



GBase 8s DB-Access 用户指南



GBase 8s DB-Access 用户指南，南大通用数据技术股份有限公司

GBase 版权所有©2004-2021，保留所有权利

版权声明

本文档所涉及的软件著作权及其他知识产权已依法进行了相关注册、登记，由南大通用数据技术股份有限公司合法拥有，受《中华人民共和国著作权法》、《计算机软件保护条例》、《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的南大通用数据技术股份有限公司的版权信息由南大通用数据技术股份有限公司合法拥有，受法律的保护，南大通用数据技术股份有限公司对本文档可能涉及到的非南大通用数据技术股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅，并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经南大通用数据技术股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将视为侵权，南大通用数据技术股份有限公司具有依法追究其责任的权利。

本文档中包含的信息如有更新，恕不另行通知。您对本文档的任何问题，可直接向南大通用数据技术股份有限公司告知或查询。

通讯方式

南大通用数据技术股份有限公司

天津市高新区开华道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

GBASE[®] 是南大通用数据技术股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由南大通用数据技术股份有限公司合法拥有，受法律保护。未经南大通用数据技术股份有限公司书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯南大通用数据技术股份有限公司商标权的，南大通用数据技术股份有限公司将依法追究其法律责任。

目 录

1 简介	1
1.1 本简介内容.....	1
1.2 关于语言环境的假设.....	1
1.3 符合行业标准.....	2
1.4 如何阅读语法图.....	2
1.5 示例代码约定.....	3
2 DB-Access 入门	5
2.1 GBase 8s 服务器 DB-Access 实用程序的需求.....	5
2.1.1 环境变量.....	5
2.2 GBase 8s Client Software Development Kit DB-Access 实用程序的需求.....	6
2.3 演示数据库.....	7
2.3.1 创建演示数据库.....	7
2.3.2 dbaccessdemo 命令：创建演示数据库.....	8
2.4 启动 DB-Access	9
2.4.1 dbaccess 命令：启动 DB-Access	9
2.4.2 在不使用菜单的情况下以交互方式运行 DB-Access.....	17
3 全屏菜单界面.....	21
3.1 查询语言选项.....	21
3.1.1 SQL 编辑器	22
3.1.2 系统编辑器.....	23
3.1.3 “运行”选项支持的语句.....	23
3.1.4 重定向查询结果.....	29
3.1.5 选择现有 SQL 语句	29
3.1.6 保存当前 SQL 语句	30
3.1.7 支持 SPL 例程.....	31
3.1.8 出现错误时发生何事.....	33
3.2 数据库选项.....	33
3.2.1 可用数据库列表.....	34
3.2.2 检索非缺省语言环境信息.....	35
3.2.3 关闭数据库.....	36
3.3 表选项.....	37
3.3.1 显示表信息.....	38
3.4 连接选项和会话选项.....	39

3.4.1 隐式关闭.....	40
4 附录	41
4.1 如何阅读 SQL 语句的联机帮助	41
4.2 演示 SQL	42
4.2.1 用于关系数据库模型的 SQL 文件	43
4.2.2 用于维数据库模型的 SQL 文件	56
4.2.3 用于对象关系数据库模型的用户定义的例程	62

1 简介

1.1 本简介内容

本出版物描述了如何使用 DB-Access 实用程序来从 GBase 8s 数据库服务器访问、修改和检索信息。

重要： 将 DB-Access 与 GBase 8s 数据库服务器的当前版本一起使用。如果您将 DB-Access 用于不同版本的数据库服务器，那么您将可能获得不一致的结果，例如：当您不支持长标识的版本与支持长标识的版本一起使用时。

本出版物是为以下用户编写的：

- 数据库用户
- 数据库管理员
- 数据库应用程序程序员

本出版物假定您有以下知识背景：

- 对于计算机、操作系统和操作系统提供的实用程序的应用知识
- 使用关系数据库的相关经验或者了解数据库概念
- 一些计算机编程经验

1.2 关于语言环境的假设

GBase 8s 产品可以支持多种语言、文化和代码集。由给定地域和编码中语言使用的与数字数据、货币、日期和时间的字符集、整理和表示法相关的所有信息都集中在称为 Global Language Support (GLS) 语言环境的单个环境中。

GBase 8s OLE DB Provider 遵循用于日期、时间和货币的 ISO 字符串格式，如 Microsoft[™] OLE DB 标准所定义。可以通过设置 GBase 8s 环境变量或注册表项（如 DBDATE）来覆盖该缺省值。

如果在 GBase 8s 环境中使用简单网络管理协议 (SNMP)，请注意，这些协议 (SNMPv1 和 SNMPv2) 仅识别英语代码集。

本出版物中的示例在编写时假设您使用以下某种语言环境：UNIX[™] 平台上的 en_us.8859-1 (ISO 8859-1)。这些语言环境支持用于显示和输入日期、时间、数字和货币值的美国英语格式约定。它们还支持 ISO 8859-1 代码集（在 UNIX 和 Linux[™] 上），这些代码集包括 ASCII 代码集以及很多 8 位字符（如 é、è 和 ñ）。

如果您计划在数据或 SQL 标识中使用其他语言环境中的字符，或者希望符合字符数据的其他整理规则，那么可以指定其他语言环境。

1.3 符合行业标准

GBase 8s 产品符合各种标准。

基于 SQL 的 GBase 8s 产品完全兼容 SQL-92 入门级（发布为 ANSI X3.135-1992），这与 ISO 9075:1992 完全相同。另外，GBase 8s 数据库服务器的许多功能都遵守 SQL-92 中级和完全级别以及 X/Open SQL 公共应用程序环境 (CAE) 标准。

1.4 如何阅读语法图

语法图使用特殊组件描述语句和命令的语法。

从左到右，从上到下跟随线的路径阅读语法图。

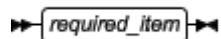
此右侧双箭头加直线符号 \Rightarrow 表示语句开始。

右侧箭头符号 \rightarrow 表示语句延续到下一行。

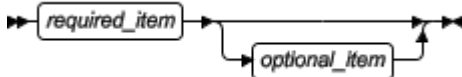
右箭头加直线符号 \rightarrow 表示语句继续上一行的内容。

直线、右箭头加左箭头符号 \Rightarrow 表示语句结束。

必需项出现在水平线（主路径）中。



可选项出现在主路径下方。

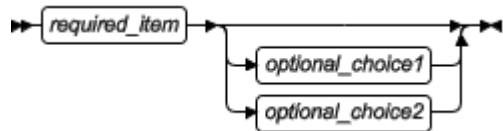


如果可以从两个或多个项中选择，那么它们以堆栈的方式表示。

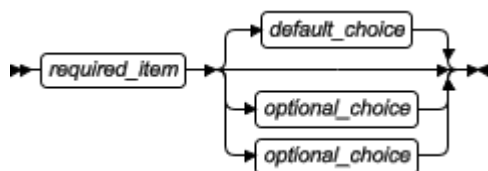
如果必须选择其中一项，那么堆栈中的一项出现在主路径上。



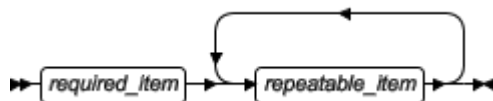
如果从中选择的项是可选的，那么整个堆栈出现在主路径下方。



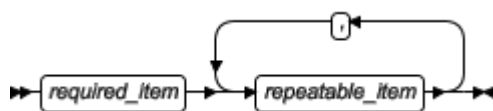
如果缺省其中一项，则它会在主路径上方显示，剩余的选项将会显示在下方。



返回左侧的箭头，在主线之上，表示该项可重复。在此情况下，重复项必须用一个或多个空格隔开。



如果重复的箭头包含一个逗号，那么您必须使用逗号分隔重复的项。

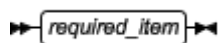


堆栈上方重复的箭头表示可以从堆栈的项目中进行多个选择或者重复一个选择。

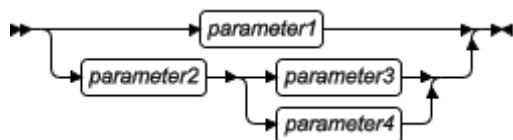
SQL 关键字以大写字母出现（例如：**FROM**）。它们必须严格按照所显示的拼写。变量以小写字母出现（例如：`column-name`）。它们表示用户在语句中提供的名称或值。

如果出现了标点符号、括号、算术运算符或其它这样的符号，那么必须将它们作为语法的一部分输入。

某些时候，一个变量表示一个语句段。例如：在以下语法图中，变量 `parameter-block` 表示已标记为 `parameter-block` 的语句段：



`parameter-block`:



1.5 示例代码约定

SQL 代码的示例在整个出版物中出现。除非另有说明，代码不特定于任何单个的 GBase 8s 应用程序开发工具。

如果示例中仅列出 SQL 语句，那么它们将不用分号定界。例如：您可能看到以下示例中的代码：

```
CONNECT TO stores_demo
```

```
...  
  
DELETE FROM customer  
  WHERE customer_num = 121  
  
...  
  
COMMIT WORK  
DISCONNECT CURRENT
```

要将此 SQL 代码用于特定产品，必须应用该产品的语法规则。例如，如果使用的是 SQL API，那么必须在每条语句的开头使用 EXEC SQL，并在每条语句的结尾使用分号（或其他合适的定界符）。如果使用的是 DB - Access，那么必须用分号将多条语句隔开。

提示：代码示例中的省略点表示在整个应用程序中将添加更多的代码，但是不必显示它以描述正在讨论的概念。

有关使用特定应用程序开发工具或 SQL API 的 SQL 语句的详细指导，请参阅您的产品文档。

2 DB-Access 入门

DB-Access 提供了用于输入、运行和调试结构化查询语言 (SQL) 语句与存储过程语言 (SPL) 例程的菜单驱动型界面。您还可以从命令行以交互方式运行 DB-Access。

使用 SQL 和 SPL 命令可执行数据定义任务（例如，指定表中数据列的数量和类型）和数据管理任务（例如，存储、查看和更改表数据）。

您可以使用 DB-Access 执行下列方面的数据库处理：

- 运行很少执行的特别查询
- 连接至一个或多个数据库、在数据库和外部文本文件之间传输数据以及显示有关数据库的信息
- 显示数据库的系统目录表和信息模式
- 练习 GBase 8s SQL 指南：教程中提供的 SQL 和 SPL 语句和示例
- 测试您要存储以供在生产环境中使用的应用程序
- 创建演示数据库

重要：DB-Access 未计划用作应用程序开发环境。当在 DB-Access 中运行 SQL 语句时，不能有条件地转移或循环执行这些语句。

GBase 8s 服务器和 GBase 8s Client Software Development Kit 随附了 DB-Access 实用程序。

2.1 GBase 8s 服务器 DB-Access 实用程序的需求

启动 DB-Access 之前，请先准备 GBase 8s 服务器环境。

- 启动 GBase 8s 服务器随附的 DB-Access 实用程序之前，请先执行以下任务：
- 设置环境变量
- 如果需要全球化，请设置 Global Language Support (GLS) 语言环境
- 启动数据库服务器

要保护 DB-Access 与 GBase 8s 的连接，可以使用安全套接字层 (SSL) 协议。

2.1.1 环境变量

作为安装和设置过程的一部分，系统或数据库管理员会设置某些环境变量，这些变量使 GBase 8s 产品可以在特定的操作系统环境中工作。

在 UNIX[™] 操作系统上使用 DB-Access，那么路径中必须包含 \$GBS_HOME/bin。操作系统使用该路径来定位初始化脚本和 dbaccess 可执行文件。

在 UNIX 环境中，数据库服务器必须具有 GBASEDBTERM 环境变量所列终端中的相应终端，并对其进行设置。

除非 GBASEDBTERM 环境变量设置为 termcap 文件，否则 DB-Access 将使用 terminfo 目录中的终端定义。如果 DB-Access 无法根据 GBASEDBTERM 设置来初始化菜单，那么 DB-Access 将尝试使用其他设置。例如，如果 DB-Access 无法使用 terminfo 目录初始化菜单，那么 DB-Access 将使用 termcap 文件来启动菜单。

您可以设置下列可选环境变量：

DBACCNOIGN

如果在菜单方式下运行 LOAD 命令，此变量将回滚未完成的事务。

DBCENTURY

为仅具有两位数年份的 DATE 和 DATETIME 值（例如 04/15/12）设置适当的扩展。

DBDATE

指定 DATE 值的用户格式。

DBEDIT

设置缺省 DB-Access 文本编辑器，而不更改与操作系统 shell 关联的缺省文本编辑器。

有关 DB-Access 如何将您指定的文本编辑器用作缺省文本编辑器的更多信息，请参阅系统编辑器。

DBFLTMASK

在 14 个字符的缓冲区内设置数据类型为 FLOAT、SMALLFLOAT 和 DECIMAL 的缺省浮点值。

此变量只会影响数字的 DB-Access 显示大小。

DELIMIDENT

使数据库服务器将双引号 (") 文本解释为标识而非字符串。

IFX_LONGID

确定客户机应用程序是否可以处理长标识。

如果使用 IFX_LONGID 环境变量来支持最多 128 字节的 SQL 标识，那么 DB-Access 的某些错误、警告或其他消息可能会截断在标识中包含多于 18 个字节的数据对象名称。为了避免此类截断，可以不声明包含 18 个字节以上的名称。

2.2 GBase 8s Client Software Development Kit

DB-Access 实用程序的需求

在客户机上启动 DB-Access 之前，请先设置客户机环境。

客户机上的 DB-Access 实用程序可以直接访问 Client SDK 与其建立了客户机/服务器连接的 GBase 8s 数据库。

启动 Client SDK 随附的 DB-Access 实用程序之前，请先执行以下任务：

- 设置 sqlhosts 信息。
- 将 GBS_HOME 环境变量设置为 Client SDK 安装目录。

2.3 演示数据库

您可以练习将 DB-Access 与演示数据库配合使用。

如果使用 GBase 8s 演示数据库，那么可以添加、删除或更改提供的数据和脚本。还可以将数据库复原为其原始状态。

您可以配置以下演示数据库：

- stores_demo 数据库举例说明了具有关于虚拟批发体育用品分销商的表以及示例中使用的其他表的关系模式。包含电力使用量和地理位置的表举例说明了时间系列和空间信息。GBase 8s 手册中的许多示例基于 stores_demo 数据库。
- superstores_demo 数据库举例说明了对象关系模式。superstores_demo 数据库包含扩展数据类型、类型和表继承以及用户定义的例程的示例。

用于安装演示数据库的脚本位于 \$GBS_HOME/bin 目录（在 UNIX™ 上）。

某些操作系统要求您具有运行 SQL 命令文件的执行许可权，在 DB-Access 中打开这些文件或其内容的读许可权，或具有保存修改过的文件和新文件的写许可权。使用 UNIX chmod 命令可允许执行初始化脚本所安装的 SQL 文件。

演示脚本是针对缺省语言环境设计的。如果您使用的是非缺省语言环境（例如 en_us.utf8 ），那么一些诸如 SET COLLATION 语句的功能可能不能正常发挥作用。

2.3.1 创建演示数据库

通过运行 dbaccessdemo 命令可创建演示数据库。

创建演示数据库时，该脚本会确认您是否要复制样本 SQL 命令文件。演示数据库包括的命令文件具有 .sql 扩展名，并包含您可以使用的样本 SQL 语句。

要创建演示数据库：

1. 创建目录。

必须对所创建路径名中的每个目录具有 UNIX™ 读许可权和执行许可权。

2. 将目录更改为新目录并运行 `dbaccessdemo` 或 `dbaccessdemo_ud` 命令。
3. 创建数据库时，初始化脚本在屏幕上显示了一系列消息。按 Y 以将命令文件复制到创建的目录中。

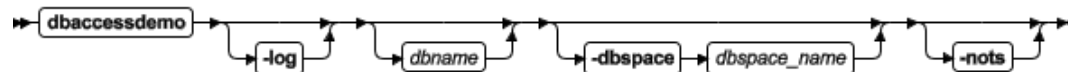
将创建演示数据库。您是该数据库的所有者和数据库管理员（DBA）。

如果要废弃对数据库或命令文件进行的更改，请重新运行 `dbaccessdemo` 或 `dbaccessdemo_ud` 命令，然后按 Y 以将现有命令文件替换为原始版本。

2.3.2 dbaccessdemo 命令：创建演示数据库

使用 `dbaccessdemo` 或 `dbaccessdemo_ud` 命令可创建演示数据库。

`stores_demo` 的语法



`superstores_demo` 的语法



-log

请求演示数据库的事务日志记录。

dbname

替代缺省数据库名称。必须遵循标识命名准则。

-dbspace

请求演示数据库的特定数据库空间位置。

dbspace_name

储存演示数据库。如果未指定数据库空间名称，那么在缺省情况下会将数据库的数据放入根数据库空间中。要创建数据库空间，请使用 `onspaces` 实用程序。

-nots

阻止在 `stores_demo` 数据库中创建时间系列表。

示例

以下命令创建名为 `stores_demo` 的数据库：

```
dbaccessdemo
```

下列示例创建名为 demo_db 的 stores_demo 数据库实例：

```
dbaccessdemo demo_db
```

以下命令初始化 stores_demo 数据库，并且还启动日志事务：

```
dbaccessdemo -log
```

下列命令在 dbspace_2 中创建名为 demo_db 的 stores_demo 数据库实例：

```
dbaccessdemo demo_db -dbspace dbspace_2
```

以下命令创建名为 superstores_demo 的数据库：

```
dbaccessdemo_ud
```

2.4 启动 DB-Access

通过从命令行运行 dbaccess 命令可启动 DB-Access。您可以选择是使用 DB-Access 菜单界面还是使用命令行界面。

您可以通过以下方式启动并使用 DB-Access：

- 从主菜单启动 DB-Access。
- 从特定菜单或屏幕启动 DB-Access。
- 在未显示 DB-Access 菜单的情况下，运行包含 SQL 语句的文件。
- 在没有菜单界面的情况下，从命令行以交互方式启动并运行 DB-Access。

如果 UNIX™ 上的 TERM、TERMCAP 或 TERMINFO 环境变量无法使 DB-Access 识别您使用的终端类型，那么主菜单不会显示。相反，会显示与下列文本类似的消息：

```
未知的终端类型。
```

如果在 UNIX 终端上使用窗口界面，请从非滚动控制台窗口发出 dbaccess 命令。

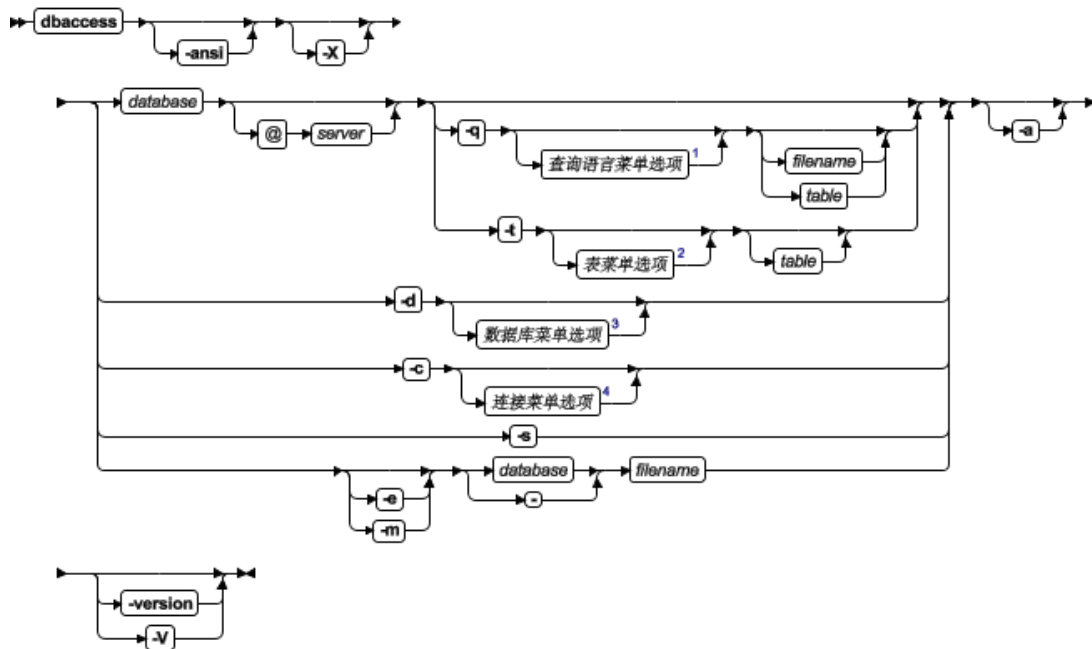
技巧： 如果操作系统找不到 dbaccess，请在程序名前面包含完整路径，如下所示：

```
$GBS_HOME/bin/dbaccess
```

2.4.1 dbaccess 命令：启动 DB-Access

使用 dbaccess 命令可启动 DB-Access。包含选项可指定数据库或命令文件，或者转至特定菜单屏幕。

语法



不带选项的 dbaccess 命令会启动主菜单，但不会选择任何数据库，也不会激活任何选项。您可从主菜单中选择子菜单。

-ansi

导致 DB-Access 在遇到对符合 ANSI 标准的语法的 GBase 8s 扩展时生成警告。有关更多信息，请参阅示例：检查 ANSI 符合性。

-a

在遇到第一个错误后直接停止进程。在遇到第一个错误后停止运行中的进程可以使数据更加一致。

-c

将“连接”菜单作为顶级菜单而启动。

-d

将“数据库”菜单作为顶级菜单而启动。

-e

回送 filename 指定的命令文件中的每一行。

-m

显示与命令文件中 SQL 语句有关的多个服务器级别生成的所有错误消息。

-q

将“查询语言”菜单（SQL 菜单）作为顶级菜单而启动。

-s

连接至 DB-Access 主菜单并显示有关当前会话的信息。

此信息包括数据库服务器名称、数据库服务器类型、主计算机、服务器功能和其他设置。

-t

将“表”菜单作为顶级菜单而启动。

-V

显示 DB-Access 的版本号和序列号而不启动应用程序。-V 不能与任何其他选项一起使用。

-version

显示 DB-Access 的版本号和构建信息（包括 GLS 库版本）而不启动应用程序。-version 不能与任何其他选项一起使用。

-X

激活用于 LOAD 和 UNLOAD 语句的十六进制格式。

database

您希望 DB-Access 在当前会话启动时连接到的数据库的名称。连字符（-）指示数据库已在命令文件的 DATABASE 语句中指定。

filename

指定一个命令文件以通过 SQL 菜单装入。

server

数据库服务器的名称。

table

指定数据库中的表。

如果从命令行中指定的子菜单或选项退出，那么将直接退出至操作系统命令行。

连接菜单选项

dbaccess 命令的“连接”菜单选项表示“连接”菜单的快捷键。

-cc

选择“连接”菜单上的“连接”选项。

-cd

选择“连接”菜单上的“断开连接”选项。

数据库菜单选项

dbaccess 命令的“数据库”菜单选项表示“数据库”菜单的快捷键。

-dc

选择“数据库”菜单上的“创建”选项。

-dcl

转至“创建数据库”菜单上的“日志”选项

-dd

选择“数据库”菜单上的“删除”选项。

-di

选择“数据库”菜单上的“信息”选项。通过此选项，您可以按下列方法添加另一字母以转至下一层菜单并查看：

-dib

当前数据库的数据库空间信息

-din

当前数据库的 NLS 信息

-dip

当前数据库中的存储过程

如果任何 -di 选项前均未附带数据库名称，那么必须从“选择数据库”屏幕选择当前数据库。

-dl

选择“数据库”菜单上的“关闭”选项。

-ds

选择“数据库”菜单上的“选择”选项。

查询语言菜单选项

dbaccess 命令的“查询语言”菜单选项表示“查询语言”菜单的快捷键。

-qc

选择 SQL 菜单上的“选择”选项。

-qd

选择 SQL 菜单上的“删除”选项。

-qi

选择 SQL 菜单上的“信息”选项。通过此选项，您可以如以下列表中所示添加另一字母（并指定表），以转至下一层菜单并查看：

-qic

表中的列

-qif

有关表的分段存储策略的信息

-qig

有关表中触发器的信息

-qii

表上的索引

-qio

表约束

-qip

表上的访问特权

-qir

表上的表级别引用特权

-qis

表状态信息

如果 -qi 选项未附带表名，那么必须从“表信息”屏幕中选择一个表名。

-qm

选择 SQL 菜单上的“修改”选项。

-qn

选择 SQL 菜单上的“新建”选项。

-qs

选择 SQL 菜单上的“保存”选项。

-qu

选择 SQL 菜单上的“使用编辑器”选项。

如果 -q 选项前未附带数据库名称，那么必须从“选择数据库”屏幕选择当前数据库。

在“查询语言”菜单上选择“修改”选项时，必须首先从“选择”菜单选择要修改的命令文件。然后，“修改”屏幕才会显示并显示文本。

限制： 不能直接转至 SQL 菜单上的“运行”或“输出”选项。尝试进行此操作会导致错误消息。

表菜单选项

dbaccess 命令的“表”菜单选项表示“表”菜单的快捷键。

-ta

选择“表”菜单上的“变更”选项。

-tc

选择“表”菜单上的“创建”选项。

-td

选择“表”菜单上的“删除”选项。

-ti

选择“表”菜单上的“信息”选项。通过此选项，您可以如以下列表中所示添加另一字母（并指定表），以转至下一层菜单并查看：

-tic

表中的列

-tif

有关表的分段存储策略的信息

-tig

有关表中触发器的信息

-tii

表上的索引

-tio

表约束

-tip

表上的访问特权

-tir

表上的表级别引用特权

-tis

表状态信息

如果 **-ti** 选项未附带表名，那么必须从“表信息”屏幕中选择一个表名。

如果 **-t** 选项前未附带数据库名称，那么必须从“选择数据库”屏幕选择当前数据库。

示例：为数据库启动 DB-Access

此示例显示如何启动 DB-Access 并指定要连接到的数据库。

假定联机的数据库服务器包含一个名为 `mystores` 的数据库。要使 `mystores` 数据库成为当前数据库，通过下列命令启动 DB-Access：

```
dbaccess mystores
```

您可以指定未联机的数据库服务器上的数据库。例如：下列任一命令均会选择 `xyz` 数据库服务器上的 `newstores` 数据库：

```
dbaccess newstores@xyz
```

```
dbaccess //xyz/newstores
```

当 DB-Access 启动时，指定的数据库和数据库服务器名称显示在虚线上，如下图所示。

图：带有数据库和数据库服务器名称的 DB-Access 主菜单

```
DB-Access:   查询语言  连接  数据库  表  会话  退出
```

```
----- newstores@xyz ----- 按 CTRL-W 以获得帮助 ---
```

示例：运行命令文件

此示例显示如何启动 DB-Access 并运行包含 SQL 语句的命令文件。

以下样本命令在 mystores 数据库上运行名为 sel_stock.sql 的文件中的 SQL 语句：

```
dbaccess mystores sel_stock
```

以下样本命令在 sel_all.sql 文件指定的数据库上运行该文件中的 SQL 语句：

```
dbaccess - sel_all.sql
```

某些操作系统要求您具有运行 SQL 命令文件的执行许可权，在 DB-Access 中打开这些文件或其内容的读许可权，或具有保存修改过的文件或新文件的写许可权。

使用 UNIX™ chmod 命令可允许执行初始化脚本所安装的 SQL 文件。

示例：查看信息模式

此示例显示如何启动 DB-Access 并查看指定数据库的信息模式。

\$GBS_HOME/etc 目录中的 xpg4_is.sql 文件会创建信息模式并为指定的数据库安装视图。

下列命令创建数据库 mystores 的信息模式：

```
dbaccess mystores $GBS_HOME/etc/xpg4_is.sql
```

信息模式将符合 X/Open XPG4 且具有 GBase 8s 扩展的四个仅供参考的视图添加到数据库。运行 xpg4_is.sql 之后，使用 DB-Access 检索有关指定数据库中您有权存取的表和列的信息。

提示： 不要在 ANSI 数据库上安装符合 XPG4 标准的视图，因为符合 XPG4 标准的视图与 SQL 标准委员会所定义的符合 ANSI 标准的信息模式视图在格式上有很大差别。

示例：检查 ANSI 符合性

此示例显示如何启动 DB-Access 并检查数据库是否符合 ANSI 标准。

要检查 SQL 语句是否符合 ANSI 标准，请包括 -ansi 选项或设置 DBANSIWARN 环境变量。将 -ansi 选项与其他 dbaccess 选项配合使用，例如 -dc（用于创建数据库）、-tc 或 -ta（用于创建或改变表）或 -qcfilename（用于选择命令文件）。在 DB-Access 创建数据库 research 时，下列命令可检查 ANSI 符合性：

```
dbaccess -ansi -dc research
```

如果设置了 DBANSIWARN 环境变量，那么不需要在命令行上指定 -ansi 选项。

在下列情况下，DB-Access 会显示 SQLSTATE 值和警告：

- 包括 `-ansi` 选项或设置 `DBANSIWARN` 环境变量。
- 存取或创建 ANSI 数据库。
- 在命令行方式下运行 DB-Access 或指定 `.sql` 输入文件。
- 运行 SQL 语句将生成警告而不是错误。

示例：以十六进制格式显示不可打印字符

此示例启动 DB-Access 并激活十六进制装入和卸载格式 (XLUF)，这样 LOAD 和 UNLOAD SQL 语句就可以通过十六进制格式对不可打印的 ASCII 符号进行格式化。

以下命令将为 `mystores` 数据库激活 XLUF 格式：

```
dbaccess -X mystores
```

UNLOAD 语句生成的 `.unl` 文件包含十六进制格式更改。

2.4.2 在不使用菜单的情况下以交互方式运行 DB-Access

如果不希望使用菜单，并且没有预编译的 SQL 文件，请使用键盘或标准输入设备从命令行输入 SQL 语句。

在不使用菜单自变量并且将连字符作为最后一个自变量的情况下启动 DB-Access 时，DB-Access 会处理来自标准输入设备（在 UNIX™ 上）的命令。DB-Access 会读取您指示输入完成之前所输入的内容。然后，DB-Access 会处理您的输入，并将结果写入标准输出设备（在 UNIX 上）。

DB-Access 以交互方式读取和运行来自终端键盘的 SQL 语句。当 DB-Access 交互式运行时，大于 (>) 提示符标记可输入下一个 SQL 语句的行。

当输入分号 (;) 以结束单个 SQL 语句时，DB-Access 将处理该语句。当按 CTRL-D 以结束交互式会话时，DB-Access 停止运行。下列示例显示了交互式会话中的用户输入和结果：

```
dbaccess - -  
  
>database stores_demo;  
  
Database selected.  
  
>select count(*) from systables;  
  
(count(*))
```

```
21

1 row(s) retrieved.

>^D

dbaccess - -
>database stores_demo;

Database selected.

>select count(*) from systables;

(count(*))

21

1 row(s) retrieved.

>^D
```

批处理命令输入 (UNIX)

可以使用直接插入 shell 脚本来提供一个或多个 SQL 语句。例如，可以使用具有直接插入标准输入文件的 UNIX C、Bourne 或 Korn shell：

```
dbaccess mystores- <<EOT!

select avg(customer_num) from customer

where fname matches '[A-G]*';

EOT!
```

可以使用管道来提供 SQL 语句，如此 UNIX 示例所示：

```
echo 'select count(*) from systables' | dbaccess mystores
```

DB-Access 将任何以感叹号(!)开始的行解释为 shell 命令。可以将 shell 转义行与 SQL 语句混合，并将其放入 SQL 语句中，如下所示：

```
dbaccess mystores -
>select
!echo hello
>hello
count(*) from systables;
>
(count(*))

      21

1 row(s) retrieved.
>
```

以交互方式连接到数据库环境

可以在以交互方式发出的 SQL 语句中使用 CONNECT . . . USER 语法。但是，当连接至缺省数据库服务器时，DB-Access 不支持 CONNECT 语句的 USER 子句。

当您在交互方式下将 USER ‘user identifier’ 子句包括在 CONNECT 语句中时，DB-Access 会提示您输入密码。

下列两个命令示例显示了如何在交互方式下连接至数据库服务器。第一个示例使用未指定用户标识的 CONNECT 语句。

```
dbaccess -nohistory- -
> connect to '@starfish';

Connected.
```

如果在 CONNECT 语句中包括 USER 子句（如第二个示例所示），DB-Access 使用回送禁止

来提示您输入密码：

```
> connect to '@starfish' user 'marae';
```

```
ENTER PASSWORD:
```

```
Connected.
```

限制： 为了安全起见，不要在密码可以被看到的屏幕上输入密码。另外，当交互式使用 DB-Access 时，不要在 CONNECT 语句中包括 USING password 子句。如果处于交互方式下，并在提示之前尝试输入密码，那么将会显示错误消息。

您可以在包括 USER 子句的 DB-Access 文件中运行 CONNECT 语句的 USER 子句。下列示例使用包含 CONNECT 语句（具有 USING 子句）的命令文件连接至数据库服务器：

```
dbaccess - connfile.sql
```

重要： 包含以下语句的 SQL 命令文件受到保护，除 USER 子句所标识的 user_id 之外的其他任何人都无法访问。

```
CONNECT TO '@dbserver' USER 'user_id' USING password
```

对于 UNIX™，以下示例使用 shell 文件连接至数据库服务器。DB-Access 提示您输入密码。

```
dbaccess - - <<\!
```

```
connect to '@starfish' user 'marae';
```

```
!
```

```
ENTER PASSWORD:
```

此处，分隔引号保留了数据库服务器名称中和用户授权标识中的字母大小写。

3 全屏菜单界面

DB-Access 全屏菜单界面将指导您逐步运行 SQL 语句。

DB-Access 用户界面结合了下列功能部件：

- 菜单的层次结构
- 屏幕，提示您作出简要响应以及从选择列表中作出选择
- 上下文“帮助”屏幕
- 交互式模式编辑器，可帮助您构造表
- SQL 程序员环境，包括以下功能部件：
 - 内置 SQL 编辑器，可在其中输入和修改 SQL 和 SPL 语句
 - 一个选项，可用于选择使用所选的另一编辑器
 - 数据库服务器语法检查程序和运行时调试器
 - 存储、检索和执行 SQL 与 SPL 例程
- 选择数据库查询和报告的输出

3.1 查询语言选项

使用“查询语言”选项可输入、修改、保存、检索和运行 SQL 语句。DB-Access 将语句（如有）保留在编辑器中。这些语句被称为当前语句。

使用“查询语言”选项：

- 了解 SQL 和 SPL。

例如：使用“查询语言”选项练习 GBase 8s SQL 指南：教程 中的示例。

- 作为 DB-Access 模式编辑器的备用方法创建和改变表结构。
- 选择、显示、添加、更新和删除数据。

SQL 菜单具有以下选项：

选项	用途
新建	清除 SQL 编辑器中的当前语句和位置光标。

选项	用途
运行	运行当前 SQL 语句。将显示一条消息，或者显示查询所检索到的数据及检索到的行数。
修改	允许您在 SQL 编辑器中修改当前 SQL 语句。
使用编辑器	启动系统编辑器，以便您可以修改当前语句或创建新的语句。“使用编辑器”可与“新建”和“修改”互换使用。
输出	将运行选项输出重定向至文件、打印机或系统管道。
选择	列出 SQL 命令文件，以便您可以选择要运行或修改的文件。
保存	将当前 SQL 语句保存到文件中，以供以后使用。
信息	显示表信息，如列、索引、特权、约束、触发器、状态和分段存储策略。
删除	删除指定的 SQL 命令文件。
退出	返回到主菜单。

3.1.1 SQL 编辑器

选择“新建”或“修改”选项时，将显示 SQL 编辑器。可以根据需要输入任意行数的文本。虽然您可能会受到系统内存约束的限制，但是不会受到屏幕大小的限制。使用 SQL 编辑器时，可以根据需要输入任意行数的文本。虽然您可能会受到系统内存约束或最大 SQL 语句大小 64 KB 的限制，但是不会受到屏幕大小的限制。如果未使用“保存”选项来保存输入的语句，那么选择清除 SQL 编辑器的选项（例如“新建”或“选择”）时，将删除这些语句。

SQL 编辑器在一行中显示的字符不会超过 80 个，并且不会换行。

- 如果选择的现有命令文件中一行的字符扩展到第 80 列之外，那么 DB-Access 在第 80 列中显示百分比符号（%）来指示溢出。您看不到百分号之后的所有字符，但是语句会正确运行。
- 如果在新的命令文件中输入字符，从而使行扩展到第 80 列之外，那么 DB-Access 覆盖第 80 列中的所有字符。您看不到溢出，并且语句不能正确运行。

要使全部文本显示在屏幕上，请在每行前 80 个字符中的合理位置按 Enter 键。

如果必须输入超过 80 个字符的带引号字符串（例如插入到长 CHAR 列），请使用系统编

辑器而不是 SQL 编辑器。

如果要在文本中包括注释：

- 对符合 ANSI 标准的数据库使用双减号。
- 每个注释行以双减号（--）注释指示符开头。注释指示符跨越整行。
- 对不符合 ANSI 标准的数据库使用花括号（{}）。将整个注释指示符包括在花括号之间。

3.1.2 系统编辑器

当要输入或修改长 SQL 语句或一系列语句时，相对于 SQL 编辑器来说，您可能认为系统编辑器更灵活，对它也更熟悉。从 SQL 菜单选择“使用编辑器”选项以使用系统编辑器。

如果未设置 DBEDIT 环境变量，必须选择要用于会话的文本编辑器。如果选择“使用编辑器”，那么 DB-Access 将针对每个会话提示您进行一次接受或覆盖缺省系统编辑器的操作。

DB-Access 显示的缺省编辑器取决于您为操作系统建立的首选项：

- 公共 UNIX[™] 系统编辑器是 vi 和 ex。
- 如果您将文本编辑器用作系统缺省程序，那么您必须将 .sql 文件保存为文本。

按 Enter 键以选择在 USE-EDITOR 提示符后指定的缺省编辑器。要使用其他编辑器，输入该编辑器的名称，然后按 Enter 键。

3.1.3 “运行”选项支持的语句

退出编辑器屏幕之后，SQL 菜单将再次打开，同时突出显示“运行”选项，并且在屏幕底部显示语句文本。您可以使用“运行”选项来运行大部分 SQL 语句。

要运行未列出的语句，请使用 SQL 菜单选项“新建”（或“使用编辑器”）和“保存”来输入并保存语句，然后从命令行运行已保存的文件。

以下是可使用“运行”选项运行的 SQL 语句的列表。

- ALLOCATE COLLECTION
- ALLOCATE DESCRIPTOR
- ALLOCATE ROW
- ALTER ACCESS_METHOD
- ALTER FRAGMENT

- ALTER FUNCTION
- ALTER INDEX
- ALTER PROCEDURE
- ALTER ROUTINE
- ALTER SECURITY LABEL COMPONENT
- ALTER SEQUENCE
- ALTER TABLE
- BEGIN WORK
- CLOSE
- CLOSE DATABASE
- COMMIT WORK
- CONNECT
- CREATE ACCESS_METHOD
- CREATE AGGREGATE
- CREATE CAST
- CREATE DATABASE
- CREATE DISTINCT TYPE
- CREATE EXTERNAL TABLE
- CREATE FUNCTION
- CREATE FUNCTION FROM
- CREATE INDEX
- CREATE OPAQUE TYPE
- CREATE OPCLASS
- CREATE PROCEDURE
- CREATE ROLE
- CREATE ROUTINE FROM

- CREATE ROW TYPE
- CREATE SCHEMA
- CREATE SECURITY LABEL COMPONENT
- CREATE SECURITY LABEL
- CREATE SECURITY POLICY
- CREATE SEQUENCE
- CREATE SYNONYM
- CREATE TABLE
- CREATE TRIGGER
- CREATE VIEW
- CREATE XDATASOURCE
- CREATE XDATASOURCE TYPE
- DATABASE
- DEALLOCATE COLLECTION
- DEALLOCATE DESCRIPTOR
- DEALLOCATE ROW
- DECLARE
- DELETE
- DESCRIBE
- DESCRIBE INPUT
- DISCONNECT
- DROP ACCESS METHOD
- DROP AGGREGATE
- DROP CAST
- DROP DATABASE
- DROP FUNCTION

- DROP INDEX
- DROP OPAQUE TYPE
- DROP OPCLASS
- DROP PROCEDURE
- DROP ROLE
- DROP ROW TYPE
- DROP SECURITY LABEL COMPONENT/POLICY/LABEL
- DROP SEQUENCE
- DROP SYNONYM
- DROP TABLE
- DROP TRIGGER
- DROP TYPE
- DROP VIEW
- DROP XADATASOURCE
- DROP XADATASOURCE TYPE
- EXECUTE
- EXECUTE FUNCTION
- EXECUTE IMMEDIATE
- EXECUTE PROCEDURE
- FETCH
- FLUSH
- FREE
- GET DESCRIPTOR
- GET DIAGNOSTICS
- GRANT
- GRANT DBSECADM

- GRANT DEFAULT ROLE
- GRANT EXEMPTION
- GRANT FRAGMENT
- GRANT SECURITY LABEL
- INFO
- INSERT
- LOAD
- LOCK TABLE
- MERGE
- OPEN
- OUTPUT
- PREPARE
- PUT
- RENAME COLUMN
- RENAME DATABASE
- RENAME INDEX
- RENAME SEQUENCE
- RENAME TABLE
- REVOKE
- REVOKE DBSECADM
- REVOKE DEFAULT ROLE
- REVOKE EXEMPTION
- REVOKE FRAGMENT
- REVOKE SECURITY LABEL
- ROLLBACK WORK
- SAVE EXTERNAL DIRECTIVES

- SELECT
- SET AUTOFREE
- SET COLLATION
- SET CONNECTION
- SET CONSTRAINTS
- SET DATASKIP
- SET DEBUG FILE TO
- SET DEFERRED PREPARE
- SET DESCRIPTOR
- SET ENCRYPTION PASSWORD
- SET ENVIRONMENT
- SET EXPLAIN
- SET ISOLATION
- SET LOCK MODE
- SET LOG
- SET OPTIMIZATION
- SET PDQPRIORITY
- SET ROLE
- SET SESSION AUTHORIZATION
- SET STATEMENT CACHE
- SET TRANSACTION
- START VIOLATIONS TABLE
- STOP VIOLATIONS TABLE
- TRUNCATE
- UNLOAD
- UNLOCK TABLE

- UPDATE
- UPDATE STATISTICS
- WHENEVER

3.1.4 重定向查询结果

SELECT 语句的输出通常显示在屏幕上。您可以使用 SQL 菜单上的“输出”选项将查询结果路由到打印机，将其存储在系统文件中，或将其通过管道输送到程序。“输出”选项产生的结果与 SQL 的 OUTPUT 语句的结果相同。

SELECT 语句必须作为当前语句出现在屏幕上。从 SQL 菜单选择“输出”选项以显示“输出”菜单。

您有以下输出选项：

- 将查询结果直接发送到打印机。DB-Access 将结果发送到缺省打印机，并在屏幕的底部显示一条指示检索行数的消息。查询结果不会显示在屏幕上。您可以设置 DBPRINT 环境变量，以指定缺省打印机。
- 将查询结果写入新文件，或将结果追加到现有文件。如果 DB-Access 提示您输入文件名时，您未指定路径，那么文件将存储在启动 DB-Access 时所在的目录中。
- 将查询结果发送到管道。指定目标程序，如 more，以便通过它发送输出。DB-Access 将结果发送到该管道。

在 UNIX™ 系统上，您必须具有运行目标程序的许可权。

3.1.5 选择现有 SQL 语句

将 SQL 语句保存到命令文件中后，可以随时检索该命令文件并运行或编辑 SQL 语句。

选择 SQL 菜单上的“选择”选项以显示“选择”屏幕，该屏幕上具有您可以访问的命令文件的列表。这些文件具有扩展名 .sql（虽然未显示扩展名）。例如，下图列出了演示数据库中的命令文件。

图：列出当前 .sql 文件的“选择”屏幕

选择 >>

使用方向键选择命令文件，或输入名称，然后按 Enter 键。

----- mystores@dbserver1 ----- 按 CTRL-W 以获得帮助 -----

alt_cat	c_state	d_trig	sel_ojoin1
c_calls	c_stock	d_view	sel_ojoin2
c_cat	c_stores	del_stock	sel_ojoin3
c_custom	c_table	ins_table	sel_ojoin4
c_index	c_trig	opt_disk	sel_order
c_items	c_type	sel_agg	sel_sub
c_manuf	c_view1	sel_all	sel_union
c_orders	c_view2	sel_group	upd_table
c_proc	d_proc	sel_join	

如果当前数据库不存在，那么列表包括当前目录以及 DBPATH 环境变量指定的任何目录中的所有命令文件。

重要： 此列表仅包括那些具有 .sql 扩展名的文件名。如果在 DB-Access 外部创建 SQL 文件，但在保存时未使用 .sql 扩展名，那么该文件不会显示在供选择的文件列表中。

DB-Access 仅可以识别存储在从中启动 DB-Access 的目录中的文件。如果“选择”命令生成空列表，而您确信有命令文件，请退出 DB-Access，将目录更改为包含 .sql 文件的目录，然后重新启动 DB-Access。

3.1.6 保存当前 SQL 语句

可以将 SQL 语句保存在文件中以备以后使用，例如从命令行运行语句或通过 SQL 菜单上的“选择”选项检索保存的语句。

要将当前的一个或多个 SQL 语句保存在文件中，请选择 SQL 菜单上的“保存”选项。输

入命令文件的名称：

- 使用 1 - 10 个字符。以字母开头，然后使用字母、数字和下划线（_）的任意组合。
按 Enter 键以保存文件。
- UNIX[™]：文件名区分大小写。文件 orders 与 Orders 或 ORDERS 并不相同。

DB-Access 会向文件名追加扩展名 .sql。例如：如果将文件命名为 cust1，那么 DB-Access 用名称 cust1.sql 存储文件。“选择”屏幕仍然会列出 cust1，但是如果从命令行列出目录文件，操作系统会将同一文件标识为 cust1.sql。

3.1.7 支持 SPL 例程

您可以从 SQL 菜单创建并运行以 SPL 编写的例程。

可以将 SPL 例程存储到单独的命令文件中，然后从应用程序进行调用或将其作为独立程序运行。创建 SPL 例程之后，可以在 DB-Access 中通过相应的 SQL 语句运行该例程。下列示例对步骤进行了详细说明。

创建并运行 SQL 例程

1. 要创建例程文本，直接在“新建”屏幕或“使用编辑器”屏幕中输入。在 CREATE PROCEDURE 语句的语句块中输入例程的 SPL 和 SQL 语句。

如果例程返回值，请使用 CREATE FUNCTION 语句。

2. 使用“运行”选项来创建例程并在 sysprocedures 系统目录表中对其进行注册。
3. 使用“新建”屏幕输入 EXECUTE PROCEDURE 语句，此语句将指定要运行的例程。

如果使用 GBase 8s 并通过 CREATE FUNCTION 语句创建例程，那么输入 EXECUTE FUNCTION 语句来运行函数。

4. 使用“运行”选项来运行例程并显示结果。

例如，演示数据库随附的 c_proc.sql 命令文件包含 SPL。要能够运行例程，请先将 c_proc.sql 文件中的词 procedure 更改为 function，因为该例程会返回值。然后使用“选择”选项，并选择 c_proc。

首先必须在数据库中注册该例程。如下图所示，选择“运行”选项。

图：在 SQL 菜单上显示 SPL 例程的文本

SQL: N/新建 R/运行 M/修改 U/使用编辑器 O/输出 C/选择 S/保存 I/信息
D/删除 E/退出

运行当前的 SQL 语句。

----- mydata@mynewdb ----- 按 CTRL-W 以获得帮助 -----

```
create function read_address (lastname char(15))

    returning char(15), char(15), char(20), char(15),char(2), char(5);

    define p_fname, p_city char(15);

    define p_add char(20);

    define p_state char(2);

    define p_zip char(5);

    select fname, address1, city, state, zipcode

        into p_fname, p_add, p_city, p_state, p_zip

        from customer

        where lname = lastname;

    return p_fname, lastname, p_add, p_city, p_state, p_zip;

end procedure;
```

DB-Access 显示一条消息以指示数据库服务器已创建例程。要运行例程，请从 SQL 菜单选择“新建”，然后输入相应的 EXECUTE 语句。在以下示例中，用户请求其姓氏为 Pauli 的客户的地址：

```
EXECUTE PROCEDURE read_address ("Pauli")
```

在“新建”屏幕中输入 EXECUTE PROCEDURE 或 EXECUTE FUNCTION 语句之后，按 Esc 键返回到 SQL 菜单。从 SQL 菜单选择“运行”以运行例程。下图显示了运行例程的结果。

图：在 SQL 菜单上运行 SPL 例程的结果

```
SQL:   N/新建  R/运行  M/修改  U/使用编辑器  O/输出  C/选择  S/保存  I/信息
       D/删除  E/退出
```

运行当前的 SQL 语句。

----- mydata@mynewdb ----- 按 CTRL-W 以获得帮助 -----

Ludwig

Pauli

213 Erstwild Court

Sunnyvale

CA

94086

提示： SPL 例程以可执行文件格式存储在系统目录表中。使用“数据库信息”菜单上的“例程”选项显示当前数据库中例程的列表或显示指定例程的文本。

3.1.8 出现错误时发生何事

如果 SQL 语句中有任何语法或输入错误，那么 DB-Access 不会处理该语句。相反，它会继续显示语句的文本以及描述错误的消息。

如果出现执行或运行时错误，那么 DB-Access 会继续处理语句并返回错误消息。例如：如果尝试创建已经创建的表，那么下列消息会显示在屏幕底部：

310: 数据库中已有表 (mavis.mystock)。

如果尝试运行包含多个 SQL 语句的语句，那么可能不会立即看到错误消息。例如：如果第一个语句是正确运行的 SELECT 语句，而下一语句包含输入错误，那么第一个语句检索到的数据首先显示在屏幕上，然后才会显示第二个语句的错误消息。

DB-Access 检测到错误时，处理将停止，并将突出显示 SQL 菜单上的“修改”选项。选择下列方法之一来纠正语句：

- 按 Enter 键以选择“修改”，这样会返回到 SQL 编辑器。
- 选择“使用编辑器”选项来使用所选择的缺省编辑器。

3.2 数据库选项

使用“数据库”选项可处理数据库和事务。

使用“数据库”选项可执行以下操作：

- 创建数据库或选择数据库。

您所处理的数据库称为当前数据库。

- 检索并显示有关数据库的信息，如可用的数据库空间和例程的文本。
- 删除现有的数据库或关闭当前数据库。
- 落实或回滚事务。

您只能访问位于当前数据库服务器上的数据库。要选择某一数据库服务器作为当前数据库服务器，您可以在启动 DB-Access 时指定数据库服务器，可以使用“连接”菜单，也可以从 SQL 菜单运行 CONNECT 语句。如果您未显式选择数据库服务器，那么 DB-Access 使用 sqlhosts 文件中的第一行 dbservername 条目指定的缺省数据库服务器作为当前数据库。

如果在另一数据库已打开时选择或创建数据库，那么 DB-Access 将在使所选择数据库成为当前数据库或新数据库之前关闭该数据库。

“数据库”菜单显示以下选项：

选项	用途
选择	使数据库成为当前数据库
创建	构建新数据库并使该数据库成为当前数据库
信息	显示有关当前数据库的信息
删除	从系统中除去数据库。不能删除当前数据库。
关闭	关闭当前数据库
退出	退出“数据库”菜单并返回到主菜单

3. 2. 1 可用数据库列表

选择“选择”选项时，将打开“选择数据库”屏幕。可用数据库列表中的第一个数据库将突出显示，同时显示数据库服务器的名称。

列表按数据库服务器的字母顺序排列，然后按每个数据库服务器的数据库字母顺序排列。在“选择数据库”屏幕上最多可以显示 512 个数据库名称。

重要： 在“选择数据库”屏幕中，数据库名限制为 18 个字符。如果数据库名称长度超过 18 个字符，那么您将看到名称的前 17 个字符后跟一个“+”号。输入“+”号以在 vi 中

显示完整的长名称。要从 vi 中退出，按 ESC ZZ。

显示的可用数据库的列表取决于两个因素：

- 某些环境变量的设置。
 - 如果您使用一个数据库服务器，那么 DB-Access 会显示当前数据库服务器上 and DBPATH 设置中所有数据库的名称。
- 当前连接。例如：
 - 如果不存在显式连接，那么 DB-Access 会显示 DBPATH 设置中的数据库。
 - 如果存在当前显式连接，那么将显示属于当前数据库服务器的 DBPATH 设置中的所有数据库。

3.2.2 检索非缺省语言环境信息

全球化影响列表在 DB-Access 中显示的顺序。全球化允许显示非英语语言数据并进行适当的排序。GBase 8s 支持使用 Global Language Support (GLS) 语言环境进行全球化。较早的数据库服务器版本使用本机语言支持 (NLS) 来实现此目的。

如果当前数据库支持全球化，那么您可以在“数据库信息”菜单上选择 NLS 选项以显示有关整理顺序和 CType (字符分类类型) 的信息，如下图所示。

图：显示了全球化信息的“数据库信息”菜单

```
数据库信息:  B/数据库空间  N/NLS  R/例程  D/数据库  E/退出

显示数据库的 NLS 信息。

----- - stores_demo ----- 按 CTRL-W 以获得帮助 -----

fr_fr.8859-1 整理顺序

CType
```

DB-Access 在“数据库信息”菜单上未提供用于显示 GLS 整理顺序和字符分类类型的选项。要获取有关为数据库服务器启用的 GLS 语言环境的信息，请使用 SQL 编辑器输入以下查询：

```
SELECT tabname, site FROM systables
```

```
WHERE tabid = 90 OR tabid = 91
```

具有 tabid 90 的行存储数据库语言环境的 COLLATION 类别。具有 tabid 91 的行存储数据库语言环境的 CTYPE 类别。下图显示了针对缺省美国英语语言环境的上述查询的结果。

图：检索 GLS 信息

```
SQL:  N/新建  R/运行  M/修改  U/使用编辑器  O/输出  C/选择  S/保存  I/信息
D/删除  E/退出

运行当前的 SQL 语句。

----- mydata@mynewdb ----- 按 CTRL-W 以获得帮助 ---

tabname          GL_COLLATE
site              en_US.819

tabname          GL_CTYPE
site              en_US.819

检索到 2 行
```

3. 2. 3 关闭数据库

要关闭当前数据库，请从“数据库”菜单中选择“关闭”选项。

如果开始某个事务，但未将其落实或回滚，然后尝试关闭包含事务的数据库，那么将打开“事务”菜单。在关闭当前数据库之前，“事务”菜单会确保您落实或回滚活动事务。

重要： 请小心选择选项。如果选择“落实”，可能会落实不必要的事务；如果选择“回滚”，肯定会丢失所有新事务。

在未首先终止事务的情况下，每当您试图打开新数据库或尝试退出 DB-Access 菜单系统时，也会打开“事务”菜单。

重要： 如果在符合 ANSI 标准的数据库中开始事务，但未发出 COMMIT 语句或 ROLLBACK 语句，然后尝试使用非菜单方式关闭数据库，那么 DB-Access 将为您落实该事务。如果不想落实事务，那么从命令行中发出 ROLLBACK 语句和 CLOSE DATABASE 语句。

3.3 表选项

使用“表”选项可处理表。

要在不进行 SQL 编程的情况下执行以下任何表管理任务，请使用“表”选项：

- 创建新表
- 定义新表或现有表的分段存储策略
- 改变、删除或显示有关现有表的信息

使用“表”菜单选项，如下表所示：

选项	用途
创建	允许您定义新表的结构。“创建表”菜单提供了用于内置数据类型的数据类型选项。要使用某种扩展数据类型（如智能大对象、用户定义的（不透明）数据类型或集合数据类型）来定义列，请使用 SQL 菜单输入并运行 CREATE TABLE 语句。DB-Access 只能构造以升序排列的非集群 B 型树列索引。如果需要散列或混合分段存储，请使用 SQL 菜单输入并运行 CREATE TABLE 或 ALTER TABLE 语句。
改变	允许您改变现有表的结构，包括列、分段存储和约束。您必须具有“改变”特权才能成功改变表。要使用 LOAD 语句将数据插入表中，您必须对该表具有“插入”和“选择”特权。
信息	显示有关表结构的信息
删除	从数据库中删除表
移动	将表从当前数据库移动到另一个数据库。
退出	返回到 DB-Access 主菜单

“创建表”和“变更表”菜单都具有下表中描述的不同选项：

选项	用途
添加	显示模式编辑器，您可以从中将新列添加至表
修改	显示使用“添加”选项定义的列，以便能够在构建表之前修改列结构

选项	用途
删除	从表中删除现有列
屏幕	在模式编辑器中显示列定义的下一屏
表选项	使您可以显示和选择新表的存储空间。显示用于设置新表的分段存储策略的选项。使您可以设置新表的扩展数据块大小和锁定方式。为现有分段表添加或删除行标识。
约束	使您可以定义主键约束、外键约束、检查约束和唯一约束以及设置缺省列值
退出	构建、重新构建或废弃使用其他选项指定的模式和结构，然后返回到“表”菜单

重要： 必须使用空格键在菜单选项之间移动，因为方向键控制模式编辑器中的光标移动。

3.3.1 显示表信息

使用“表”菜单上的“信息”选项可显示有关表结构的信息。

注意以下各项：

- 如果您不是表所有者，那么表名带有所有者名称前缀，如 `june.clients`。
- 如果表列表不能在一个屏幕中完全显示，那么最后一个条目为省略号 (...)。使用方向键来突出显示省略号，将会显示表名的下一页。
- 如果启用全球化，那么表名列表将依据创建数据库时定义的数据库整理规则进行排序。因此，使用不同 DB-Access 排序序列的不同用户将看到按相同顺序列出的数据库中的表名。

要请求有关不同数据库服务器上表的信息，请在提示符处使用格式 `database@server:table` 或 `database@server:owner.table`。以下示例请求有关 `customer` 表的信息，该表由 `dba` 在数据库服务器 `topend` 上的 `accounts` 数据库中创建：

```
INFO FOR TABLE >> accounts@topend:dba.customer
```

“信息”菜单具有下列选项：

选项	用途
列	按列名列出数据类型，并指示哪些列可以包含空值

选项	用途
索引	描述为指定表定义的每个索引
特权	列出对指定表具有“选择”、“更新”、“插入”、“删除”、“索引”或“改变”特权的用户
引用	列出对指定表具有表级别“引用”特权的用户以及他们可以引用的列的名称
状态	列出当前表的表名、所有者、行大小、行和列数以及创建日期
约束	显示引用约束、主键约束、唯一约束和检查约束以及指定表中的列的缺省值
触发器	显示指定触发器的标题和正文信息
表	重新显示“表信息”菜单，以便您可以选择另一个表进行检查
分段	列出指定给表的分段数据库空间，并且对于基于表达式的分段存储，显示指定给每个数据库空间的表达式
退出	返回到“表”菜单

提示： 从“创建表”菜单中，使用“表选项”来查看扩展块和锁定方式信息，或发出 SELECT 语句来列出 systables 系统目录表中的表描述。

3.4 连接选项和会话选项

如果要连接至特定数据库服务器和数据库或显式地与当前数据库环境断开连接，那么使用“连接”选项。使用“会话”选项来显示有关当前 DB-Access 会话的信息。

数据库服务器检查由客户机传递的客户机语言环境信息，验证数据库语言环境，并且确定用于在客户机和数据库之间传输数据的服务器处理语言环境。

DB-Access 与 GBase 8s 之间的连接可以使用安全套接字层（SSL）协议，这种通信协议确保了通过网络传输的数据的私密性和完整性。

“连接”菜单显示以下选项：

选项	用途
连接	连接到数据库环境。要存取特定数据库，您必须具有许可权。

选项	用途
断开连接	与当前数据库环境断开连接
退出	返回到 DB-Access 主菜单

使用“连接”选项时，“选择数据库”屏幕将按字母顺序列出指定数据库服务器上的所有可用数据库。“选择数据库”屏幕上的数据库列表取决于当前连接。例如：

- 如果不存在当前连接或者当前连接是隐式缺省连接，那么将显示 DBPATH 环境变量设置中列出的所有数据库。
- 如果存在当前显式连接，那么显示附属于当前服务器的 DBPATH 中的所有数据库。

3.4.1 隐式关闭

当您连接到新环境时，DB-Access 将关闭所有打开的连接或数据库。

在以下情况下，DB-Access 将关闭所有打开的连接或数据库：

- 当您连接至新的数据库环境而没有显式地与当前环境断开连接时，DB-Access 将执行隐式断开连接并关闭数据库。
- 当您连接到 database@server，然后关闭数据库时，数据库服务器将保持连接。
- 当您连接至数据库服务器并打开数据库然后关闭数据库时，数据库服务器保持连接。
- 如果您打开数据库，然后尝试连接至数据库服务器，那么 DB-Access 执行隐式断开连接并关闭数据库。

只允许一个连接。您必须与同打开的数据库相关联的数据库服务器断开连接或关闭数据库，然后才能连接至另一个数据库服务器。

如果 DB-Access 必须关闭仍有未完成事务的数据库，它将提示您落实或回滚这些事务。

4 附录

4.1 如何阅读 SQL 语句的联机帮助

特定约定用于在 DB-Access 联机帮助屏幕中表示 SQL 语句的语法。

可以按以下两种方法之一请求 SQL 语句的联机帮助：

- 突出显示 SQL 菜单上的“新建”、“修改”或“使用编辑器”选项，然后按 CTRL-W。
- 在位于 SQL 菜单的“新建”或“修改”屏幕时，按 CTRL-W。

请求 DB-Access 中 SQL 语句联机帮助时显示的语法图的格式与 GBase 8s SQL 指南：语法 中的语法图不同。

以下列表中描述了控制 DB-Access 联机帮助屏幕中的 SQL 语句语法的约定和规则。

ABC

SQL 语句中以大写字母显示的任何术语均为关键字。准确地输入关键字，忽略大小写，如下示例中所示：

```
CREATE SYNONYM synonym-name
```

此语法指示您必须输入关键字 CREATE SYNONYM 或 create synonym，而不能添加或删除空格或字母。

abc

用一个值替换以小写字母显示的任何术语。在上一个示例中，用一个值替换 synonym-name。

()

如同显示的那样输入任何圆括号。它们是 SQL 语句语法的一部分，而不是特殊符号。

[]

不要输入方括号作为语句的一部分。它们括起语句中可选的任何部分。例如：

```
CREATE [TEMP] TABLE
```

此语法指示您可以输入 CREATE TABLE，也可以输入 CREATE TEMP TABLE。

竖线指示在几个选项中进行选择。例如：

```
[VANILLA | CHOCOLATE [MINT] | STRAWBERRY]
```

此语法指示您可以输入 VANILLA、CHOCOLATE 或 STRAWBERRY，并且如果输入 CHOCOLATE，还可以输入 MINT。

{ }

当您必须在几个选项中只选择一个选项时，这些选项被括在花括号中，并用竖线分开。例如：

```
{GUAVA | MANGO | PASSIONFRUIT}
```

此语法指示您必须输入 GUAVA、MANGO 或 PASSIONFRUIT，但不能输入多于一个选项。

...

省略号指示您可以输入不确定数目的附加项，如一项后面紧跟省略号。例如：

```
old-column-name
```

```
...
```

此语法指示您可以在第一个列名后输入一系列现有的列名。

4.2 演示 SQL

DB-Access 随附的各种命令文件。

命令文件在命令行中显示时均带有扩展名 .sql，但在 SQL “选择” 菜单中显示时则不带扩展名。

这些命令文件中的关键字以大写字母显示，以使 SQL 语句易于阅读。实际命令文件中的关键字为小写。

重要： 尽管在本附录中命令文件是按字母顺序列出的，但是如果您按该顺序运行用于创建表的命令文件，将导致错误。表的创建顺序由于链接这些表的引用约束而至关重要。

选择 SQL 菜单上的“选择”选项时，将打开“选择”屏幕。屏幕将显示可以访问的命令文件的列表，与下图显示的屏幕类似。这些文件是 stores_demo 数据库附带的。其他 .sql 文件将在本附录的后面部分中进行描述。

图：“选择”屏幕上列出的命令文件

```
选择 >>
```

使用方向键选择命令文件，或输入名称，然后按 Enter 键。

----- stores_demo @dbserver1 ----- 按 CTRL-W 以获得帮助 ---

alt_cat	c_state	d_trig	sel_ojoin1
c_calls	c_stock	d_view	sel_ojoin2
c_cat	c_stores_demo	del_stock	sel_ojoin3
c_custom	c_table	ins_table	sel_ojoin4
c_index	c_trig	opt_disk	sel_order
c_items	c_type	sel_agg	sel_sub
c_manuf	c_view1	sel_all	sel_union
c_orders	c_view2	sel_group	upd_table
c_proc	d_proc	sel_join	

如果未看到演示数据库附带的命令文件，请检查以下各项：

- 运行演示数据库初始化脚本时，是否已将演示 SQL 命令文件复制到当前目录？如果未复制，请重新运行初始化脚本以复制它们。
- 是否已从安装了演示 SQL 命令文件的目录启动 DB-Access？如果不是，请退出 DB-Access，更改为适当的目录，并再次启动 DB-Access。

将这些命令文件与 DB-Access 一起使用来练习使用 SQL 和演示数据库。无论何时要刷新数据库表和 SQL 文件，都可以重新运行演示数据库初始化脚本。

4.2.1 用于关系数据库模型的 SQL 文件

您可以在 stores_demo 演示数据库上运行样本 SQL 命令文件。

alt_cat.sql 命令文件

以下命令文件改变 catalog 表。它删除 catalog 表上的现有约束 aa，并添加一个指定级联删除的新约束 ab。您可以使用此命令文件，然后再使用 del_stock.sql 命令文件在包含记录的数据库上练习级联删除。

```
ALTER TABLE catalog DROP CONSTRAINT aa;

ALTER TABLE catalog ADD CONSTRAINT

    (FOREIGN KEY (stock_num, manu_code) REFERENCES stock

      ON DELETE CASCADE CONSTRAINT ab);
```

c_calls.sql 命令文件

以下命令文件创建 cust_calls 表：

```
CREATE TABLE cust_calls

(

    customer_num          INTEGER,

    call_dtime            DATETIME YEAR TO MINUTE,

    user_id               CHAR(18) DEFAULT USER,

    call_code             CHAR(1),

    call_descr            CHAR(240),

    res_dtime             DATETIME YEAR TO MINUTE,

    res_descr             CHAR(240),

    PRIMARY KEY (customer_num, call_dtime),

    FOREIGN KEY (customer_num) REFERENCES customer (customer_num),

    FOREIGN KEY (call_code) REFERENCES call_type (call_code)

);
```

c_cat.sql 命令文件

以下命令文件创建 catalog 表。它包含约束 aa，该约束使您可以通过在包含日志记录的数据库上运行 alt_cat.sql 和 del_stock.sql 命令文件中的 SQL 语句来练习级联删除。

```
CREATE TABLE catalog

(

    catalog_num          SERIAL(10001),
```



```
stock_num      SMALLINT NOT NULL,  
manu_code      CHAR(3) NOT NULL,  
cat_descr      TEXT,  
cat_picture    BYTE,  
cat_advert     VARCHAR(255, 65),  
PRIMARY KEY (catalog_num),  
FOREIGN KEY (stock_num, manu_code) REFERENCES stock  
CONSTRAINT aa  
);
```

c_custom.sql 命令文件

以下命令文件创建 customer 表：

```
CREATE TABLE customer  
(  
    customer_num      SERIAL(101),  
    fname             CHAR(15),  
    lname             CHAR(15),  
    company            CHAR(20),  
    address1           CHAR(20),  
    address2           CHAR(20),  
    city               CHAR(15),  
    state              CHAR(2),  
    zipcode            CHAR(5),  
    phone              CHAR(18),  
    PRIMARY KEY (customer_num)  
);
```

c_index.sql 命令文件

以下命令文件在 customer 表的 zipcode 列上创建索引：

```
CREATE INDEX zip_ix ON customer (zipcode);
```

c_items.sql 命令文件

以下命令文件创建 items 表：

```
CREATE TABLE items
(
    item_num          SMALLINT,
    order_num         INTEGER,
    stock_num         SMALLINT NOT NULL,
    manu_code         CHAR(3) NOT NULL,
    quantity          SMALLINT CHECK (quantity >= 1),
    total_price       MONEY(8),
    PRIMARY KEY (item_num, order_num),
    FOREIGN KEY (order_num) REFERENCES orders (order_num),
    FOREIGN KEY (stock_num, manu_code) REFERENCES stock
        (stock_num, manu_code)
);
```

c_manuf.sql 命令文件

以下命令文件创建 manufact 表：

```
CREATE TABLE manufact
(
    manu_code         CHAR(3),
    manu_name         CHAR(15),
    lead_time         INTERVAL DAY(3) TO DAY,
    PRIMARY KEY (manu_code)
);
```

c_orders.sql 文件

以下命令文件创建 orders 表：

```
CREATE TABLE orders
(
    order_num         SERIAL(1001),
    order_date        DATE,
```

```
customer_num      INTEGER NOT NULL,
ship_instruct     CHAR(40),
backlog           CHAR(1),
po_num            CHAR(10),
ship_date         DATE,
ship_weight       DECIMAL(8,2),
ship_charge       MONEY(6),
paid_date         DATE,
PRIMARY KEY (order_num),
FOREIGN KEY (customer_num) REFERENCES customer (customer_num)
);
```

c_proc.sql 命令文件

以下命令文件创建 SPL 例程。它读取客户的全名和地址，并将姓氏作为其唯一的自变量。

此例程显示 CREATE PROCEDURE 的旧用法。

为了符合 GBase 8s 首选的 SQL 标准，如果要从例程中返回值，请定义函数。

```
CREATE PROCEDURE read_address (lastname CHAR(15))
    RETURNING CHAR(15), CHAR(15), CHAR(20), CHAR(15), CHAR(2),
CHAR(5);

    DEFINE p_fname, p_city CHAR(15);
    DEFINE p_add CHAR(20);
    DEFINE p_state CHAR(2);
    DEFINE p_zip CHAR(5);

    SELECT fname, address1, city, state, zipcode
        INTO p_fname, p_add, p_city, p_state, p_zip
    FROM customer
        WHERE lname = lastname;

    RETURN p_fname, lastname, p_add, p_city, p_state, p_zip;
```

```
END PROCEDURE;
```

c_state 命令文件

以下命令文件创建 state 表：

```
CREATE TABLE state
(
    code          CHAR(2),
    sname         CHAR(15),
    PRIMARY KEY (code)
);
```

c_stock.sql 命令文件

以下命令文件创建 stock 表：

```
CREATE TABLE stock
(
    stock_num      SMALLINT,
    manu_code      CHAR(3),
    description     CHAR(15),
    unit_price      MONEY(6),
    unit           CHAR(4),
    unit_descr      CHAR(15),
    PRIMARY KEY (stock_num, manu_code),
    FOREIGN KEY (manu_code) REFERENCES manufact
);
```

c_stores.sql 命令文件

以下命令文件创建 stores_demo 数据库：

```
CREATE DATABASE stores_demo;
```

c_table.sql 命令文件

以下命令文件创建名为 restock 的数据库，然后在该数据库中创建名为 sports 的定制表：

```
CREATE DATABASE restock;
```

```
CREATE TABLE sports
(
    catalog_no      SERIAL UNIQUE,
    stock_no        SMALLINT,
    mfg_code        CHAR(5),
    mfg_name        CHAR(20),
    phone           CHAR(18),
    descript        VARCHAR(255)
);
```

c_trig.sql 命令文件

以下命令文件创建名为 log_record 的表，然后创建名为 upqty_i 的用于更新表的触发器：

```
CREATE TABLE log_record
(
    item_num        SMALLINT,
    ord_num         INTEGER,
    username        CHARACTER(8),
    update_time     DATETIME YEAR TO MINUTE,
    old_qty         SMALLINT,
    new_qty         SMALLINT);

CREATE TRIGGER upqty_i
UPDATE OF quantity ON items
REFERENCING OLD AS pre_upd
              NEW AS post_upd
FOR EACH ROW(INSERT INTO log_record
              VALUES (pre_upd.item_num, pre_upd.order_num, USER, CURRENT,
                      pre_upd.quantity, post_upd.quantity));
```

c_type.sql 命令文件

以下命令文件创建 call_type 表：

```
CREATE TABLE call_type
```

```
(  
    call_code          CHAR(1),  
    code_descr         CHAR(30),  
    PRIMARY KEY (call_code)  
);
```

c_view1.sql 命令文件

以下命令文件在单个表上创建名为 `custview` 的视图，并将视图上的特权授予 `public`。它包括 `WITH CHECK OPTION` 关键字，用于验证通过视图对基础表所做的任何更改是否未破坏视图的定义。

```
CREATE VIEW custview (firstname, lastname, company, city) AS  
  
    SELECT fname, lname, company, city  
  
        FROM customer  
  
            WHERE city = 'Redwood City'  
  
                WITH CHECK OPTION;  
  
GRANT DELETE, INSERT, SELECT, UPDATE  
  
    ON custview  
  
    TO public;
```

c_view2.sql 命令文件

以下命令文件在 `orders` 和 `items` 表上创建视图：

```
CREATE VIEW someorders (custnum,ocustnum,newprice) AS  
  
    SELECT orders.order_num, items.order_num,  
  
            items.total_price*1.5  
  
        FROM orders, items  
  
            WHERE orders.order_num = items.order_num  
  
                AND items.total_price > 100.00;
```

d_proc.sql 命令文件

以下命令文件删除 `c_proc.sql` 命令文件创建的 SPL 例程：

```
DROP PROCEDURE read_address;
```

d_trig.sql 命令文件

以下命令文件删除 c_trig.sql 命令文件所创建的触发器：

```
DROP TRIGGER upqty_i;
```

d_view.sql 命令文件

以下命令文件删除 c_view1.sql 命令文件所创建的名称为 custview 的视图：

```
DROP VIEW custview;
```

del_stock.sql 命令文件

以下命令文件从库存编号为 102 的 stock 表中删除行。此删除将级联至 catalog 表（尽管相关制造商代码将保留在 manufact 表中）。可在 alt_cat.sql 命令文件之后使用 del_stock.sql 命令文件，用于在包含记录的数据库上练习级联删除。

```
DELETE FROM stock WHERE stock_num = 102;
```

运行 alt_cat.sql 和 del_stock.sql 命令文件中的 SQL 语句之后，对 catalog 表发出以下查询以验证是否已删除行：

```
SELECT * FROM catalog WHERE stock_num = 102;
```

stores_demo 数据库已更改。您可能要重新运行 dbaccessdemo 脚本来重新构建原始数据库。

ins_table.sql 命令文件

以下命令文件在 c_table.sql 命令文件创建的 sports 表中插入一行：

```
INSERT INTO sports  
  
VALUES (0,18,'PARKR', 'Parker Products', '503-555-1212',  
  
'Heavy-weight cotton canvas gi, designed for aikido or  
  
judo but suitable for karate. Quilted top with side ties,  
  
drawstring waist on pants. White with white belt.  
  
Pre-washed for minimum shrinkage. Sizes 3-6.');
```

opt_disk.sql 命令文件

以下命令文件提供光盘子系统上的 SELECT 语句的示例。它包括支持光学存储器的只读 family() 和 volume() 运算符。（这仅在使用 Optical Subsystem 时可用。）

查询生成包含自行车头盔图形的卷列表。将为每个包含自行车头盔图形的数据行生成一行输出（family 和 volume）。family() 运算符返回存储光学 BLOB 列的光学系列产品的名称，而 volume() 返回存储光学 BLOB 列的卷的编号。这些函数仅对存储在光学介质上的

数据有效。

```
SELECT family(cat_picture), volume(cat_picture)
      FROM catalog
      WHERE stock_num = 110;
```

sel_agg.sql 命令文件

以下命令文件中的 SELECT 语句使用聚集函数查询表数据。它在单个语句中组合了聚集函数 MAX 和 MIN。

```
SELECT MAX (ship_charge), MIN (ship_charge)
      FROM orders;
```

sel_all.sql 命令文件

以下命令文件示例包含可在交互式 SQL 的 GBase 8s 实现中使用的所有七个 SELECT 语句子句。此 SELECT 语句将 orders 表和 items 表连接在一起。它还使用显示标号、表别名和用作列指示符的整数；对数据进行分组和排序；并将结果放入临时表中。

```
SELECT o.order_num, SUM (i.total_price) price,
      paid_date - order_date span
      FROM orders o, items i
      WHERE o.order_date > '01/01/90'
            AND o.customer_num > 110
            AND o.order_num = i.order_num
      GROUP BY 1, 3
      HAVING COUNT (*) < 5
      ORDER BY 3
      INTO TEMP temptab1;
```

sel_group.sql 命令文件

以下命令文件示例包括 GROUP BY 和 HAVING 子句。HAVING 子句通常在组形成之后通过将一个或多个限定条件应用到组来补充 GROUP BY 子句，这与 WHERE 子句限定单独行的方法类似。（使用 HAVING 子句的一个优点在于：您可以在搜索条件中包括聚集，但是不能在 WHERE 子句的搜索条件中包括聚集。）

每个 HAVING 子句将组的一个列或聚集表达式与组的另一个聚集表达式或常量比较。您可以使用 HAVING 子句将条件置于组列表中的列值和聚集值上。


```
SELECT order_num, COUNT(*) number, AVG (total_price) average  
  
FROM items  
  
GROUP BY order_num  
  
HAVING COUNT(*) > 2;
```

sel_join.sql 命令文件

以下命令文件示例对 customer 表和 cust_calls 表使用简单连接。此查询仅返回显示已调用客户服务的客户的那些行。

```
SELECT c.customer_num, c.lname, c.company,  
  
       c.phone, u.call_dtime, u.call_descr  
  
FROM customer c, cust_calls u  
  
WHERE c.customer_num = u.customer_num;
```

sel_ojoin1.sql 命令文件

以下命令文件示例对两个表使用简单外连接。在 cust_calls 表前使用关键字 OUTER 可以使它成为从属表。外连接导致查询返回有关所有客户的信息，即使他们未调用客户服务。将检索 customer 控制表中的所有行，并且将空值指定给 cust_calls 从属表中对应的行。

```
SELECT c.customer_num, c.lname, c.company,  
  
       c.phone, u.call_dtime, u.call_descr  
  
FROM customer c, OUTER cust_calls u  
  
WHERE c.customer_num = u.customer_num;
```

sel_ojoin2.sql 命令文件

以下命令文件示例创建一个外连接，它是至第三个表的简单连接的结果。外连接的第二种类型称作嵌套的简单连接。

此查询首先在 orders 表和 items 表上执行简单连接，并检索有关包含 KAR 或 SHM 的 manu_code 的项的所有订单的信息。然后，它执行外连接，将此信息与 customer 控制表中的数据组合起来。可选的 ORDER BY 子句对数据进行重组。

```
SELECT c.customer_num, c.lname, o.order_num,  
  
       i.stock_num, i.manu_code, i.quantity  
  
FROM customer c, OUTER (orders o, items i)  
  
WHERE c.customer_num = o.customer_num  
  
      AND o.order_num = i.order_num
```

```
AND manu_code IN ('KAR', 'SHM')

ORDER BY lname;
```

sel_ojoin3.sql 命令文件

以下 SELECT 语句示例是外连接的第三种类型，称为嵌套的外连接。它通过创建外连接来查询表数据，该外连接是至第三个表的外连接的结果。

此查询首先在 orders 表和 items 表上执行外连接，并检索有关包含 KAR 或 SHM 的 manu_code 的项的所有订单的信息。然后，它执行外连接，将此信息与 customer 控制表中的数据组合起来。此查询保留上一示例所除去的订单编号，并为不包含具有任一制造商代码的项的订单返回行。可选的 ORDER BY 子句对数据进行重组。

```
SELECT c.customer_num, lname, o.order_num,
       stock_num, manu_code, quantity
FROM customer c, OUTER (orders o, OUTER items i)
WHERE c.customer_num = o.customer_num
      AND o.order_num = i.order_num
      AND manu_code IN ('KAR', 'SHM')
ORDER BY lname;
```

sel_ojoin4.sql 命令文件

以下示例使用第四种类型的外连接查询表数据。此查询显示外连接，该外连接是每两个表与第三个表的外连接的结果。在此类型的外连接中，连接关系可能仅存在于控制表和从属表之间。

此查询单独地将从属表 orders 和 cust_calls 连接到 customer 控制表，但不会将两个从属表连接起来。（INTO TEMP 子句选择结果放入临时表。）

```
SELECT c.customer_num, lname, o.order_num,
       order_date, call_dtime
FROM customer c, OUTER orders o, OUTER cust_calls x
WHERE c.customer_num = o.customer_num
      AND c.customer_num = x.customer_num
      INTO temp service;
```

sel_order.sql 命令文件

以下示例使用 ORDER BY 和 WHERE 子句进行查询。在此 SELECT 语句中，通过比较

'bicycle%' (LIKE 条件或 MATCHES 条件的 'bicycle*') 来指定后跟任意顺序的零或更多字符的字母 bicycle。它通过添加另一个将 PRC 的 manu_code 排除在外的比较条件来进一步缩小搜索范围。

```
SELECT * FROM stock
    WHERE description LIKE 'bicycle%'
        AND manu_code NOT LIKE 'PRC'
    ORDER BY description, manu_code;
```

sel_sub.sql 命令文件

以下示例使用子查询进行查询。此自连接使用相关子查询进行检索，并列出了已订购的价格最高的 10 项。

```
SELECT order_num, total_price
    FROM items a
    WHERE 10 >
        (SELECT COUNT (*)
            FROM items b
            WHERE b.total_price < a.total_price)
    ORDER BY total_price;
```

sel_union.sql 命令文件

以下示例使用 UNION 子句来查询两个表中的数据。复合查询对 stock 表和 items 表中的 stock_num 列和 manu_code 列执行联合。语句选择单价低于 \$25.00 或订购数量大于 3 的项，并列出了它们的 stock_num 和 manu_code。

```
SELECT DISTINCT stock_num, manu_code
    FROM stock
    WHERE unit_price < 25.00

UNION

SELECT stock_num, manu_code
    FROM items
```

```
WHERE quantity > 3;
```

upd_table.sql 命令文件

以下示例更新 c_table.sql 命令文件创建的 sports 表：

```
UPDATE sports  
  
SET phone = '808-555-1212'  
  
WHERE mfg_code = 'PARKR';
```

4.2.2 用于维数据库模型的 SQL 文件

通过运行用于创建 sales_demo 数据库的脚本，可以为数据仓储应用程序实施维数据库。

sales_demo 数据库基于 stores_demo 模式和数据。

要创建 sales_demo 数据库：

1. 使用以下命令创建 stores_demo 数据库：

```
dbaccessdemo -log
```

2. 确保 createdw.sql 和 loaddw.sql 文件与 loaddw.sql 所使用的扩展名为 .unl 的文件位于同一目录中。
3. 运行 createdw.sql 文件。
4. 运行 loaddw.sql 文件。

createdw.sql 文件

此文件创建包含记录的新 sales_demo 数据库，然后在该数据库内创建表。它包含以下语句：

```
create database sales_demo with log;  
  
create table product (  
    product_code integer,  
    product_name char(31),  
    vendor_code char(3),  
    vendor_name char(15),  
    product_line_code smallint,  
    product_line_name char(15));
```

```
create table customer (  
    customer_code integer,  
    customer_name char(31),  
    company_name char(20));
```

```
create table sales (  
    customer_code integer,  
    district_code smallint,  
    time_code integer,  
    product_code integer,  
    units_sold smallint,  
    revenue money (8,2),  
    cost money (8,2),  
    net_profit money(8,2));
```

```
create table time  
(  
    time_code int,  
    order_date date,  
    month_code smallint,  
    month_name char(10),  
    quarter_code smallint,  
    quarter_name char(10),  
    year integer  
);
```

```
create table geography (
```

```
district_code serial,  
district_name char(15),  
state_code char(2),  
state_name char(18),  
region smallint);
```

loadw.sql 文件

此文件包含从两个源装入数据所必需的命令：

- 演示目录中带有扩展名 .unl 的文件
- 从 stores_demo 数据库中选择的数据

loadw.sql 中的这些 SQL 语句完成以下操作：

```
connect to "stores_demo ";  
  
load from "add_orders.unl"  
    insert into stores_demo :orders;  
  
load from 'add_items.unl'  
    insert into stores_demo :items;  
  
  
connect to "sales_demo";  
  
load from 'costs.unl'  
    insert into cost;  
  
load from 'time.unl'  
    insert into time;  
  
  
insert into geography(district_name, state_code, state_name)  
    select distinct c.city, s.code, s.sname  
from stores_demo :customer c, stores_demo :state s  
    where c.state = s.code;  
  
update geography      -- converts state_code values to region values  
    set region = 1
```

```
where state_code = "CA";

update geography

set region = 2

where state_code <> "CA";

insert into customer (customer_code, customer_name, company_name)

select c.customer_num, trim(c.fname) || " " || c.lname, c.company

from stores_demo :customer c;

insert into product (product_code, product_name, vendor_code,

vendor_name, product_line_code, product_line_name)

select a.catalog_num,

trim(m.manu_name) || " " || s.description,

m.manu_code, m.manu_name, s.stock_num, s.description

from stores_demo :catalog a, stores_demo :manufact m,

stores_demo :stock s

where a.stock_num = s.stock_num and

a.manu_code = s.manu_code and

s.manu_code = m.manu_code;

insert into sales (customer_code, district_code,

time_code, product_code,

units_sold, revenue, cost, net_profit)

select c.customer_num, g.district_code, t.time_code, p.product_code,

SUM(i.quantity), SUM(i.total_price),

SUM(i.quantity * x.cost),

SUM(i.total_price) - SUM(i.quantity * x.cost)

from stores_demo :customer c, geography g, time t,

product p,

stores_demo :items i, stores_demo :orders o, cost x
```

```
where c.customer_num = o.customer_num and
      o.order_num = i.order_num and
      p.product_line_code = i.stock_num and
      p.vendor_code = i.manu_code and
      t.order_date = o.order_date and
      p.product_code = x.product_code and
      c.city = g.district_name
GROUP BY 1,2,3,4;

connect to "stores_demo ";
load from 'add_orders.unl'
      insert into stores_demo :orders;
load from 'add_items.unl'
      insert into stores_demo :items;

connect to "sales_demo";
load from 'costs.unl'
      insert into cost;
load from 'time.unl'
      insert into time;

insert into geography(district_name, state_code, state_name)
      select distinct c.city, s.code, s.sname
from stores_demo :customer c, stores_demo :state s
      where c.state = s.code;
update geography      -- converts state_code values to region values
      set region = 1
      where state_code = "CA";
update geography
```



```
set region = 2

where state_code <> "CA";

insert into customer (customer_code, customer_name, company_name)

select c.customer_num, trim(c.fname) || " " || c.lname, c.company

from stores_demo :customer c;

insert into product (product_code, product_name, vendor_code,

vendor_name, product_line_code, product_line_name)

select a.catalog_num,

trim(m.manu_name) || " " || s.description,

m.manu_code, m.manu_name, s.stock_num, s.description

from stores_demo :catalog a, stores_demo :manufact m,

stores_demo :stock s

where a.stock_num = s.stock_num and

a.manu_code = s.manu_code and

s.manu_code = m.manu_code;

insert into sales (customer_code, district_code,

time_code, product_code,

units_sold, revenue, cost, net_profit)

select c.customer_num, g.district_code, t.time_code, p.product_code,

SUM(i.quantity), SUM(i.total_price),

SUM(i.quantity * x.cost),

SUM(i.total_price) - SUM(i.quantity * x.cost)

from stores_demo :customer c, geography g, time t, product p,

stores_demo :items i, stores_demo :orders o, cost x

where c.customer_num = o.customer_num and

o.order_num = i.order_num and
```

```
p.product_line_code = i.stock_num and  
p.vendor_code = i.manu_code and  
t.order_date = o.order_date and  
p.product_code = x.product_code and  
c.city = g.district_name  
GROUP BY 1,2,3,4;
```

4.2.3 用于对象关系数据库模型的用户定义的例程

您可以在 `superstores_demo` 数据库上运行用户定义的例程样本。

`superstores_demo` 数据库未取代 `stores_demo` 数据库。两个数据库均可用。

`superstores_demo` 数据库模式与 `stores_demo` 较早的版本不兼容。在许多情况下，由于表的差异，您不能针对 `superstores_demo` 的表使用为 `stores_demo` 开发的测试查询。

没有专门与 `superstores_demo` 关联的 SQL 命令文件。但是，有一些用户定义的例程可以在 SQL 编辑器或系统编辑器中运行。

`superstores_demo` 数据库包括以下功能的示例：

- 集合类型：SET 和 LIST
- 已命名行类型：location_t、loc_us_t 和 loc_non_us_t
- 未命名行类型
- 类型和表继承
- 内置数据类型：BOOLEAN、SERIAL8 和 INT8
- 单值数据类型：percent
- 智能大对象：BLOB 和 CLOB

`superstores_demo` 数据库具有行类型和表来支持下表继承层次结构：

- customer/retail_customer
- customer/whlsale_customer
- location/location_us
- location/location_non_us

The logo for GBASE, featuring the word "GBASE" in a bold, red, sans-serif font, followed by a registered trademark symbol (®). To the left of the text is a solid red square.

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

