GBASE®

GBase 8s 导入导出工具指南

www.gbase.cn

GBase 8s 导入导出工具指南，南大通用数据技术股份有限公司

版权声明

免责声明

通讯方式

南大通用数据技术股份有限公司

天津华苑产业高新区开华道 22 号普天创新产业园东塔 20-23 层

电话：400-013-9696　　邮箱：info@gbase.cn

商标声明

目　录

1 介绍

关于本手册

本手册描述 GBase 8s 提供的多种导入导出工具使用说明及语法介绍，本手册主要针对以下用户：

有下列背景的数据库服务器管理员

对于计算机、操作系统以及操作系统提供的实用程序的应用知识

使用关系数据库的经验，或熟悉数据库概念

一定的数据库管理、操作系统管理和网络管理经验

本手册介绍的导入导出工具包括：**dbexport/dbimport**、**dbload**、**dbschema**、**load/unload**、**gload/ungload**。

2 dbexport 和 dbimport 实用程序

dbexport 和 **dbimport** 实用程序向存储在磁盘或磁带中的文本文件导入或导出数据库。

dbexport 实用程序将整个数据库卸载到文本文件并创建模式文件。可以通过 **dbimport**，使用该模式文件在另一个 GBase 8s 环境中重新创建数据库模式，并且您还可以编辑模式文件以修改 **dbimport** 创建的数据库。

如果您无法使用 **gunload** 和 **gload** 实用程序并且您想要将数据库（带或不带其模式文件）卸载到磁盘或磁带，那么您可以使用 **dbexport** 和 **dbimport** 实用程序。

dbimport 实用程序用于创建数据库，它使用磁带或磁盘上文本文件中的数据来装入数据库。输入文件由用来重新创建数据库的模式文件和包含数据库数据的数据文件组成。通常使用 dbexport 实用程序生成输入文件，但您也可以使用任何适当格式的输入文件。

注意： 导入数据库时，请使用与创建该数据库时使用的相同的环境变

量，否则可能得到预料之外的结果。如果有任何使用与 **dbimport** 用的设置不同的设置创建的分段存储表达式、检查约束、触发器或用户定义的例程, 则无法使用单个的导入准确地重新产生数据库。

- 如果有任何使用与 **dbimport** 用的设置不同的设置创建的分段存储表达式、检查约束、触发器或用户定义的例程, 则无法使用单个的导入准确地重新产生数据库。
- 如果将数据从较早版本数据库服务器移动到新版本的数据库服务器，则可能会发生预料之外的结果。一些配置参数或环境变量被弃用，或者它们的缺省值被更改。例如，假定在原始数据库中缺省创建了附加索引。在当前版本的数据库服务器中，会缺省创建拆离索引。如果您需要维持原始的行为。可以在运行 **dbimport** 实用程序之前将 DEFAULT_ATTACH 环境变量设置为 1 。

适用于以下情况的要求和限制:

压缩数据

dbexport 实用程序会解压压缩的数据。您必须在使用 **dbimport** 实用程序导入数据后重新压缩这些数据。

日期值

日志值使用四位数记年。对象的日期上下文包含创建对象的日期、DBCENTURY 和 GL_DATE 环境变量的值以及一些其他的环境变量。如果导入期间的日期上下文与创建这些对象时的上下文不同，您可能会得到显式的错误，或无法找到您的数据，或检查约束不像期望的那样工作。许多这些问题并不会生成错误。

提示：缺省情况下，dbexport 以四位数记年法导出日期，除非环境变量被设置为可以覆盖那些格式的值。

高可用集群

不能在 HDR 辅助服务器或共享磁盘（SD）辅助服务器上使用 **dbexport** 实用程序。只有当服务器设置为停止应用日志文件，远程独立（RS）服务器上才

支持 **dbexport** 实用程序。使用 **STOP_APPLY** 配置参数停止日志文件的应用。索引的可更改的辅助服务器上都支持 **dbimport** 实用程序

基于标签的访问控制（LBAC）

当导出 LBAC 保护的数据时，限制导出的数据为 LBAC 证书允许您读取的数据。如果 LBAC 证书不允许您读取一行，则该行不能导出，并且不会返回错误。要导出所有的行，您必须可以查看到这些行。

NLSCASE 模式

无论您的源数据库的 **NLSCASE** 模式是 **SENSITIVE** 还是 **INSENSITIVE**，可以通过始终迁移到与源数据库相同 NLSCASE 模式的目标数据库，来降低区分大小写问题的风险。对于包含具有大小写变化的 **NCHAR** 和 **NVARCHAR** 数据值（例如，“GBASE”、“gbase”或“Gbase”）列的表，在迁移后可能遇到以下不同：

- 与迁移前的相同查询的结果相比，ORDER BY 和排序操作可以在查询结果集中生成不同顺序的合格行。
- 如果任何索引或约束键列包含不同大小写的格式的相同字符串，则在迁移之前数据符合的唯一索引和参照约束可能会在新数据库中具有完整性违规。
- 使用条件运算符应用于 NCHAR 或 NVARCHAR 值的谓词的查询可能会在迁移后返回来自相同数据的不同结果。

非缺省的数据库语言环境

如果数据库使用非缺省的语言环境并且 **GL_DATETIME** 环境变量具有非缺省设置，那么必须将 **USE_DTENV** 环境变量的值设置为值 1，然后才能使用 **dbexport** 或 **dbimport** 实用程序正确处理本地化的 DATETIME 值。

表上的 select 触发器

在使用 **dbexport** 实用程序导出数据库之前，必须禁用 SELECT 触发器。**dbexport** 实用程序在导出期间会运行 SELECT 语句。该 SELECT 语句触发器

可能会修改数据库内容。

GBase MQ 扩展的虚拟表

MQCreateVtiRead() 、**MQCreateVtiReceive()** 和 **MCQCreateVtiWrite()** 函数创建虚拟表，将它映射到合适的 WebSphere® MQ 消息队列。当 dbexport 实用程序卸载数据时，它移除 WebSphere® MQ 队列的消息。在使用 **dbexport** 实用程序之前，请移除任何 MQ 虚拟表。在使用 **dbimport** 实用程序加载数据库后，可以使用适当的函数在目标数据库上创建表。

支持 DATABASE.OBJECT 之后

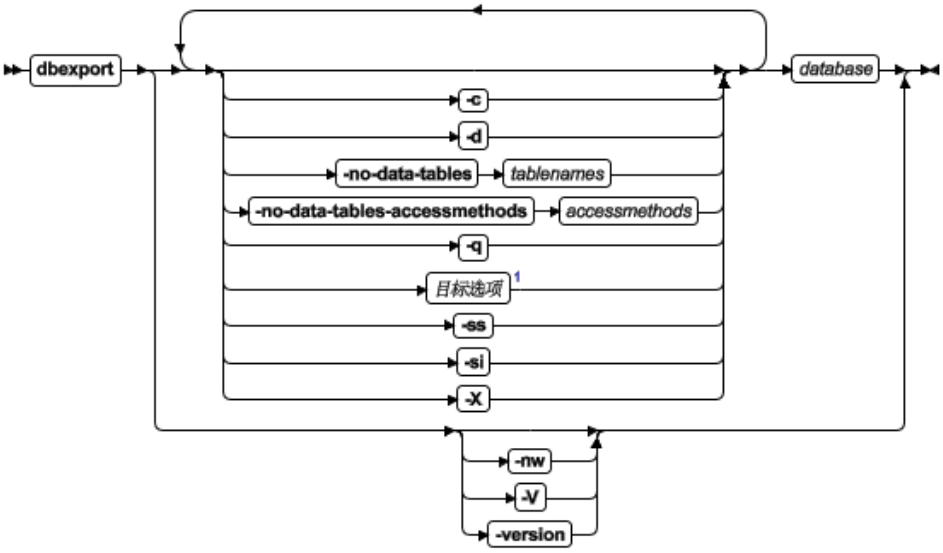
dbexpor/dbimport 执行导出操作时，数据库对象名前缀不再增加 owner.引用，如果需要导出的数据库中存在 ORACLE 模式下创建的包、存储过程、函数等数据库对象，导出文件中会增加库名_ora.sql 脚本专门用于保存以上数据库对象；其他导出文件保持不变。执行导入操作时，如果存在库名_ora.sql 脚本文件，会自动增加 ORACLE 模式下创建的包、存储过程、函数等数据库对象的导入。

dbexpor/dbimport 导出导入工具使用方法保持不变。执行导出操作时，如果需要导出在 ORACLE 模式下创建的如表、视图、同义词、索引、物化视图、存储过程等数据库对象，对应内容会导出在 库名_ora.sql 脚本 专门用于保存 Oracle 模式数据库对象；gbase 模式下，导出文件保持不变。执行导入操作时，如果存在 库名_ora.sql 脚本文件，会自动增加 ORACLE 模式下创建的对象导入。

需注意，如有 Oracle 兼容需求，请保持在 Oracle 模式下使用。如存在 gbase 模式下创建如视图、触发器，但基表为 Oracle 模式下创建等情况。导入场景由于优先导入 gbase 脚本，可能报错对象不存在。

2.1 dbexport 命令的语法

dbexport 命令将数据库卸载到文本文件，以便以后将这些文本文件导入到另一个数据库中。该命令还可以创建模式文件。



元素	用途	重要注意事项
-c	使 dbexport 完成导出，除非发生致命错误	**参考**：有关此选项的详细信息，请参阅 dbexport 错误。
-d	使 dbexport 只导出简单大对象描述符，不导出简单大对象数据	**参考**：有关简单大对象描述符的信息，请参阅 *GBase 8s Optical Subsystem Guide*。 **限制**：SE 不支持。
-q	隐藏错误消息、警告和生成的 SQL 数据定义语句的显示	无。
-ss	对指定数据库中的所有表生成特定于数据库服务器的信息	**参考**：有关此选项的详细信息，请参阅 dbexport 特定于服务器的信息。

-si	为非分段表排除索引存储子句的生成 -si 选项仅在与 -ss 选项一起使用时才可用。	**参考：**有关此选项的详细信息，请参阅 dbexport 特定于服务器的信息。
-no-data-tables	防止为指定的表导出数据。仅导出指定表的定义。	接受以逗号分隔的表的名称列表，这些表的数据将不会被导出。 **缺省行为：**只导出 tsinstanceTable 表的定义，而不会导出数据。会导出所有其它表的数据和定义。
-no-data-tables accessmethods	防止使用指定的访问方法卸载数据。	接收以逗号分隔的访问方法的名称列表。不会卸载使用这些访问的表。 **缺省值：** ts_rts_vtam, ts_vtam 使用 ts_rts_vtam 和 ts_vtam 访问方法的表不会被卸载。
-X	识别字符字段中的 HEX 二进制数据	无。
-nw	生成用于在未指定所有者的情况下创建数据库的 SQL	无。
-V	显示软件版本号和序列号	无。
-version	扩展 -V 选项以显示有关构建操作系统、构建号和构建日期的其他信息	无。
database	指定希望导出的数据库的名称	**其他信息：**如果语言环境设置为使用多字节字符，那么可对数据库名称使用多字节字符。 **参考：**如果希望使用比数据库的简单

		名称更多的内容，请参阅《*GBase 8s SQL 指南：语法*》的『数据库名称』一节。
-l	指定数据库在线导出时不用锁定数据库	此参数会按照脏读隔离级别导出指定数据库

您必须具有 DBA 特权或以用户 **gbasedbt** 的身份登录才能导出数据库。

Global Language Support： 当环境变量设置正确时，如《GBase 8s GLS 用户指南》中所述，**dbexport** 可处理数据中的外来字符并从 GLS 数据库导出数据。有关更多信息，请参阅数据库重命名。

可以设置 IFX_UNLOAD_EILSEQ_MODE 环境变量来支持 **dbexport** 使用对于环境中指定的语言环境无效的字符数据。

可将定界标识用于 **dbexport** 实用程序。该实用程序会检测诸如关键字、混合大小写或具有特殊字符的数据库对象，并且实用程序还在它们周围括上双引号。

除了数据文件和模式文件，**dbexport** 会在当前目录中创建名为 **dbexport.out** 的消息文件。此文件包含错误消息、警告以及它生成的 SQL 数据定义语句的显示。也会将相同的材料写到标准输出，除非指定了 **-q** 选项。

导出期间，将以互斥方式锁定数据库。如果 **dbexport** 无法获得互斥锁定，它将显示诊断消息并退出。此时需要指定 **-l** 选项。

提示： **dbexport** 实用程序可创建大于 2 GB 的文件。要支持这样大的文件，请确保操作系统文件大小限制设置得足够高。例如：在 **UNIX™** 上，将 **ulimit** 设置为不受限制。

示例

以下命令导出 customer 数据库中所有表的定义，但是不导出数据：

```
dbexport -no-data-tables -no-data-tables-accessmethods customer
```

示例

以下命令将在不指定所有者的情况下，为 customer 数据库生成模式和数据：

```
dbexport customer –nw
```

示例

以下命令将按照脏读隔离级别导出指定数据库，在/work/exports 目录下生成 customer.exp 目录：

```
dbexport -c -l /work/exports customer
```

2.1.1 终止 dbexport 实用程序

您可以随时停止 **dbexport** 实用程序。

要取消 **dbexport**，请按中断键。

dbexport 实用程序在其终止前会请求确认。

2.1.2 dbexport 错误

dbexport -c 选项使 **dbexport** 完成导出，除非发生致命错误。

即使您使用了 **-c** 选项，如果发生以下任一致命错误，**dbexport** 仍会中断处理：

- **dbexport** 无法打开指定的磁带。
- **dbexport** 找到磁带或磁盘的坏写。
- 已使用的命令参数无效。
- **dbexport** 无法打开数据库，或者没有执行此操作的系统许可权。
- 在调用期间指定名称的子目录已存在。

2.1.3 dbexport 特定于服务器的信息

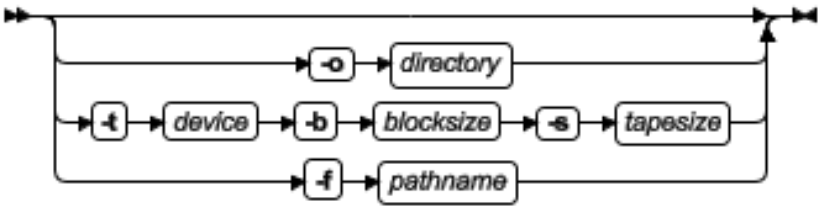
dbexport -ss 选项生成特定于服务器的信息。此选项指定最初扩展数据块和下一扩展数据块的大小、分段存储信息（如果表被分段）、锁定方式、表的数据库空间、任何简单大对象的 Blob 空间以及任何智能大对象的数据库空间。

只有与 **-ss** 选项一起使用时才可用的 **dbexport -si** 选项不能为非分段表生成索引存储子句。

2.1.4 dbexport 目标选项

dbexport 实用程序支持磁盘和磁带目标选项。

目标选项



元素	用途	重要注意事项
-b *blocksize*	指定磁带设备的块大小（以千字节计）	无。
-f *pathname*	指定希望存储模式文件的路径名（如果要在磁带上存储数据文件）	**其他信息：**路径名可以是完整的路径名或文件名。如果只给出文件名，那么文件将存储在当前目录中。
-o *directory*	指定磁盘上的目录，dbexport 将在该目录中创建	**限制：**指定为*目录名*的目录必须存在。

	database.exp 目录。 该目录保留 dbexport 为***数据库***创建的数据文件和模式文件。	
-s *tapesize*	指定您可在磁带上存储的数据量（以千字节计）	**其他信息：** 要写到磁带的末尾，请将 *tapesize* 指定为 0。 如果您不指定 0，那么最大 *tapesize* 是 2 097 151 KB。
-t *device*	指定您希望存储文本文件和可能存储模式文件的磁带设备路径名。	-t 选项不允许您指定远程磁带设备。

写到磁盘时，**dbexport** 将在 **-o** 选项指定的目录中创建 **database.exp** 子目录。 **dbexport** 实用程序为数据库中的每个表创建带 **.unl** 扩展名的文件。模式文件写入文件 **database.sql。 .unl** 和 **.sql** 文件位于 **database.exp** 目录中。

如果没有为数据和模式文件指定目标，将把子目录 **database.exp** 放置在当前工作目录中。

将数据文件写入磁带时，可以使用 **-f** 选项将模式文件存储到磁盘。不需要将模式文件命名为 **database.sql**。您可任意取名。

仅 UNIX/Linux

对于 UNIX™ 或 Linux™ 上的非 SE 数据库服务器，该命令为：

```
dbexport //finland/reports
```

以下命令将数据库 **stores_demo** 导出到磁带，该磁带的块大小为 16 KB 且容量为 24000 KB。该命令还可以将模式文件写入到 **/tmp/stores_demo.imp**。

```
dbexport -t /dev/rmt0 -b 16 -s 24000 -f /tmp/stores_demo.imp
        stores_demo
```

以下命令将同一 **stores_demo** 数据库导出到名为 **/work/exports/stores_demo.exp** 的目录。结果模式文件为 **/work/exports/stores_demo.exp/stores_demo.sql**。

```
dbexport -o /work/exports stores_demo
```

2.2 dbexport 创建的模式文件的内容

dbexport 实用程序可以创建模式文件。此文件包含您重新创建导出的数据库所需要的 SQL 语句。

您可编辑该模式文件以修改数据库的模式。

如果使用 **-ss** 选项，那么模式文件将包含特定于服务器的信息，例如最初和下一扩展数据块的大小、分段存储信息、锁定方式、每个表驻留的数据库空间、每个简单大对象列驻留的 Blob 空间以及智能大对象的数据库空间。不会保留以下信息：

- 数据库的日志记录方式

  有关日志记录方式的信息，请参阅《GBase 8s SQL 指南：参考》。

- SERIAL 列的起始值

模式文件中的语句，这些语句创建表、视图、索引、分区段表、索引和角色，授予最初创建数据库的用户名称执行这些操作的特权。使用这种方法，原所有者将保留数据库的 DBA 特权并成为所有表、索引和视图的所有者。另外，执行 **dbimport** 命令的用户也将具有数据库的 DBA 特权。

dbexport 创建的模式文件包含以括号包围的注释，注释包含有关表中的行、列、索引数的信息以及有关卸载文件的信息。**dbimport** 实用程序使用这些注释中的信息来装入数据库。

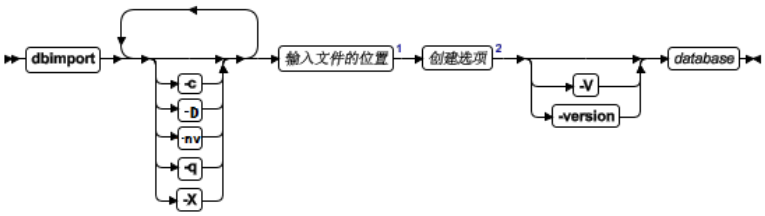
卸载文件中的行数必须匹配模式文件中对应的卸载注释。如果在卸载文件中更改了行数而没有更改模式文件中的行数，将发生不匹配。

注意： 不要删除模式文件中的任何注释，不要更改现有注释或添加任何新注释。如果您更改或添加新注释，**dbimport** 实用程序可能停止或产生不可预测的的结果。

如果从卸载文件中删除了行，请使用卸载文件中正确的行数更新模式文件中的注释。然后，**dbimport** 将成功执行。

2.3 dbimport 命令的语法

dbimport 命令将先前导出的数据导入到另一个数据库。



元素	用途	重要注意事项
-c	即使出现某些非致命错误，也可以完成数据导入。	**参考**：有关更多信息，请参阅 dbimport 错误和警告。
-D	如果未在 CREATE TABLE 语句中指定扩展区大小，请在导入操作期间为第一个和后续扩展区指定 16 KB 的缺省扩展区大小。	如果在 CREATE TABLE 语句中指定了扩展区大小，则忽略此选项。 缺省值有助于确保在导入操作指定的 dbspace 中有足够的空间可用。此选项可防止在导入操作期间自动计算扩展区的大小，并且在以下情况下尤其有用：

		导入包含具有大的最大行大小的列的表时，如 LVARCHAR 列。 在没有 -ss 选项的情况下运行 dbexport 命令后导入数据时，-ss 选项指定有关 extent 大小的服务器特定信息。
-nv	当 **dbimport -nv** 命令正在运行时，不会检查 ALTER TABLE ADD CONSTRAINT 在启用或过滤模式下创建的具有外键约束的表是否违规，就像您还指定了 NOVALIDATE 一样。	通过绕过对引用约束的检查，此选项可以减少已经符合其外键约束的非常大的表的迁移时间。在 ALTER TABLE ADD CONSTRAINT 语句完成后，NOVALIDATE 模式不会持续。
-q	隐藏错误消息、警告和生成的 SQL 数据定义语句的显示	无。
-V	显示软件版本号和序列号	无。
-version	扩展 -V 选项以显示有关构建操作系统、构建号和构建日期的其他信息	无。
-X	识别字符字段中的 HEX 二进制数据	无。
database	声明要创建的数据库的名称	**其他信息：**如果希望使用比数据库的简单名称更多的内容，请参阅《*GBase 8s SQL 指南：语法*》中的『数据库名称』一节。

dbimport 实用程序可使用来自以下位置选项的文件：

- 位于磁盘上的所有输入文件。
- 位于磁带上的所有输入文件。
- 位于磁盘上的模式文件以及位于磁带上的数据文件。

要点： 不要在您的输入文件中放置注释。当 **dbimport** 实用程序读取注释时可能导致无法预料的结果。

dbimport 实用程序针对导入的 GBase 8s 数据库服务器（排除 SE）支持以下任务：

- 指定数据库将驻留的数据库空间
- 创建具有无缓冲日志记录的符合 ANSI 的数据库
- 创建支持显式事务的数据库（具有缓冲或无缓冲日志记录）
- 创建未记录的数据库
- 通过用于 NCHAR 和 NVARCHAR 字符串的 NLS 不区分大小写属性创建数据库。

运行 dbimport 的用户被授予对新创建的数据库的 DBA 特权。dbimport 进程在装入每个表时将锁定该表并在完成装入时解锁该表。

Global Language Support： 当 GLS 环境变量正确设置时，如《GBase 8s GLS 用户指南》所述，**dbimport** 可以将数据导入到支持 GLS 的数据库服务器版本。

2.3.1 终止 dbimport 实用程序

您可以随时停止 **dbimport** 实用程序。

要取消 **dbimport** 实用程序，请按中断键。

dbimport 实用程序在其终止前会请求确认。

2.3.2 dbimport 错误和警告

dbimport -c 选项使 **dbimport** 实用程序完成导出，除非发生致命错误。

如果在 **dbimport** 命令中包含 **-c** 选项，那么 **dbimport** 忽略以下错误：

- 包含过多列的数据行
- 不能在表上放置锁
- 不能释放锁

即使您使用了 **-c** 选项，如果发生以下任一致命错误，**dbimport** 仍会中断进程：

- 无法打开指定的磁带设备
- 到磁带或磁盘的坏写
- 无效的命令参数
- 无法打开数据库或无系统许可权
- 无法转换数据

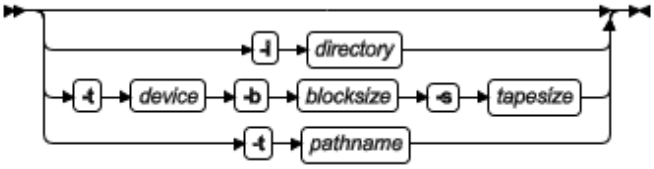
dbimport 实用程序在当前目录中创建名为 **dbimport.out** 的消息文件。此文件包含与 **dbimport** 处理有关的任何错误消息和警告。相同的消息也会写到标准输出，除非您指定 **-q** 选项。

2.3.3 dbimport 输入文件位置选项

输入文件的位置指定 **database.exp** 目录的位置，该位置包含 **dbimport** 实用程序将导入的文件。

如果未指定输入文件的位置，**dbimport** 将在当前目录下的 **database.exp** 目录中搜索数据文件并在 **database.exp/database.sql** 中搜索模式文件。

dbimport 输入文件位置



元素	用途	重要注意事项
-b *blocksize*	指定磁带设备的块大小（以千字节计）	如果正从磁带导入，那么必须使用与导出数据库时使用的相同块大小。
-f *pathname*	指定 dbimport 可在何处找到用于输入的模式文件，该文件用来在从磁带读取数据文件时创建数据库	**其他信息：**如果使用 -f 选项来导出数据库，通常应使用与 dbexport 命令中指定的同一路径名。如果您只指定文件名，dbimport 将在当前目录的 .exp 子目录中查找文件。

-i *directory*	指定磁盘上 *database*.exp 目录的完整路径名，该目录中包含 dbimport 用来创建和装入新数据库的输入数据文件和模式文件。目录名必须与数据库名相同。	**其他信息：**此目录必须与您使用 dbexport -o 选项指定的目录相同。如果更改了目录名，那么还要重命名数据库。
-s *tapesize*	指定您可在磁带上存储的数据量（以千字节计）	**其他信息：**要读取到磁带的末尾，请指定磁带大小为 0 如果正从磁带导入，那么必须使用与导出数据库时使用的相同磁带大小。如果未将 *tapesize* 指定为 0，那么最大的 *tapesize* 是 2097151 KB。
-t *device*	指定包含输入文件的磁带设备的路径名	-t 选项*不*允许您指定远程磁带设备。

在 UNIX™ 或 Linux™ 上显示输入文件位置的示例

要从块大小为 16 KB 且容量为 24 000 KB 的磁带导入 stores_demo 数据库，请发出以下命令：

```
dbimport -c   -t /dev/rmt0 -b 16 -s 24000 -f
        /tmp/stores_demo.imp stores_demo
```

从 /tmp/stores_demo.imp 读取模式文件。

要从 /work/exports 目录下的 stores_demo.exp 目录导入 stores_demo 数据库，请发出以下命令：

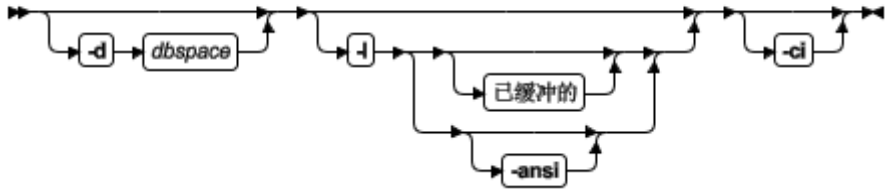
```
dbimport -c -i /work/exports stores_demo
```

模式文件假定为/work/exports/stores_demo.exp/stores_demo.sql。

2.3.4 dbimport 创建选项

dbimport 实用程序支持用于创建数据库、为该数据库指定数据库空间、定义日志记录以及（可选）将 ANSI/ISO 符合性和/或 NLS 不区分大小写性指定为数据库属性的选项。

创建选项



元素	用途	重要注意事项
-ansi	创建符合 ANSI/ISO 的数据库，在该数据库中对事务日志记录启用 ANSI/ISO 规则。否则，缺省情况下数据库使用显式事务。	如果省略 -ansi 选项，数据库将使用显式事务。 **其他信息：**有关符合 ANSI/ISO 的数据库的更多信息，请参阅《*GBase 8s SQL 指南：参考*》。
-ci	指定 NLS 不区分大小写的属性。否则，缺省情况下数据库区分大小写。	**其他信息：**请参阅《*GBase 8s SQL 指南：语法*》和《*GBase 8s SQL 指南：参考*》中 NLS 不区分大小写的属性的描

		述。
-d *dbspace*	指定要创建数据库的数据库空间。	如果省略此元素，缺省位置为**根**数据库空间
-l	为导入的数据库建立无缓冲事务日志记录。如果省略 -l 标志，将不对数据库进行日志记录。	**参考：**有关更多信息，请参阅数据库日志记录方式。
-l buffered	为导入的数据库建立缓冲事务日志记录。如果包含 -l 而省略 buffered，数据库会使用无缓冲日志记录。	**参考：**有关更多信息，请参阅数据库日志记录方式。

如果在数据库服务器中创建了包含分区的表或索引分段，那么在导入单个数据库空间中包含多个分区的数据库时必须使用包含分区名称的语法。有关语法的详细信息，请参阅《GBase 8s SQL 指南：语法》。

显示 dbimport 创建选项的示例（UNIX™ 或 Linux™）

要从 **/usr/gbasedbt/port/stores_demo.exp** 目录导入 **stores_demo** 数据库，请发出以下命令：

```
dbimport -c stores_demo -i /usr/gbasedbt/port -l -ansi
```

新数据库符合 ANSI/ISO。

下一个示例以类似方式从 **/usr/gbasedbt/port/stores_demo.exp** 目录导入 **stores_demo** 数据库。 导入的数据库使用缓冲事务日志记录和显式事务。**-ci** 标志指定在查询中以及在对数据类型为 NCHAR 和 NVARCHAR 的列和字符串执行的其他操作中不区分大小写：

```
dbimport -c stores_demo -i /usr/gbasedbt/port -l buffered -ci
```

数据库属性的 **-ansi** 和 **-ci** 选项不是互斥的。可以指定符合 ANSI/ISO 且同时满足 NLS 不区分大小写的条件的数据库，如以下 **dbimport** 命令示例中所示：

```
dbimport -c stores_demo -i /usr/gbasedbt/port -l -ansi -ci
```

2.3.5 数据库日志记录方式

因为模式文件中未保留日志记录方式，所以您可以在使用 **dbimport** 实用程序导入数据库时指定日志记录信息。

使用 **dbimport** 时，您可以指定以下任何日志记录选项：

- 具有无缓冲日志记录的符合 ANSI 的数据库
- 无缓冲日志记录
- 缓冲日志记录
- 无日志记录

有关更多信息，请参阅 **dbimport** 创建选项。

-l 选项与 CREATE DATABASE 语句的日志记录子句等价，如下所示：

- 省略任何 **-l** 选项相当于省略 WITH LOG 子句。
- **-l** 选项与 WITH LOG 子句等价。
- **-l buffered** 选项与 WITH BUFFERED LOG 等价。
- **-l -ansi** 选项相当于 WITH LOG MODE ANSI 子句，并表示无缓冲日志记录。

2.3.6 数据库重命名

dbimport 实用程序赋予新数据库与导出的数据库相同的名称。如果将数据库导出到磁带，当使用 **dbimport** 导入时无法更改其名称。如果将数据库导出到磁盘，那么可更改数据库名称。

您可以使用 RENAME DATABASE 语句来更改数据库名称。

更改数据库名称的替代方法

以下示例显示更改数据库名称的替代方法。在此示例中，假设 **dbexport** 将数据库 **stores_demo** 卸载到目录 **/work/exports/stores_demo.exp** 中。因此，数据文件（.unl 文件）存储在 **/work/exports/stores_demo.exp**，而模式文件是 **/work/exports/stores_demo.exp/stores_demo.sql**。

要在 UNIX™ 或 Linux™ 上将数据库名称更改为新名称：

1. 更改 **.exp** 目录的名称。即，将 **/work/exports/stores_demo.exp** 更改为 **/work/exports/newname.exp**。

2. 更改模式文件名称。即，将 **/work/exports/stores_demo.exp/stores_demo.sql** 更改为 **/work/exports/stores_demo.exp/newname.sql**。 请不要更改 **.unl** 文件的名称。

3. 使用以下命令导入数据库：

```
dbimport -i /work/exports newname
```

2.4 使用 dbimport 更改数据库语言环境

您可以使用 **dbimport** 实用程序更改数据库的语言环境。

要更改数据库的语言环境：

1. 将 **DB_LOCALE** 语言环境设置为当前数据库语言环境的名称。
2. 在数据库上运行 **dbexport**。
3. 使用 DROP DATABASE 语句删除具有当前语言环境名称的数据库。
4. 将 **DB_LOCALE** 环境变量设置为该数据库的期望的语言环境。
5. 运行 **dbimport**，以创建带有期望的语言环境的新数据库并将数据导入此数据库。

2.5 简单大对象

当 dbimport、dbexport 和 DB-Access 实用程序处理简单大对象数据时，它们为该数据在临时目录中创建临时文件。

从包含简单大对象的表导出或导入数据前，您必须有以下各项之一：

- 当前活动驱动器上的 \tmp 目录
- DBTEMP 环境变量设置为指向可用于临时存储简单大对象的目录

注意： 如果表在列中有 CLOB 或 BLOB，您将无法使用 dbexport 将表导出到磁带。如果表在列中具有用户定义的类型，使用 dbexport 将表导出到磁带可能产生无法预料的结果（依赖于用户定义类型的导出函数）。导出的 CLOB 大小以十六进制格式存储在卸载文件中。

3 dbload 实用程序

dbload 实用程序将数据装入 GBase 8s 产品创建的数据库或表中。它将数据从一个或多个文本文件传送到一个或多个现有表中。

此实用程序支持 所有 GBase 8s 版本中的新数据类型。

先决条件： 如果数据库包含基于访问控制 (LBAC) 对象，那么 **dbload** 实用程序只能装入您的安全标号控制其列安全标号或行安全标号的那些行。如果要装入整个表，那么必须具有写入所有标注的行和列的必要 LABC 凭证。有关 LBAC 对象的更多信息，请参阅 GBase 8s 安全性能指南 和《GBase 8s SQL 指南：语法》。

不能在高可用性集群中的辅助服务器上使用 **dbload** 实用程序。

当您使用 **dbload** 实用程序时，可以操纵正在装入的数据文件或访问正在装入的数据库。尽可能使用 LOAD 语句，它比 **dbload** 要快。

dbload 实用程序给了您极大的灵活性，但它没有其他方法快，并且您必须准备一个命令文件来控制输入。您可将多种格式的数据用于 **dbload**。

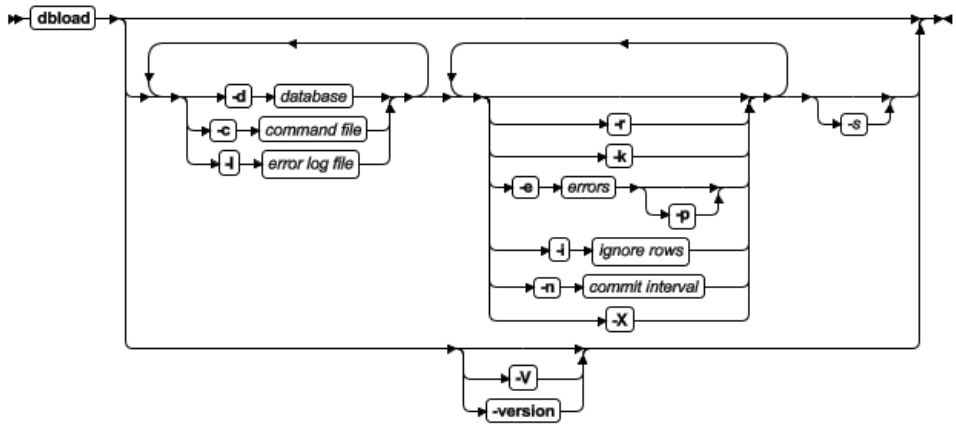
dbload 实用程序与 LOAD 语句相比，有以下优点：

- 可使用 **dbload** 从使用通过多种格式排列创建的输入文件中装入数据。**dbload** 命令文件可适应来自完全不同的数据库管理系统的数据。
- 通过指示 **dbload** 进行读取但忽略 x 个行，可在装入中指定一个起始点。
- 可指定批处理大小，这样在每插入 x 个行之后就落实该插入。
- 可限制读取的坏行数，超过该数则 **dbload** 结束。

dbload 灵活性的代价是时间和花在创建 **dbload** 命令文件上的精力，这是 **dbload** 运行所要求的。输入文件不作为 **dbload** 命令行的一部分进行指定，也不是要插入数据的表。此信息包含在命令文件中。

3.1 dbload 命令的语法

dbload 命令将数据装入数据库或表。



元素	用途	重要注意事项
-c *command file*	指定 dbload 命令文件的文件名或路径名	**参考：**有关构建命令文件的信息，请参阅 dbload

		实用程序的命令文件。
-d *database*	指定要接收数据的数据库名称	**其他信息：**如果希望使用比数据库的简单名称更多的内容，请参阅《*GBase 8s SQL 指南：语法*》的『数据库名称』一节。
-e *errors*	指定终止前 dbload 读取的坏行数。*errors* 的缺省值为 10。	**参考：**有关更多信息，请参阅装入操作期间坏行限制。
-i *ignore rows*	指定在输入文件中要忽略的行数	**参考：**有关更多信息，请参阅装入操作期间要忽略的行数。
-k	指示 dbload 在装入操作期间，以互斥方式锁定命令文件中列出的表	**参考：**有关更多信息，请参阅装入操作期间表锁定。 您不能将 -k 选项和 -r 选项一起使用，因为 -r 选项指定在装入操作期间不锁定任何表。
-l *error log file*	指定错误日志文件的文件名或路径名	如果指定现有文件，那么将覆盖它的内容。如果您指定的文件不存在，dbload 将创建该文件。 **其他信息：**错误日志文件存储诊断信息以及 dbload 无法插入到数据库中的任何输入文件行。

-n *commit interval*	以行数指定落实间隔 缺省间隔为 100 行。	**其他信息：**如果您的数据库支持事务，dbload 将在读取并插入指定数量的新行后落实事务。每次落实后会出现一条消息。 **参考：**有关事务的信息，请参阅《*GBase 8s SQL 指南：教程*》。
-p	如果坏行数超过限制将提示要求指示信息	**参考：**有关更多信息，请参阅装入操作期间坏行限制。
-r	阻止 dbload 在装入期间锁定表，这样就在装入期间允许其他用户更新表中的数据	**其他信息：**有关更多信息，请参阅装入操作期间表锁定。 您不能将 -r 选项和 -k 选项一起使用，因为 -r 选项指定在装入操作期间不锁定任何表，而 -k 选项指定以互斥方式锁定表。
-s	检查命令文件中的语句语法而不插入数据	**其他信息：**标准输出显示命令文件，并在任何发现错误的位置标识该错误。
-V	显示软件版本号和序列号	无。
-version	扩展 -V 选项以显示有关构建操作系统、构建号和构建日期的其他信息	无。
-X	识别字符字段中的 HEX 二进制数据	无。

提示： 如果指定了一部分（但不是全部）所需信息，**dbload** 将提示

您提供附加规范。数据库名称、命令文件和错误日志文件都是需要的。如果漏掉了所有这三个选项，将接收到错误消息。

dbload 命令示例

以下命令将数据装入名为 **finland** 的数据库服务器上的 **turku** 目录中的 **stores_demo** 数据库：

```
dbload -d //finland/turku/stores_demo -c commands -l errlog
```

3.1.1 装入操作期间表锁定

dbload -k 选项在装入操作期间会覆盖缺省表锁定方式。**-k** 选项指示 **dbload** 以互斥方式而非共享方式锁定表。

如果未指定 **-k** 选项，将以共享方式锁定在命令文件中指定的表。当以共享方式锁定表时，数据库服务器在将行插入表中时仍必须获得行或页互斥锁。

指定 **-k** 选项时，数据库服务器在整个表上放置互斥锁。**-k** 选项提高了大型装入的性能，因为数据库服务器在装入操作期间插入行时不需要获得行或页上的互斥锁定。

如果未指定 **-r** 选项，那么装入期间将锁定在命令文件中指定的表，这样其他用户就不能更新该表中的数据。表锁定减少了装入期间需要的锁数量，但也降低了并发性。如果计划装入大量行，请在非高峰时间使用表锁定并装入。

3.1.2 装入操作期间要忽略的行数

dbload -i 选项指定在 **dbload** 开始处理数据之前，会在输入文件中忽略的换行符的数目。

如果您最近的 **dbload** 会话过早结束，此选项是非常有用的。

例如：如果 **dbload** 在插入 240 行输入后结束，那么如果将忽略行数设置为 240，您可以从第 241 行再次开始装入。

如果输入文件中的头信息在数据记录前面，那么 **-i** 选项也会很有用。

3.1.3 装入操作期间坏行限制

dbload -e 选项使您可以指定在 **dbload** 终止前允许的坏行数。

如果将错误数设置为正整数，那么当 **dbload** 读取（错误数 + 1）个坏行时将终止。如果将错误数设置为 0，那么当 **dbload** 读取第一个坏行时就将终止。

如果 **dbload** 超过了坏行限制，并且指定了 **-p** 选项，**dbload** 在终止前将提示您输入指令。提示将询问您是想回滚还是想落实自上一事务以后插入的所有行。

如果 **dbload** 超过了坏行限制，但未指定 **-p** 选项，**dbload** 将落实自上一事务以后插入的所有行。

3.1.4 终止 dbload 实用程序

如果按中断键，那么 **dbload** 将终止并废弃已插入但还没有落实给数据库的任何新行（如果数据库具有事务）。

3.1.5 dbload 实用程序的名称和对象准则

当使用 **dbload** 实用程序时，您必须遵循指定网络名以及处理简单大对象、索引和定界标识的准则。

表 1. dbload 实用程序的名称和对象准则

对象	准则

网络名	如果已联网，请在数据库名中包含数据库服务器名和目录路径，以指定另一数据库服务器上的数据库。
简单大对象	只要简单大对象在文本文件中，您就可使用 dbload 实用程序装入简单大对象。
索引	索引的存在将大大影响 dbload 实用程序装入数据的速度。为了获得最佳性能，运行 dbload 之前请删除接收数据的表上任何的索引。您可在 dbload 完成后创建新索引。
对象	**准则**
定界标识	可将定界标识用于 dbload 实用程序。该实用程序会检测诸如关键字、混合大小写或具有特殊字符的数据库对象，并在它们周围括上双引号。 如果您最近的 dbload 会话过早结束，请在命令行语法中指定起始行号，以从文件中的下一记录还原装入。

3.2 dbload 实用程序的命令文件

使用 **dbload** 实用程序之前，必须创建一个命令文件，该文件命名输入数据文件以及接收数据的表。命令文件将来自一个或多个输入文件的字段映射到您数据库中的一个或多个表的列。

该命令文件只包含 FILE 和 INSERT 语句。每个 FILE 语句命名一个输入数据文件。FILE 语句还定义来自输入文件的已插入到表中的数据字段。每个 INSERT 语句命名一个用来接收数据的表。INSERT 语句还定义 **dbload** 如何将 FILE 语句中描述的数据放入表列中。

在命令文件中，FILE 语句可以下面的形式出现：

- 定界符格式
- 字符位置格式

FILE 语句有 4,096 字节的大小限制。

当输入数据行中的每个字段使用相同的定界符且每行都以换行字符结束时，请使用定界符格式的 FILE 语句。此格式是典型的带可变长度字段的数据行。只要数据行符合定界符和换行要求，您也可将定界符格式的 FILE 语句用于长度固定的字段。定界符格式的 FILE 和 INSERT 语句比字符位置格式易于使用。

当无法用定界符来进行标识且必须使用输入行中的字符位置来标识输入数据字段时，请使用字符位置格式的 FILE 语句。例如：使用此形式来指示第一输入数据字段从字符位置 1 开始并继续直到字符位置 20。如果必须将字符串转换为空值，您也可使用此形式。例如：如果输入数据文件使用空格序列来指示空值，那么如果您希望指示 **dbload** 在出现空格字符串的每个地方替换为空时，您必须使用此形式。

您可在单个命令文件中使用两种形式的 FILE 语句。但为清楚起见，以下部分将两种形式分开描述。

3.2.1 定界符格式的 FILE 和 INSERT 语句

为 **dbload** 实用程序定义信息的 FILE 和 INSERT 语句会采用定界符格式。

以下 **dbload** 命令文件的示例演示了 FILE 和 INSERT 语句的简单定界符格式。此示例是基于 **stores_demo** 数据库的。一条 UNLOAD 语句创建了三个输入数据文件：**stock.unl**、**customer.unl** 和 **manufact.unl**。

```
FILE stock.unl DELIMITER '|' 6;
INSERT INTO stock;
FILE customer.unl DELIMITER '|' 10;
INSERT INTO customer;
FILE manufact.unl DELIMITER '|' 3;
INSERT INTO manufact;
```

要查看 **.unl** 输入数据文件，请参阅目录 **$GBASEDBTDIR/demo/prod_name**（UNIX™ 、Linux™）

3.2.1.1 定界符格式的语法

定界符格式的语法指定字段定界符、输入文件和每行数据中的字段数。

下图显示了定界符 FILE 语句的语法。

元素	用途	重要注意事项
c	为特定输入文件指定作为字段定界符的字符	如果由 **c** 指定的定界符在输入文件的任何地方以文字字符出现，那么在输入文件中必须在该字符前加上反斜杠 (\)。例如：如果 **c** 的值指定为方括号 ([)，那么您必须在输入文件中出现的任何文字方括号之前放置反斜杠。同样，必须在输入文件中出现的任何反斜杠之前放置附加的反斜杠。
filename	指定输入文件	无。
nfields	指示每个数据行中的字段数	无。

dbload 实用程序将序列名 **f01、f02、f03** 等（依此类推）指定给输入文件中的字段。您看不到这些名称，但是如果您在关联的 INSERT 语句中引用这些字段来指定值列表，那么必须使用 **f01、f02、f03** 格式。有关详细信息，请参阅如何编写定界符格式的 **dbload** 命令文件。

两个连续的定界符定义了一个空字段。作为预防措施，您可紧接在标志每个数据行结束的换行字符前面放置一个定界符。如果数据行的最后字段有数据，那么必须使用定界符。如果您省略此定界符，那么无论何时当数据行的最后字段不为空时都将导致错误。

插入的数据类型对应于显式或缺省列列表。如果数据字段宽度与其对应的字符列宽度不同，将使数据符合列宽度。即，如果插入数据达不到列宽度将对其填充空格，或如果超过列宽度，那么会将其截短。

如果命名的列数少于表中的列数，**dbload** 将向未命名的列插入表创建指定的缺省值。如果未指定缺省值，**dbload** 将尝试插入空值。如果这个尝试违反了

非空约束或唯一约束，那么插入操作将失败且将返回错误消息。

如果 INSERT 语句省略列名，那么 INSERT 缺省指定语句中指定的表中的每一列。如果 INSERT 语句省略了 VALUES 子句，那么 INSERT 缺省指定先前的 FILE 语句的每个字段。

如果列出的（或由缺省隐含的）列名数不匹配列出的（或由缺省隐含的）值数，将导致错误。

dbload INSERT 语句的语法类似于 SQL 中的 INSERT 语句，不同之处在于，在 **dbload** 中，INSERT 语句无法与 SELECT 语句合并使用。

不要在 **dbload** 命令文件中使用 INSERT INTO 语句的 CURRENT、TODAY 和 USER 关键字；它们在 **dbload** 命令文件中不受支持。这些关键字只在 SQL 中受支持。

例如：以下 **dbload** 命令不受支持：

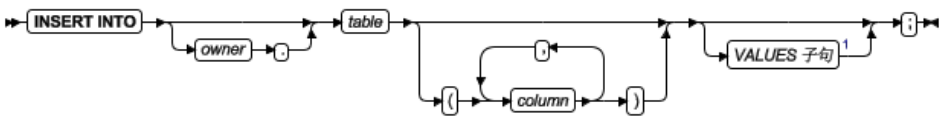
```
FILE "testtbl2.unl" DELIMITER '|' 1;
INSERT INTO testtbl    (testuser, testtime, testfield)
                       VALUES    ('kae', CURRENT, f01);
```

首先装入现有数据，然后写 SQL 查询，以使用当前时间、日期或用户登录来插入或更新数据。您可写以下 SQL 语句：

```
INSERT INTO testtbl    (testuser, testtime, testfield)
                       VALUES    ('kae', CURRENT, f01);
```

CURRENT 关键字返回系统日期和时间。TODAY 关键字返回系统日期。USER 关键字返回用户登录名称。

下图显示了定界符格式的 **dbload** INSERT 语句的语法。



元素	用途	重要注意事项
column	指定接收新数据的列	无。

owner.	指定表所有者的用户名	无。
table	指定接收新数据的表	无。

使用此命令文件运行 **dbload** 的用户必须在命名的表上具有 Insert 特权。

3.2.1.2 如何编写定界符格式的 dbload 命令文件

命令文件必须包含所需的元素，其中包括定界符。

以下示例中的 FILE 语句将 **stock.unl** 数据行描述为包含六个字段，每个字段以竖条 (|) 作为定界符进行分隔。

```
FILE stock.unl DELIMITER '|' 6;
INSERT INTO stock;
```

两个连续的定界符定义了一个空字段。作为预防措施，您可紧接在标志每个数据行结束的换行字符前面放置一个定界符。如果数据行的最后字段有数据，那么必须使用定界符。如果您省略此定界符将导致错误。

将 FILE 语句与以下示例中出现在输入文件 **stock.unl** 中的数据行进行比较。（由于最后的字段后面没有跟定界符，所以如果任何数据行以空字段结束，那么都将导致错误。）

```
1|SMT|baseball gloves|450.00|case|10 gloves/case
2|HRO|baseball|126.00|case|24/case
3|SHK|baseball bat|240.00|case|12/case
```

示例 INSERT 语句只包含需要的元素。由于省略了列的列表，INSERT 语句默认要将值插入 **stock** 表中的每个字段中。由于省略了 VALUES 子句，INSERT 语句默认在最新的 FILE 语句中定义了每个字段的输入值。此 INSERT 语句是有效的，因为 **stock** 表包含六个字段，与 FILE 语句定义的值数量相对应。

以下示例显示了从此 INSERT 语句插入到 **stock** 中的第一个数据行。

字段	列	值

f01	stock_num	1
f02	manu_code	SMT
f03	描述	baseball gloves
字段	**列**	**值**
f04	unit_price	450.00
f05	unit	case
f06	unit_descr	10 gloves/case

以下示例中的 FILE 和 INSERT 语句演示了更加复杂的 INSERT 语句语法：

```
FILE stock.unl DELIMITER '|' 6;
INSERT INTO new_stock (col1, col2, col3, col5, col6)
        VALUES (f01, f03, f02, f05, 'autographed');
```

在此示例中，VALUES 子句使用 **dbload** 自动指定的字段名称。您必须使用字母 f 后跟数字来引用自动指定的字段名称：**f01、f02、f10、f100、f999、f1000** 等等。所有其他格式都是不正确的。

提示： 前九个字段必须包含零：f01、f02、...、f09。

用户更改了列名、数据的顺序以及新的 **stock** 表中 **col6** 的意义。由于 **new_stock** 中的第四列 (**col4**) 未在列列表中进行命名，所以新数据行在 **col4** 位置包含空值（假定该列允许空值）。如果没有为 **col4** 指定缺省值，插入值将为空值。

下表显示了从此 INSERT 语句插入到 **new_stock** 中的第一个数据行。

列	值
col1	1
col2	baseball gloves

col3	SMT
col4	null
col5	Case
col6	Autographed

3.2.2 字符位置格式的 FILE 和 INSERT 语句

为 **dbload** 实用程序定义信息的 FILE 和 INSERT 语句会采用字符位置格式。

本主题中的示例基于输入数据文件 **cust_loc_data**，其包含 **customer** 表的最后四列（**city**、**state**、**zipcode** 和 **phone**）。将输入文件中的字段用空格填充以创建数据行，在这些数据行中，数据字段的位置和字符数在所有数据行中都相等。这些字段的定义分别是 CHAR(15)、CHAR(2)、CHAR(5) 和 CHAR(12)。图 1 显示了字符位置以及 **cust_loc_data** 文件中的五个示例数据行。

图: 样本数据文件

```
                  12                  3
1234567890123456789012345678901234
Sunnyvale       CA94086408-789-8075
Denver          CO80219303-936-7731
Blue Island     NY60406312-944-5691
Brighton        MA02135617-232-4159
Tempe           AZ85253xxx-xxx-xxxx
```

以下 **dbload** 命令文件的示例演示了 FILE 和 INSERT 语句的字符位置格式。示例包含两个新表（**cust_address** 和 **cust_sort**）来接收数据。为了此示例的用途，**cust_address** 包含四列，列的列表省略了第二列。**cust_sort** 表包含两列。

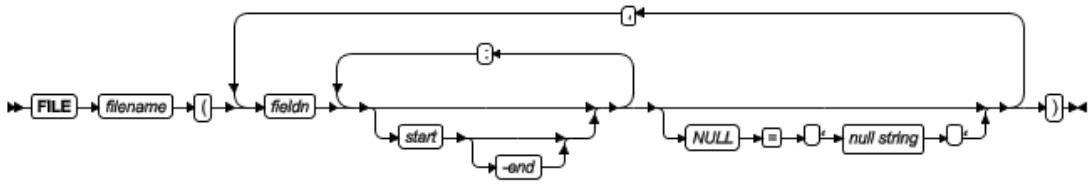
```
FILE cust_loc_data
        (city 1-15,
           state 16-17,
```

```
        area_cd 23-25 NULL = 'xxx',
        phone 23-34 NULL = 'xxx-xxx-xxxx',
        zip 18-22,
        state_area 16-17 : 23-25);
INSERT INTO cust_address (col1, col3, col4)
    VALUES (city, state, zip);
INSERT INTO cust_sort
    VALUES (area_cd, zip);
```

3.2.2.1 字符位置格式的语法

字符位置格式的语法指定一些信息，其中包括数据行中字符位置范围的开始字符位置和结束字符位置。

下图显示了字符位置 FILE 语句的语法。



元素	用途	重要注意事项
-end	指示数据行中结束字符位置范围的字符位置	*end* 值前必须有连字符。
fieldn	为正使用字符位置范围定义的数据字段指定名称	无。
filename	指定输入文件的名称。	无。
null string	指定 dbload 必须以空值替代的数据值	必须是加引号的字符串。
start	指示数据行中开始一定范围的字符位置的字符	无。

	位置。如果您指定 *start* 而没有指定 *end*，那么它将代表单个字符。	

您可在数据字段定义或不同字段中重复相同的字符位置。

引用的 null string 的作用域是您定义它的数据字段。您可为每个允许空条目的字段定义显式的空字符串。

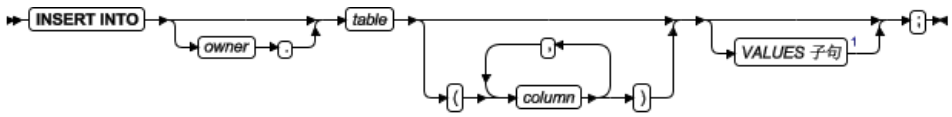
插入的数据类型对应于显式或缺省列列表。在数据字段宽度与其对应的字符列不同时，如果列较宽，那么将在插入值中填充空格；如果字段较宽，那么将会截断插入值。

如果命名的列数少于表中的列数，**dbload** 将插入为未命名的列指定的缺省值。如果未指定缺省值，**dbload** 将尝试插入空值。如果整个尝试违反了非空约束或唯一约束，插入操作将失败且将返回错误消息。

如果 INSERT 语句省略列名，那么 INSERT 缺省指定语句中指定的表中的每一列。如果 INSERT 语句省略了 VALUES 子句，那么 INSERT 缺省指定先前的 FILE 语句的每个字段。

如果列出的（或由缺省隐含的）列名数不匹配列出的（或由缺省隐含的）值数，将导致错误。

dbload INSERT 语句的语法类似于 SQL 中的 INSERT 语句，不同之处在于，在 **dbload** 中，INSERT 语句无法与 SELECT 语句合并使用。下图显示了字符位置格式的 **dbload** INSERT 语句的语法。



元素	用途	重要注意事项
column	指定接收新数据的列	无。

owner.	指定表所有者的用户名	无。
table	指定接收新数据的表	无。

字符位置格式的语法与定界符格式的语法相同。

使用此命令文件运行 **dbload** 的用户必须在命名的表上具有 Insert 特权。

3.2.2.2 如何编写字符位置格式的 dbload 命令文件

命令文件必须定义数据字段，并使用字符位置来定义每个字段的长度。

以下示例中的 FILE 语句定义了 **cust_loc_data** 表数据行中的六个数据字段。

```
FILE cust_loc_data
        (city 1-15,
           state 16-17,
           area_cd 23-25 NULL = 'xxx',
           phone 23-34 NULL = 'xxx-xxx-xxxx',
           zip 18-22,
           state_area 16-17 : 23-25);
INSERT INTO cust_address (col1, col3, col4)
        VALUES (city, state, zip);
```

语句命名了字段并使用字符位置来定义每个字段的长度。将前面示例中的 FILE 语句与下图中的数据行进行比较。

图: 样本数据文件

FILE 语句定义了以下数据字段，它们从样本数据文件中的数据行派生而来。

列	来自数据行 1 的值	来自数据行 2 的值
city	Sunnyvale++++++	Tempe++++++++++
列	来自数据行 1 的值	来自数据行 2 的值
state	CA	AZ
area_cd	408	null
phone	408-789-8075	null
zip	94086	85253
state_area	CA408	AZxxx

为 **phone** 和 **area_cd** 字段定义的空字符串在这些列中生成空值，但不影响存储在 **state_area** 列中的值。

该 INSERT 语句将从 FILE 语句派生而来的字段名和值用作值列表输入。请考虑以下 INSERT 语句：

```
INSERT INTO cust_address (col1, col3, col4)
        VALUES (city, state, zip);
```

该 INSERT 语句使用样本数据文件中的数据，并且 FILE 语句将以下信息放入 **cust_address** 表中。

列	来自数据行 1 的值	来自数据行 2 的值
col1	Sunnyvale++++++	Tempe++++++++++
col2	null	null
col3	CA	AZ
col4	94086	85253

由于没有命名 **cust_address** 中的第二列 **(col2)**，所以新的数据行将包含空值（假定该列允许空值）。

请考虑以下 INSERT 语句：

```
INSERT INTO cust_sort
        VALUES (area_cd, zip);
```

该 INSERT 语句将以下数据行插入 cust_sort 表中。

列	来自数据行 1 的值	来自数据行 2 的值
col1	408	null
col2	94086	85253

由于没有提供列列表，**dbload** 从系统目录读取 **cust_sort** 中所有列的名称。（您不能将数据插入临时表，因为临时表不会进入系统目录。）前面的 FILE 语句的字段名称指定了要装入每列中的值。您无需对每个 INSERT 语句使用一个 FILE 语句。

3.3 装入复杂数据类型的命令文件

您可以创建 dbload 命令文件，将包含复杂数据类型的列装入表。

可将以下数据类型用于 dbload：

- BLOB 或 CLOB
- ROW 类型中的 SET

不可将以下数据类型用于 dbload 实用程序：

- ROW 类型中的 CLOB 或 BLOB
- SET 中的 ROW 类型

要点： 所有装载实用程序（dbexport、dbimport、dbload、gload、gunload 和 onxfer）都依赖导出和导入功能。如果在写用户定义的数据类型时未定义此功能，那么无法使用这些实用程序。

如果数据的表示包含句柄，那么在一个数据类型中装入另一个新的数据类型可能会导致问题。如果字符串表示数据，那么您应可装入它。

您可以将 dbload 与命名行类型、未命名行类型、集合以及列表一起使用。

3.3.1 将 dbload 实用程序用于命名行类型

将命名行类型用于 dbload 实用程序的过程与将其他复杂数据类型用于 dbload 的过程稍有不同，因为命名行类型实际上是用户定义的数据类型。

假定您具有名称为 person 的表，其包含一个带命名行类型的列。再假定 person_t 命名行类型包含六个字段：name、address、city、state、zip 和 bdate。

以下语法显示了如何创建在本示例中使用的命名行类型和表：

```
CREATE ROW TYPE person_t
    (
            name VARCHAR(30) NOT NULL,
            address VARCHAR(20),
            city VARCHAR(20),
            state CHAR(2),
            zip VARCHAR(9),
            bdate DATE
    );
```

```
CREATE TABLE person of TYPE person_t;
```

要为命名行类型（或为任何用户定义的数据类型）装入数据

1. 使用 UNLOAD 语句将表卸载到输入文件。在本示例中，输入文件将命名行类型看成六个单独的字段：

```
Brown, James|13 First St.|San Francisco|CA|94070|01/04/1940|
Karen Smith|1820 Elm Ave #100|Fremont|CA|94502|01/13/1983|
```

2. 使用 **dbschema** 实用程序捕捉表模式和行类型。必须使用 **dbschema -u** 选项来获得命名行类型。

```
dbschema -d stores_demo -u person_t > schema.sql
dbschema -d stores_demo -t person > schema.sql
```

3. 使用 DB-Access 在新数据库中重新创建 **person** 表。

   有关详细步骤，请参阅使用 **dbschema** 输出作为 DB-Access 输入。

4. 创建 **dbload** 命令文件。此 **dbload** 命令文件将两行插入新数据库中的 **person** 表。

```
FILE person.unl DELIMITER '|' 6;
INSERT INTO person;
```

   此 **dbload** 示例显示了如何将新数据行插入 **person** 表。该 INSERT 语句和 **dbload** 命令文件中的行数必须匹配：

```
FILE person.unl DELIMITER '|' 6;
    INSERT INTO person
    VALUES ('Jones, Richard', '95 East Ave.',
                                'Philadelphia', 'PA',
    '19115',
    '03/15/97');
```

5. 运行 **dbload** 命令：

```
dbload -d newdb -c uds_command -l errlog
```

提示： 要找到包含命名行类型的卸载表中的字段数，可计算每个竖条（|）定界符之间的字段数。

3.3.2 将 dbload 实用程序用于未命名行类型

您可以将未命名行类型用于 dbload 实用程序，这些类型是使用 ROW 构造函数创建的，并会定义列或字段的类型。

在以下示例中，devtest 表包含两个带未命名行类型的列 s_name 和 s_address。 s_name 列包含三个字段：f_name、m_init 和 l_name。s_address 列包含四个字段：street、city、state 和 zip。

```
CREATE TABLE devtest
(
s_name ROW(f_name varchar(20), m_init char(1), l_name varchar(20)
not null),
s_address ROW(street varchar(20), city varchar(20), state char(20),
zip varchar(9)
);
```

来自 devtest 表的数据卸载到 devtest.unl 文件。每个数据行包含两个定界字段，每个未命名行类型一个。ROW 构造函数放在每个未命名行类型之前，如下：

```
ROW('Jim','K','Johnson')|ROW('10 Grove St.','Eldorado','CA','94108')|
ROW('Maria','E','Martinez')|ROW('2387 West Wilton
Ave.','Hershey','PA','17033')|
```

此 dbload 示例显示了如何将包含未命名行类型的数据插入 devtest 表。在每个未命名行类型两边放上双引号，否则插入无法工作。

```
FILE devtest.unl DELIMITER '|' 2;
```

```
INSERT INTO devtest (s_name, s_address)
VALUES ("row('Stephen', 'M', 'Wu')",
    "row('1200 Grand Ave.', 'Richmond', 'OR', '97200')");
```

3.3.3 将 dbload 实用程序用于集合数据类型

您可以将集合数据类型（例如 SET、LIST 和 MULTISET）用于 dbload 实用程序。

3.3.3.1 SET 数据类型示例

SET 数据类型是存储唯一元素的无序集合类型。SET 数据类型中的元素个数可以变化，但不允许空值。

以下语句创建了一个表，在该表中，children 列定义为 SET：

```
CREATE TABLE employee
    (
            name char(30),
            address char(40),
            children SET (varchar(30) NOT NULL)
    );
```

来自 employee 表的数据卸载到 employee.unl 文件。每个数据行包含四个定界的字段。 第一个集合包含三个元素（Karen、Lauren 和 Andrea），然而第二个集合包含四个元素。SET 构造函数放在每个 SET 数据行之前。

```
Muriel|5555 SW Merry
Sailing Dr.|02/06/1926|SET{'Karen','Lauren','Andrea'}|
        Larry|1234 Indian Lane|07/31/1927|SET{'Martha',
            'Melissa','Craig','Larry'}|
```

此 dbload 示例显示了如何将包含 SET 数据类型的数据插入新数据库中的 employee 表中。在每个 SET 数据类型两边放上双引号，否则插入无法工作。

```
FILE employee.unl DELIMITER '|' 4;
INSERT INTO employee
VALUES ('Marvin', '10734 Pardee', '06/17/27',
        "SET{'Joe', 'Ann'}");
```

3.3.3.2 LIST 数据类型示例

LIST 数据类型是存储有序的非唯一元素的集合类型；也就是说，它允许元素值重复。

以下语句创建了一个表，在该表中，month_sales 列定义为 LIST：

```
CREATE TABLE sales_person
    (
            name CHAR(30),
            month_sales LIST(MONEY NOT NULL)
    );
```

来自 sales_person 表的数据卸载到 sales.unl 文件。每个数据行包含两个定界的字段，如下：

```
Jane Doe|LIST{'4.00','20.45','000.99'}|
Big Earner|LIST{'0000.00','00000.00','999.99'}|
```

此 dbload 示例显示了如何将包含 LIST 数据类型的数据插入新数据库中的 sales_person 表中。在每个 LIST 数据类型两边使用双引号，否则插入操作无法进行。

```
FILE sales_person.unl DELIMITER '|' 2;
INSERT INTO sales_person
VALUES ('Jenny Chow', "{587900, 600000}");
```

您可以用类似的方式装入多个集合。

4 dbschema 实用程序

dbschema 实用程序显示复制数据库对象所需的 SQL 语句（模式）。

您也可以将 dbschema 实用程序用于以下用途：

- 显示 UPDATE STATISTICS 语句创建的分发。
- 显示 Information Schema 视图的模式
- 显示用于创建对象（例如数据库、表、序列、同义词、存储空间、块、日志、角色和特权）的模式
- 显示用于创建对象（例如数据库、表、序列、同义词、角色和特权）的模式
- 显示为数据库中的一个或多个表存储的分发信息
- 显示有关用户定义的数据类型和行类型的信息

在获取数据库的模式之后，您可以将 dbschema 输出重定向到通过 DB-Access 使用的文件。

如果配置了 UPDATABLE_SECONDARY 配置参数和 STOP_APPLY 配置参数，那么所有只读辅助服务器上都支持 dbschema 实用程序。

所有可更新的辅助服务器上都支持 dbschema 实用程序。

只读辅助服务器上也支持 dbschema 实用程序。但是，dbschema 实用程序在这些服务器上运行时，会显示一条警告消息。

 **注意：** 使用 dbschema 实用程序可以增加数据库中的序列对象，而在生成的数字中创建间隔则可能不是那些需要序列化整数的应用程序所期望的。

支持 Oracle 模式之后 dbschema 实用程序增加-o 参数,-o 对象名(例如：-o pkg1）导出指定数据库对象，-o all 导出 ORACLE 模式下创建的所有包、存储过程、函数等数据库对象，导出内容格式中不再包括 owner.引用。导入时使用方法与目前用法保持一致。

Oracle 模式下创建的表、视图、触发器等数据库对象相应导出内容格式以 Oracle 为准，不再包括 owner. 引用。GBase 模式下创建的数据库对象依旧包含 owner. 引用。导出语法为原生或 Oracle 兼容取决于创建对象时所在模式，并且将导出对应 set environment sqlmode 模式设置语句。

4.1 dbschema 输出中的对象方式和违例检测

dbschema 实用程序的输出显示对象方式并支持违例检测。

dbschema 输出显示：

- 非空规范后的非空约束的名称。
- 可使用该实用程序的输出作为输入来创建另一数据库。如果在两个数据库中没有对非空约束使用相同名称，那么可能导致问题。
- 处在禁用状态的对象的对象方式。这些对象可以是约束、触发器或索引。
- 处在过滤状态的对象的对象方式。这些对象可以是约束或唯一索引。
- 与基本表关联的违例和诊断表（如果违例和诊断表是为基本表启动的）。

有关对象方式和违例检测的更多信息，请参阅《GBase 8s SQL 指南：语法》中的 SET、START VIOLATIONS TABLE 和 STOP VIOLATIONS TABLE 语句。

4.2 使用 dbschema 实用程序的准则

可将定界标识用于 dbschema 实用程序。dbschema 实用程序检测诸如关键字、混合大小写或具有特殊字符的数据库对象，并将这些数据库对象括在双引

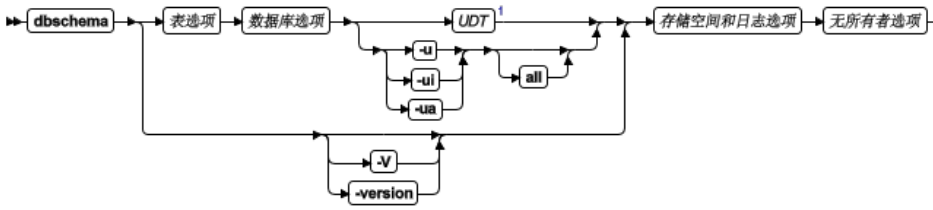
号中。

Global Language Support： 在使用 dbschema 实用程序之前，您必须禁用 SELECT 触发器并且正确设置 GLS 环境变量。

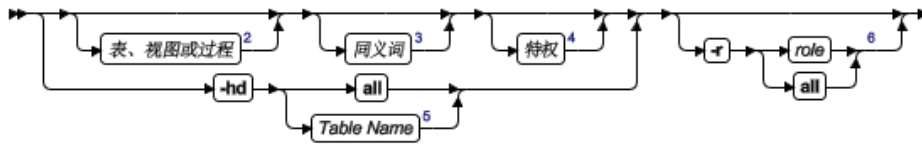
当正确设置了 GLS 环境变量后（如《GBase 8s GLS 用户指南》所述），dbschema 实用程序就可以处理外来字符。

4.3 dbschema 命令的语法

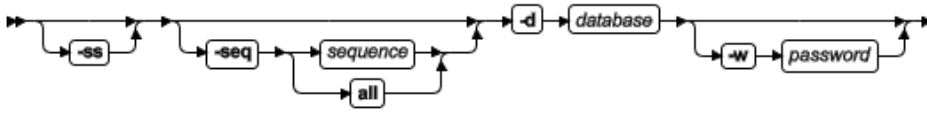
dbschema 命令显示了复制指定数据库对象所需的 SQL 语句（模式）。该命令还显示 UPDATE STATISTICS 语句创建的分发。



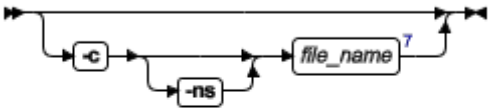
表选项



数据库选项



存储空间和日志选项



无所有者选项

元素	用途	更多信息
all	指示 dbschema 包含数据库中的所有表或序列对象，或分发显示中的所有用户定义的数据类型	无。
元素	**用途**	**更多信息**
-c *file_name*	生成用于复制存储空间、块、物理日志和逻辑日志的命令。	如果使用 -c 元素而不使用 -ns 元素，数据库服务器会生成 SQL 管理 API 命令。 如果同时使用 -c 元素和 -ns 元素，数据库服务器会生成 gspaces 或 glogadmin 命令。

-d *database*	指定模式应用于的数据库。数据库可在远程数据库服务器上。	**参考：**如果希望限定*数据库*的名称，请参阅《*GBase 8s SQL 指南：语法*》的“数据库名称”主题。
filename	指定包含 dbschema 输出的文件的名称	如果省略文件名，dbschema 会将输出发送到屏幕。如果指定文件名，dbschema 将创建名为 *filename* 的文件来包含 dbschema 输出。
-hd	将分发显示为数据值	如果对表名指定了 ALL 关键字，那么将显示数据库中所有表的分发。
元素	**用途**	**更多信息**
-it	设置在 dbschema 查询目录表时用于 dbschema 的隔离类型。隔离类型为： • DR = 脏读 • CR = 已落实读 • CS = 游标稳定性 • CRU = 具有 RETAIN® UPDATE LOCKS 的已落实读 • CSU = 具有 RETAIN UPDATE LOCKS 的游	此选项不会显示其他任何信息。

	标稳定性 • DRU = 具有 RETAIN UPDATE LOCKS 的脏读 • LC = 已落实读（上次落实） • RR = 可重复读	
-l	将锁定方式设置为在 dbschema 查询目录表时等待 dbschema 的秒*数*。	此选项不会显示其他任何信息。
-ns	生成用于复制存储空间、块、物理日志和逻辑日志的 gspaces 或 glogadmin 实用程序命令。	在命令中，-c 元素必须放在 -ns 元素之前。
-nw	生成用于在未指定所有者的情况下创建对象的 SQL。	-nw 元素也是 dbexport 命令选项。
-q	禁止头中的数据库版本。	此可选元素放在其他元素之前。
-r	生成有关角色创建的信息。	有关详细信息，请参阅角色创建。
元素	**用途**	**更多信息**
-seq*sequence*	生成 DDL 语句来定义指定的序列对象	无。
-ss	生成特定于服务器的信息	如果没有生成表模式，那么将忽略此选项。
-si	为非分段表排除索引存储子句的生成	此选项只有和 -ss 选项一起使用时才可用。
-sl *length*	指定未格式化的 CREATE TABLE 和 ALTER TABLE 语句的最大长度（以字节为单位）。	注意
-u	打印函数、强制转型和用户定义的数据类型的定义	指定 -u all 以包含分发列表中的所有表。

-ua	打印用户定义的数据类型（包括对某个数据类型定义的所有函数和类型）的定义。	无。
-ui	打印用户定义的数据类型的定义（包括类型继承）	无。
-V	显示软件版本号和序列号	无。
-version	扩展 -V 选项以显示有关构建版本、主机、操作系统、构建号和构建日期以及 GLS 版本的其他信息。	无。
-w *password*	指定数据库密码（如果有）。	

您必须是 DBA 或对数据库具有 Connect 或 Resource 特权，才能对数据库运行 **dbschema**。

示例

以下命令将生成具有 customer 数据库中所有表或序列对象的模式，但不指定所有者：

```
dbschema –d customer all -nw
```

4.3.1 数据库模式创建

您可创建整个数据库的模式或数据库的一部分的模式。

使用 dbschema 实用程序选项可执行以下操作：

- 对于特定的表或整个数据库，按所有者显示 CREATE SYNONYM 语句。
- 对于特定的表或整个数据库，显示 CREATE TABLE、CREATE VIEW、CREATE FUNCTION 或 CREATE PROCEDURE 语句。
- 对于数据库或特定表，显示影响指定用户或影响所有用户的所有 GRANT 特权语句。用户可以是用户名或角色名。
- 显示用户定义的数据类型和行数据类型（带或不带类型继承）。

- 显示定义指定序列 对象或定义数据库中所有序列对象的 CREATE SEQUENCE 语句。

使用 dbschema 且只指定了数据库名称时，等同于使用带所有选项（除了 -hd 和 -ss 选项）的 dbschema。另外，如果为数据库创建了“信息模式”视图，那么将显示此模式。例如：以下两个命令是等价的：

```
dbschema -d stores_demo
dbschema -s all -p all -t all -f all -d stores_demo
```

dbschema 所显示的 CREATE TABLE 语句包含的 SERIAL 字段不指定起始值。使用模式文件创建的新 SERIAL 字段具有起始值 1 ，不论原始数据库中它们的起始值是多少都是如此。如果起始值不可接受，那么必须修改模式文件。

4.3.1.1 在 UNIX 或 Linux 网络中创建数据库的模式

dbschema -d 选项用于在 UNIX™ 或 Linux™ 网络上创建并显示数据库的模式。

可以在任何可访问的非 SE GBase 8s 数据库服务器上指定数据库。

以下命令显示了 stores_demo 数据库的模式，该数据库在 UNIX 或 Linux 系统控制台上的 finland 数据库服务器上：

```
dbschema -d //finland/stores_demo
```

4.3.1.2 更改对象所有者

您可以编辑 **dbschema** 输出以更改新对象的所有者。

当 **dbschema** 实用程序生成任何 CREATE TABLE、CREATE INDEX、CREATE SYNONYM、CREATE VIEW、CREATE SEQUENCE、CREATE

PROCEDURE、CREATE FUNCTION 或 GRANT 语句并重新生成任何唯一、引用或检查约束时，它使用 owner.object 约定。结果，如果您使用 **dbschema** 输出来创建新对象（表、索引、视图、过程、约束、序列或同义词），原对象的所有者将拥有新对象。如果希望更改新对象的所有者，那么在将 **dbschema** 输出作为 SQL 脚本运行前，必须编辑它。

如果还指定了存储编译时警告的文件的路径名，那么可使用 **dbschema** 的输出来创建新的函数。此路径名将显示在 **dbschema** 输出中。

有关 CREATE TABLE、CREATE INDEX、CREATE SYNONYM、CREATE VIEW、CREATE SEQUENCE、CREATE PROCEDURE、CREATE FUNCTION 和 GRANT 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

4.3.2 dbschema 特定于服务器的信息

dbschema -ss 选项生成特定于服务器的信息。在除 SE 以外的所有 GBase 8s 数据库服务器中，**-ss** 选项总是生成锁定方式、扩展数据块大小以及数据库空间名称（如果数据库空间名称与数据库的数据库空间不同）。另外，如果表被分段，那么 **-ss** 选项将显示有关分段存储策略的信息。

指定 **dbschema -ss** 选项时，输出还会显示为特定用户或在整个模式中发出的所有 GRANT FRAGMENT 语句。

只有和 **-ss 选项**一起使用时才可用的 -si 选项为非分段表排除索引存储子句的生成。

如果数据库空间包含多重分区，将在输出中显示数据库空间分区名称。

有关分段级权限的信息，请参阅《GBase 8s SQL 指南：语法》中的 GRANT FRAGMENT 和 REVOKE FRAGMENT 语句。

4.3.3 用户定义的数据类型和复杂数据类型

dbschema -u 选项将显示数据库包含的任何用户定义数据类型和复杂数据类型的定义。子选项 **i** 将类型继承添加到 **dbschema -u** 选项显示的信息中。

以下命令显示 **stork** 数据库的所有用户定义数据类型和复杂数据类型：

```
dbschema -d stork -u all
```

使用指定选项 -u all 运行的 **dbschema** 的输出可能如以下示例所示：

```
create row type 'gbasedbt'.person_t
    (
        name varchar(30, 10) not null,
        address varchar(20, 10),
        city varchar(20, 10),
        state char(2),
        zip integer,
        bdate date
    );
create row type 'gbasedbt'.employee_t
    (
        salary integer,
        manager varchar(30, 10)
        ) under person_t;
```

以下命令显示了 **stork** 数据库中 **person_t** 表的用户定义数据类型和复杂数据类型以及它们的类型继承：

```
dbschema -d stork -ui person_t
```

使用选项 -ui person_t 运行的 **dbschema** 的输出可能如以下示例所示：

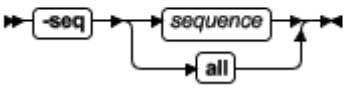
```
create row type 'gbasedbt'.person_t
    (
```

```
        name varchar(30, 10) not null,
        address varchar(20, 10),
        city varchar(20, 10),
        state char(2),
        zip integer,
        bdate date
    );
create row type 'gbasedbt'.employee_t
    (
        salary integer,
        manager varchar(30, 10)
        ) under person_t;
create row type 'gbasedbt'.sales_rep_t
    (
        rep_num integer,
        region_num integer,
        commission decimal(16),
        home_office boolean
        ) under employee_t;
```

4.3.4 序列创建

dbschema -seq sequence 命令会生成有关序列创建的信息。

以下语法图分段显示序列创建。



元素	用途	重要注意事项
-seq *sequence*	显示定义 *sequence* 的 CREATE SEQUENCE 语句	无。
-seq all	显示数据库的所有 CREATE SEQUENCE 语句	无。

使用选项 **-seq** sequitur 运行 **dbschema** 可能产生下面的输出：

```
CREATE SEQUENCE sequitur INCREMENT 10 START 100 NOCACHE CYCLE
```

有关 CREATE SEQUENCE 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

4.3.5 同义词创建

dbschema -s 命令会生成有关同义词创建的信息。

以下语法图分段显示同义词的创建。

同义词

元素	用途	重要注意事项
-s *ownername*	显示 *ownername* 拥有的 CREATE SYNONYM 语句	无。
-s all	显示指定数据库、表或视图的所有 CREATE SYNONYM 语句	无。

使用指定选项 -s alice 运行的 **dbschema** 的输出可能如以下示例所示：

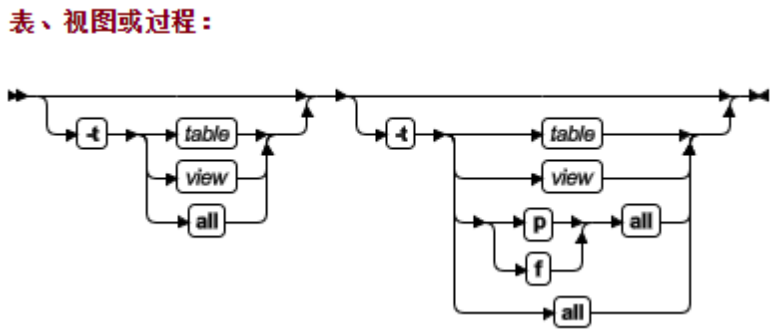
```
CREATE SYNONYM 'alice'.cust FOR 'alice'.customer
```

有关 CREATE SYNONYM 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

4.3.6 表、视图或过程创建

几个 **dbschema** 选项可以生成有关创建表、视图和过程的信息。

以下语法图显示表、视图和过程的创建。



元素	用途	重要注意事项
-f all	将 SQL 语句的输出仅限于复制所有函数和过程	无。

	的那些语句	
-f *function*	将 SQL 语句的输出仅限于复制指定函数的那些语句	无。
-f *procedure*	将 SQL 语句的输出仅限于复制指定过程的那些语句	无。
-ff all	将 SQL 语句的输出仅限于复制所有函数的那些语句	无。
-fp all	将 SQL 语句的输出仅限于复制所有过程的那些语句	无。
-t *table*	将 SQL 语句的输出仅限于复制指定表的那些语句	无。
-t *view*	将 SQL 语句的输出仅限于复制指定视图的那些语句	无。
-t all	在 SQL 语句的输出中包含复制所有表和视图的所有语句	无。

有关 CREATE PROCEDURE 语句和 CREATE FUNCTION 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

4.3.7 表信息

dbschema -ss 命令会检索有关分段表、锁定方式以及扩展数据块大小的信息。

以下 **dbschema** 输出显示了为分段表指定的表达式。

```
{ TABLE “sallyc”.t1 row size = 8 number of columns = 1 index size =
0 }
```

```
create table “sallyc”.t1
(
c1 integer
) fragment by expression
(c1 < 100 ) in db1 ,
((c1 >= 100 ) AND (c1 < 200 ) ) in db2 ,
remainder in db4
extent size 16 next size 16 lock mode page;
revoke all on “sallyc”.t1 from “public”;
```

以下 **dbschema** 输出显示有关分区分段表中的分区的信息。

```
DBSCHEMA Schema Utility grant dba to "sqlqa";
{ TABLE "sqlqa".t1 row size = 24 number of columns = 2 index size =
13 }
create table "sqlqa".t1
(
c1 integer,
c2 char(20)
)
fragment by expression
partition part_1 (c1 = 10 ) in dbs1 ,
partition part_2 (c1 = 20 ) in dbs1 ,
partition part_3 (c1 = 30 ) in dbs1 ,
partition part_4 (c1 = 40 ) in dbs1 ,
partition part_5 (c1 = 50 ) in dbs1
extent size 16 next size 16 lock mode page;
```

4.3.8 存储空间、块和日志创建

dbschema -c 命令会生成 SQL 管理 API 命令，用于复制存储空间、块、逻辑日志和物理日志。如果使用 **dbschema** -c -ns 命令，数据库服务器会生成 **gspaces** 或 **glogadmin** 实用程序命令，用于复制存储空间、块、物理日志和逻辑日志。

例如：

- 运行以下命令可生成名为 **dbschema**1.out 的文件，其中包含用于以 SQL 管理 API 格式复制存储空间、块、物理日志和逻辑日志的命令：

  ```
  dbschema -c dbschema1.out
  ```

- 运行以下命令可生成名为 **dbschema**2.out 的文件，其中包含用于以 **gspaces** 和 **glogadmin** 实用程序格式复制存储空间、块、物理日志和逻辑日志的命令：

  ```
  dbschema -c -ns dbschema2.out
  ```

（可选）运行命令时，先指定 **-q**，再指定 -c 或 -c -ns，以禁止数据库版本。例如，指定：

```
dbschema -q -c -ns dbschema3.out
```

4.3.8.1 创建存储空间、块和日志的样本输出

dbschema -c 或 **dbschema -c -ns** 命令的输出包含所有 SQL 管理 API 或 **gspaces** 和 **glogadmin** 实用程序命令，可用于复制存储空间、块和日志。

SQL 管理 API 格式的输出示例

```
# Dbspace 1 -- Chunk 1
EXECUTE FUNCTION TASK ('create dbspace', 'rootdbs',
 '/export/home/gbasedbt/data/rootdbs1150fc4', '200000',
```

```
 '0', '2', '500', '100')

# Dbspace 2 -- Chunk 2
EXECUTE FUNCTION TASK ('create dbspace', 'datadbs1',
 '/export/home/gbasedbt/data/datadbs1150fc4', '5000000',
 '0', '2', '100', '100')

# Dbspace 3 -- Chunk 3
EXECUTE FUNCTION TASK ('create dbspace', 'datadbs2',
 '/export/home/gbasedbt/data/datadbs2150fc4', '5000000',
 '0', '2', '100', '100')

# Dbspace 4 -- Chunk 4
EXECUTE FUNCTION TASK ('create dbspace', 'datadbs3',
 '/export/home/gbasedbt/data/datadbs3_1150fc4', '80000',
 '16', '8', '400', '400')
EXECUTE FUNCTION TASK ('start mirror', 'datadbs3',
 '/export/home/gbasedbt/data/datadbs3_1150fc4', '80000',
 '16', '/export/home/gbasedbt/data/mdatadbs3_1150fc4', '16')

# Dbspace 5 -- Chunk 5
EXECUTE FUNCTION TASK ('create tempdbspace', 'tempdbs',
 '/export/home/gbasedbt/data/tempdbs_1150fc4', '1000',
 '0', '2', '100', '100')

# Dbspace 6 -- Chunk 6
EXECUTE FUNCTION TASK ('create sbspace', 'sbspace',
```

```
 '/export/home/gbasedbt/data/sbspace_1150fc4',
 '1000', '0')

# Dbspace 6 -- Chunk 7
EXECUTE FUNCTION TASK ('add chunk', 'sbspace',
 '/export/home/gbasedbt/data/sbspace_1_1150fc4',
 '1000', '0')

# Dbspace 7 -- Chunk 8
EXECUTE FUNCTION TASK ('create blobspace', 'blobdbs',
 '/export/home/gbasedbt/data/blobdbs_1150fc4',
 '1000', '0', '4')

# External Space 1
EXECUTE FUNCTION TASK ('create extspace', 'extspace',
'/export/home/gbasedbt/data/extspac_1150fc4')

# Physical Log
EXECUTE FUNCTION TASK ('alter plog', 'rootdbs', '60000')

# Logical Log 1
EXECUTE FUNCTION TASK ('add log', 'rootdbs', '10000')
```

gspaces 和 glogadmin 实用程序格式的输出示例

```
# Dbspace 1 -- Chunk 1
gspaces -c -d rootdbs -k 2 -p
 /export/home/gbasedbt/data/rootdbs1150fc4
```

```
 -o 0 -s 200000 -en 500 -ef 100


# Dbspace 2 -- Chunk 2
gspaces -c -d datadbs1 -k 2 -p
 /export/home/gbasedbt/data/datadbs1150fc4
 -o 0 -s 5000000 -en 100 -ef 100


# Dbspace 3 -- Chunk 3
gspaces -c -d datadbs2 -k 2 -p
 /export/home/gbasedbt/data/datadbs2150fc4
 -o 0 -s 5000000 -en 100 -ef 100


 Dbspace 4 -- Chunk 4
gspaces -c -d datadbs3 -k 8
 -p /export/home/gbasedbt/data/datadbs3_1150fc4
 -o 16 -s 80000 -en 400 -ef 400
 -m /export/home/gbasedbt/data/mdatadbs3_1150fc4 16


# Dbspace 5 -- Chunk 5
gspaces -c -d tempdbs -k 2 -t -p
 /export/home/gbasedbt/data/tempdbs_1150fc4 -o 0 -s 1000


# Dbspace 6 -- Chunk 6
gspaces -c -S sbspace -p
 /export/home/gbasedbt/data/sbspace_1150fc4
 -o 0 -s 1000 -Ms 500
```

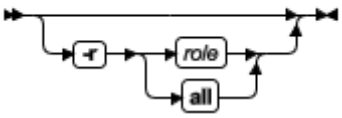
```
# Dbspace 7 -- Chunk 7
gspaces -c -b blobdbs -g 4 -p
 /export/home/gbasedbt/data/blobdbs_1150fc4 -o 0 -s 1000

# External Space 1
gspaces -c -x extspace -l
 /export/home/gbasedbt/data/extspac_1150fc4

# Logical Log 1
glogadmin -a -d rootdbs -s 10000
```

4.3.9 角色创建

dbschema -ss 命令会生成有关角色创建的信息。

以下语法图显示角色的创建。

角色



元素	用途	重要注意事项
-r *role*	显示复制和授权指定角色所需的 CREATE ROLE 和 GRANT 语句。	不能使用-r 选项指定用户或角色的列表。您可指定一个角色或所有角色。SE 不支持 -r 选项。
-r all	显示复制和授权所有角	无

	色所需的所有 CREATE ROLE 和 GRANT 语句。	

以下 **dbschema** 命令和输出显示创建了角色 **calen** 并被授予 **cathl**、**judith** 和 **sallyc**：

```
sharky% dbschema -r calen -d stores_demo


DBSCHEMA Schema 实用程序
软件序列号 RDS#N000000
create role calen;


grant calen to cathl with grant option;
grant calen to judith ;
grant calen to sallyc ;
```

4.3.10 特权

dbschema -p 命令会生成有关特权的信息。

以下语法图分段显示特权信息。

特权

元素	用途	重要注意事项
-p *user*	显示授予特权给 *user* 的 GRANT 语句，其中 *user* 是用户名或角色名。只指定一个	不能使用 -p 选项指定特定的用户列表。可指定一个用户或角色或指定所有用户或角色。

	用户或角色	
-p all	显示指定数据库、表或视图的所有用户或指定表的所有角色的 GRANT 语句。	无。

输出还显示对指定用户或角色或（使用 **all** 选项时）整个模式发出的所有 GRANT FRAGMENT 语句。

4.3.10.1 授予特权

您可以生成有关 GRANT 语句授权者的 **dbschema** 信息。

在 **dbschema** 输出中，AS 关键字指示 GRANT 语句的授权者。以下示例输出指示 **norma** 发出了 GRANT 语句：

```
GRANT ALL ON 'tom'.customer TO 'claire' AS 'norma'
```

当 **dbschema** 输出中出现 GRANT 和 AS 关键字时，您可能需要授予特权，才能将 **dbschema** 输出作为 SQL 脚本运行。根据前面的示例输出行，以下条件必须为真，您才能将该语句作为脚本的一部分运行：

- 用户 **norma** 必须具有到数据库的 Connect 特权。
- 用户 **norma** 必须具有表 **tom.customer** 的所有特权 WITH GRANT OPTION。

有关 GRANT、GRANT FRAGMENT 和 REVOKE FRAGMENT 语句的更多信息，请参阅《GBase 8s SQL 指南：语法》。

4.3.10.2 显示角色的特权信息

您可以生成有关为特定角色所授予的特权的 **dbschema** 信息。

角色是在授权给该角色的数据库对象上的拥有特权的分类。DBA 可对某个

角色指定相关工作任务的特权（例如工程师），然后将该角色授权给用户，而不是将相同的特权集授予每个用户。创建角色后，DBA 就可使用 GRANT 语句将角色授权给用户或授权给其他角色。

例如，使用以下 **dbschema** 命令显示授予给 **calen** 角色的特权。

```
sharky% dbschema -p calen -d stores_demo
```

dbschema 实用程序显示的信息的示例为：

```
grant alter on table1 to 'calen'
```

4.3.11 dbschema 输出中表的分发信息

带有表名的 **dbschema** -hd 命令会检索为数据库表存储的分发信息。如果对表名指定了 ALL 关键字，那么将显示数据库中所有表的分发。

在 **dbimport** 操作执行过程中会为非不透明列中的主要索引自动创建分发信息。以 MEDIUM 或 HIGH 方式运行 UPDATE STATISTICS 语句，以创建有关包含以下索引类型的表的分发信息：

- Virtual Index Interface (VII) 或功能索引
- 用户定义数据类型的列上的索引
- 内置的不透明数据类型的列上的索引（例如 BOOLEAN 或 LVARCHAR）

当 UPDATE STATISTICS 以 MEDIUM 或 HIGH 方式对表运行时，如果您使用了 SAMPLING SIZE 关键字，那么 **dbschema** 实用程序的输出将显示分发信息。

有关 UPDATE STATISTICS 语句的信息，请参阅《GBase 8s SQL 指南：语法》。

分发的 **dbschema** 的输出在以下部分中提供：

- 分发描述

- 分发信息
- 溢出信息

dbschema 输出的各部分将在以下各节解释。作为示例，讨论使用以下名为 **invoices** 的虚构表的分发。此表包含 165 行（包括重复的行）。

您可使用与以下示例类似的方法调用 **dbschema** 生成本讨论的输出：

```
dbschema -hd invoices -d pubs_stores_demo
```

4.3.11.1 显示分发信息的 dbschema 输出示例

dbschema 输出可以显示已为指定表创建的数据分发，以及生成分发的 UPDATE STATISTICS 语句运行的日期。

以下 **dbschema** 输出的示例显示分发信息。

```
cathl.invoices.invoice_num 的分发

High 方式，10.000000 分辨率

--- DISTRIBUTION ---

      (                    5)
1: (  16,       7,        11)
2: (  16,       6,        17)
3: (  16,       8,        25)
4: (  16,       8,        38)
5: (  16,       7,        52)
6: (  16,       8,        73)
7: (  16,      12,        95)
```

```
      8: (    16,       12,                 139)
      9: (    16,       11,                 182)
     10: (    10,        5,                 200)

--- OVERFLOW ---

      1: (     5,                  56)
      2: (     6,                  63)
}
```

示例中分发信息的描述

dbschema 输出示例的第一部分描述了已为指定表创建了何种数据分发。以下示例中指出了表的名称：

```
cathl.invoices.invoice_num 的分发
```

输出是针对 **invoices** 表的，该表由用户 **cathl** 所有。本数据分发描述了列 invoice_num。如果表具有构建在多个列上的分发，**dbschema** 将单独地列出每个列的分发。

dbschema 输出还显示运行 UPDATE STATISTICS 语句（用于生成分发）的日期。您可使用此日期得知您的分发已过时多久。

输出的描述部分的最后一行描述了创建分发的方式（MEDIUM 或 HIGH）以及分辨率。如果使用 Medium 方式创建分发，还将列出样本的可信度。例如：如果使用 HIGH 方式、分辨率 10 运行 UPDATE STATISTICS 语句，最后一行将如以下示例所示：

```
High 方式，10.000000 分辨率
```

4.3.11.2 dbschema 输出中的分发信息

dbschema 输出中的分发信息描述了为分发创建的二进制文件，表中和每

个二进制文件中的值的范围以及每个二进制文件中的不同值的数量。

请考虑以下示例：

```
(                                5)
    1: (    16,         7,              11)
    2: (    16,         6,              17)
    3: (    16,         8,              25)
    4: (    16,         8,              38)
    5: (    16,         7,              52)
    6: (    16,         8,              73)
    7: (    16,        12,              95)
    8: (    16,        12,             139)
    9: (    16,        11,             182)
   10: (    10,         5,             200)
```

最右边列中的第一个值是此列中的最小值。在本示例中，它是 5 。

左边的列显示了二进制文件号，在本示例中是 1 到 10。括号中的第一个数字显示了二进制文件中有多少值。对于该表，是总行数 (165) 的 10%，四舍五入为 16。所有二进制文件（除了最后一个二进制文件）的第一个数字都相同。最后一行可能具有较小的值，指示它不具有一样多的行值。在本示例中，所有二进制文件（除了最后一个二进制文件）都包含 16 行，最后一个二进制文件包含 10 行。

括号中的中间列指示此二进制文件中包含多少不同的值。因此，如果 16 个值的二进制文件中有 11 个不同的值，它暗示这些值中的一个或多个至少重复了一次。

括号中的右边列是二进制文件中的最大值。最后一个二进制文件中的最大值也是表中的最大值。对于本示例，最后一个二进制文件中的最大值为 200 。

4.3.11.3 dbschema 输出中的溢出信息

dbschema 输出的最后部分显示了具有许多重复的值。

所指示的值的重复次数必须大于一个临界量，该临界量大约分辨率的 25% 乘以行数。如果留在一般分发数据中，重复使分发倾斜，所以将它们从分发移到一个单独的列表中，如以下示例所示：

```
--- OVERFLOW ---


        1: (     5,                  56)
        2: (     6,                  63)
```

对于本示例，临界量是 0.25 * 0.10 * 165 或 4.125。因此，任何重复 5 次或更多次的值都将列在溢出部分。此分发中的两个值在表中重复了 5 次或更多次：值 56 重复了 5 次，而值 63 重复了 6 次。

4.4 使用 dbschema 输出作为 DB-Access 输入

您可使用 dbschema 实用程序来获取数据库的模式并将 dbschema 输出重定向到文件。稍候，您可以将文件导入 DB-Access，并使用 DB-Access 在新数据库中重新创建模式。

4.4.1 将表插入到 dbschema 输出文件

您可以将 CREATE TABLE 语句插入 **dbschema** 输出文件，并将此输出用作 DB-Access 输入。

以下示例将 customer 表的 CREATE TABLE 语句复制到 **dbschema** 输出文件 **tab.sql** 中：

```
dbschema -d db -t customer > tab.sql
```

从输出文件 **tab.sql** 除去有关 **dbschema** 的头信息，然后使用 DB-Access 在另一个数据库中重新创建表，如下所示：

```
dbaccess db1 tab.sql
```

4.4.2 重新创建数据库模式

您可使用 **dbschema** 和 DB-Access 从数据库保存模式，然后在另一数据库中重新创建该模式。**dbschema** 输出文件可包含创建整个数据库的语句。

保存数据库模式并重新创建数据库：

1. 使用 **dbschema** 将模式保存到输出文件（例如 db.sql）：

   ```
   dbschema -d db > db.sql
   ```

   还可使用 -ss 选项生成特定于服务器的信息：

   ```
   dbschema -d db -ss > db.sql
   ```

2. 从输出文件除去有关 **dbschema** 的头信息（如果有的话）。

3. 在输出文件的开头添加 CREATE DATABASE 语句或使用 DB-Access 来创建新数据库。

4. 使用 DB-Access 在新数据库中重新创建模式：

   ```
   dbaccess - db.sql
   ```

   使用 **db.sql** 在不同的数据库服务器上创建数据库时，请确认数据库空间存在。

数据库 **db** 和 **testdb** 名称不同但具有相同的模式。

5 LOAD 和 UNLOAD 语句

您可以使用 SQL LOAD 和 UNLOAD 语句来移动数据。LOAD 语句速度较快且较易于使用，但它只接受指定的数据格式。通常可将使用 UNLOAD 语句准备好的数据用于 LOAD 语句。

您可以使用 DB-Access 中的 UNLOAD 语句从表中将选定的行卸载到文本文件。

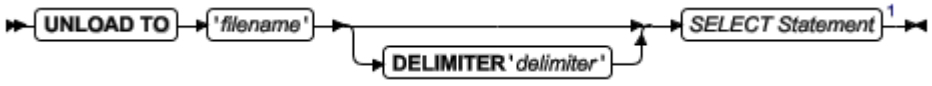
UNLOAD 语句允许您在卸载数据时对其进行操作，但它要求将数据卸载到磁盘上的文件而非磁带。如果卸载到磁盘文件，可能需要使用 UNIX™ 、Linux ™实用程序将这些文件装入到磁带上。

要装入表，请使用 LOAD 或 **dbload**。要操纵正在装入的数据文件或在装入数据库时要对其进行访问，请使用 **dbload** 实用程序。灵活性是以花在创建 **dbload** 命令文件上的时间以及较慢的执行速度为代价的。尽可能使用 LOAD 语句，它比 **dbload** 要快。

如果数据库包含基于标签的访问控制 (LBAC) 对象，那么只能装入或卸载您的安全标签控制其列安全标签或行安全标签的那些行。如果要装入或卸载整个表，那么必须具有写入/读取所有标注的行和列的必要 LBAC 凭证。有关 LBAC 对象的更多信息，请参阅 GBase 8s 安全性能指南 和《GBase 8s SQL 指南：语法》。

5.1 UNLOAD 语句的语法

DB-Access 中的 UNLOAD 语句可从表中将选定的行卸载到文本文件。



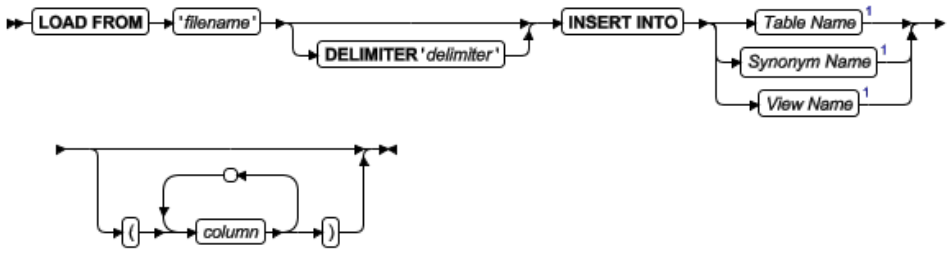
元素	用途	重要注意事项
delimiter	用作定界符的字符	**要求：**请参阅定界符格式的语法

filename	指定输入文件	无。

此语法图仅用于快速参考。有关 UNLOAD 语句的语法和使用的详细信息，请参阅《GBase 8s SQL 指南：语法》。

5.2 LOAD 语句的语法

DB-Access 中的 LOAD 语句可将行附加到数据库的现有表中。



元素	用途	重要注意事项
column	从 *filename* 接收数据的列的名称	必须是指定表或视图中的列。
delimiter	用作定界符的字符	请参阅定界符格式的语法

此语法图仅用于快速参考。有关 LOAD 语句的语法和使用的详细信息，请参阅《GBase 8s SQL 指南：语法》。

5.3 用于支持多字节代码集的语言环境的 load 和 unload 语句

对于支持多字节代码集的语言环境，确保接收字符数据的任何列的声明大小（以字节计）都足够存储整个数据字符串。

对于某些语言环境，对于最长的数据字符串，这可能需要最多达 4 倍的逻

辑字符数。

5.4 用于非缺省语言环境和 GL_DATETIME 环境变量的 load 和 unload 语句

在非缺省语言环境中，装入或卸载 DATETIME 值的操作可能会受 **GL_DATETIME** 和 **USE_DTENV** 环境变量设置的影响。

如果数据库使用非缺省的语言环境并且 **GL_DATETIME** 环境变量有非缺省的设置，那么必须先将 **USE_DTENV** 环境变量设置为值 1，然后才能正确处理本地化的 DATETIME 值，方法是使用 LOAD 或 UNLOAD 语句，或者使用 **dbimport** 或 **dbexport** 实用程序，或者在 DML 操作中对 CREATE EXTERNAL TABLE 语句定义的对象进行处理。

5.5 支持多字符定界符

支持可将定界符设置为多个字符。用户在导入和导出数据时，可以在导入导出语句中设置多字符为 DELIMITER 定界符，也可以通过设置 DBDELIMITER 环境变量为多字符来实现。

使用 DELIMITER 子句可指定定界符，该定界符用于隔开包含在输入表中某一行中的每一列中的数据。可以指定水平制表符或空格（=ASCII 32）作为定界符号。

定界符可以是单个字符或多个字符，但是字符个数不得超过 4 个。并且，不能包含下列项：

- 反斜杠（\）
- 十六进制数字（0 至 9、a 至 f、A 至 F）
- 不可见字符（除空格、水平制表符之外的所有空白字符，包括换行符、换页符、回车符、垂直制表符）

如果省略此子句，则数据库服务器检查 DBDELIMITER 环境变量。如果尚未设置 DBDELIMITER 环境变量，则缺省的定界符是竖线（|）。

以下示例指定分号（;）作为定界符。该示例使用 Windows 文件命名约定。

```
LOAD FROM 'C:\data\loadfile' DELIMITER ';' INSERT INTO orders;
```

设置 DBDELIMITER 环境变量，可以指定与 LOAD 及 UNLOAD 和与 DBEXPORT 实用程序语句配合使用的字段定界符。

例如，要将字段定界符更改为两个加号，可按如下方式设置 DBDELIMITER

```
EXPORT DBDELIMITER='++'
```

6 gunload 和 gload 实用程序

gunload 和 **gload** 实用程序提供在相同平台上使用相同数据库服务器的计算机之间移动数据的最快方法。

例如：假设您的站点购买了性能更强的 UNIX™ 计算机，使用户更快地进行访问。您需要将现有数据库传送到新计算机上的新数据库服务器。使用 **gunload** 从第一个数据库服务器卸载数据，然后使用 **gload** 将数据装入第二个数据库服务器。两个数据库服务器都必须具有相同或兼容的版本号。您可移动整个数据库或只移动选定的表，但无法修改数据库模式。

gunload 实用程序可比 **dbexport** 或 UNLOAD 语句更快地卸载数据，因为 **gunload** 以二进制格式和以页大小为单位复制数据。**gload** 实用程序使用 **gunload** 实用程序创建的磁带或文件并重新创建数据库或表。

gunload 和 **gload** 实用程序比 **dbimport**、 **dbload** 或 LOAD 要快，但相比之下灵活性小很多，并且不允许您修改数据库模式或从一个操作系统（或数据库服务器）版本移动到另一个。

6.1 有关何时使用 gunload 和 gload 实用程序的准则

仅当满足特定条件时，才能使用 **gunload** 和 **gload**。

仅当您对以下每个问题的答案都是是时，才可以使用 **gunload** 和 **gload**。如果您的答案是否，那么不能使用 **gunload** 和 **gload**。

仅当对每个问题的答案都为“是”时，才可以使用 gunload 和 gload
目标数据库服务器在同一硬件平台上吗？
想要移动到相同版本的另一数据库服务器吗？
想要保持现有数据库模式而不进行更改吗？
想要移动整个数据库或整个表吗？
页面映像是兼容的吗？
数值表示方法是相同的吗？

无法使用 gunload 和 gload 实用程序的情况

- 您无法使用 **gunload** 和 **gload**：
- 在 GLS 和非 GLS 数据库之间移动数据。
- 将压缩的数据从一个数据库移到另一个。
- 在使用 **gload** 和 **gunload** 实用程序之前，您必须将压缩表和分段中的数据解压缩。
- 移动外部表或包含外部表的数据库。
- 使用 **gunload** 实用程序之前，必须删除所有外部表。
- 移动包含扩展或智能大对象数据类型的表和数据库

6.2 使用 gload 和 gunload 实用程序的要求

gload 和 **gunload** 实用程序有其局限性。这些实用程序只能用于在同一操作系统上相同版本的数据库服务器之间移动数据。您无法修改数据库模式，必

须关闭日志记录，并且实用程序可能很难使用。

gload 和 **gunload** 实用程序具有以下要求：

- 源数据库和目标数据库必须来自相同版本的数据库服务器。不能使用 **gload** 和 **gunload** 实用程序将数据从一个版本移动到另一个。
- 您不能使用 **gload** 和 **gunload** 在不同类型的数据库服务器之间移动数据。
- **gload** 命令必须与相应的 **gunload** 命令（它卸载相同的表或 **gload** 引用的表）的作用域相同。例如，您无法做到先用 **gunload** 来卸载整个数据库，然后再使用 **gload** 从该数据库仅装入表子集。
- 如果数据库包含扩展或智能大对象数据类型，那么不要使用 **gload** 和 **gunload** 来移动数据。（而是使用 HPL 来移动数据。）
- 因为 **gload** 读取的磁带包含以磁盘页大小为单位存储的二进制数据，所以源数据库驻留的计算机（使用 **gunload** 的计算机）和目标数据库将驻留的计算机（使用 **gload** 的计算机）必须具有相同的页大小、相同的数字数据表示法，针对结构和单位具有相同的字节对齐方式。
- 您不能使用 **gload** 和 **gunload** 在非 GLS 和 GLS 语言环境之间移动数据。
- 不能在高可用性集群中的服务器上使用 **gload** 和 **gunload**。
- 如果您压缩了表或分段，那么不能使用 **gload** 和 **gunload**。

如果 NLS 和 GLS 语言环境是相同的，那么可使用 **gunload** 和 **gload** 在数据库间移动数据。例如，如果 NLS 和 GLS 表都是使用相同的法语语言环境创建的，那么 **gload** 和 **gunload** 可以移动数据。但是，如果用户 A 在服务器 A 上具有法语语言环境 NLS 表并试图将数据装入服务器 B 上的德语语言环境 GLS 表，那么 **gload** 将报告错误。

如果页大小不同，**gload** 将失败。如果两台计算机上的对齐或数字数据类型不同（例如：最重要的字节在最后而不是第一个或者使用不同的浮点类型表

示法），那么数据页面的内容会得到错误的解释。

6.3 gunload 和 gload 实用程序工作方式

可以从数据库卸载数据的 **gunload** 实用程序会将数据库或表写入磁带或磁盘上的文件中。**gload** 实用程序将使用 **gunload** 命令创建的数据装入数据库服务器。

gunload 实用程序以磁盘页为单位以二进制格式卸载数据，这使此实用程序比 **dbexport** 更有效。

您可使用 **gunload** 实用程序在具有相同版本的数据库服务器的计算机间移动数据。

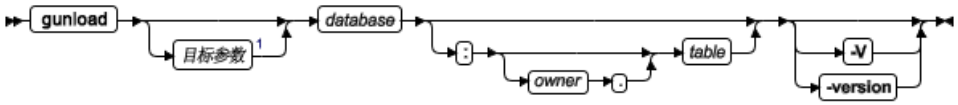
要点： 您不能使用 **gload** 和 **gunload** 实用程序将数据从数据库服务器的一个版本移动到另一版本，也不能在不同类型的数据库服务器之间移动。此外，**gload** 命令必须与相应的 **gunload** 命令（它卸载相同的表或 **gload** 引用的表）的作用域相同。例如，您无法做到先用 **gunload** 来卸载整个数据库，然后再使用 **gload** 从该数据库仅装入表子集。

gload 实用程序在指定的数据库空间中创建数据库或表。 然后，**gload** 实用程序向它装入来自 **gunload** 实用程序创建的输入磁带或磁盘文件的数据。

装入期间，您可将存储在 Blobspace 中的简单大对象移动到另一 Blobspace 。

6.4 gunload 命令的语法

gunload 命令从数据库卸载数据并将数据库或表写入磁带或磁盘上的文件中。



元素	用途	重要注意事项
database	指定数据库的名称	**其他信息：**不能使用数据库服务器名（*database@dbservername*）限定数据库名称。 **参考：**语法必须与“标识”段一致；请参阅《*GBase 8s SQL 指南：语法*》。
owner.	指定表的所有者	**其他信息：**所有者名称不能包含无效字符。 **参考：**有关路径名的语法，请参阅操作系统文档。
table	指定表名	**要求：**表必须存在。 **参考：**语法必须与“表名”段一致；请参阅《*GBase 8s SQL 指南：语法*》。

如果未指定任何目标参数选项，那么 **gunload** 将使用 TAPEDEV 指定的设备。Block 大小和磁带大小分别是 TAPEBLK 和 TAPESIZE 指定的值。（有关 TAPEDEV、TAPEBLK 和 TAPESIZE 的信息，请参阅《GBase 8s 管理员参考》。）

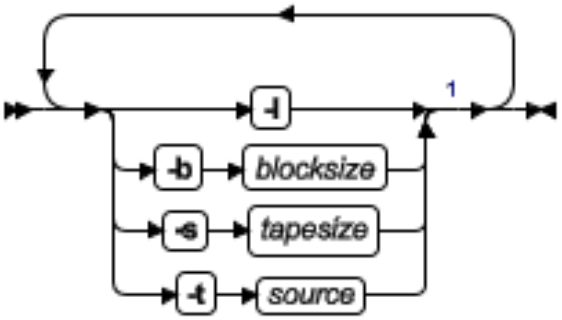
-V 选项显示软件版本号和序列号。**-version** 选项扩展了 **-V** 选项，以显示有关构建操作系统、构建号和构建日期的其他信息。

6.4.1 gunload 目标参数

gunload 实用程序支持磁带或文件目标选项。

以下语法图分段显示 **gunload** 目标参数

目标参数



元素	用途	重要注意事项
-b *blocksize*	指定磁带设备的块大小（以千字节计）	**要求：***blocksize* 必须是整数。 **其他信息：**此选项覆盖了 TAPEBLK 或 LTAPEBLK 中的缺省值。
-l	指示 gunload 分别从 LTAPEDEV、LTAPEBLK 和 LTAPESIZE 读取磁带设备、块大小和磁带大小的值	无。
-s *tapesize*	指定存储在磁带上的数据量（以千字节计）	**要求：***tapesize* 必须是整数。要写到磁带的末尾，请将磁带大小指定为 0。 如果您不指定 0，那么最大的 *tapesize* 是

		2 097 151 KB。 **其他信息：**此选项覆盖 TAPESIZE 或 LTAPESIZE 中的缺省值。
-t *source*	指定磁盘上文件的路径名或安装输入磁带的磁带设备的路径名	**其他信息：**此选项覆盖 TAPEDEV 或 LTAPEDEV 指定的磁带设备。路径名必须是有效的路径名。

6.4.2 影响 gunload 的约束

当使用 **gunload** 实用程序时，您必须注意可影响 **gunload** 磁带上的数据装入方式的约束。

以下约束适用于 **gunload**：

- 必须将 **gunload** 磁带上的数据装入您的数据库服务器管理的数据库或表中。
- 如果数据库包含扩展数据类型，则不能使用 **gunload** 和 **gload** 。
- 您要将 **gunload** 写入的磁带装入的计算机必须与原计算机有相同的页大小以及相同的数字数据表示法。
- 您必须使用相同版本的数据库服务器的 **gload** 实用程序读取 **gunload** 创建的文件。不能使用 **gunload** 和 **gload** 将数据从一种版本移动到另一种版本。
- 卸载完整的数据库时，您不能修改数据库对象（例如表、索引和视图）的所有权，直到完成重新装入数据库为止。
- 卸载和装入表时，**gunload** 不保留与原表相关联的访问特权、同义词、视图、约束、触发器或缺省值。运行 **gunload** 之前，请使用 **dbschema** 实用程序获取访问特权、同义词、视图、约束、触发器和缺省值的列表。完成装入表之后，请使用 **dbschema** 重新创建表的特定信息。

6.4.3 数据库或表卸载特权

要卸载数据库，您必须对数据库具有 DBA 特权或是用户 **gbasedbt**。要卸载表，您必须拥有该表、对表所驻留的数据库具有 DBA 特权或是用户 **gbasedbt**。

对 **gunload** 和 **gload**，用户 **root** 不具有特殊的特权。

6.4.4 与数据库一起卸载的表

如果卸载数据库，将卸载数据库中的所有表（包括系统目录表）。

同时也将卸载该数据库中的所有表的所有触发器、SPL 例程、缺省值、约束和同义词。

6.4.5 与表一起卸载的数据

如果卸载表，**gunload** 将从 **systables**、**systables**、**syscolumns**、**sysindexes** 和 **sysblobs** 系统目录表卸载表数据和信息。

卸载表时，**gunload** 不会卸载有关与表关联的约束、触发器或缺省值的信息。另外，将不会卸载为表定义的访问特权以及与该表相关联的同义词或视图。

6.4.6 卸载操作期间锁定

卸载操作期间，以共享方式锁定数据库或表。如果 **gunload** 无法获得共享锁定，那么将返回错误。

gload 实用程序在指定的 **dbspace** 中创建数据库或表。然后，**gload** 实用

程序向它装入来自 **gunload** 实用程序创建的输入磁带或磁盘文件的数据。

6.5 日志记录方式

gunload 实用程序不保留数据库的日志记录方式。使用 **gload** 装入数据库后，您可使数据库符合 ANSI 或添加日志记录。

有关日志记录方式的信息，请参阅《GBase 8s SQL 指南：语法》。

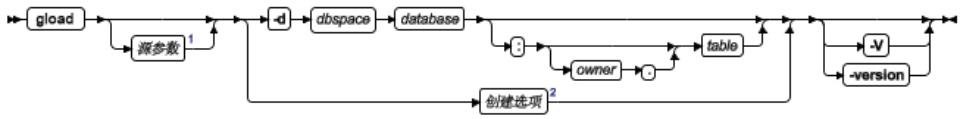
装入期间，您可将存储在 Blob 空间中的简单大对象移动到另一 Blob 空间。

如果没有指定任何源参数选项，**gload** 将使用 TAPEDEV 指定的设备。块大小和磁带大小分别是 TAPEBLK 和 TAPESIZE 指定的值。（有关 TAPEDEV、TAPEBLK 和 TAPESIZE 的更多信息，请参阅《GBase 8s 管理员指南》。）

如果没有指定创建选项，**gload** 将在根数据库空间中存储数据库或表。

6.6 gload 命令的语法

gload 命令将使用 **gunload** 命令创建的数据装入数据库服务器。



元素	用途	重要注意事项

-d *dbspace*	将数据库或表装入指定数据库	正在装入的磁带必须包含指定的数据库或表。
database	指定数据库的名称	数据库名不能包含数据库服务器名称，例如 *database@dbservername*。 **参考**：语法必须与“标识”段一致；请参阅《*GBase 8s SQL 指南：语法*》。
元素	**用途**	**重要注意事项**
owner.	指定表的所有者	所有者名称不能包含无效字符。 **参考**：有关路径名的语法，请参阅操作系统文档。
table	指定表名	表必须存在。 **参考**：语法必须与“表名”段一致；请参阅《*GBase 8s SQL 指南：语法*》。

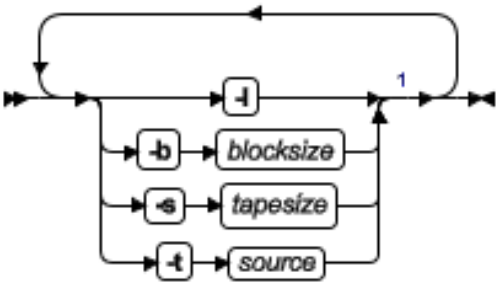
-V 选项显示软件版本号和序列号。**-version** 选项扩展了 **-V** 选项，以显示有关构建操作系统、构建号和构建日期的其他信息。

6.6.1 gload 源参数

gload 命令含有指定磁带或文件源相关信息的选项。

以下语法图分段显示 **gload** 源参数。

源参数



元素	用途	重要注意事项
-b *blocksize*	指定磁带设备的块大小（以千字节计）	**要求：**无符号整数。必须指定磁带设备的块大小。 **其他信息：**此选项覆盖了 TAPEBLK 或 LTAPEBLK 中的缺省值。
-l	指示 gload 分别从配置参数 LTAPEDEV、LTAPEBLK 和 LTAPESIZE 读取磁带设备、块大小和磁带大小的值	**其他信息：**如果您先指定 -l，然后指定 -b、-s 或 -t，指定的值将覆盖配置文件中的值。
-s *tapesize*	以千字节为单位指定数据库服务器可在磁带上存储的数据量	**要求：**无符号整数。要写到磁带的末尾，请将磁带大小指定为 0。 如果您不指定 0，那么

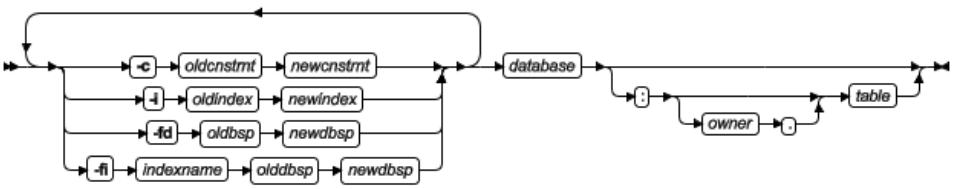
		最大的 *tapesize* 是 2 097 151 KB。 **其他信息：**此选项覆盖 TAPESIZE 或 LTAPESIZE 中的缺省值。
-t *source*	指定磁盘上文件的路径名或安装输入磁带的磁带设备的路径名	必须是合法路径名。 **其他信息：**此选项覆盖 TAPEDEV 或 LTAPEDEV 指定的磁带设备。 **参考：**有关路径名的语法，请参阅操作系统文档。

6.6.2 gload 创建选项

gload 命令包含用于重新创建数据库的信息。

以下语法图分段显示 **gload** 创建选项。

创建选项



元素	用途	重要注意事项

-c *oldcnstrnt newcnstrnt*	指示 gload 重命名指定的约束。	无。
-i *oldindex newindex*	指示 gload 在磁盘上存储索引时重命名表索引。	**其他信息：**在装入期间使用 -i 选项重命名索引以避免与现有的索引名称冲突。 **参考：**语法必须与“标识”段一致；请参阅《*GBase 8s SQL 指南：语法*》。
-fd *olddbsp newdbsp*	将数据分段从一个数据库空间移到另一个。	新数据库空间必须存在且必须不包含表的另一个数据分段。 **其他信息：**此选项用于并行数据查询 (PDQ) 和表分段存储。
-fi *indexname olddbs newdbsp*	将索引分段从一个数据库空间移到另一个。	新数据库空间必须存在且必须不包含表的另一个索引分段。 **其他信息：**此选项用于 PDQ 和表分段存储。
database	指定数据库的名称	**要求：**数据库名不能包含数据库服务器名称，例如 *database@dbservername* 。 **参考：**语法必须与“标识”段一致；请参阅《*GBase 8s SQL 指南：语法*》。
owner.	指定表的所有者	**要求：**所有者名称不能包含无效字符。 **参考：**有关路径名的语法，请参

		阅操作系统文档。
table	指定表名	**要求**：表必须不存在。 **参考**：语法必须与“表名”段一致；请参阅《*GBase 8s SQL 指南：语法*》。

如果没有为非分段表指定任何创建选项，**gload** 实用程序将在根数据库空间中存储数据库或表。

若为分段表，**gunload** 会保留分段存储表达式以供 **gload** 在以后使用。因此，导入表的分段方式与原始表相同。

只要使用唯一的对，您就可以随时以任何顺序使用 **-c**、**-i**、**-fd** 和 **-fi** 选项。

6.6.3 影响 gunload 的约束

gload 实用程序比 **dbimport**、**dbload** 或 LOAD 方法执行得要快。为了获得较高的性能，**gload** 具有一定约束。

gload 实用程序具有以下约束：

- **gload** 实用程序只创建新的数据库或表；运行 **gload** 之前，您必须删除或重命名相同名称的现有数据库或表。执行期间，**gload** 实用程序的提示将询问您是否要重命名 Blobspaces。
- 装入期间 **gload** 实用程序在数据库中的每个表上放置共享锁定。即使您无法正确更新带锁的表行，数据库还是可用于查询。
- 装入完整的数据库时，运行 **gload** 的用户将成为数据库的所有者。
- **gload** 实用程序创建的数据库不带日志记录；您必须在 **gload** 装入数据库后启动日志记录。
- 使用 **gload** 将表装入已日志记录的数据库时，操作期间必须关闭数据库的日志记录。

- 对于分段的表，会保留数据库空间指定，除非您使用 **-fn** 选项将其覆盖。
- 对于非分段的表，如果未使用 **-d** 选项指定目标数据库空间，那么 **gload** 实用程序会尝试在根数据库空间中存储表。如果由于页大小不同而无法将表存储在根数据库空间或使用 **-d** 选项指定的数据库空间中，那么 **gload** 实用程序会尝试使用数据库空间编号与最初所卸载表的数据库空间编号相同的数据库空间。如果此数据库空间仍然具有不同的页面大小，那么装入操作将失败。

6.6.4 装入期间的日志记录

使用 **gload** 实用程序从 **gunload** 输入磁带创建表时，**gload** 只可以将信息装入不带日志记录的数据库。因此，将表装入现有的、已日志记录的数据库之前，必须结束该数据库的日志记录。

您可能还需要在非高峰时间进行装入。否则可能填满逻辑日志文件或花费过多的共享内存资源。装入表后，在继续数据库记录日志之前，请创建 0 级数据库空间备份。

使用 **gload** 从 **gunload** 输入磁带创建数据库时，得到的数据库不符合 ANSI 且不使用事务日志记录。装入数据库后可使数据库符合 ANSI 或添加日志记录。

gload 实用程序在事务中执行它的所有装入。如果错误发生，此功能允许回滚更改。

6.6.5 将简单大对象移至 Blobspace

如果装入的表包含存储在 Blobspace 中的简单大对象，**gload** 实用程序将询问您是否希望将它们移动到另一个 Blobspace。

如果您回答是，**gload** 将显示当创建磁带时存储该简单大对象的 Blobspace 名称。然后它将请求您输入您希望用来存储该简单大对象的 Blobspace 名称。

如果输入有效的 Blobspace 名称，**gload** 将把表中所有的简单大对象列移动到新的 Blobspace。否则，**gload** 将再次提示您输入有效的 Blobspace 名称。

6.6.6 所有权和特权

装入新数据库时，运行 **gload** 实用程序的用户成为所有者。数据库中（表、视图和索引）的所有权与使用 **gunload** 将数据库卸载到磁带时的所有权相同。

要装入表，您必须在数据库上拥有 Resource 特权。当 **gload** 装入新表时，运行 **gload** 的用户将成为所有者，除非您在表名中指定了所有者。（要在表名中指定所有者，您需要对数据库有 DBA 特权。）

gunload 实用程序不保留同义词或访问特权。要获取已定义同义词或访问特权的列表，请在运行 **gunload** 之前使用 **dbschema** 实用程序（在 **dbschema** 实用程序中描述）。

6.6.7 装入操作期间互斥锁定

装入操作期间，**gload** 实用程序放置对新数据库或表的互斥锁定。

装入过程作为单个事务继续进行，而当发生错误或系统故障时，**gload** 将删除新的数据库或表。

6.7 使用 gunload 和 gload 实用程序在计算机之间移动数据库

您可以使用 **gunload** 和 **gload** 实用程序将完整的数据库从一台计算机移动到另一台。

将数据库从一台计算机移动到另一台计算机：

1. 确保两台计算机上的页大小、数字表示以及结构和联合上的字节对齐相同。

   在某些 UNIX™ 系统上，页面大小为 2 KB。页大小是 GBase 8s 的特征。有关页大小的信息，请参阅《GBase 8s 管理员指南》。数字表示和字节对齐是操作系统特征。有关数字表示和字节对齐的信息，请参阅操作系统手册。

2. 决定存储已卸载数据的地方：

   - **在磁盘上**。为 **gunload** 创建一个空文件来保存数据。确保您拥有此文件的写许可权。
   - **在磁带上**。使用在 ONCONFIG 配置文件中由 TAPEDEV 或 LTAPEDEV 配置参数指定的磁带设备和特征，或指定另一个磁带设备。请确保您指定的磁带设备可用于 **gunload**。但是，如果将 TAPEDEV 配置参数设置为 STDIO，那么 **gunload** 实用程序将无法卸载数据。

3. 运行 **gcheck** 实用程序以确保数据库的一致性。

   有关 **gcheck** 的信息，请参阅《GBase 8s 管理员参考》。

4. 运行 **gunload** 实用程序从数据库卸载数据。

   有关 **gunload** 命令语法的详细信息，请参阅 **gunload** 命令的语法。

5. 如果需要，将存储介质（磁带或磁盘）传送到新计算机。

   如果两台计算机位于同一网络上，您可远程读取或写入数据。

6. 运行 **gload** 实用程序将数据装入新数据库。

   有关 **gload** 命令语法的详细信息，请参阅 **gload** 命令的语法。

7. 为新数据库设置日志记录状态。

   有关日志记录状态的信息，请参阅《GBase 8s 管理员指南》。

8. 如果需要，更改数据库的 DBA 特权。

9. 创建新数据库的 0 级备份。

6.8 使用 gunload 和 gload 实用程序在计算机之间移动表

您可以使用 **gunload** 和 **gload** 实用程序将表从同一台计算机上的一个数据库空间移动到另一个。

将表从一台计算机移动到另一台计算机：

1. 确保两台计算机上的页大小、数字表示以及结构和联合上的字节对齐相同。（在某些 UNIX™ 系统上，页大小为 2 KB。）

2. 决定存储已卸载数据的地方。

3. 运行 gcheck 实用程序以确保数据库的一致性。

4. 如果希望保存表的触发器、访问特权、SPL 例程、缺省值、约束和同义词，请运行 dbschema 实用程序。

5. 运行 gunload 实用程序。

   有关 gunload 命令语法的详细信息，请参阅 gunload 命令的语法。

6. 如果需要，将存储介质传送到新计算机。

7. 如果表包含存储在 Blobspace 中的简单大对象，那么确定在何处存储简单大对象。如果需要，创建新的 Blobspace。

8. 关闭日志记录。

   装入表时，必须关闭目标数据库上的日志记录。（创建和装入整个数据库时，日志记录的状态并不重要。）

9. 运行 gload 实用程序。

有关 gload 命令语法的详细信息，请参阅 gload 命令的语法。

10. 创建已修改数据库的 0 级备份。
11. 打开日志记录（如果您需要日志记录）。
12. 如果希望恢复表的触发器、访问特权、SPL 例程、缺省值、未保留的约束以及同义词，请运行 dbschema 实用程序或手动重新创建这些对象。

即使是单个表，主键或缺省值之类的约束也都会保留。外键、访问特权、SPL 例程和同义词不会保留。

6.9 使用 gunload 和 gload 实用程序在数据库空间之间移动表

您可以使用 **gunload** 和 **gload** 实用程序将表从同一台计算机上的一个数据库空间移动到另一个。

将表从同一台计算机上的一个数据库空间移动到另一个数据库空间：

1. 运行 **gunload** 实用程序卸载表。
2. 有关 **gunload** 命令语法的详细信息，请参阅 **gunload** 命令的语法。
3. 关闭日志记录。
4. 装入表时，必须关闭目标数据库上的日志记录。
5. 运行 **gload** 实用程序。
6. 在 **gload** 命令中指定新表名和新数据库空间名称。
7. 有关 **gload** 命令语法的详细信息，请参阅 **gload** 命令的语法。
8. 如果数据成功装入，删除旧数据库空间中的旧表，并将新表重命名为旧表名。
9. 创建已修改数据库的 0 级备份。

10. 打开日志记录（如果您需要日志记录）。

7 gadmin 实用程序还原选项

使用 **gadmin** 实用程序的 **-b** 选项可还原到所转换的较旧版本的数据库服务器。

7.1 gadmin -b 命令的作用

转换数据库服务器时，若干修改将使数据库的格式无法兼容较旧的版本。gadmin -b 命令可以修改数据，这样较早版本的数据库服务器就可以访问这些数据。

转换数据库服务器时，若干修改将使数据库的格式无法兼容较旧的版本。gadmin -b 命令可以修改数据，这样较早版本的数据库服务器就可以访问这些数据。在某些情况下，不同版本之间的数据库格式是兼容的，因此不需要 gadmin -b 命令。输入 gadmin -b 可查看数据库服务器可用的选项的用法消息。

该实用程序不会还原不影响兼容性的对数据布局所作的更改。

在用户可以使用较早版本的数据库服务器存取数据之前，您必须还原数据库。

有关其他 **gadmin** 选项的信息，请参阅《GBase 8s 管理员参考》中的 gadmin 实用程序。

7.2 使用 gadmin -b 命令进行准备

使用 gadmin -b 命令之前，请通知用户您打算将数据库服务器脱机。还原实用程序会强制除去所有用户并关闭数据库服务器。

gadmin -b 命令包含隐式的 **-yuk** 命令。

请确保已将 **GBASEDBTSERVER** 环境变量设置为正确的数据库服务器。

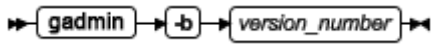
仅 UNIX/Linux

您必须是用户 **root** 或用户 gbasedbt 才能运行 **gadmin**。

7.3 gadmin -b 命令的语法

gadmin -b 命令可以将数据库恢复到转换前的数据库版本。您无法使用此命令还原到服务器的任何其他版本。

转换数据库服务器时，若干修改将使数据库的格式无法兼容较旧的版本。gadmin -b 命令可以修改数据，这样较早版本的数据库服务器就可以访问这些数据。在某些情况下，不同版本之间的数据库格式是兼容的，因此不需要 gadmin -b 命令。输入 gadmin -b 可查看数据库服务器可用的选项的用法消息。



元素	用途
-b *version_number*	将数据库|还原到指定的版本。

V8.5	PSM_DEBUG	针对您的环境（例如，单个会话）指定在存储管理器调试日志中打印的调试信息量，从而覆盖 PSM_DEBUG 配置参数的值。
V8.5	PSM_DEBUG_LOG	针对您的环境（例如，单个会话）指定存储管理器调试日志的位置，从而覆盖 PSM_DEBUG_LOG 配置参数的值。
V8.5	PSM_LOG_POOL	针对您的环境（例如，单个会话）更改存储管理器放置备份和复原日志数据的池的名称，从而覆盖

GBASE®

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微信二维码

技术支持热线：400-013-9696