_BLOB_TOO_SHORT 提供的视频缓冲区太小
38V05	DMB0106E	MMDB_SQLSTATE_VIDEO_BAD_AVI_HDR 提供了无效的 AVI 文件
38V07	DMB0106E	MMDB_SQLSTATE_VIDEO_BAD_QT_HDR 提供了无效的 Quicktime 文件

表 16. SQLSTATE 代码和相关联的消息号 (续)

SQLSTATE	消息号。	描述
38600	DMB0075E DMB0101E DMB0102E DMB0103E DMB0210E	MMDB_SQLSTATE_INVALID_INPUT UDF 的输入参数无效
38601	DMB0009E	MMDB_SQLSTATE_MALLOC_FAIL 内存分配失败
38602	DMB0386E	MMDB_SQLSTATE_CANNOT_COLLOCATE 不能与用户数据连用
38603	DMB0077E	MMDB_SQLSTATE_READ_FILE_FAIL 不能读文件
38604	DMB0080E	MMDB_SQLSTATE_WRITE_FILE_FAIL 不能写文件
38610	DMB0070E	MMDB_SQLSTATE_INVALID_HANDLE 媒体列包含无效数据
38611	DMB0073E	MMDB_SQLSTATE_INVALID_SESSION_HANDLE 无效的 UDF 会话句柄
38612	DMB0074E	MMDB_SQLSTATE_INVALID_STATEMENT_HANDLE 无效的 UDF 语句句柄
38613	DMB0083E	MMDB_SQLSTATE_INVALID_IMPORT_REQUEST 导入请求无效
38615	DMB0071E	MMDB_SQLSTATE_CONNECT_FAIL 连接数据库时出错
38617	DMB0071E	MMDB_SQLSTATE_ALLOC_STMT_FAIL 分配新语句句柄时出错
38618	DMB0208E DMB0138E	MMDB_SQLSTATE_FREE_STMT_FAIL 释放语句时出错
38619	DMB0208E DMB0132E	MMDB_SQLSTATE_BIND_FAIL 绑定时出错
38620	DMB0208E	MMDB_SQLSTATE_BIND_COLUMN_FAIL 绑定列时出错
38621	DMB0208E	MMDB_SQLSTATE_BIND_FILE_FAIL 绑定文件时出错
38622	DMB0208E DMB0132E	MMDB_SQLSTATE_SET_PARAM_FAIL 设置参数时出错
38623	DMB0208E DMB0131E	MMDB_SQLSTATE_PREPARE_FAIL 准备 SQL 语句时出错
38624	DMB0208E DMB0133E DMB0172E	MMDB_SQLSTATE_EXECUTE_FAIL 执行语句时出错
38625	DMB0208E DMB0133E	MMDB_SQLSTATE_EXEC_DIRECT_FAIL 直接在 UDF 中执行 SQL 语句时出错

SQLSTATE

表 16. SQLSTATE 代码和相关联的消息号 (续)

SQLSTATE	消息号。	描述
38626	DMB0208E DMB0133E	MMDB_SQLSTATE_FETCH_FAIL 检索下一数据行时出错
38627	DMB0208E	MMDB_SQLSTATE_COMMIT_FAIL 落实事务时出错
38628	DMB0208E	MMDB_SQLSTATE_GET_LENGTH_FAIL 检索字符串值的长度时出错
38629	DMB0208E	MMDB_SQLSTATE_GET_SUBSTRING_FAIL 检索字符串值的一部分时出错
38650	DMB0077E DMB0079E	MMDB_SQLSTATE_COPY_BLOB_2_FILE_FAIL 将 BLOB 复制至文件时出错
38651	DMB0086E	MMDB_SQLSTATE_BLOB_BUFFER_TOO_SMALL 提供的缓冲区太小
38652	DMB0082E	MMDB_SQLSTATE_BUILD_HANDLE 构造媒体列数据时出错
38653	DMB0083E	MMDB_SQLSTATE_INVALID_INSERT_VIA_SELECT 通过选择的插入请求无效
38654	DMB0081E	MMDB_SQLSTATE_INVALID_OFFSET_SIZE 偏移大小无效
38655	DMB0068E	MMDB_SQLSTATE_METATABLE_DOESNOT_EXIST 必需的元数据表不存在
38670	DMB0134E DMB0103E	MMDB_SQLSTATE_UNKNOWN_FORMAT 存储的介质具有未知的格式
38671	DMB0135E	MMDB_SQLSTATE_CREATE_THUMBNAIL_FAIL 创建缩略图时出错
38672	DMB0114E	MMDB_SQLSTATE_FORMAT_CONVERSION_FAIL 转换文件格式时出错
38673	DMB0363E	MMDB_SQLSTATE_INVALID_UPDATE 调用不引用表的 Update UDF 时出错
38674	DMB0361E	MMDB_SQLSTATE_NOT_ENABLED 当 Import UDF 应用于未对 Extender 启用的列时出错
38675	DMB0129E	MMDB_SQLSTATE_VIDEO_INTERNAL Video Extender UDF 中发生内部错误
38676	DMB0129E	MMDB_SQLSTATE_AUDIO_INTERNAL Audio Extender UDF 中发生错误内部
38677	DMB0129E	MMDB_SQLSTATE_IMAGE_INTERNAL
38678	DMB0089E DMB0208E	MMDB_SQLSTATE_BASE_INTERNAL_ERROR 公共层中发生内部错误
38681	DMB0108E	MMDB_SQLSTATE_IMPORT_ENV_NOT_SETUP 用于导入的环境变量未正确设置
38682	DMB0111E	MMDB_SQLSTATE_STORE_ENV_NOT_SETUP 用于存储操作的环境变量未正确设置
38683	DMB0107E	MMDB_SQLSTATE_EXPORT_ENV_NOT_SETUP 用于导出操作的环境变量未正确设置

表 16. SQLSTATE 代码和相关联的消息号 (续)

SQLSTATE	消息号。	描述
38684	DMB0117E	MMDB_SQLSTATE_TEMP_ENV_NOT_SETUP 用于创建临时文件的环境变量未正确设置
38686	DMB0109E	MMDB_SQLSTATE_CANT_RESOLVE_IMPORT_FILE 解析导入文件名时出错
38687	DMB0112E	MMDB_SQLSTATE_CANT_RESOLVE_STORE_FILE 解析存储文件名时出错
38688	DMB0110E	MMDB_SQLSTATE_CANT_RESOLVE_EXPORT_FILE 解析导出文件名时出错
38689	DMB0116E	MMDB_SQLSTATE_CANT_CREATE_TMP_FILE 创建临时文件时出错
38690	DMB0076E	MMDB_SQLSTATE_OPEN_IMPORT_FILE_FAIL 打不开导入文件
38691	DMB0115E	MMDB_SQLSTATE_OPEN_STORE_FILE_FAIL 打不开导入文件
38692	DMB0114E	MMDB_SQLSTATE_OPEN_EXPORT_FILE_FAIL 打不开导出文件
38693	DMB0118E	MMDB_SQLSTATE_OPEN_TEMP_FILE_FAIL 打不开临时文件
38694	DMB0117E	MMDB_SQLSTATE_OPEN_CONTENT_FILE_FAIL 打不开内容文件
38695	DMB0135E	MMDB_SQLSTATE_OPEN_THUMBNAIL_FILE_FAIL 打不开缩略图文件
38696	DMB0135E	MMDB_SQLSTATE_READ_THUMBNAIL_FILE_FAIL 不能读缩略图文件
38697	DMB0207E	MMDB_SQLSTATE_OVERWRITE_NOT_ALLOWED 不能执行覆盖操作
38699	DMB0171E	MMDB_SQLSTATE_QUERY_NAME_NOT_FOUND 找不到具有该名称的查询
38700		MMDB_SQLSTATE_NO_MANAGEBLOB
38701		MMDB_SQLSTATE_UDFLOCATOR_FAIL
38702		MMDB_SQLSTATE_SQL_FAIL
38703		MMDB_SQLSTATE_INVALID_UPDATE
38704		MMDB_SQLSTATE_NOT_ENABLED
38705	DMB0366E DMB0382E	MMDB_SQLSTATE_QBIC_QUERY_FAIL_TO_BUILD 构建查询时发生故障
38706	DMB0205E	MMDB_SQLSTATE_QBIC_TABLE_COLUMN_PAIR_NOT_VALID 尝试存取 QBIC 目录时发生故障。或者是在该目录中找不到图像句柄, 或者是表名与列名的组合没有目录。
38707	DMB0383E	MMDB_SQLSTATE_QBIC_QUERY_EXECUTE_FAILED 运行查询时发生故障
38708		MMDB_SQLSTATE_QBIC_UKNOWN_ERROR QBIC 中发生未知的故障

SQLSTATE

表 16. SQLSTATE 代码和相关联的消息号 (续)

SQLSTATE	消息号。	描述
38709	DMB0208E	MMDB_COPY_FILE_TO_LOCATOR_FAILURE 将文件复制至 LOB 定位器时发生故障
38710	DMB0534E	MMDB_SQLSTATE_QBIC_UNSUPPORTED_UDF 该 UDF 不受支持。

消息

DMB0001E 未连接 DB2 Extender 服务器。原因: ″<code>″。

原因: 尝试的操作试图要求运行 DB2 Extender 服务。

操作: 在服务器上，在命令行上对操作系统运行 DMBSTART 命令。

DMB0003W 正对此会话运行 DB2 Extender 跟踪设施。

原因: 跟踪设施用尽了系统资源。

操作: 若系统的性能受到影响，则您可能要关闭跟踪。

DMB0004I 只有实例所有者才可运行此程序: ″<name>″。

原因: 必须由创建实例的用户标识启动 DB2 Extender 服务器。

操作: 由在其下创建实例的用户标识运行 DMBSTART 命令。

DMB0005E 未对 ″<Extender-name>″ Extender 启用当前数据库。

原因: 尝试要求对特定 DB2 Extender 启用数据库的操作。例如，若要对 DB2IMAGE 数据启用表，则必须先对 DB2IMAGE 数据启用存储该表的数据库。

操作: 对想要的 Extender 数据类型启用数据库，并再试。

DMB0006E 用户 ″<name>″ 无权调用此 API。

原因: 尝试从某用户标识调用应用程序编程接口，但该用户标识不具有该 API 所需的权限级别。

操作: 从另一用户标识运行应用程序，或请数据库管理员更改初始用户标识的权限级别。

DMB0007E 未对 Extender ″<Extender-name>″ 启用用户表 ″<table-name>″。

原因: 未对 DB2 Extender 启用尝试对其执行操作的表。例如，在可对音频启用表中的列之前，必须启用该表以存放音频数据。

操作: 确保首先对 Extender 启用该表。然后启用列。

DMB0008E 运行存储过程 ″<name>″ 时出错。

原因: 消息中标识的存储过程中有错误，或环境有问题。

操作: 验证应用程序并再试。

DMB0009E 内存分配错误。

原因: 系统无法分配支持尝试的操作所需的内存。

操作: 验证系统是否有足够的内存来完成该操作。

DMB0010E 先前已对 UDT ″<name>″ 定义了 ″<Extender-name>″ Extender。

原因: 已将用户定义类型（UDT）名用于对另一 DB2 Extender 定义的 UDT。

操作: 为 UDT 选择另一名称。

DMB0011E 不能对 ″<Extender-name>″ Extender 启用用户列 ″<column-name>″。用户列的定义与和 Extender 相关联的单值类型 ″MMDBSYS.<name>″ 不兼容。

原因: 未对消息中显示的数据类型定义指示的列，因此不能对该 Extender 启用它。

操作: 确保已使用与 Extender 相对应的数据类型定义了要启用的列。

DMB0012E 指定的用户表 ″<table-name>″ 不存在。

原因: 不存在具有指定名称的表。

操作: 检查表的名称并检查该表是否存在。

DMB0013E 未对表 ″<table-name>″ 定义列 ″<column-name>″。

原因: 尝试的操作引用了不存在于标识的表中的列名。

操作: 检查表和列的名称。

DMB0014W 已对 ″<Extender-name>″ Extender 启用了用户表 ″<table-name>″ 中的列 ″<column-name>″。

原因: 试图对 Extender 启用列，但已对该 Extender 启用了该列。

操作: 不需要任何操作。

DMB0015W 已对 Extender ″<Extender-name>″ 启用了数据库。

原因: 试图对 Extender 启用数据库，但已对该 Extender 启用了该数据库。

操作: 不需要任何操作。

DMB0016W 已对 ″<Extender-name>″ Extender 启用了用户表 ″<table-name>″。

原因: 试图对 Extender 启用表，但已对该 Extender 启用了该表。

操作: 不需要任何操作。

DMB0017E 已对 ″<Extender-name>″ Extender 启用了用户表 ″<table-name>″。但最少一个相关联的元数据表 ″<table-name>″ 或 ″<table-name>″ 不存在。

原因: 与该表相关联的一个或多个管理支持（元数据）表已损坏或毁坏。在没有这些元数据表的情况下，不能将用户表用于该 Extender 类型的数据。

操作: 禁用用户表，并对 Extender 重新启用它。

DMB0018E 系统不能为表 ″<table-name>″ 中的列 ″<column-name>″ 创建唯一触发器名。

原因: 当系统尝试启用用户表中的列时，在创建 DB2 Extender 使用的触发器期间出错。

操作: 重复该操作。若再次出错，与数据库管理员联系，然后与 IBM 服务中心联系。

DMB0019I Extender ″<Extender-name>″ 的表 ″<table-name>″ 中引用了 ″<Count>″ 个文件。

原因: 此消息显示特定 Extender 的用户表引用的外部媒体文件数。

操作: 不需要任何操作。

DMB0020I ″<Extender-name>″ Extender 的表模式 ″<name>″ 中引用了 ″<Count>″ 个文件。

原因: 此消息显示具有特定模式名的用户表引用的外部媒体文件数。

操作: 不需要任何操作。

DMB0021I ″<extender-name>″ Extender 的表 ″<table-name>″ 中引用了 ″<count>″ 个不可存取的文件。

原因: 此消息显示特定 Extender 的用户表引用的，但不可存取的外部媒体文件数。这些文件可能已被删除。

操作: 不需要任何操作。

DMB0022I ″<Extender-name>″ Extender 引用了 ″<count>″ 个不可存取的文件。

原因: 此消息显示外部媒体文件数，它们:

- 由当前数据库中的用户表引用。
- 是特定 Extender 介质类型（如视频）的。
- 不可存取；例如，这些文件可能已被删除。

操作: 不需要任何操作。

DMB0023I Extender ″<Extender-name>″ 的具有表模式 ″<name>″ 的表中引用了 ″<count>″ 个不可存取的文件。

原因: 此消息显示具有特定模式名的用户表引用的，但不可存取的外部媒体文件数。这些文件可能已被删除。这些消息还指示对其启用了表的 Extender 的数目。

操作: 不需要任何操作。

DMB0024I 已对 “<count>” 个 Extender 启用了当前数据库。

原因: 此消息列示对其启用了当前数据库的 DB2 Extender 的数目。

操作: 不需要任何操作。

DMB0025I 已对 ″<count>″ 个 Extender 启用了表 ″<table-name>″。

原因: 此消息列示对其启用了指示的表的 DB2 Extender 的数目。

操作: 不需要任何操作。

DMB0026I 已对 “<count>” 个 Extender 启用了表 “<table-name>” 中的列 “<column-name>”。

原因: 此消息列示对其启用了指示的列的 DB2 Extender 的数目。

操作: 不需要任何操作。

DMB0027I 已对 Extender ″<Extender-name>″ 启用了当前数据库。

原因: 此消息指示对其启用了当前数据库的 DB2 Extender。

操作: 不需要任何操作。

DMB0028I 已对 Extender ″<Extender-name>″ 启用了表 ″<table-name>″。

原因: 此消息指示允许用户表存放的介质数据类型。

操作: 不需要任何操作。

DMB0029I 已对 Extender ″<Extender-name>″ 启用了表 ″<table-name>″ 中的列 ″<column-name>″。

原因: 此消息指示允许用户列存放的介质数据类型。

操作: 不需要任何操作。

DMB0030E 不能对 ″<Extender-name>″ Extender 启用当前数据库。RC = ″<code>″。

原因: 该数据库不存在，或您无权启用该数据库。

操作: 确保该数据库存在，且您有权启用该数据库。

DMB0031E 不能对 ″<Extender-name>″ Extender 启用表。RC = ″<code>″。

原因: 数据库不存在，或未启用该表，或您无权启用该表。

操作: 确保数据库存在，且对 Extender 启用了数据库和表。确保您有权启用该表。

DMB0032E 不能对 ″<Extender-name>″ Extender 启用列。RC = ″<code>″。

原因: 该列不是使用此 Extender 的数据类型定义的，或该列不存在，或未启用表，或您无权启用该列。

操作: 确保该列是使用正确的数据类型定义的。确保表已启用，且您有权启用该列。

DMB0033E 您无权运行此命令。

原因: 您的用户标识不具有运行该命令所需的权限级别。

操作: 从另一用户标识运行该命令，或请数据库管理员更改您的当前用户标识的权限级别。

DMB0034I 数据库 ″<database-name>″ 的 “DB2 Extender 服务器” 已成功启动。

原因: 当前数据库的服务器启动成功。

操作: 不需要任何操作。

DMB0035I 数据库 ″<database-name>″ 的 DB2 Extender 服务器已停止。

原因: 当前数据库的服务器已成功停止。

操作: 不需要任何操作。

DMB0036E 不能启动或停止 DB2 Extender 服务器。DB2 Extender 服务器守护程序可能不在运行。与数据库管理员联系。

原因: 启动或停止 DB2 Extender 服务器时出错。DB2 Extender 服务器守护程序可能不在运行。

操作: 请联系数据库管理员。

DMB0037E USE 会话句柄无效。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0038E USE 语句句柄无效。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0039E USE 错误: ″<error>″。

原因: 发生了内部错误。

操作: 遵循相关错误消息中包含的指示信息，并重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0040E SQL 错误: ″<error>″

原因: 发生了内部错误。

操作: 遵循相关错误消息中包含的指示信息，并重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0041W 已使用新指定的表空间对 ″<Extender-name>″ Extender 重新启用了当前数据库。

原因: 尽管先前启用了当前数据库，但它使用另一表空间。现在用管理支持表的新表空间启用了该数据库。

操作: 不需要任何操作。

DMB0042E 未对 ″<Extender-name>″ Extender 启用表 ″<table-name>″ 中的列 ″<column-name>″。

原因: 未对 Extender 启用指示的尝试对其进行操作的列。例如，您可能尝试禁用当前未对指示的 Extender 启用的列。

操作: 确保已对消息中指示的 Extender 启用了该列。

DMB0043I 已对 ″<Extender-name>″ Extender 禁用当前数据库。

原因: 禁用操作成功。

操作: 不需要任何操作。

DMB0044I 已对 ″<Extender-name>″ Extender 禁用表 ″<table-name>″。

原因: 禁用操作成功。

操作: 不需要任何操作。

DMB0045I 已对 ″<Extender-name>″ Extender 禁用表 ″<table-name>″ 中的列 ″<column-name>″。

原因: 禁用操作成功。

操作: 不需要任何操作。

DMB0046E 不能对 ″<Extender-name>″ Extender 禁用当前数据库。RC = ″<code>″。

原因: 该数据库不存在，或未对该 Extender 启用，或您无权禁用该数据库。

操作: 确保该数据库存在，且已对 Extender 启用。确保您有权禁用该数据库。

DMB0047E 不能对 ″<Extender-name>″ Extender 禁用表。RC = ″<code>″。

原因: 该表不存在，或未对该 Extender 启用，或您无权禁用该表。

操作: 确保该表存在，且已对 Extender 启用。确保您有权禁用该表。

DMB0048E 不能对 ″<Extender-name>″ Extender 禁用列。RC = ″<code>″

原因: 未对消息中指示的 Extender 启用该列，因此不能对该 Extender 禁用它。

操作: 验证 Extender 的名称，以及用户列是否是您要禁用的列。

DMB0049E 您无权运行此命令。

原因: 您的用户标识不具有运行该命令所需的权限级别。

操作: 从另一用户标识运行该应用程序，或请数据库管理员更改您的当前用户标识的权限级别。

DMB0050E 您对表 ″<table-name>″ 不具有 ″<authority-level>″ 权限。

原因: 该操作所需的权限级别比进行尝试的用户标识的权限级别高。

操作: 从具有正确权限的用户标识执行该操作，或请数据库管理员更改您的当前用户标识的权限级别。

DMB0051E 媒体文件头不正确。

原因: 系统不能读取或打开媒体文件头。该文件已损坏，或它不是媒体文件。

操作: 验证该文件是否是媒体文件，且是否未损坏。

DMB0052I 数据库 ″<database-name>″ 的 DB2 Extender 服务器已成功启动。

原因: 服务器已成功启动。

操作: 不需要任何操作。

DMB0053I 数据库 ″<database-name>″ 的 DB2 Extender 服务器已成功停止。

原因: 服务器已成功停止。

操作: 不需要任何操作。

DMB0054E DB2 Extender 服务器不能连接至数据库或分配 DB2 语句句柄。数据库 "<database-name>" 的 DB2 Extender 服务器可能不在运行。

原因: DB2 Extender 服务器不能连接至数据库或分配 DB2 语句句柄。该数据库的 DB2 Extender 服务器可能不在运行。

操作: 确保该数据库的 DB2 Extender 服务器正在运行。若不在运行，则对该数据库启动特定的 Extender 服务器，或请系统管理员重新启动 Extender 服务。

DMB0055I "command-name" 命令成功完成。

原因: 该命令成功完成。

操作: 不需要任何操作。

DMB0056E 在 "<keyword>" 后面找到意外的标记 "<token>"。预期的标记可包括: <Extendername>。

原因: 该命令需要的是 DB2 Extender 的名称，而不是消息中指示的标记。

操作: 遵循命令的语法，并再试。

DMB0057E 表空间 "<table-space-name>" 无效。

原因: 消息中的表空间不存在。

操作: 验证表空间的名称以及它是否存在。

DMB0058I "<Extender-name>" Extender 引用了 "<Count>" 个文件。

原因: 此消息显示特定 Extender 引用的外部媒体文件数。

操作: 不需要任何操作。

DMB0059E "<Name>" 对 DB2 Extender 来说不是有效名称。有效的 Extender 名包括 "<Extender-name,>" DB2VIDEO、DB2AUDIO 和 DB2IMAGE。

原因: Extender 名拼写错误。

操作: 验证 Extender 名。

DMB0060E "<function>" 的正确语法是: "<syntax>"。

原因: 输入的命令的语法错误。

操作: 遵循消息中描述的语法。

DMB0061E "<keyword>" 后面的表名 "<name>" 无效。

原因: 该命令期望表名。

操作: 遵循命令的语法，并再试。

DMB0062E "<keyword>" 后面的列名 "<name>" 无效。

原因: 该命令期望列名。

操作: 遵循命令的语法，并再试。

DMB0064E 系统不识别 "<keyword>" 后面的标记 "<token>"。

原因: 该命令需要的是除消息中指示的标记之外的其它信息。

操作: 遵循命令的语法，并再试。

DMB0065E "<keyword>" 后面的用户标识 "<identifier>" 无效。

原因: 该命令期望有效的用户标识。

操作: 验证想要的用户标识，并再试。

DMB0066E "<keyword>" 后面的密码 "<password>" 无效。

原因: 该命令需要的是有效的用户标识，而不是消息中指示的标记。

操作: 验证密码，并再试。

DMB0067E 输入的命令无效。

原因: 命令名拼写错误，或语法错误。

操作: 遵循命令的语法，并再试。

DMB0068E 元数据表不存在。

原因: 该功能尝试使用应对数据对象存在的管理支持（元数据）表。该元数据表可能已损坏或被擦除。

操作: 检查名称，并验证元数据表是否存在。若元数据表被意外擦除或损坏，则禁用而后重新启用数据对象。

DMB0069E DBname "<name>" 无效。

原因: 不存在具有该名称的数据库。

操作: 检查名称，并验证数据库是否存在。

DMB0070E 句柄无效。

原因: 应用程序中传送的句柄值可能已毁坏。

操作: 验证应用程序以确保未更改 Extender 句柄值。

DMB0071E 不能连接 ″<database-name>″。

原因: 该数据库的 DB2 Extender 服务器可能未启动。

操作: 检查服务器的状态。若该服务器不在运行，则在 DMB 命令行使用 START SERVER 命令重新启动它。

DMB0072E UDF SQL 服务器不能从 DB 断开。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0073E USE 会话句柄无效。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0074E USE 语句句柄无效。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0075E 指定文件名。

原因: 该操作需要媒体文件名。

操作: 输入媒体文件的名称。

DMB0076E 打不开导入文件。

原因: 导入文件已丢失或已损坏。

操作: 验证导入文件的名称和存在情况。

DMB0077E 不能打开 / 读取内容文件。

原因: Extender 句柄指向不存在或已毁坏的文件。Extender 变得不可存取该文件。

操作: 使用 FILENAME UDF 来查找文件的名称，或验证该内容文件是否存在。

DMB0078E 不能创建导出文件。

原因: 导出文件已丢失或已毁坏。

操作: 验证导出文件的名称和存在情况。

DMB0079E 不能将 BLOB 复制至文件。

原因: 该文件不能接受 BLOB。可能没有足够的存储空间来容纳 BLOB。

操作: 将 BLOB 的大小与可用的存储量相比较，必要时增大存储量。

DMB0080E 不能写文件。

原因: 该文件已损坏或不存在，或名称拼写不正确。

操作: 验证文件的名称和存在情况。

DMB0081E 偏移或大小无效。

原因: 该操作在数据结构中找不到期望的数据。字段的大小或偏移不正确。

操作: 验证偏移和字段的大小。

DMB0082E 不能构建句柄。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0083E ″<Extender-name>″ 与 ″<Extender-name>″ 不兼容。

原因: 在此用法中，消息中指定的两个 Extender 不兼容。全查询或子查询的插入操作无效。

操作: 确保对其启用目标对象的 Extender 就是对其启用源对象的 Extender。

DMB0084E 导入请求无效: 文件名、内容、存储器类型。

原因: 导入操作失败。文件名、内容或存储器类型无效。

操作: 检查数据，并再试。

DMB0085E 更新请求无效: 文件名、内容、存储器类型。

原因: 更新操作失败。文件名、内容或存储器类型无效。

操作: 检查数据，并再试。

DMB0086E 请求的大小太大。

原因: 请求的大小大于 UDF 的最大 blob 大小。

操作: 请求较小的大小。

DMB0087E 文件名无效。

原因: 没有具有该名称的文件。

操作: 验证文件的名称和存在情况。

DMB0088E 句柄值是 NULL。

原因: UDF 期望非空句柄。

操作: 确保应用程序获取有效的句柄，并将其传送给 UDF。

DMB0089E 句柄值不存在。

原因: 传送给 UDF 的句柄无效。

操作: 确保应用程序传送有效的句柄。

DMB0090E 数据截断。

原因: 文件或缓冲区太小，不能接受数据。

操作: 增大文件或缓冲区的大小。

DMB0091W 内容已在文件中。

原因: 文件已有内容。该内容将被覆盖。

操作: 不需要任何操作。

DMB0092E 对列 ″<column-name>″ 尝试的插入操作无效。已对 ″<Extender-name>″ Extender 启用了该列。

原因: 正在插入的数据的数据类型与对其启用该列的 Extender 的数据类型不同。

操作: 确保对其启用目标对象的 Extender 就是对其启用源对象的 Extender。

DMB0093E 对列 ″<column-name>″ 尝试的更新操作无效。已对 ″<Extender-name>″ Extender 启用了该列。

原因: 正在更新的数据的数据类型与对其启用该列的 Extender 的数据类型不同。

操作: 确保对其启用目标对象的 Extender 就是对其启用源对象的 Extender。

DMB0094I 表 ″<table-name>″ 不存在。

原因: 系统找不到具有该名称的表。它可能存在于另一数据库中。

操作: 不需要任何操作。

DMB0095W 未对 ″<Extender-name>″ Extender 启用表 ″<table-name>″。

原因: 未对该 Extender 启用该表。

操作: 不需要任何操作。

DMB0096W 未对 ″<Extender-name>″ Extender 启用表 ″<table-name>″ 中的列 ″<column-name>″。

原因: 系统应启用该列。

操作: 不需要任何操作。

DMB0097W 未对 ″<Extender-name>″ Extender 启用当前数据库。

原因: 系统期望启用该数据库。

操作: 对消息中指示的 Extender 启用该数据库。

DMB0098E 用户对表 ″<table-name>″ 不具有 ″<authority-level>″ 权限。

原因: 该操作所需的权限级别比进行尝试的用户标识的权限级别高。

操作: 从拥有该表的用户标识执行该操作，或请数据库管理员更改您的当前用户标识的权限级别。

DMB0099E 不能落实事务。

原因: 当前数据库的 Extender 服务器可能已停止。

操作: 检查服务器的状态。若服务器不在运行，则在 db2ext 命令行使用 START SERVER 命令重新启动它。

DMB0100E ″<name>″ 不是有效的表名。

原因: 不存在具有该名称的表。

操作: 验证表的名称和存在情况，并再试。

DMB0101E 无效的 NULL 参数。

原因: 命令期望非空参数。

操作: 检查语法并再试。

DMB0102E 无效的存储类型。

原因: 对于 DB2 Extender，存储类型指示在何处存储介质数据。

操作: 指定 0 来指示外部（在文件中），指定 1 来指示内部（在数据库中）。

DMB0103E 不受支持的格式。

原因: DB2 Extender 不支持此对象的格式。

操作: 将此对象转换为受支持的格式。

DMB0104E 视频内容缓冲区太小。

原因: 视频剪辑对于分配给它的缓冲区来说太大。

操作: 分配更大的缓冲区。

DMB0105E MPEG1 头文件无效。

原因: MPEG1 文件头丢失或损坏。

操作: 验证该文件是否是 MPEG1 文件。

DMB0106E AVI 头文件无效。

原因: AVI 文件头丢失或损坏。

操作: 验证该文件是否是 AVI 文件。

DMB0107E 未设置导出环境。

原因: 在 DB2 Extender 中，未正确设置导出环境的环境变量。

操作: 确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0108E 未设置导入环境。

原因: 在 DB2 Extender 中，未正确设置导入环境的环境变量。

操作: 确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0109E 不能解析导入文件。

原因: 没有具有该名称的导入文件。

操作: 验证该文件的名称和存在情况，并确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0110E 不能解析导出文件。

原因: 没有具有该名称的导出文件。

操作: 验证该文件的名称和存在情况，并确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0111E 未设置存储环境。

原因: 未正确设置存储环境的环境变量。

操作: 确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0112E 不能解析存储文件。

原因: 没有具有该名称的存储文件。

操作: 验证该文件的名称和存在情况，并确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0113E 打不开导入文件。

原因: 该文件可能被别人锁定，或该文件丢失或损坏。

操作: 验证该文件的名称、存在情况和状态，以及您的权限级别。

DMB0114E 打不开导出文件。

原因: 该文件可能被别人锁定，或该文件丢失或损坏。

操作: 验证该文件的名称、存在情况和状态，以及您的权限级别。

DMB0115E 打不开存储文件。

原因: 系统尝试写文件，但该文件已存在。服务器无权覆盖该文件。

操作: 验证该文件的名称、存在情况和状态，以及您的权限级别。

DMB0116E 不能创建临时文件。

原因: 可能没有足够的存储空间来创建临时文件。

操作: 确保正确设置了 Extender 的临时环境变量。必要时增大存储量。

DMB0117E 未设置临时环境。

原因: 未正确设置临时环境的环境变量。

操作: 确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0118E 打不开临时文件。

原因: 该临时文件可能已被覆盖或损坏。

操作: 确保环境变量设置正确，如第 475 页的附录 A，『设置 DB2 Extender 的环境变量』中所述。

DMB0119I dmbsrv 服务器为 ″<name>″ 启动了 ″<count>″ 个连接。

原因: 此消息指示服务器启动时进行了多少个连接。

操作: 不需要任何操作。

DMB0120E dmbsrv 服务器未能为 ″<name>″ 启动 ″<count>″ 个连接。

原因: DB2 可能未启动，或数据库可能不存在，或系统可能用完了许可连接。

操作: 确保 DB2 已启动且数据库存在。若问题仍然存在，与 IBM 联系以获取更多许可证。

DMB0121I dmbsrv 服务器正在为 ″<name>″ 启动 ″<count>″ 个连接。

原因: 此消息指示服务器启动时进行了多少个连接。

操作: 不需要任何操作。

DMB0122I dmbssd 服务器已就绪。

原因: 该服务器已准备好运行应用程序。

操作: 不需要任何操作。

DMB0129E 无效操作: ″<operation-name>″。

原因: 不存在具有该名称的命令或 API。

操作: 验证命令或 API，并再试。

DMB0130E 列 ″<column-name>″ 未能与 SQL 语句相连。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0131E SQL 准备语句失败。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0132E SQL 设置参数失败。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0133E SQL 执行语句失败。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0134E 文件格式转换失败。

原因: 格式转换不支持存储的多媒体数据的格式。

操作: 不能转换此文件的格式。

DMB0135E 不能打开 / 读缩略图。

原因: 缩略图文件可能已丢失或毁坏。

操作: 验证缩略图文件的名称、存在情况和完整性。

DMB0136E 找不到绑定文件。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0137E 不能连接 DB ″<database-name>″

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0138E 不能释放 SQL 语句。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0139E ″<keyword>″ 后面的特征名 ″<name>″ 无效。

原因: Image Extender 在此命令中期望有效的特征名。

操作: 用有效的特征名再试该命令。有效的特征名包括:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

DMB0141E ″<keyword>″ 后面的限定符 ″<identifier>″ 无效。

原因: 系统不能标识命令中的限定符。

操作: 检查限定符，并再试。

DMB0142E 未打开目录。

原因: 在 DB2 Extender 中，当前命令需要打开 QBIC 目录。

操作: 用 OPEN QBIC CATALOG 命令打开 QBIC 目录。

DMB0143I 在表 ″<table-name>″ 的列 ″<column-name>″ 中的 QBIC 目录中，已将自动编目设置为 ″<status>″。有 ″<count>″ 个特征:

原因: 此消息指示在特定图像列的 QBIC 目录中定义的特征数，以及是否打开了自动编目。

操作: 不需要任何操作。

DMB0145E 查询句柄无效。

原因: API 调用中使用的查询句柄无效。

操作: 检查应用程序，查看是否获取了正确的查询句柄。

DMB0146E 特征类名 ″<feature-class>″ 无效。

原因: 没有具有该名称的特征类。有效的特征名包括:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

操作: 更正特征的名称，并再试。

DMB0147E 特征类名 ″<feature-class>″ 丢失或无效。

原因: 有效的特征名包括:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

操作: 更正特征的名称，并再试。

DMB0148E 特征 ″<feature-name>″ 已是查询的一个成员。

原因: 查询已支持消息中指示的特征。

操作: 不需要任何操作。

DMB0149E 特征 ″<feature-name>″ 不是查询的成员。

原因: 该查询未包括指定的特征名。

操作: 要在调用存取特征的其它 API 之前将该特征添加至查询，使用 QbQueryAddFeature API。

DMB0150E 系统不能分配内存。

原因: 系统无法分配支持尝试的操作所需的内存。

操作: 验证系统是否有足够的内存来完成该操作。

DMB0151E 指向返回值的指针是 NULL。

原因: 因为指向返回值的指针一定不能是 NULL，所以 API 调用未成功完成。

操作: 确保向 API 调用提供有效的参数，并正确地遵循语法。

DMB0152E 指向列表返回值的指针是 NULL。

原因: 因为指向返回值的指针一定不能是 NULL，所以 API 调用未成功完成。

操作: 确保向 API 调用提供有效的参数，并正确地遵循语法。

DMB0153E 作用域参数已保留，且必须是 0。

原因: 该参数留给将来使用。

操作: 将作用域设置为 0。

DMB0154E 指向特征类名的指针无效。

原因: API 调用期望指向输入特征类名的有效指针。

操作: 确保向 API 调用提供有效的参数，并正确地遵循语法。

DMB0155I 已将缓冲区大小零传送给 ″<function-name>″ 函数。

原因: API 调用需要缓冲区返回信息。

操作: 不需要任何操作。

DMB0156E QbImageSource 指针是 NULL。

原因: NULL 空值指示不应更改结构。

操作: 不需要任何操作。

DMB0157E QbImageSource 类型 ″<type>″ 无效。

原因: 此 DB2 Extender API 引用的数据结构的数据类型有错。

操作: 该结构的数据类型应是 QbImageSource。

DMB0159E 指向 QbImageSource 图像缓冲区的指针是 NULL。

原因: API 调用期望返回指针。

操作: 检查应用程序，查看是否正确指定了 API 调用和缓冲区。

DMB0160I 图像缓冲区或文件的长度是零。

原因: 长度是零。

操作: 不需要任何操作。

DMB0161E 指向表和／或列名的指针是 NULL。

原因: API 调用期望提供指针。

操作: 检查应用程序，查看是否正确指定了 API 调用的输入。

DMB0162I 将 requestedHits 设置为零。

原因: 当 requestedHits 设置为零时，不能返回任何信息。

操作: 不需要任何操作。

DMB0163I 该函数尚不受支持。

原因: 该函数尚不受支持。

操作: 不需要任何操作。

DMB0164E 系统不能处理查询（<query-name>）。

原因: 创建查询时出错。

操作: 检查命令或 API 的输入，并再试。

DMB0165E 系统不能运行查询（<query-name>）。

原因: 创建查询时出错。

操作: 检查命令或 API 的输入，并再试。

DMB0166E 执行 ″<name>″ 时，在 ″<name>″ 中找到语句错误: ″<error>″

原因: 发生内部 IBM 错误。

操作: 请联系数据库管理员。

DMB0167E 读 QbGenericImageDataClass（<error>）时出错。

原因: 发生内部 IBM 错误。

操作: 请联系数据库管理员。

DMB0168E 在搜索之前，未设置查询的特征 ″<feature-name>″。

原因: 因为未对查询指定任何特征，所以它未运行。

操作: 使用 QbAddFeature API 或 ADD QBIC FEATURE 命令将特征添加至查询。

DMB0169E “调用层接口”中发生以下错误: ″<error>″。

原因: CLI 错误。

操作: 遵循消息文本中的指示信息。

DMB0170E 查询名 ″<query-name>″ 已在使用。

原因: 存在具有该名称的另一查询。

操作: 选择另一名称。

DMB0171E 未存储查询名 ″<query-name>″。

原因: 在创建查询之后，系统未能存储它。

操作: 确保您具有写权限，且有足够的存储量来存储查询。

DMB0172E SQL 错误: ″<error>″。

原因: 发生了内部错误。

操作: 遵循相关错误消息中包含的指示信息，并重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0173E 该目录已打开，但是只读的: ″<catalog-name>″。

原因: 因为别人已以写方式打开了目录，或您对其不具有写权限，所以不能更新该目录。

操作: 等到其他用户完成，从另一用户标识运行应用程序，或请数据库管理员更改您的当前用户标识的权限级别。

DMB0174E 发生系统错误: ″<error>″。

原因: 发生内部 IBM 错误。

操作: 遵循相关错误消息中包含的指示信息，并重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0175I 找不到图像: ″<information>″。

原因: 找不到与查询相匹配的图像。数据库可能是空的。

操作: 不需要任何操作。

DMB0176I 列已有 QBIC 目录: ″<table-name column-name>″。

原因: 存在具有该名称的另一目录。

操作: 不需要任何操作。

DMB0177E 系统打不开目录; 错误消息是 ″<error>″。

原因: 目录已损坏。

操作: 遵循消息文本中的指示信息。

DMB0178E 系统不能删除目录; 错误消息是: ″<error>″。

原因: 该目录不存在, 或它已损坏。

操作: 验证目录的名称和存在情况, 并再试。

DMB0179E 目录句柄无效: ″<error>″。

原因: API 调用中使用的目录句柄无效。

操作: 检查应用程序, 查看是否获取了正确的目录句柄。

DMB0180I 对目录的存取被拒绝: ″<error>″。

原因: 存取被拒绝。

操作: 不需要任何操作。

DMB0181I 目录在使用中, ″<error>″。

原因: 另一操作正在使用此目录。

操作: 不需要任何操作。

DMB0184I 已启动跟踪:

原因: 已启动跟踪。

操作: 不需要任何操作。

DMB0185I 尚未启动跟踪。

原因: 尚未启动跟踪。

操作: 不需要任何操作。

DMB0186I 在 ″<time>″, 已从目录 ″<directory-name>″ 中打开了跟踪。跟踪文件是 ″<file-name>″。已写 ″<Count> 个字节的跟踪数据。

原因: 跟踪已打开。

操作: 不需要任何操作。

DMB0187E 因为系统不能为写操作打开文件 ″<file-name>″, 所以不能建立通信。

原因: 您不是环境变量 DB2INSTANCE 描述的当前实例的所有者, 或环境变量 (如 DB2MMTOP) 设置不正确。

操作: 用拥有实例的用户标识注册。验证环境变量是否设置正确。

DMB0188I 创建跟踪守护程序时出错: ″<error>″

原因: 发生了内部错误。

操作: 重复该操作。若再次出错, 与 IBM 服务中心联系。

DMB0189I 已成功启动跟踪:

原因: 已启动跟踪。

操作: 不需要任何操作。

DMB0190E 不能启动跟踪。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错, 与 IBM 服务中心联系。

DMB0191E 需要设置环境变量 ″<name>″。

原因: 系统配置不正确。

操作: 设置该变量, 并再试。

DMB0192I 跟踪已成功关闭。

原因: 跟踪已关闭。

操作: 不需要任何操作。

DMB0193E 系统不能写文件 ″<file-name>″。

原因: 您对指定文件的目录不具有写权限。

操作: 请与数据库管理员联系以获取权限。

DMB0194E 系统不能读文件 ″<file-name>″。

原因: 该文件不存在，或您对该文件不具有读权限。

操作: 确保该文件存在，且您对该文件具有读权限。

DMB0198E 输入文件中的跟踪码 ″<code>″ 未知。输入文件可能已毁坏。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0199E 您对任何引用表都不具有 ″<authority-level>″ 权限。

原因: 您的用户标识不具有该操作所需的权限级别。

操作: 从另一用户标识执行该操作，或请数据库管理员更改您的当前用户标识的权限级别。

DMB0200W 您对最少一个引用表不具有 ″<authority-level>″ 权限。

原因: 您的用户标识不具有某些表所需的权限级别。

若正在列示引用的文件，则列示的文件是您对其具有“选择”权限的表所引用的文件。若系统上有您对其不具有“选择”权限的表，则不列示它们引用的文件。

若正在重组元数据，则系统仅重组您对其具有“控制”权限的表的元数据。

操作: 要获取所有文件，从另一用户标识执行该操作，或请数据库管理员更改您的当前用户标识的权限级别。

DMB0201I 已存在具有该名称的特征: ″<feature-name>″。

原因: QBIC 目录中已包括具有该名称的特征。

操作: 不需要任何操作。

DMB0202E 特征名无效: ″<feature-name>″。

原因: 没有具有该名称的特征类。有效的特征名包括:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

操作: 更正特征的名称，并再试。

DMB0203E 特征找不到: ″<feature-name>″。

原因: 没有具有该名称的特征类，或 QBIC 目录中未包括该特征类。有效的特征名包括:

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

操作: 更正特征的名称，并再试。

DMB0204E 未对 DB2IMAGE 启用列: ″<column-name>″。

原因: 未对 Image Extender 启用该列。

操作: 确保已对 Image Extender 启用了该列。

DMB0205E 找不到 ″<table-name column-name>″ 和目录。

原因: 没有与指定列相关联的 QBIC 目录。

操作: 在执行任何其它 QBIC 操作之前，创建该列的 QBIC 目录。

DMB0206W 未对任何 Extender 启用指定的列。

原因: 该列可能不存在，或其数据类型可能与 Extender 不兼容。

操作: 验证是否已使用正确的数据类型定义了该列。

DMB0207E 不能覆盖文件。

原因: 该文件已存在，但 EXPORT UDF 不能覆盖它。

操作: 将该文件调至另一文件名，或允许 EXPORT UDF 覆盖该文件。

DMB0208E sqlcode=<code> clistate=<code>。

原因: 发生了内部错误。

操作: 重复该操作。若再次出错，与 IBM 服务中心联系。

DMB0209E 无效的音频头文件。

原因: 音频文件头丢失或损坏。

操作: 验证 DB2 Extender 是否支持该音频文件的格式。

DMB0211W 文件存在，未被覆盖。

原因: 指定的目标文件已存在，未被覆盖。

操作: 不需要任何操作。

DMB0212E resultType 参数已保留，且必须是 0。

原因: 该参数留给将来使用。

操作: 将 resultType 设置为 0。

DMB0214E 指向查询名的指针无效。

原因: API 调用期望指向输入查询名的有效指针。

操作: 确保向 API 调用提供有效的参数，并正确地遵循语法。

DMB0352E 未初始化命令行环境。

原因: 未初始化命令行环境，不能运行 db2ext 命令行处理器。（此消息仅适用于 Windows NT 和 Windows 95 环境。）

操作: 发出 db2cmd 命令以打开 DB2CLP 窗口，然后发出 db2ext 命令以在该窗口中运行 db2 命令行处理器。

DMB0353E 不能与 db2ext 命令行处理器的后台处理通信。

原因: db2ext 命令行处理器的后台处理正在运行，但 db2ext 命令行处理器不能与其通信。

操作: 尝试在另一窗口中发出 db2ext 命令。

DMB0354E 不能启动 db2ext 命令行处理器的后台处理。

原因: db2ext 命令行处理器的后台处理正在运行，但 db2ext 命令行处理器不能与其通信。

操作: 检查后台处理的可执行模块（db2extb 或 db2extb.exe）是否存在，且其目录是否在 PATH 环境变量中。

DMB0355E db2ext 命令行处理器的后台处理超时。

原因: db2ext 命令行处理器的后台处理已成功启动，但 db2ext 命令行处理器不能在允许的时间限制内与其通信。

操作: 尝试在另一窗口中发出 db2ext 命令。

DMB0356E 不能与 db2ext 命令行处理器的后台处理通信。

原因: db2ext 命令行处理器将请求发送至其后台处理，但未接收到该请求。

操作: 确保 db2ext 命令行处理器的后台处理仍在运行。

DMB0357E db2ext 命令行处理器的后台处理不响应。

原因: db2ext 命令行处理器将请求发送至其后台处理，但后台处理未在允许的时间限制内作出响应。

操作: 确保 db2ext 命令行处理器的后台处理仍在运行。

DMB0359E 未在超时期内创建 db2ext 命令行处理器后台处理请求队列或输入队列。

原因: db2ext 命令行处理器的后台处理未在允许的时间限制内创建信息队列。（此消息仅适用于 UNIX 环境。）

操作: 确保 DB2 实例主目录所驻留的磁盘未满（后台处理需要此主目录来创建信息队列的文件）。若该磁盘未满，检查是否启动了太多的 db2extb 进程。若正在许多窗口中运行 db2ext 命令行处理器，则这是有可能的。首次以命令方式发出 db2ext 命令行处理器请求时，在窗口中启动后台处理。当不再需要 db2ext 命令行处理器时，务必发出命令 `db2ext terminate` 结束它。仅当发出 `terminate` 命令时，才删除后端处理的信息队列。

DMB0361E 列或表未启用。

原因: 指定了IMPORT UDF，但未对 Extender 启用指定的表列。

操作: 启用该表列，并再试。

DMB0363E 丢失表和列名。

原因: 调用了UPDATE UDF，但未指定表。

操作: 指定表并再试。

DMB0364E 先前已对表空间 "<tablespace-name>" 定义了 Extender "<Extender-name>"。

原因: 已使用与指定的表空间不同的表空间对 Extender 启用了指定的数据库、表或列。

操作: 检查表空间规范是否正确。

DMB0365E 您对 "<schema-name>"."<table-name>" 的 "<metadata-table-name>" 和 "<metadata-table-name >" 不具有 CONTROL 特权。

原因: 因为您对指定用户表的元数据表不具有必需的 CONTROL 特权，所以您的请求被拒绝。

操作: 请数据库管理员授予您元数据表的 CONTROL 特权。

DMB0366E 期望特征名。

原因: 查询字符串中期望特征名。

操作: 更正查询字符串，并再试。

DMB0367E 期望颜色|颜色直方图|文件。

原因: 查询字符串中期望“颜色”、“直方图”或“文件”。

操作: 更正查询字符串，并再试。

DMB0368E 期望 ', '。

原因: 查询字符串中期望 ', '。

操作: 更正查询字符串，并再试。

DMB0369E 文件无效。

原因: 查询字符串中指定的文件无效。

操作: 更正查询字符串，并再试。

DMB0370E 期望文件名。

原因: 查询字符串中期望文件名。

操作: 更正查询字符串，并再试。

DMB0371E 期望服务器 / 客户机。

原因: 查询字符串中期望“服务器”或“客户机”。

操作: 更正查询字符串，并再试。

DMB0372E 期望 '('。

原因: 查询字符串中期望 '('。

操作: 更正查询字符串，并再试。

DMB0373E 期望 ')'。

原因: 查询字符串中期望 ')'。

操作: 更正查询字符串，并再试。

DMB0374E 期望百分比。

原因: 查询字符串中期望百分比值。

操作: 更正查询字符串，并再试。

DMB0375E 期望颜色。

原因: 查询字符串中期望红色、绿色和蓝色值。

操作: 更正查询字符串，并再试。

DMB0376E 期望 '='。

原因: 查询字符串中期望 '='。

操作: 更正查询字符串，并再试。

DMB0377E 期望 '<'。

原因: 查询字符串中期望 '<'。

操作: 更正查询字符串，并再试。

DMB0378E 期望 '>'。

原因: 查询字符串中期望 '>'。

操作: 更正查询字符串，并再试。

DMB0379E 期望 'and'。

原因: 查询字符串中期望 'and'。

操作: 更正查询字符串，并再试。

DMB0380E 期望权重。

原因: 查询字符串中期望权重。

操作: 更正查询字符串，并再试。

DMB0381E 未设置特征。

原因: 未将特征添加至 QBIC 目录。

操作: 将特征添加至 QBIC 目录，并重新编目图像。

DMB0382E 未能构建查询。

原因: 当前数据库的 Extender 服务器可能已停止。

操作: 检查服务器的状态。若服务器不在运行，则在 db2ext 命令行使用 START SERVER 命令重新启动它。

DMB0383E 未能执行查询。

原因: 当前数据库的 Extender 服务器可能已停止。

操作: 检查服务器的状态。若服务器不在运行，则在 db2ext 命令行使用 START SERVER 命令重新启动它。

DMB0384E 未能获取下一项。

原因: 已到达列表末尾。

操作: 确保应用程序不尝试检索列表末尾后面的项。

DMB0386E 不能与用户数据连用

原因: SQL API sqluihsh() 已返回非零的返回码。

操作: 重试。若问题仍然存在，与 IBM 支持联系。

DMB0387E 指定表空间的节点组与用户表的节点组不同。

原因: 为了启用表而作为输入传送的一个或多个表空间（即，元数据表、索引或 BLOB 的表空间）是在与定义用户表的节点组不同的节点组上定义的。

操作: 将启用在与用户表相同的节点组上定义的表空间。

DMB0388E 未在同一节点组上定义正规、长型或索引表空间。

原因: 为了启用数据库而作为输入传送的一个或多个表空间（即，元数据表、索引或 BLOB 的表空间）是在与其它表空间不同的节点组上定义的。

操作: 使用在同一节点组上定义的表空间。

DMB0389W 指定的表空间的节点组不包括所有分区服务器。

原因: 作为输入传送的表空间是在不包括所有分区服务器的节点组上定义的。

操作: 不需要任何操作。但是，若表空间是在涉及所有分区服务器的节点组上定义的，则IMPORT 和 UPDATE UDF 将会更有效执行。若 Extender 应用程序以 BLOB 格式存储介质内容，则这特别显著。

DMB0391I 仅当 DB2 UDB 客户机正在访问 DB2 UDB 服务器时，才可运行此命令。

原因: db2ext 命令行处理器未与 DB2 UDB 服务器相连，或 DB2 UDB 客户机未启动 db2ext 命令行处理器。例如，仅当 db2ext 命令行处理器与 “DB2 非扩充企业版”服务器相连时，命令 START SERVER 才有效。

操作: 不要在当前客户机／服务器配置中发出此命令。

DMB0392I 仅当 DB2 UDB 客户机正在访问“DB2 UDB 扩充企业版”服务器时，才可运行该命令。例如，仅当 db2ext 命令行处理器与“DB2 扩充企业版”服务器相连时，命令 DISCONNECT SERVER 才有效。

原因: db2ext 命令行处理器未与 “DB2 UDB 扩充企业版”服务器相连，或未从 DB2 UDB 客户机启动 db2ext 命令行处理器。

操作: 不要在当前客户机／服务器配置中发出此命令。

DMB0402E 仅当应用程序与 DB2 ″<server-type>″ 服务器相连时，命令 ″<command-name>″ 的选项 ″<option-name>″ 才有效。

原因: 因为 db2ext 命令行处理器未与支持该选项的服务器类型相连，所以指定的参数无效。例如，仅当 db2ext 命令行处理器与 “DB2 扩充企业版” 服务器相连时，才可指定带参数 NODENUM <nodenum> 的命令 GET SERVER STATUS。

操作: 不要在当前客户机／服务器配置中发出此命令参数。

DMB0411E 无效的基端口

原因: 实例创建期间，输入了无效的 TCP/IP 端口号作为基端口。

操作: 正确的语法是 dmbicrt -r:base_port,end_port -t:base_port,end_port。更正参数，并重试该命令。

DMB0412E 无效的结束端口

原因: 实例创建期间，输入了不正确的 TCP/IP 端口号作为结束端口。

操作: 正确的语法是 dmbicrt -r:base_port,end_port -t:base_port,end_port。更正参数，并重试该命令。

DMB0413E 无法解析 DB2 Extender 安装路径。

原因: 实例创建程序找不到环境变量 ″DMBPATH″ 的值。

操作: 设置变量 ″DMBPATH″，并重试该应用程序。

DMB0414E 无法解析计算机主机名。

原因: 尝试解析计算机的名称时遇到内部错误。

操作: 与 IBM 支持联系。

DMB0415E 无法解析此机器的节点号。

原因: 文件 ″db2nodes.cfg″ 中未列示正在其上运行实例创建的机器。

操作: 将该机器添加至 ″db2nodes.cfg″，并重试该应用程序。

DMB0416E 必须由 root 用户启动此程序。无法继续。

原因: 正在其下运行程序的用户标识不具有 root 用户权限。

操作: 作为 root 用户注册，并重试该应用程序。

DMB0417E 必须由具有管理员权限的用户执行此程序。无法继续。

原因: 在其下正在运行程序的用户标识不具有管理员权限。

操作: 用具有管理权限的用户标识注册，并重试该应用程序。

DMB0418E 无法获取关于用户的信息: "<userid>"。

原因: 尝试获取与正在创建的实例相关联的用户信息时发生内部错误。

操作: 确保有与正在创建的实例同名的有效用户标识，并重试该应用程序。

DMB0419E 无法创建 AIV Extender 目录 "<directory_name>。返回码 = <code>

原因: 尝试创建指定目录时出错。返回码表示从操作系统返回的错误。

操作: 确保在目录名中指定的文件系统 / 驱动器存在，且该许可权允许创建目录。

DMB0420E 无法为 AIV Extender 目录 "<directory_name>" 创建链接。返回码 = <code>

原因: 尝试创建指定的符号链接时出错。返回码表示从操作系统返回的错误。

操作: 确保在目录名中指定的文件系统 / 驱动器存在，且该许可权允许创建链接。

DMB0421E 打不开文件: "<file_name>"。返回码 = <code>

原因: 尝试打开指定文件时出错。返回码表示从操作系统返回的错误。

操作: 确保该文件存在，且许可权允许打开文件。

DMB0422E 无法写入文件: "<file_name>"。返回码 = <code>

原因: 尝试写入指定文件时出错。返回码表示从操作系统返回的错误。

操作: 确保该文件存在，且许可权允许写文件。

DMB0424E 找不到 "db2nodes.cfg" 文件。

原因: 未能找到 DB2 文件 "db2nodes.cfg"。

操作: 确保已安装正确版本的 "DB2 UDB 扩充企业版"，并重试该应用程序。

DMB0426E 错误: "<error_code>" 打开键 "<registry_key>"。

原因: 尝试打开指定的注册表键时出错。

操作: 记录返回码，并与 IBM 支持联系。

DMB0427E 概要文件注册表中未设置变量 "<variable>"。

原因: 未在 Windows NT 注册表中找到指定的值。

操作: 确保指定了有效的 DB2 Extender 变量的名称。

DMB0430E 找不到 DB2 注册表值

原因: 找不到 DB2 使用的注册表值。

操作: 确保已安装正确版本的 "DB2 UDB 扩充企业版"，并重试该应用程序。

DMB0431E 无法创建 Extender 注册表键: "<registry_key>"。

原因: 尝试创建 Extender 注册表键时发生内部错误。

操作: 与 IBM 支持联系。

DMB0432E 无法设置 Extender 注册表键的值: "<registry_key>"。

原因: 尝试设置 Extender 注册表键值时发生内部错误。

操作: 与 IBM 支持联系。

DMB0435E 无法存取控制文件 "<control_file>"。

原因: 找不到指定的控制文件。

操作: 与 IBM 支持联系。

DMB0443E 无法打开目录 "<directory_name>"。返回 = <code>

原因: 尝试打开指定的目录时出错。返回码表示从操作系统返回的错误。

操作: 确保在目录名中指定的文件系统 / 驱动器存在，且该许可权允许打开目录。

DMB0449W -q:datapath 是 DB2 Extender 实例创建所必需的。

原因: 尝试创建 DB2 Extender 实例时，未指定 `-q` 参数。

操作: 指定该参数，并重试该应用程序。

DMB0450W 一个或多个指定的 ″<port>″ 端口已在使用中。

原因: 服务文件中已将指定为供 DB2 Extender 使用的端口列示为在使用中。

操作: 指定未在使用的一个或多个端口，并重试该应用程序。

DMB0452E 在 ″db2nodes.cfg″ 文件中找不到节点号 ″<node_num>″。

原因: 在 db2nodes.cfg 文件中找不到此机器的节点号。

操作: 将节点号添加至 db2nodes.cfg 文件，并重试该应用程序。

DMB0460W 无法确定 TCP/IP 端口是否可用。

原因: 尝试验证指定的 TCP/IP 端口是否已在使用中时出错。

操作: 确保服务文件中未将指定的端口列示为正由另一应用程序使用。

DMB0462E 无法初始化此节点。返回码 = <code>

原因: 尝试初始化当前节点时，Extender 启动遇到错误。

操作: 与 IBM 支持联系。

DMB0495E 此版本的 AIV Extender 不支持长名称。

原因: 您在调用 Extender 管理 API 或发出 db2ext 命令行命令时指定了长标识符。在此版本的 AIV Extender 中，支持的标识符的最大长度是:

- 本地授权标识（AUTHID）— 8 个字符
- 表模式（TABSCHEMA）— 8 个字符
- 表名（TABNAME）— 18 个字符
- 列名 — 18 个字符

检查 API 调用或命令，务必使用短标识符。

DMB0496E 指定了无效的表名或列名。

原因: 您在调用 Extender 管理 API 或发出 db2ext 命令行命令时指定了无效的标识符。最可能的原因是标识符名太长。查阅*快速入门*一书以了解有关 UDB Db2 中的名称长度的信息。

检查 API 调用或命令，务必使用短标识符。

DMB497E DB2MMDATAPATH 上的存取被拒绝。

原因: （仅限于 EEE）您指定了并非在所有节点上都可存取的目录或共享名。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取。检查您创建实例时指定的目录或共享名是否在所有节点上都存在并可存取。

DMB498E DB2MMDATAPATH 路径的最少一个部分不是目录。

原因: （仅限于 EEE）您指定的目录或共享名并不是节点上的目录。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取。检查您创建实例时指定的目录或共享名是否在所有节点上都存在并可存取。

DMB499E DB2MMDATAPATH 字符串太长。

原因: （仅限于 EEE）您指定的目录或共享名导致变量 DB2MMDATAPATH 太长。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取。检查您创建实例时指定的目录或共享名是否正确，且它是否在所有节点上都存在并可存取。

DMB500E DB2MMDATAPATH 目录不存在。

原因: （仅限于 EEE）您指定的目录或共享名在节点上不存在。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取。检查您创建实例时指定的目录或共享名是否在所有节点上都存在并可存取。

DMB501E DB2MMDATAPATH 上发生未知的 stat() 错误。

原因: （仅限于 EEE）尝试存取此环境变量中的目录或共享名时遇到问题。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取。检查您创建实例时指定的目录或共享名是否在所有节点上都存在并可存取。

DMB502E DB2MMDATAPATH 存在，但不是目录。

原因: （仅限于 EEE）您指定了一个目录或共享名的名称，但该名称不是所有节点上的目录或共享名的名称。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取。检查您创建实例时指定的目录或共享名是否在所有节点上都存在并可存取。

DMB503E DB2MMDATAPATH 存在，但不可读。

原因: （仅限于 **EEE**）您指定了目录或共享名，但不能在所有节点上从中读取。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取。检查您创建实例时指定的目录或共享名是否在所有节点上都存在并可存取。

DMB504E DB2MMDATAPATH 存在，但不可写。

原因: （仅限于 **EEE**）您指定了目录或共享名，但不能在所有节点上写入它们。创建 DB2 Extender 实例时指定的目录或共享名必须存在，且必须在所有节点上都可存取和读／写。检查您创建实例时指定的目录或共享名是否在所有节点上都存在并可存取。

DMB504E 未设置 DB2MMDATAPATH。

原因: （仅限于 **EEE**）创建 DB2 Extender 实例时，未设置环境变量 DB2MMDATAPATH。若这是新的 DB2 Extender 实例，则使用 DMBIDROP 删除该实例；然后重建它，并正确地指定 `-q` 选项。

若这不是新的 DB2 Extender 实例，则:

- 在 UNIX 环境中:
  1. 检查该目录是否正确，且是否在所有节点上都存在并可存取
  2. 将 *$INSTHOME*/dmb/dmbprofile 修改为将 DB2MMDATAPATH 作为目录导出。
- 在 Windows 环境中:
  1. 检查目录的共享名是否正确，且该目录是否在所有节点上都存在且可存取
  2. 在 IAV Extender 实例注册表中，添加注册表项 DB2MMDATAPATH，并将该共享名用作值。键为

     \\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2 Extenders

     \PROFILE*instance_name*\DB2MMDATAPATH。

DMB506E 未设置实例名。

原因: 当运行 DMBSTART 时，DB2INSTANCE 环境变量未设置。在用 DMBSTART 启动 DB2 Extender 之前，确保 DB2START 正确工作。

DMB507E dmbssd *name* node *arguments*

原因: 内部错误。与 IBM 代表联系。

DMB508E 节点号必须大于或等于 0。

原因: 内部错误。与 IBM 代表联系。

DMB509E 此程序不应手工启动。

原因: 内部错误。与 IBM 代表联系。

DMB512E 用法: *arguments* dmbInstName。

原因: 内部错误。与 IBM 代表联系。

DMB513E *Name* 不是有效实例名。

原因: 您在尝试删除 DB2 Extender 实例时指定的名称未被识别为实例名。检查您是否指定了正确的实例名，且是否存在具有该名称的目录 *$INSTHOME*/dmb

DMB514I 此实例上既未安装服务器也未安装客户机。

原因: 您尝试删除 DB2 Extender 实例，但尚未安装 Extender。进行检查，确保安装正确，且安装目录尚未被重命名。

DMB515I 未删除 DB2 实例。可以调用 DB2IDROP 来删除该 DB2 实例。

原因: 当删除 DB2 Extender 实例时，未删除相关联的 DB2 实例。使用 DB2IDROP 来删除该 DB2 实例。

DMB518E 预期不到的错误。函数 = *function name*, 返回码 = *return_code*。

原因: 当您尝试创建或删除 DB2 Extender 实例时，遇到预期不到的错误。检查安装和设置是否正确。

DMB520E 您不能作为 root 用户执行此程序。

原因: 检查您是否具有正确的权限来执行此操作。

DMB521E 更改 *name* 的许可权的尝试失败。

原因: 确保您有正确的权限来更改许可权。

DMB522E 更改 *name* 的所有权的尝试失败。

原因: 确保您有正确的权限来更改所有权。

DMB523E 更改 *name* 的组所有权的尝试失败。

原因: 确保您有正确的权限来更改组所有权。

DMB524E 文件或目录 *name* 已存在。

原因: 已存在具有您指定的名称的文件或目录。选择另一名称并重新运行该命令。

DMB525E 创建 *name* 的尝试失败。

原因: 检查您是否具有正确的权限来执行此操作。

DMB526E 文件或目录 *name* 丢失。

原因: 找不到指示的文件或目录。检查您是否指定了有效的文件或目录名。

DMB527E 将文件或或目录 *name* 复制至 *name* 的尝试失败。

原因: 检查您是否具有正确的权限来复制该文件或目录; 检查是否有足够的空间可用于复制。

DMB528E 用户标识 *user_ID* 无效。

原因: 您指定了无效的用户标识。检查该用户标识，并重新运行该命令。

DMB529E 用户标识 *user_ID* 的主组 *group* 无效。

原因: 您对该用户标识指定了无效的主组。检查主组是否正确，并重新运行该命令。

DMB530E 实例名 *name* 无效。

原因: 当尝试创建或使用实例时，您指定了无效的名称。检查实例名是否有效，并重新运行该命令。

DMB531E 操作系统 *name*, 版本 *version_number* 不受支持。

原因: 您尝试在不受支持的操作系统版本上运行此命令。检查执行此操作的需求。

DMB535E 不能存取指定的文件。

原因: 在运行此命令之前，检查您是否对该文件拥有存取权。

DMB0533E 分区数据库服务器环境不支持 API <*API name*>。

原因: 不能在分区数据库环境中使用指定的 API。

操作: 参考有关指定的 API 的章节以了解有关如何在应用程序中处理此函数的信息。

DMB0534E 不受支持的 UDF。

原因: 不能在分区数据库环境中使用该用户定义函数。

操作: 检查信息 SQL0443N，以确定是哪一个 UDF 遇到了问题。参考有关该 UDF 的章节以了解有关如何在应用程序中处理此函数的信息。

诊断跟踪

DB2 Extender 包括一个记录 Extender 服务器活动的跟踪设施。仅应在 IBM 服务人员的指导下使用跟踪设施。

跟踪设施在服务文件中记录关于各种事件的信息，如进入或退出 DB2 Extender 组件，或 DB2 Extender 组件返回错误代码。因为它记录许多事件的信息，所以仅应在必要时（例如调查错误情况时）才使用跟踪设施。此外，您应在使用跟踪功能时限制活动的应用程序数。限制活动应用程序数可使隔离问题原因变得容易。

使用 DMBTRC 命令来控制跟踪。可以在 AIX 服务器或 Windows NT 或更高版本服务器的命令行上发出此命令。您必须具有 SYSADM、SYSCTRL 或 SYSMINT 权限才能发出该命令。

使用 DMBTRC 命令来:

- 启动跟踪
- 停止跟踪
- 重新编排跟踪信息的格式，使它更容易阅读
- 显示跟踪状态

启动跟踪

可输入以下命令以启动跟踪:

dmbtrc on *path*

其中 *path* 是将包含跟踪信息的服务器文件的路径。

例如，以下命令启动跟踪:

```
dmbtrc on /tmp/trace.txt
```

停止跟踪

可输入以下命令以停止跟踪:

dmbtrc off

重新格式化跟踪信息

跟踪信息是以二进制格式记录的。可重新编排该信息的格式，并通过输入以下命令，使其有更好的可读性:

dmbtrc format *input_file output_file*

其中，*input_file* 是包含二进制格式的跟踪信息的文件，*output_file* 是将包含重新格式化后的信息的文件。*output_file*参数是可选的；如果不指定它，则重新格式化过的信息会显示在屏幕上。

例如，以下命令会重新格式化跟踪信息:

```
dmbtrc format /tmp/trace.txt /tmp/fmttrace.txt
```

显示跟踪状态

使用命令:

```
dmbtrc info
```

来显示下列跟踪状态信息:

- 跟踪设施打开或关闭
- 包含跟踪信息的文件的路径

跟踪

附录 A. 设置 DB2 Extender 的环境变量

DB2 Extender 提供了存储、检索或更新图像、音频或视频对象时如何指定文件名的灵活性。还可灵活地指定程序来显示或播放从数据库表中检索到的图像、音频和视频对象。

如何使用环境变量来解析文件名

虽然可对存储、检索和更新操作指定全限定文件名（即，后跟文件名的完整路径），但最好指定相对文件名。在 AIX、HP-UX 或 Solaris 中，相对文件名是任何不以斜杠开始的文件名；在 Windows 中，相对文件名是任何不以后跟冒号和反斜杠的驱动器盘符开始的文件名。

如果您指定相对文件名， Extender 将使用各种客户机和服务器环境变量中的目录规范来解析文件名。这允许在客户机／服务器环境中移动文件，而不必更改文件名。每次移动文件时，都必须要更改全限定文件名。

表 17 列示并描述了为了供 Image Extender、Audio Extender 和 Video Extender 解析文件名使用而可设置的环境变量。

表 17. DB2 Extender 的环境变量

Image Extender	Audio Extender	Video Extender	描述
		服务器环境变量	
DB2IMAGEPATH	DB2AUDIOPATH	DB2VIDEOPATH	用来解析源于服务器文件的存储、检索和更新操作的源文件名。
DB2IMAGESTORE	DB2AUDIOSTORE	DB2VIDEOSTORE	用来解析对服务器文件的存储和更新操作的目标文件名。
DB2IMAGEEXPORT	DB2AUDIOEXPORT	DB2VIDEOEXPORT	用来解析对服务器文件的检索操作的目标文件名。
DB2IMAGETEMP			用来解析创建临时服务器文件的操作的目标文件名。但是，若指定了 TMP 环境变量，则使用目录 TMP 来解析文件名。
		客户机环境变量	
DB2IMAGEPATH	DB2AUDIOPATH	DB2VIDEOPATH	用来解析对客户机文件的显示和播放操作的源文件名。
DB2IMAGETEMP	DB2AUDIOTEMP	DB2VIDEOTEMP	用来解析创建临时客户机文件的操作的目标文件名。但是，若指定了 TMP 环境变量，则使用目录 TMP 来解析文件名。

若不为特定的 Extender 设置适当的环境变量，则 Extender 将使用下列环境变量来解析文件名:

环境变量	描述
DB2MMPATH	用来解析存储、检索和更新操作的源文件名。
DB2MMSTORE	用来解析存储和更新操作的目标文件名。
DB2MMEXPORT	用来解析检索操作的目标文件名。

© Copyright IBM Corp. 1998, 2003

DB2MMTEMP	用来解析创建临时文件的操作的文件名。

如何使用环境变量来标识显示或播放程序

除解析文件名之外，环境变量还用来标识显示 Image Extender 检索到的图像对象，并播放 Audio Extender 和 Video Extender 检索到的音频或视频对象的程序。分别使用 DBiBrowse、DBaPlay 和 DBvPlay API 来显示或播放这些对象。使用每个 API 时，可指定显示或播放程序，或者，可指示要让缺省程序显示或播放对象。

DB2 Extender 使用客户机中的下列环境变量来标识缺省显示或播放程序:

环境变量	描述
DB2IMAGEBROWSER	用来标识缺省图像显示程序。
DB2AUDIOPLAYER	用来标识缺省音频播放程序。
DB2VIDEOPLAYER	用来标识缺省视频播放程序。

如何使用 DB2MMDATAPATH 环境变量（仅限于 EEE）

DB2 Extender 使用 DB2MMDATAPATH 环境变量来解析分区数据库环境中各种操作的位置。例如，DB2 Image Extender 使用 DB2MMDATAPATH 的值在分区数据库环境中存储 QBIC 数据。

在创建 DB2 Extender 实例时设置了 DB2MMDATAPATH，如第 9 页的『DMBICRT』和下列安装“自述”文件中所述:

- aixeee 目录中的 install.txt（在 AIX 中安装 DB2Extender 以与“DB2 扩充企业版”一起使用）
- soleee 目录中的 install.txt（在“Solaris 操作环境”中安装 DB2 Extender 以与“DB2 扩充企业版”一起使用）

如何使用 DB2MMDATAPATH 的一个示例是关于存储 QBIC 特征和索引数据的。在 UNIX 中，DB2 Image Extender 在以下目录中存储此 QBIC 数据:

db2mmdatapath /NODE*node_num*/QBIC/*database_name*

其中，*db2mmdatapath* 是 DB2MMDATAPATH 环境变量的值，*node_num* 是节点号，而 *database_name* 是数据库名。

考虑以下 AIX 示例。假设 DB2MMDATAPATH 设置为 /localfs/dmbdata。并假设名为 Sample 的数据库分区在节点 0、2 和 5 中。将在下列目录中存储 Sample 数据库的 QBIC 数据。

节点 0: /localfs/dmbdata/NODE0000/QBIC/sample

节点 2: /localfs/dmbdata/NODE0002/QBIC/sample

节点 5: /localfs/dmbdata/NODE0005/QBIC/sample

设置环境变量

可在 AIX、HP-UX、Solaris、OS/2 和 Windows 中设置环境变量。

在 AIX、HP-UX、Solaris 服务器和客户机中设置环境变量

在 AIX、HP-UX 和 Solaris 中，环境变量在 C shell、Korn shell 和 Bourne shell 脚本中指定。安装 DB2 Extender 时，服务器的环境变量设置如下:

C shell

```
setenv DB2MMPATH /usr/lpp/db2ext/samples:/tmp
setenv DB2MMTEMP /tmp
setenv DB2MMSTORE /tmp
setenv DB2MMEXPORT /tmp
```

Korn 和 Bourne shell

```
 DB2MMPATH=/usr/lpp/db2ext/samples:/tmp
export DB2MMPATH

DB2MMSTORE=/tmp
export DB2MMSTORE

DB2MMEXPORT=/tmp
export DB2MMEXPORT

DB2MMTEMP=/tmp
export DB2MMTEMP
```

服务器的环境变量最初设置的值允许存取 DB2 Extender 分发的样本程序中使用的媒体文件。（参见第 481 页的附录 B, 『样本程序和媒体文件』以获取关于样本程序和媒体文件的信息。）

当在 AIX、HP-UX 或 Solaris 客户机中安装 DB2 Extender 时，客户机环境变量设置如下:

C shell

```
setenv DB2MMPATH /tmp
setenv DB2MMTEMP /tmp
```

Korn 和 Bourne shell

```
DB2MMPATH=/tmp
export DB2MMPATH

DB2MMTEMP=/tmp
export DB2MMTEMP
```

设置用来解析文件名的服务器和客户机环境变量。指定适用于环境的值。可对以 PATH 结束的环境变量指定多个目录，并用定界符分隔各个目录。仅可用一个目录来设置以 STORE、EXPORT 和 TEMP 结束的环境变量。

在 DB2IMAGEBROWSER、DB2AUDIOPLAYER 和 DB2VIDEOPLAYER 环境变量中分别指定适当的图像显示、音频播放和视频播放程序的名称。

可更改环境变量的初始设置，如下所示:

C shell

使用 SETENV 命令来设置环境变量:

```
setenv env-var directory
```

例如:

```
setenv DB2MMPATH /usr/lpp/db2ext/samples:/media
setenv DB2IMAGEPATH /Employee/pictures:/images
setenv DB2AUDIOSTORE /Employee/Sounds
setenv DB2IMAGEBROWSER 'xv %s'
```

Bourne shell

使用 EXPORT 命令来设置环境变量:

```
env-var=directory
export env-var
```

例如:

```
 DB2MMPATH=/usr/lpp/db2ext/samples:/media
export DB2MMPATH

DB2IMAGEPATH=/Employee/pictures:/images
export DB2IMAGEPATH

DB2AUDIOSTORE=/Employee/Sounds
export DB2AUDIOSTORE
```

Korn shell

使用 EXPORT 命令来设置环境变量:

```
export env-var=directory
```

例如:

```
export DB2MMPATH=/usr/lpp/db2ext/samples:/media
export DB2IMAGEPATH=/Employee/pictures:/images
export DB2AUDIOSTORE=/Employee/Sounds
```

在 Windows 服务器和客户机中设置环境变量

在 Windows 中，如何设置环境变量取决于是在非分区环境还是在分区数据库环境中（即，配合 “DB2 扩充企业版 Windows 版” ）使用 DB2 Extender。

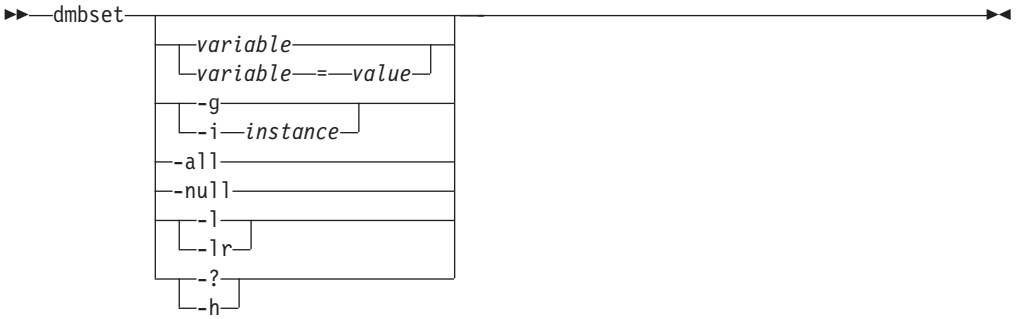
在 Windows 非分区数据库环境中设置环境变量（仅限于非 EEE）

在 Windows 中，环境变量存储在系统注册表中。可通过打开 Windows 控制面板并选择系统图标来设置变量。在 “系统属性” 对话中，选择 “环境” 选项卡。有两个包含环境变量及其值的窗口。上面的窗口显示对所有用户生效的变量。下面的窗口显示仅对当前用户生效的变量。

在 Windows 分区数据库环境中设置环境变量（仅限于 EEE）

在 Windows 分区环境中，DB2 Extender 使用的所有变量都存储在系统注册表的专用区域中。提供了称为 DMBSET 的程序来检查和更改 Extender 变量。

此程序的语法是:



要查询变量的值，输入 dmbset *变量名*。例如:

```
dmbset DB2MMPATH
```

要设置变量的值，输入: dmbset *变量名 = 值*。例如:

```
dmbset DB2MMPATH=C:\DMB\SAMPLES
```

要显示已定义实例的所有变量的值，输入 dmbset -i *实例名*。例如:

```
dmbset -i dmbinst1
```

要将值设置为空，输入 dmbset *变量名 -null*。例如:

```
dmbset DB2MMPATH -null
```

要显示所有实例使用的变量的值，输入: `dmbset -g`

要列示 DB2 Extender 使用的所有变量的名称，输入: `dmbset -lr`

要列示注册表中定义的所有实例概要文件的名称，输入: `dmbset -l`

在分区数据库环境中设置 DB2 Extender 的环境变量方面，您有很大的灵活性。例如，可用下列任何格式指定除 DB2MMDATAPATH 之外的任何环境变量的值:

- “通用命名约定”名: \\ *机器名* \ *共享名*。例如:

  ```
  \\harmony\JimsShr
  ```

- 驱动器: 路径。例如:

  ```
  f:\media
  ```

- 任何其它: *共享名* \ *目录名*。例如:

  ```
  JimsShr\images
  ```

附录 B. 样本程序和媒体文件

随 DB2 Extender 还提供了各种样本程序。样本程序使用与 Extender 一起提供的图像、音频和视频文件。大多数样本程序是用 C 语言编写的。所有 C 样本程序都是“调用层接口”（CLI）格式的。还提供了几个 Java 样本程序和一个 Net.Data 样本宏文件。

安装 DB2 Extender 时，样本程序安装在目标目录的 SAMPLES 子目录中。安装 DB2 Extender 时，图像、音频和视频文件还安装在目标目录的 SAMPLES 子目录中。 安装期间，Extender 环境变量设置为指向目标目录的 samples 子目录。

样本程序

DB2 Extender 的样本程序由许多文件组成。这些文件是:

文件	描述
enable.c	对 Audio Extender、Image Extender 和 Video Extender 启用数据库、创建表，并启用表及其列。
populate.c	将数据导入表（程序为 C 格式）
Populate.java	将数据导入表（程序为 Java 格式）
query.c	查询表中的数据（程序为 C 格式）
Query.java	查询表中的数据（程序为 Java 格式）
api.c	使用 Extender API 来查询数据库
handle.c	演示 UDF 中句柄的用法，以及如何在 SELECT 语句中进行 where 子句比较
qbcatdmo.c	创建 QBIC 目录，并将图像列编目到目录中
qbicdemo.c	查询 QBIC 目录
color.c	制作 qbicdemo.c 的颜色表声明
QbicQry.java	呈示 QBIC 查询的平均颜色和直方图颜色选择器
makesf.c	创建镜头目录文件以便与 makehtml.exe 配合使用。
makehtml.c	存取镜头目录，并创建 HTML 页面以供 Web 浏览器显示
storybrd.java	显示镜头的 applet，由 makehtml.c 生成的 HTML 页面调用
utility.c	实用程序例程
utility.h	实用程序例程的头文件
makefile.aix	在 AIX 中构建程序的制作文件
makefile.os2	在 OS/2 中构建程序的制作文件
makefile.iva	使用 IBM VisualAge C++ 在 Windows NT（或更高版本）中构建程序的制作文件
makefile.mvc	使用 Microsoft Visual C++ 在 Windows 中构建程序的制作文件
makefile.sun	在 Solaris 中构建程序的制作文件
makefile.hp	在 HP-UX 中构建程序的制作文件

© Copyright IBM Corp. 1998, 2003

样本程序

提供了下列样本程序的可执行文件。期望以显示的次序运行样本程序。

1. Enable
2. Populate
3. Query
4. API
5. Handle
6. Qbcatdmo
7. Qbicdemo
8. QbicQry
9. Makesf
10. Makehtml

随样本 Java 程序附带提供了可执行类文件（Populate.class、Query.class、QbicQry.class 和 storybrd.class）。

在运行样本程序之前，必须在服务器上创建数据库。还必须在服务器上启动 Extender 服务。要运行样本程序，输入程序名（这启动程序的可执行文件）。将提示输入数据库名、用户标识和密码。使用创建数据库的用户的用户标识和密码。

还可构建您自己的样本程序的可执行文件。要这样做，您需要:

1. 将样本程序文件复制到可写的目录。
2. 编辑制作文件，指定系统上安装了 DB2、Extender 和编译器的位置。
3. 使用 make 或 nmake 来将文件编译成可执行程序。

有关安装和使用样本程序的进一步信息，参见样本程序目录中的 README.CNT 文件。

样本图像、音频和视频文件

随 DB2 Extender 一起提供的样本图像、音频和视频文件是:

- 图像文件
  - lizzi.bmp
  - sws_stri.bmp
  - nitecry.bmp
  - ranger_r.bmp
  - fuzzblue.bmp
- 音频文件
  - lizzi.wav
  - sws_stri.wav
  - nitecry.wav
  - ranger_r.wav
  - fuzzblue.wav
- 视频文件
  - nitecry.avi

– sample.mpg

样本 Net.Data 宏文件

与 DB2 Extender 一起包括了一个名为 extender.d2w 的 Net.Data 宏文件。当由 Web 服务器运行时，宏文件执行调用 DB2 Extender UDF 的 SQL 语句。宏文件返回 Web 浏览器显示的结果。如图 30 显示的那样，每个结果页还显示为了生成结果而运行的 SQL 语句。第 484 页的图 31 显示了样本 Net.Data 宏文件的内容。

要运行样本 Net.Data 宏文件，从 Web 浏览器输入以下 URL: http://*your server*/cgi-bin/db2www/extender.d2w/startHere

其中，*your server* 是您的 Web 服务器的名称

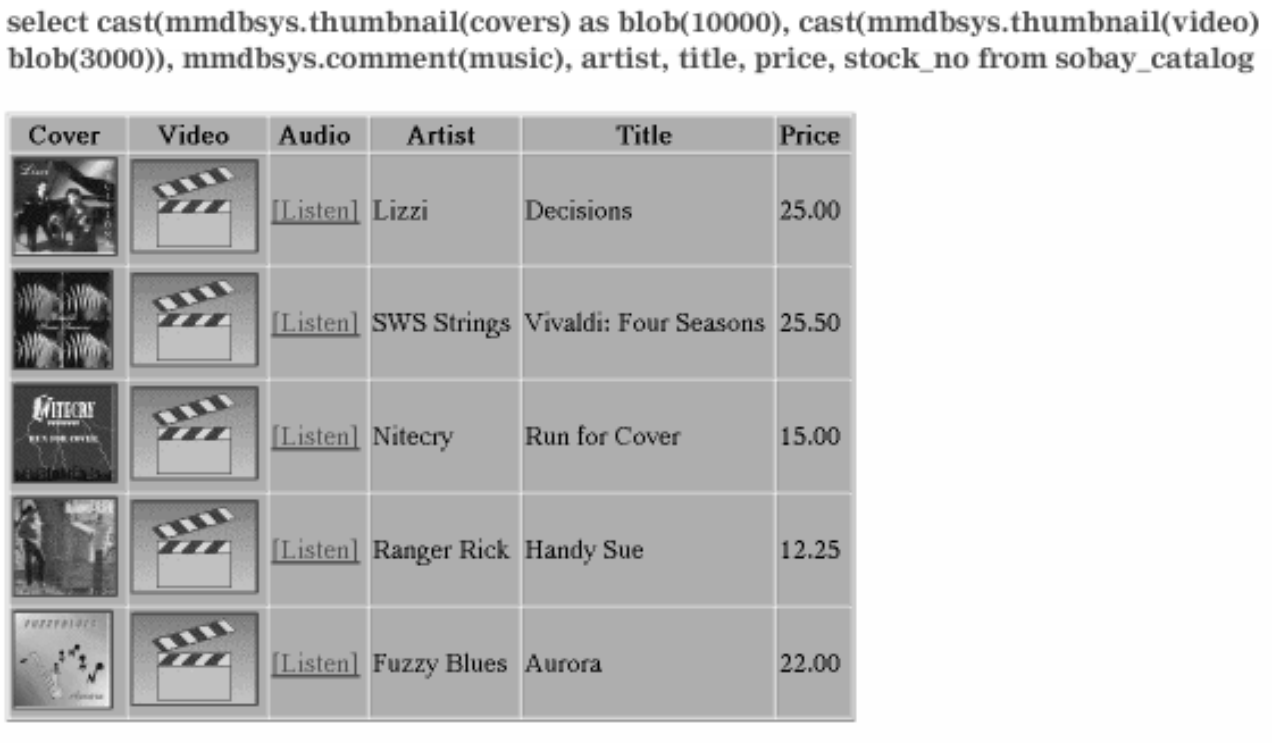


图 30. 运行样本 Net.Data 宏文件的 Web 应用程序. 每个结果页都显示为了生成结果而运行的 SQL 语句。

样本媒体文件

```
%{ ------------------------------------------------------------------------ %}
%{ Copyright International Business Machines Corporation, 1998.             %}
%{ All rights reserved.                                                     %}
%{                                                                          %}
%{ Sample Net.Data macro which shows how to call image, audio, and video    %}
%{ extender UDFs.                                                           %}
%{                                                                          %}
%{ To run, put this macro in your MACRO_PATH root, make sure the tmplobs    %}
%{ directory exists under your web server's document root, and create       %}
%{ the database to be used when running the extender sample programs        %}
%{ 'enable' and 'populate'.  Run 'enable' and 'populate'.  If you name your %}
%{ database something other than 'testdb2', you'll need to change the      %}
%{ definition of DATABASE below.  The extender environment variable         %}
%{ DB2MMEXPORT needs to be set for the instance used by Net.Data to point   %}
%{ to the webserver's <document root>/tmplobs directory.  Then restart DB2  %}
%{ and the extenders to have the variable take effect.                      %}
%{ If you are not running Net.Data's Connection Manager, you'll need to     %}
%{ provide the LOGIN and PASSWORD to the database.  If these instructions   %}
%{ seem unfamiliar to you, you should read the Net.Data documentation at    %}
%{ http://www.software.ibm.com/data/netdata/docs (or the extender documen-  %}
%{ tation on the extender sample programs).                                 %}
%{                                                                          %}
%{ To disable the showing of SQL statements, change the value of SHOWSQL    %}
%{ below to "no".                                                           %}
%{ ------------------------------------------------------------------------ %}

%{ ------------------------------------------------------------------------ %}
%{ Definitions section                                                      %}
%{ ------------------------------------------------------------------------ %}
%define{
  DATABASE="testdb2"
  SHOWSQL="yes"
%}
```

图 31. Net.Data 样本宏文件 (1/5)

```
%{ ------------------------------------------------------------------------ %}
%{ SQL functions                                                            %}
%{ ------------------------------------------------------------------------ %}
%function (DTW_SQL) startHereSQL(){
  select artist, title, stock_no, price from sobay_catalog

  %REPORT{
   <table border="2" bgcolor="#b1b1b1">
           <tr><th>Artist <th> Title <th> Stock<th> Number <th> Price </tr>
    %ROW{ <tr><td> $(V_artist) <td> $(V_title) <td>  $(V_stock_no) <td> $(V_price) <tr>
    %}
    </table>
   %}
%}

%function (DTW_SQL) addThumbsSQL(){
  select cast(mmdbsys.thumbnail(covers) as blob(10000)),
         cast(mmdbsys.thumbnail(video) as blob(3000)),
         mmdbsys.comment(music), artist, title, price, stock_no
    from sobay_catalog

  %REPORT{
   <table border="2" bgcolor="#b1b1b1">
                 <tr><th>Cover <th>Video <th>Audio <th>Artist <th>Title <th>Price </tr>
    %ROW{  <tr><td>< a href="showCover?stock_no=$(V_stock_no)"><img src="$V1)"></a>
               <td><  a href="getVideo?stock_no=$(V_stock_no)"><img src="$V2)"></a>
               <td><  a href="getAudio?stock_no=$(V_stock_no)&filename=$V3")>[Listen]</a>
               <td> $(V_artist) <td> $(V_title) <td> $(V_price) </tr>
    %}
    </table>
  %}
%}

%function (DTW_SQL) showCoverSQL(){
  select cast(mmdbsys.content(covers, 'GIF') as blob(150000)), mmdbsys.format(covers)
    from sobay_catalog
   where stock_no = '$(stock_no)'

  %REPORT{
    %ROW{  <img src="$(V1) "> <br><br><b>Original image format: $(V2)</b>%}
  %}
%}
```

图 31. Net.Data 样本宏文件 (2/5)

样本媒体文件

```
%{ The following Content call depends on DB2MMEXPORT being set properly to
   point to the tmplobs directory under the web server's document root.  %}

%function (DTW_SQL) showVideoSQL(){
  select mmdbsys.comment(video), mmdbsys.content(video, mmdbsys.comment(video), 1),
         mmdbsys.format(video)
    from sobay_catalog
   where stock_no = '$(stock_no)'

  %REPORT{
    %ROW{ <a href="/tmplobs/$(V1)"><i><b> Play Video Clip></b></i></a>
       <br><br><b>Format: $(V3) <br>(Note: NT/Win95 may not come with
          a decompressor<br>for this video format.)</br>
    %}
  %}
%}

%{ The following Content call depends on DB2MMEXPORT being set properly to
   point to the tmplobs directory under the web server's document root.  %}

%function (DTW_SQL) showAudioSQL(){
  select mmdbsys.comment(music), mmdbsys.content(music, mmdbsys.comment(music), 1),
         mmdbsys.format(music)
    from sobay_catalog
   where stock_no = '$(stock_no)'

  %REPORT{
      %ROW{ < a href="/tmplobs/$(V1) "<i><b>Play Audio Clip</b></i></a>
         <br><br><b>Format: $(V3)</b>
    %}
  %}
%}
```

图 31. Net.Data 样本宏文件 (3/5)

```
%{ ------------------------------------------------------------------------ %}
%{ HTML sections                                                            %}
%{   E.g., http://<your server>/cgi-bin/db2www/extender.d2w/startHere       %}
%{ ------------------------------------------------------------------------%}
%{   E.g., http://
%{   E.g., http://
%HTML(startHere){
<html>
   <head><title>UDB Extenders Macro Sample: Simple Row Listing</title></head>
   <body bgcolor="#ffffff">
   <font color="#3300ff" size="3"><b>If no data appears below, you might need
    to run the UDB Extender sample programs <i>enable</i> and <i>populate</i>.
    This first HTML section of the extender.d2w macro simply retrieves all the
    traditional data for all the rows in the UDB Extenders' sample database.
    %if ( "$(SHOWSQL)" == "yes" || "$(SHOWSQL)" == "YES" )
    <br><br> By default, every page generated by this macro shows the SQL used
      to generate that page.  Here is the SQL statement for this page:
    %else
      <br>
    %endif
    </b></font>
     <br>@startHereSQL()
    <br><b>Click < a href="addThumbs"><i>here</i></a> to display thumbnails
    and links to image/audio/video data.</b>
 </body>
</html>
%}

%HTML(addThumbs){
<html>
   <head><title>UDB Extenders Macro Sample: Add Thumbnails</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>This page adds album cover thumbnails
    and links to display the multimedia content of the database.  To access
    the multimedia content:
    <ul>
      <li> Click on a thumbnail of a CD cover to view a full-size image
      <li> Click on a "video thumbnail" to view a video
      <li> Click on a "[Listen]" link to listen to an audio clip
    </ul>
    </b></font> @addThumbsSQL()
    <br><b>Click < a href="startHere"><i>here</i></a> to go back to the first page.</b>
 </body>
</html>
%}
```

图 31. Net.Data 样本宏文件 (4/5)

样本媒体文件

```
%HTML(showCover){
<html>
   <head><title>UDB Extenders Macro Sample: Cover for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>For this page, the macro gets a full-size cover
image, converting the image format to GIF so that a browser can show it:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showCoverSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 </body>
</html>
%}

%HTML(getVideo){
<html>
   <head><title>UDB Extenders Macro Sample: Video clip for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>From this page, you can view a video clip:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showVideoSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 </body>
</html>
%}

%HTML(getAudio){
<html>
   <head><title>UDB Extenders Macro Sample: Audio clip for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>From this page, you can listen to an audio clip:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showAudioSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 <body>
</html>
%}
```

图 31. Net.Data 样本宏文件 (5/5)

声明

本信息是为在美国提供的产品和服务编写的。 IBM 可能在其它国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以用来代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 的产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可证。您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可证查询，请与您所在国家或地区的 IBM 知识产权部门联系，将用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

本条款不适用联合王国或任何这样的条款与当地法律不一致的国家或地区: 国际商业机器公司以“按现状”的基础提供本出版物，不附有任何形式的（无论是明示的，还是暗示的）保证，包括（但不限于）对非侵权性、适销性和适用于某特定用途的默示保证。某些国家或地区在某些交易中不允许免除明示或默示的保证，因此本条款可能不适用于您。

本信息中可能包含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；这些更改将编入本出版物的新版本中。IBM 可以随时对本出版物中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: (i) 允许在独立创建的程序和其它程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序和所有可用的许可资料均有 IBM 依据 IBM 客户协议、IBM 国际程序许可证协议或任何同等协议中的条款提供。

© Copyright IBM Corp. 1998, 2003

本资料仅用于计划目的。在所描述的产品面市之前，此处的信息可随时更改。

本资料包含用于日常商业运作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:

本资料包含用源语言编写的样本应用程序，以示范在各种操作平台上的编程技术。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。这些示例未在所有条件下经过彻底测试。因此，IBM 不能保证或暗示这些程序的可靠性、可服务性或功能。

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声明:

©（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.

如果您查看的是本资料的软拷贝，则照片和彩图可能不会显示。

编程接口信息

本出版物用于帮助您管理 DB2 Extender 并开发程序供 DB2 Extender 使用。本出版物描述了由 DB2 Extender 提供的通用编程接口和关联的指南信息。

通用编程接口允许您编写获取 DB2 Extender 的服务的程序。可以将您开发的应用程序所需要的 DB2 Extender 运行时功能部件复制到客户机或服务器。要安装运行时功能部件，参见 DB2 Extender CD-ROM 上您的操作系统的 README.TXT 文件中提供的安装指示信息。

注册商标

以下各项是 IBM 公司在美国和／或其它国家或地区的商标:

AIX	DB2 Universal Database	PS/2
DB2	IBM	QBIC
DB2 Extenders	OS/2	VisualAge

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和／或其它国家或地区的商标或注册商标。

UNIX 是 The Open Group 在美国和其它国家或地区的注册商标。

Intel 是 Intel 公司的注册商标。

其它公司、产品或服务名称可能是其它公司的商标或服务标记。

词汇表

〔A〕

按图像内容查询（Query by Image Content）（QBIC）： 由 Image Extender 提供的一种功能，它允许用户按图像的视觉特征（例如，*平均颜色（average color）* 和*材质*）来搜索图像。

〔C〕

材质（texture）： 在按图像内容进行的查询中可以使用的特征之一。它是指图像的粗糙度、对比度或方向性。

查询对象（query object）： 指定了功能部件、特征、值和特征权重，以便进行 QBIC 查询的对象。可以对该对象命名并保存它，以便在后续 QBIC 查询中使用。与查询字符串（query string）对照。

查询字符串（query string）： 指定了功能部件、特征、值和特征权重，以便进行 QBIC 查询的字符串。可以从 DB2 命令行，向查询中输入查询字符串。与查询对象（query object）对照。

触发器（trigger）： 更改一个表时要执行的一组操作的定义。触发器可用来执行下列操作，例如：验证输入数据、为最近插入的行自动生成一个值、从其它表中读取数据以进行交叉引用、或者将数据写入其它表以进行审查。通常将触发器用来进行完整性检查，或强制执行商务规则。

粗糙度（coarseness）： *材质*的一种属性，用于测量材质的等级（例如小卵石与巨石的对比）。

〔D〕

大对象（large object）（LOB）： 字节序列，其长度最大可达 2GB。 LOB 可以是三种类型：*二进制大对象*（BLOB）、*字符大对象（CLOB）* 或*双字节字符大对象（DBCLOB）*。

单值类型（distinct type）： *参见用户定义类型（user-defined type）*。

得分（score）： 一个计算值，它反映特征值与按图像内容进行的查询中指定的那些特征值的相似程度。该数目越高，匹配就越接近。该比数用来将按映象内容查询的结果进行排序。

对比度（contrast）： *材质*的一种属性，指的是图案的亮度，也是表示灰阶直方图变化的一个函数。

对象（object）： 在面向对象的编程中，是一个由数据和与该数据相关联的操作组成的抽象概念。

多分区节点组（multipartition nodegroup）： 包含多个*数据库分区服务器*的*节点组*。

〔E〕

二进制大对象（binary large object）（BLOB）： 一种二进制字符串，其最大长度可达 2 GB。图像、音频和视频对象都作为 BLOB 存储在 DB2 数据库中。

〔F〕

方向性（directionality）： *材质*的一种属性，它描述图像是否有明显的方向（比如，青草），还是近似于平滑对象（例如，玻璃）。

分区数据库（partitioned database）： 带有两个或两个以上数据库分区的数据库。用户表中的数据可以位于一个或多个数据库分区中。当一个表在多个分区上时，它的一些行存储在一个分区中，其它行存储在其它分区中。

分析（analyze）： 用来计算图像的特征的数值，并将该值添加到 QBIC 目录中。

〔G〕

故事板（storyboard）： 视频的可视摘要。Video Extender 中包括用于标识和存储视频中镜头的代表性视频帧的功能部件。可以使用这些代表帧来构建故事板。

管理支持表（administrative support table）： 一种表，DB2 Extender 用来处理用户对图像、音频和视频对象的请求。某些管理支持表标识对 Extender 启用的用户表和列。其它管理支持表包含关于已启用列中的对象的属性信息。也称为*元数据表（metadata table）*。

〔H〕

环境变量（environment variable）： 用来描述 DB2 Extender 的操作环境以及为该环境的值提供缺省值的变量。

〔J〕

吉字节（gigabyte）（GB）： 十亿个（10^9）字节。指内存容量时，表示 1 073 741 824 个字节。

渐稳（dissolve）： 当下一个视频帧的信号强度增加时，减少视频帧的信号强度。

© Copyright IBM Corp. 1998, 2003

节点组（nodegroup）： 一个或多个数据库分区的命名组。

节点（node）： 在数据库分区中，与数据库分区（database partition）是同义词。

镜头目录（shot catalog）： 用来存储关于镜头的数据（例如，在视频剪辑中，镜头的开始帧号和结束帧号）的数据库表或文件。用户可以通过 SQL 查询来存取表视图，或者存取文件中的数据。

镜头（shot）： 在两个图象更改之间的帧。

句柄（handle）： Extender 创建的字符串，用来表示表中的图像、音频或视频对象。为用户表和*管理支持表*中的对象存储一个句柄。通过这种方式，Extender 可以将存储在用户表中的句柄与存储在管理支持表中关于对象的信息相链接。

〔M〕

面向对象（object orientation）： 一种编程方法，使用这种方法，无论是实际对象还是抽象对象，在应用程序中都可以用由一组操作和数据值组成的对象来表示。例如，文档可以由文档数据以及可以对文档执行的操作（例如，归档、发送和打印）所组成的文档对象来表示。视频剪辑可以由视频对象来表示，视频对象由数据和操作（例如，播放视频剪辑或查找特定的视频帧）组成。

〔P〕

平均颜色（average color）： 颜色的一种测量方法，可以按照图像的像素中包含的平均颜色值来计算。

〔Q〕

千字节（kilobyte）（KB）： 一千个（10^3）字节。指内存容量时，表示 1024 个字节。

〔S〕

实例（instance）： 逻辑 DB2 Extender 服务器环境。在同一个工作站上可以有几个 DB2 Extender 服务器的实例，但是对于每一个 DB2 实例来说，只有一个实例。可以使用这些实例来:

将开发环境与生产环境分隔开来

将敏感信息局限于特定的一组人员。

视频剪辑（video clip）： 用胶片拍摄下的或用录像磁带录制下的资料的一部分。

视频索引（video index）： 一个文件，Video Extender 用它来查找视频剪辑中特定的*镜头*或帧。

视频（video）： 属于记录下的信息中可以看的部分。

数据库分区服务器（database partition server）： 管理*数据库分区*。数据库分区服务器由数据库管理器，以及它管理的数据集合和系统资源组成。通常，会为每个机器指定一个数据库分区服务器。

数据库分区（database partition）： 数据库的一部分，由它自己的用户数据、索引、配置文件和事务日志组成。有时称为节点或数据库节点。

双字节字符大对象（double-byte character large object）（DBCLOB）： 双字节字符的字符串，或者是单字节字符与双字节字符的组合，其中字符串最多可达 2 GB。DBCLOB 具有相关联的代码页。包括双字节字符的文本对象作为 DBCLOB 存储在 DB2 数据库中。

缩放（scaling）： 将*节点*添加至数据库以增加存储空间并提高性能。

缩略图（thumbnail）： 缩小的图像。

索引文件（index file）： 包含索引信息的文件，Video Extender 使用该索引信息来查找视频剪辑中的*镜头*或个别帧。

〔T〕

太字节（terabyte）： 一万亿（10^{12}）个字节。10 的 12 次方个字节。指内存容量时，表示 1 099 511 627 776 个字节。

特征（feature）： 图像的视觉属性，例如，*平均颜色*。

图像更改（scene change）： *视频剪辑*中的一个点，在此处，两个连续的帧之间有明显的区别。例如，若摄象机在记录视频时更改其摄象点，就会发生这种情况。

图像（image）： 图片的电子表示法。

〔W〕

位置颜色（positional color）： 指定的图像区域中的像素的*平均颜色*值。

文件引用变量（file reference variable）： 一种编程变量，对于将 LOB 移至客户机工作站上的一个文件，或者从客户机工作站上的一个文件移出 LOB 是很有用的。

〔X〕

像素（pixel）： 屏幕可以显示的图像的最小元素。

〔Y〕

音频剪辑（audio clip）： 记录下的音频资料的一部分。

音频（audio）： 属于记录下的信息中可以听的部分。

应用程序编程接口（API）（application programming interface, API）：

(1) 操作系统或可单独订购的许可程序所提供的函数接口。API 允许用高级语言编写的应用程序使用操作系统或许可程序的特定数据或函数。

(2) 在 DB2 中是界面中的一种函数，例如，获取错误消息 API。

(3) DB2 Extender 提供了一些 API，用于请求用户定义函数、管理操作、显示操作和视频画面切换检测。

用户定义函数（UDF）（user-defined function（UDF））： 由 DB2 用户定义的函数。一旦定义，该函数就可以在 SQL 查询和视频对象中使用。例如，可以创建 UDF，以便获得视频的压缩格式或者返回音频的采样速率。这提供了一种方法来定义特殊类型的对象的行为。

用户定义类型（user-defined type）（UDT）： 由 DB2 用户定义的一种数据类型。UDT 是用于区分 LOB 的。例如，可以为图像对象创建一个 UDT，而为音频对象创建另一个 UDT。尽管图像和音频对象都是作为 BLOB 来存储的，但是它们仍被作为不同于 BLOB 并且互不相同的类型来对待。

元数据表（metadata table）： *参见管理支持表（administrative support table）*。

〔Z〕

兆字节（MB）： 一百万个 （10^6）字节。指内存容量时，表示 1 048 576 个字节。

直方图颜色（histogram color）： 测量图像中的相异颜色的一种方法。每种颜色的数据分别存储在 *QBIC 目录* 中。

主变量（host variable）： 嵌入式 SQL 语句中可以引用的应用程序中的变量。主变量是在数据库与应用程序工作区之间传送数据的主要机制。

字符大对象（character large object）（CLOB）： 单字节字符的字符串，该字符串最多可达 2 GB。CLOB 具有相关联的代码页。包含单字节字符的文本对象作为 CLOB 存储在 DB2 数据库中。

A

API: *参见应用程序编程接口（application programming interface）*。

D

DB2 Extender: 一组程序中的一种，您用来存储和检索传统数字和字符数据之外的数据类型，例如，图像、音频和视频数据，以及复杂的文档。

E

Extender: *参见 DB2 Extender*。

L

LOB 定位器（LOB locator）： 存储在主变量中的小型（4 个字节）值，在程序中可以使用它来引用 DB2 数据库中的更大型 LOB。通过使用 LOB 定位器，用户可以操作 LOB，就象它存储在常规主变量中一样，而不需要在客户机和数据库服务器上的应用程序之间传送 LOB。

Q

QBIC 目录（QBIC catalog）： 一个库，用来保存有关图像的视觉特征的数据。

U

UDF: *参见用户定义函数（user-defined function）*。

UDT: *参见用户定义类型（user-defined type）*。

索引

[A]

[B]

[C]

[D]

© Copyright IBM Corp. 1998, 2003

[E]

[F]

[G]

[H]

[J]

[K]

[T]

[W]

[X]

[Y]

[Z]

A

B

C

D

E

F

G

H

I

L

M

N

O

Q

R

S

T

U

V

W

与 IBM 联系

如果您有技术问题，在与“DB2 客户支持”联系之前，请复查并执行由*《故障诊断指南》*建议的操作。此指南提出了一些信息，您可以收集这些信息来帮助“DB2 客户支持”更好地为您服务。

要获取信息或订购任何 DB2 通用数据库产品，请与当地的分支机构办公室的 IBM 代表联系或与任何授权的 IBM 软件分销商联系。

如果您住在美国，则可以拨打下列号码中的一个:

- 1-800-237-5511，以获取客户支持
- 1-888-426-4343，以了解可用的服务选项

产品信息

如果您住在美国，则可以拨打下列号码中的一个:

- 1-800-IBM-CALL（1-800-426-2255）或 1-800-3IBM-OS2（1-800-342-6672），以订购产品或获取一般信息。
- 1-800-879-2755，以订购出版物。

http://www.ibm.com/software/data/db2
DB2 万维网页面提供有关新闻、产品描述和培训计划等当前 DB2 信息。

http://www.ibm.com/software/data/support
DB2 支持 Web 页面提供对常见问题、修订、书籍和最新 DB2 技术信息的访问。

注: 此信息可能只有英语版。

http://www.ibm.com/software/data/db2/extenders
DB2 Extender Web 页面提供有关所有当前可用的 DB2 Extender 的信息。这些 DB2 Extender 包括 DB2 XML Extender、DB2 Spatial Extender 和 DB2 AVI Extender。

http://www.ibm.com/software/data/db2/extenders/support
DB2 Extender 支持 Web 页面提供对常见问题、提示和技巧、修订和文档的访问。

注: 此信息可能只有英语版。

http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi
出版物中心提供有关如何订购或下载出版物的信息。

http://www.ibm.com/certify/index.html
Professional Certification Program from IBM Web 站点提供各种 IBM 产品（包括 DB2）的证书测试信息。

在 Compuserve 上: GO IBMDB2
输入此命令以访问 IBM DB2 系列论坛。所有 DB2 产品通过这些论坛受支持。

© Copyright IBM Corp. 1998, 2003

有关如何在美国以外联系 IBM 的信息，参考 *IBM Software Support Handbook* 的 Appendix A。要访问此文档，转至以下 Web 页面：http://techsupport.services.ibm.com/guides/contacts.html

注: 在某些国家或地区，IBM 授权的经销商应联系其经销商支持机构而不是 IBM 支持中心。

S152-0598-00