GBASE

SQLServer 到 GBase 8s

迁移技术指南

www.gbase.cn

目 录

1 概述

将数据库从 SQL Server 迁移到 GBase 8s 主要包括三个步骤：数据库架构迁移(Schema/DDL)、数据迁移(Data)和应用迁移(Application)。本文将以这三个步骤为主线，介绍 GBase 8s 不同于 SQL Server 的技术特点，以及在迁移过程中的这两个数据库差异的转换方法与技巧。

与其他软件开发项目一样，数据库的迁移需要谨慎的规划以及良好的方法以确保其成功。这其中，数据库的设计至关重要，特别是关系型数据库的 Schema 设计。可以通过新的数据库技术来替代之前使用的旧方式。在进行 SQL Server 数据库迁移的时候，要按照 GBase 8s 的技术特点，进行一系列的数据库 Schema 及应用程序的调整，有助于今后应用的平稳运行。在进行迁移项目之前，一定要对上述因素有一个完整的把握，这样可以到达事半功倍的效果，同时可以在一定程度上避免不必要的麻烦。

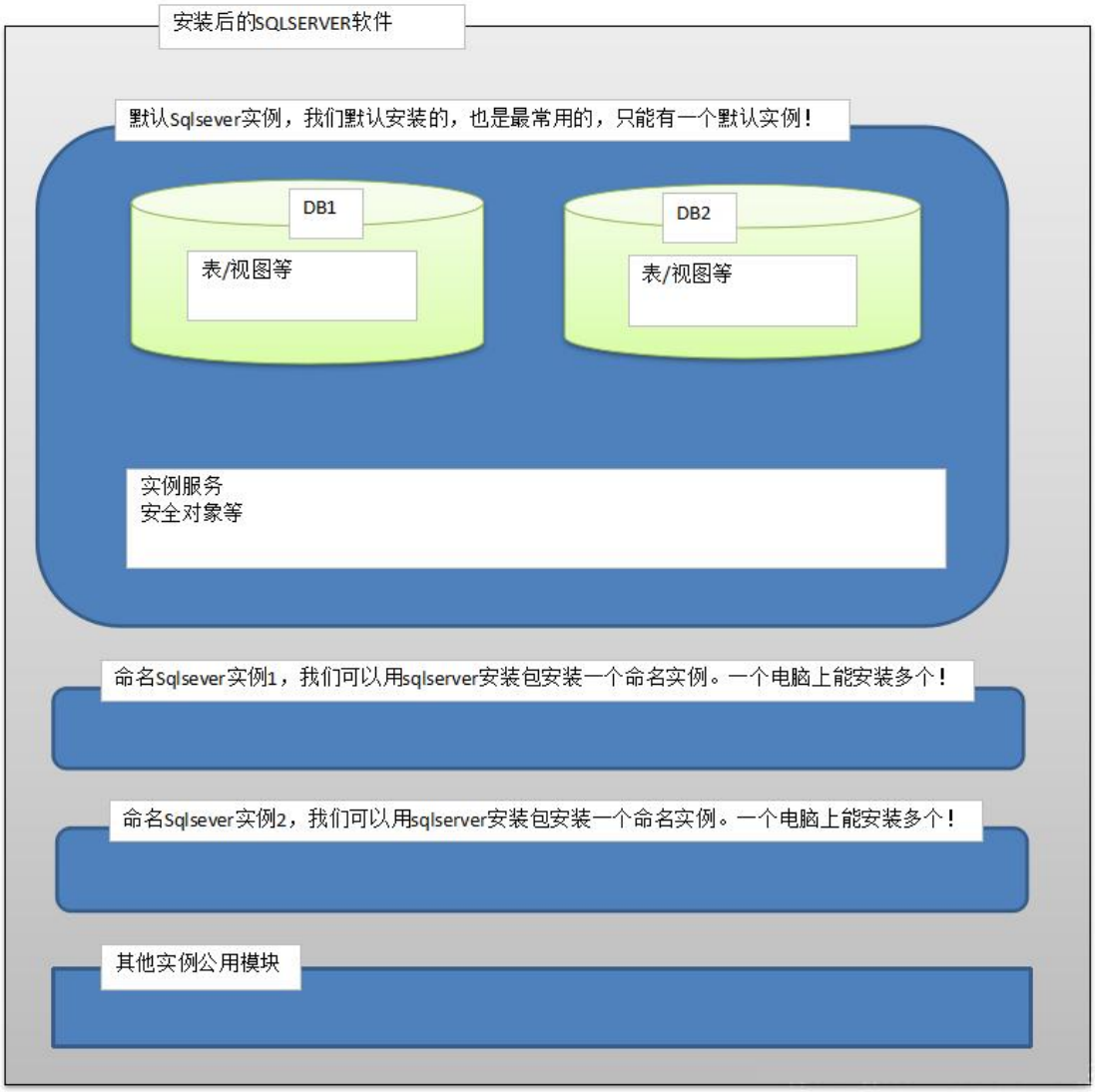
数据库对象迁移的很大一部分工作量是存储过程、函数和触发器的迁移，这部分的迁移迄今为止主要还是靠人工手动将 SQL Server 的语法及业务逻辑翻译成 GBase 8s 的语法及完全不改变业务逻辑本身，并且要尽量适应原应用程序的调用格式，尽量少的修改应用源码，这样既能保证少出错误，也同时能保证应用迁移的工作量尽可能的少。

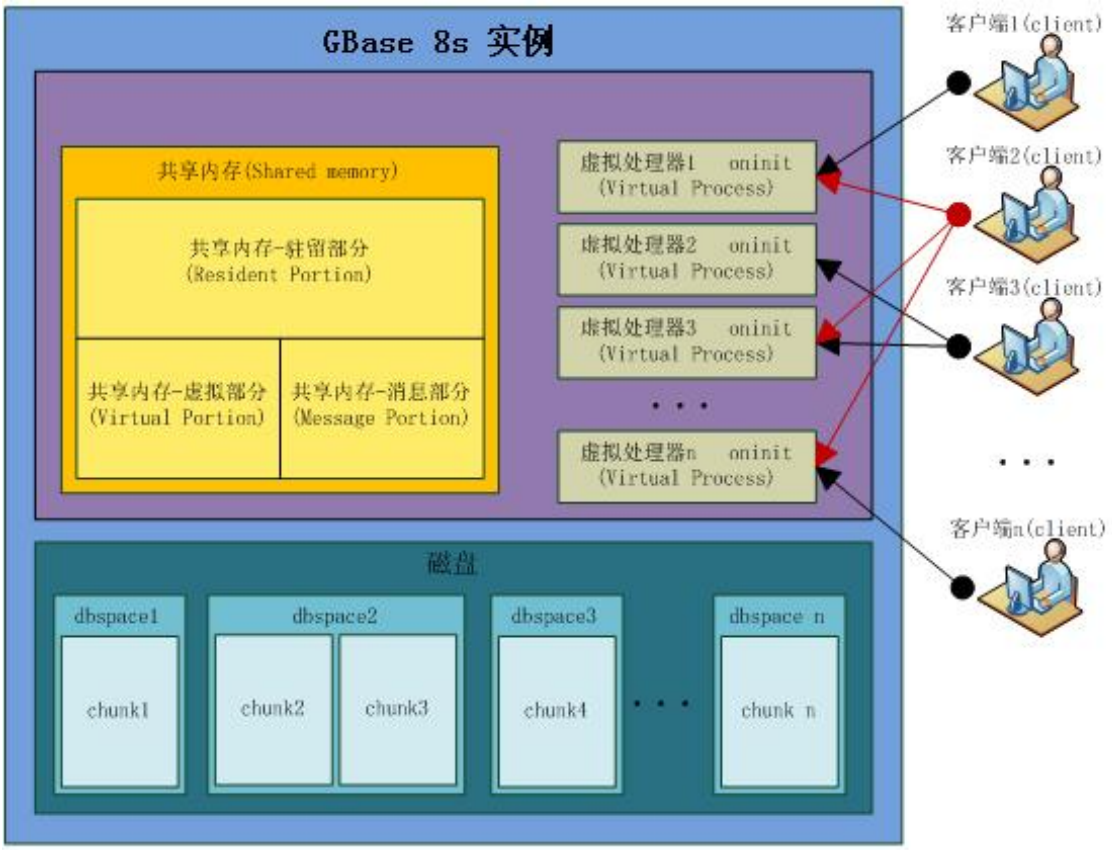
GBase 8s 支持 Oracle 和 GBase 两种模式，本文中代码示范为 GBase 模式下的 SPL 语言，如需使用 Oracle 模式下 PLSQL 语言，可参考《GBase 8s PLSQL 手册》的代码示范。

SQL Server 有多个版本，附录 A 对各版本进行了介绍对比，不同版本语法和功能也各有差异，具体细节可进一步参阅 SQL Server 官方文档，本文以 SQL Server2008 企业版和 GBase 8s V8.8 版本进行对比讲解，希望对正在或者准备进行 SQL Server 到 GBase 8s 迁移项目的工作人员有一定的指导和启发。

2 概念及对比

2.1 实例(Instance)





上面 2 个图分别展示了 SQL Server 的多实例和 GBase 8s 的一个实例的具体内容。

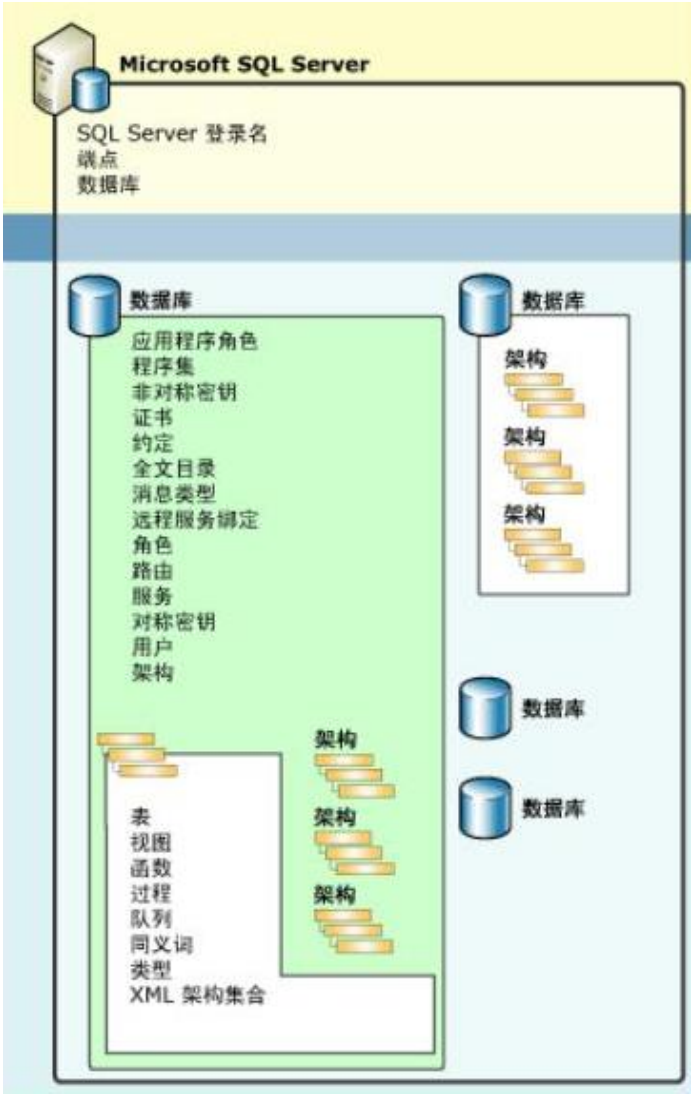
在 GBase 8s 中，实例包括共享内存、进程、存储。

在 SQL Server 中，实例是后台进程和数据库文件的集合。

GBase 8s 和 SQL Server 都支持在一台机器安装和同时运行多个实例。

2.2 数据库(Database)

GBase 8s 和 SQL Server 中的数据库(database)都是一系列包含特定用途对象的集合。从 SQL Server 2000 版本开始支持架构(Schema) ,架构在 SQL Server 中的应用并不普及，如下图：

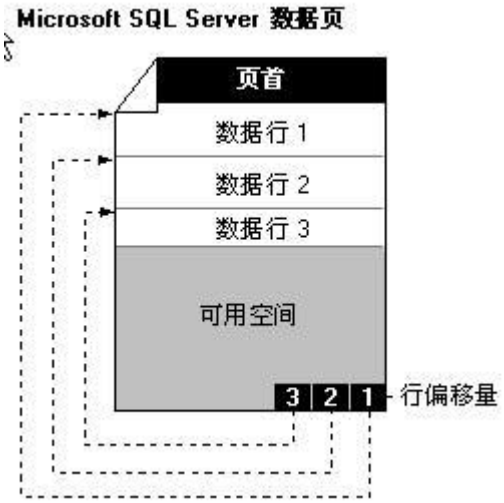


2.3 存储(Storage)

GBase 8s 中存储概念从大到小如下表

序号	GBase 8s	SQL Server
1	数据空间(dbspace)	数据文件(datafile)
2	块(Chunk)	
3	表空间(table space)	
4	区间(extent)	区(extents)
5	页(page)	页(page)

SQL Server 是基于 Windows 平台的产品，有两种数据存储文件，分别是数据文件和日志文件，其中数据文件是以 8K(=8192Byte)的页面作为存储单元。而日志文件是以日志记录作为存储单元。

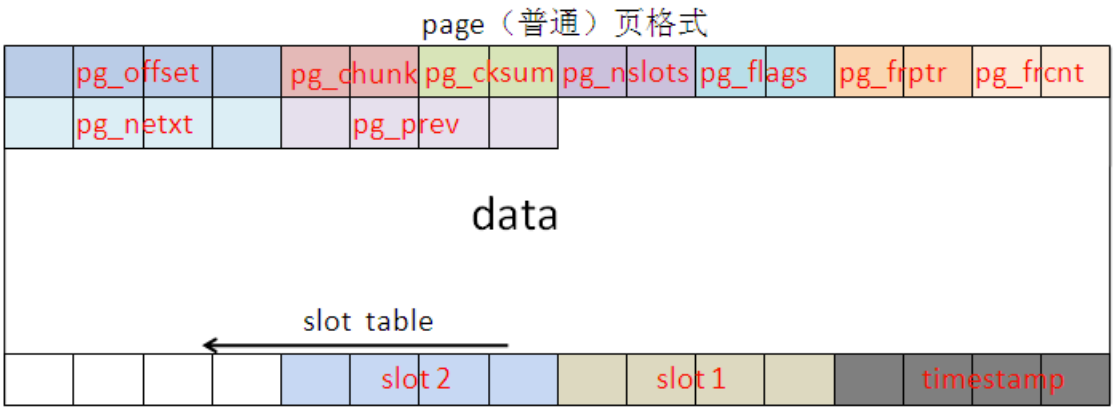


与常规书籍做类比：常规书籍中的所有内容都是写在书页上的。 与书籍类似，SQL Server 所有数据行都写在页面上。 书中的所有页都具有相同的物理大小。 同样，SQL Server 所有数据页大小均相同 - 8 KB。 书中的大多数页都包含数据（书的主要内容），某些页面包含有关内容的元数据（例如目录和索引）。 SQL Server 也是如此：大多数页包含由用户存储的实际数据行；这些称为数据页面和文本/图像页面（在特殊情况下）。 索引页包含有关数据位置的索引引用，最后有一些系统页，它们存储有关数据组织的各种元数据（PFS、GAM、SGAM、IAM、DCM、BCM 页）。 请参阅下表了解页面类型及其说明。

如前所述，在 SQL Server 中，页的大小为 8-KB。 这意味着 SQL Server 数据库中每 MB 有 128 页。 每页的开头是 96 字节的标头，用于存储有关页的系统信息。 此信息包括页码、页类型、页的可用空间以及拥有该页的对象的分配单元 ID。

区是管理空间的基本单位。 一个区是八个物理上连续的页（即 64 KB）。 这意味着 SQL Server 数据库中每兆字节有 16 个区。

SQL Server 关于页和区的功能描述与 GBase 8s 数据库基本相同，不同之处在于：GBase 8s 的页大小是 2KB（Linux）或者 4KB（Windows 或者 AIX）；页头只使用 24 字节，页尾有 4 个字节用于时间戳。一个区默认也是作个物理上的连续的页。



GBase 8s 存储相关限制

GBase 8s 存储相关限制	最大值
一页包含的数据行数	255
一个表或者分片最大的行数	4,277,659,295
一个表或者分片最大的数据页数	16,775,134
一个表或者分片最大占用的空间大小 （除 BLOB,CLOB,BYTE,TEXT 以外）	2K page size = 33,818,670,144 4K page size = 68,174,144,576 8K page size = 136,885,093,440 12K page size = 205,596,042,304 16K page size = 274,306,991,168
一行记录的最大 rowsize	32,767
表最大支持的字段数	32,767
一个复合索引支持最多的列数	16
一个函数索引的字段数	102 (for C UDRs) 341 (for SPL or Java™ UDRs)
一个索引的字节数（对于给定的页大小）	2K page size = 387 4K page size = 796 8K page size = 1615 12K page size = 2435 16K page size = 3254
单个 SQL 语句的最大长度	受限于可用内存
一个 Instance 最大支持的 database 数	21,000,000
一个 Instance 最大支持的 tables 数	477,102,080
一个 Instance 最大支持的活动用户数 active users	32,767
一个 session 最大同时访问的 database 数	Windows: 8 UNIX: 32
最大的页清理进程 page-cleaner	128
一个 dbspace 上最大支持的 partition 数	4K page size 1,048,445 2K page size 1,048,314
一个用户同时可以使用 Lock Table 锁住的表的个数	32
一个 dbspace 的最大空间	128 PB

一个 chunk 的最大空间	4 TB
一个 Instance 最多支持的 chunks 数	32,766
一个 chunk 最大存储的数据页数 pages	2,000,000,000
一个 Instance 最大支持的 dbspace 数	2047
一个 Instance 最大支持的存储空间	128 PB

2.4 锁(Locks)

建表语句中的 LOCK MODE 子句指定了 GBase 8s 的锁模式，行级锁或页级锁。GBase 8s 中默认是页级锁(可通过配置参数 DEF_TABLE_LOCKMODE 或环境变量修改这个默认值)。

GBase 8s 的锁结构存在共享内存中，而不是 dbspace 中。

SQL Server 默认锁级别为行级锁，锁的粒度如下表

资源	说明
RID	用于锁定堆中的单个行的行标识符
KEY	索引中用于保护可序列化事务中的键范围的行锁
PAGE	数据库中的 8 KB 页，例如数据页或索引页
EXTENT	一组连续的八页，例如数据页或索引页
HoBT	堆或 B 树。 用于保护没有聚集索引的表中的 B 树(索引)或堆数据页的锁
TABLE	包括所有数据和索引的整个表
FILE	数据库文件
APPLICATION	应用程序专用的资源
METADATA	元数据锁
ALLOCATION_UNIT	分配单元
DATABASE	整个数据库

2.5 隔离级别(Isolation)

GBase 8s 为应用提供的隔离级别及对应关系请参考下表。

GBASE 8S	SQL SERVER	ANSI
DIRTY READ	READ UNCOMMITTED	READ UNCOMMITTED
COMMITTED READ 默认	**READ COMMITTED 默认**	READ COMMITTED
REPEATABLE READ	REPEATABLE READ	REPEATABLE READ
REPEATABLE READ	SERIALIZABLE	SERIALIZABLE
CURSOR STABILITY	SNAPSHOT	
COMMITTED READ LAST COMMITTED	READ COMMITTED SNAPSHOT	

GBase 8s 提供 4 种主要隔离级别，它们为处理数据并发性问题提供一些机制：

脏读隔离（读未提交）

提交读隔离（读提交）

游标稳定性隔离

可重复读隔离（可重复读和可系列化）

DIRTY READ 脏读隔离级别（读未提交）

脏读隔离不需要在读取的行上使用锁，因此不检查所需的行上是否有锁。使用该隔离级别的会话可能会得到脏数据，即该数据被另一个会话更新但尚未提交。

对于非日志数据库，这是唯一可以使用的隔离级别。

以上列出的所有 3 种现象都可以在这个隔离级别上出现。

COMMITTED READ 提交读隔离级别（读提交）

提交读隔离仅允许会话读取已提交的行。它不在正在读取的行上放置锁，但检查该行是否放置了锁。因此这个隔离级别不会发生脏读。

CURSOR STABILITY 游标稳定性隔离级别

游标稳定性隔离在读取的行上放置一个共享锁，并且在锁定下一个行时释放该锁。

这个隔离级别能够阻止脏读和非重复读现象。

REPEATABLE READ 可重复读隔离级别

在 GBase 8s 中实现的可重复读隔离级别相当于 ANSI SQL 92 中的可重复读和可系列化。

这个隔离级别除了能够阻止脏读和非重复读现象之外，它还能阻止伪读现象。

如果数据库服务器需要为使用可重复读隔离的会话对表执行序列读，它将在该表上放置一个共享锁。不过，如果使用了索引扫描的话，共享锁仅能锁定与受影响的行相关的索引键。

语句 SET ISOLATION COMMITTED READ 加上 LAST COMMITTED 关键字选项会减少锁冲突的风险。此语法告诉服务器返回最近提交的行版本，即使另一个并发会话持有一个独占锁。这种隔离级别更近似于 SQL Server 的 READ COMMITTED。

于此相关的数据库参数 USELASTCOMMITTED 指定一种能使 COMMITTED READ 隔离级别的 LAST COMMITTED 功能隐性生效的隔离级别。

2.6 标识符(Identifiers)

数据库对象的名字由标识符来表示，如表名、字段名、索引名、视图名、存储过程名称等等。

GBase 8s 最大表名长度 128 个字节，但不支持“#”，“$”也不可以是表名的第一个字符，不符合 GBase 8s 命名规则的标识符，需要在移植时用正确的名称替换。

SQL Server 中标识符的首字符必须统一码（Unicode）2.0 标准中定义的字母，包括拉丁字母 a~z 和 A~Z，以及来自其他语言的字符，或者是下划线“_”、符号“@”或者数字符号“#”。要注意的是在 SQL Server 中，某些处于标识符开始位置的符号具有特殊意义。以“@”符号开始的标识符表示局部变量或参数；以一个数字符号“#”开始的标识符表示临时表或过程，如表“#gzb”表示一张临时表；以双数字符号“##”开始的标识符表示全局临时对象，如表“##gzb”则是全局临时表。标识符后续字符可以是以下 3 种：统一码（Unicode）2.0 标准中所定义的字母、来自拉丁字母或其他国家/地区脚本的十进制数字、“@”符号、美元符号“$”、数字符号“#”或下划线“_”。注意标识符不能是 Transact-SQL 的保留字也不允许嵌入空格或其它特殊字符。

SQL Server 对于保留字为字段表名称或者表名称时，可使用中括号包围使用（[ ]），而 GBase 8s 不需要。

2.7 关键字(Keywords)

GBase 8s 的关键字(保留字)有与 SQL Server 有着大量的不同，具体包括的保留字请参考附录 B。

2.8 大小写(Case sensitivity)

GBase 8s 默认大小写不敏感，但在引号中的内容则是大小写敏感的。

例如： GBase 8s 中 gbase == GBASE == GBase=="gbase" != "GBASE"。

所以在 SQL Sserver 导出建表语句中，带双引号的表名和字段名，都要加以处理才可以被 GBase 8s 使用。

SQL Server 默认也是大小写不敏感。

2.9 用户(User)

SQL Server 和 GBase 8s 均可以使用操作系统用户来进行同等的权限管理，可以使用 Grant、Revoke 命令对用户进行权限的授予与回收。不同数据库主机间的数据库访问，如同义词、跨实例视图等，需要在两台服务器上配置相应用户的信任关系，具体配置方法与使用的平台相关。

gbasedbt 用户是 GBase 8s 的超级管理员。

GBase 8s 同时允许用户通过内部授权服务（比如 Kerberos 或者 Microsoft Active Directory）来连接数据库而无需系统用户。这种方法使系统可以通过映射系统中已存在的除 gbasedbt 和 root 外的其他用户，来访问数据库。这种机制将会减少为了使用数据库而创建的大量系统用户。减少 DBA 的工作量，增加系统的安全系数。

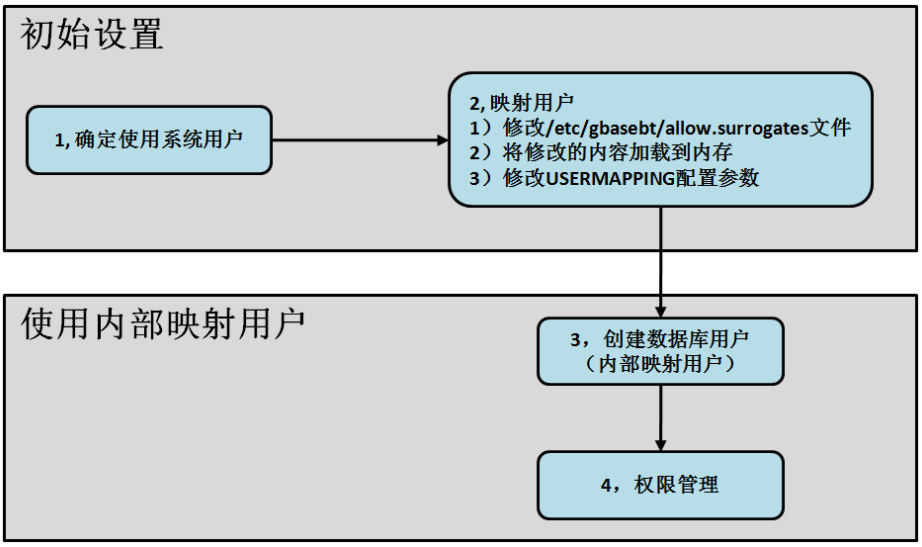


图 2-9 非系统用户访问数据库设置方法示意图

非系统用户（内部映射用户）访问数据库设置示例：

```
# 编辑 /etc/gbasedbt/allowed.surrogates 文件:
USER:daemon
# 加载到内存，修改配置文件
onmode -cache surrogates
onmode -wf USERMAPPING='ADMIN'
# 设置默认用户，指定默认目录（默认目录的权限是 777），用户密码需满足不少于 8 位含大小写字母及数字
dbaccess sysuser -
```

```
CREATE DEFAULT USER WITH PROPERTIES USER daemon HOME "/home/gbase/users" ;
CREATE USER sue WITH PASSWORD 'GBase123';
CREATE USER bale WITH PASSWORD 'GBase123';
# 授权管理
dbaccess testdb -
GRANT CONNECT TO sue;
GRANT CONNECT TO bale;
```

3 数据库对象迁移

3.1 数据类型

3.1.1 数据类型对照表

数值类型：

GBase 8s		SQL Server
SMALLINT		TINYINT SMALLINT
size：2	-32,767 到 32,767	
INT		INT
size：4	-2,147,483,647 到 2,147,483,647	
INT8 或 BIGINT		BIGINT
size：8 或 10	-9,223,372,036,854,775,807 到 9,223,372,036,854,775,807	
DECIMAL(min(p,32),min(s,26)) 或 NUMERIC(min(p,32),min(s,26))		DECIMAL(p,s) NUMERIC(p,s)
size：ceil((p+4)/2	在小数部分中最多 32 位有效数字，或在小数点的左边最多 32 位有效数字。	
MONEY(p,s)		SMALLMONEY MONEY
size：ceil((p+4)/2	在小数部分中最多 32 位有效数字，或在小数点的左边最多 32 位有效数字。	
SMALLFLOAT / REAL		REAL
size：4	6 位精度浮点数	
FLOAT		FLOAT(n)
size：8	14 位精度浮点数	
SERIAL		INT IDENTITY
size：4	-2,147,483,647 到 2,147,483,647	
SERIAL8 或 BIGSERIAL		BIGINT IDENTITY
size：8 或 10	-9,223,372,036,854,775,807 到 9,223,372,036,854,775,807	

说明：

1， 整型数值类型中，GBase 8s 相对于 SQL Server 保留了一位：最小值差 1。例：SMALLINT 的最小值，SQL Server 是-32,768，而 GBase 8s 则为-32,767。

2， 对于自增序列字段，如果自增值不是 1，则需要使用序列来实现。

字符类型：

GBase 8s		SQL Server
CHAR(n)		CHAR(n)
size：n	定长的字符串。最大 32767 字节。	
VARCHAR(n)		VARCHAR(N)
size：n + 3	变长的字符串。最大 32765 字节。	
TEXT 或 JSON		VARCHAR(max) TEXT NVARCHAR(max) NTEXT
size：< 2G	变长的字符串。最多 2GB 字节数据。	
NCHAR		NCHAR(n)
size：n	定长的字符串。最大 32767 字节。	
NVARCHAR(n)		NVARCHAR(n)
size：n + 3	变长的字符串。最大 32765 字节。	

二进制对象：

GBase 8s		SQL Server
BYTE 或 BSON		BINARY(n) VARBINARY(n) VARBINARY(max) IMAGE
size：< 2G	变长的二进制数据。最多 2GB 字节数据。	

日期时间类型：

GBase 8s		SQL Server
DATETIME YEAR TO FRACTION(5)		DATETIME DATETIME2
size：11	0001-01-01 至 9999-12-31，精度为 10 微秒	
DATETIME YEAR TO SECOND		SMALLDATETIME
size：8	0001-01-01 至 9999-12-31，精度为 1 秒	
DATE		DATE
size：4	0001-01-01 至 9999-12-31	
DATETIME HOUR TO FRACTION(5)		TIME
size：7	精度为 10 微秒。	
DATETIME YEAR TO FRACTION(5)		DATETIMEOFFSET
size：11	0001-01-01 至 9999-12-31，精度为 10 微秒	

说明：

1， GBase 8s 不支持 DATETIMEOFFSET 类型（含时区），需要改造成通用日期时间格式。

其它类型：

GBase 8s		SQL Server
BOOLEAN		**BIT**
size：1	取值为't'、'f'或者 null	
INT8 / VARCHAR(20)		**ROWVERSION TIMESTAMP**
size：		
VARCHAR(8000)		**SQL_VARIANT**
size：n		
CHAR(36)		**UNIQUEIDENTIFIER**
size：36	字段创建默认值 default(sys_guid())	
TEXT / JSON		**XML**
size：< 2GB		

说明：

1， SQL Server 中的 BIT 的取值为 1、0 或者 null，与 GBase 8s 中的 BLOOLEAN 不同，需改造；

2， SQL Server 中的 ROWVERSION/TIMESTAMP 和 SQL_VARIANT 只是使用字段接收，业务需改造；

3， SQL Server 中的 UNIQUEIDENTIFIER，需要使用字段+默认值的方式。

3.1.2 数据类型说明

数值类型：

SQL Server 和 GBase 8s 数据库的数值类型基本兼容，仅 DECIMAL(p,s)的精度有差异，SQL Server 支持 38，而 GBase 8s 只支持到 32。

整型数值需要注意最小值的差异。

字符类型

CHAR，VARCHAR 是 GBase 8s 常用的两个字符类型，CHAR 的长度是 32767，VARCHAR 的长度是 32765。

SQL Server 中的 VARCHAR（max）可使用大对象 TEXT 或者 JSON 来代替。

时间类型

GBase 8s 的常用时间类型有 DATE 和 DATETIME ，这是两个精度不同的时间类型。DATETIME 除了日期之外，还可以通过不同的定义来定义更精确的时间，最大精确十微秒，常用的精度为 DATETIME YEAR TO SECOND。

SQL Server 中的 DATETIMEOFFSET(支持时区)类型在 GBase 8s 中不被支持，需要进行相应的调整。

其它类型

SQL Server 中的 BIT、ROWVERSION/TIMESTAMP、SQL_VARIANT 和 UNIQUEIDENTIFIER 等数据需要进行相应的调整。

SQL Server 到 GBase 8s 的具体的数据类型映射，请参照附录 C。

3.2 表

3.2.1 建表语句

GBase 8s 中建表语句与 SQL Server 非常相似，细节请酌情修改。

语法

```
CREATE [TEMP] TABLE table-name
        (
        column-name
            {
            datatype
            | {BYTE|TEXT} [IN {TABLE | BLOBspace-name}]
            }
            [DEFAULT default_opts]
            [table-constraint-definition][,...]
            [column-constraint-definition][,...]
            [NOT NULL] [UNIQUE [CONSTRAINT constr-name]][,...]
            [UNIQUE (unique-column-list) [CONSTRAINT constr-name]][,...]
            [[COLUMN] SECURED WITH label-name]
        )
        [SECURITY POLICY policy-name]
        [WITH NO LOG] [IN DBspace-name] [EXTENT SIZE extent-size]
        [NEXT SIZE next-size]  [LOCK MODE {PAGE | ROW }]
```

GBase 8s 中可以对表的存储空间进行分配：EXTENT SIZE 和 NEXT SIZE 分别指定了表的初始大小和再次分配的大小。

示例：

```
CREATE TABLE dept
(
```

```
    deptno SMALLINT,
    dname VARCHAR(14),
    loc VARCHAR(13)
)
EXTENT SIZE 200 NEXT SIZE 50;
```

默认值和最小值：

GBase 8s 在没有指定存储信息时，大多数平台上的默认 first/next extent 值为 8 页，最小值为页大小的 4 倍;

GBase 8s 在 UTF-8 环境下支持中文的表名和字段名。

3.2.2 表的分片

GBase 8s 的表分片技术允许把表级数据存储在不同的物理位置，将大表分片可以提高用户相应时间，并发性能，存储性能，备份恢复性能和数据装载性能。GBase 8s 可以并行地扫描多个磁盘上的分片数据，从而实现内部查询的并行操作，因此采用“分片”技术可以提高查询效率。内部查询的并行化有助于减少对一个复杂查询的响应时间。“表分片”技术与并行数据查询（PDQ）特征联系在一起使用，数据库可以分配多个线程，从所有数据分片上并行地选取数据。此外，还可以仅仅对包含“目标数据”的数据分片进行扫描，从而大幅度地提高了整个系统效率。

表分片方式有 5 种 :轮转分片(Round-Robin) 、表达式分片(Expression-Based)、List 分片、Interval 固定间隔分片、哈希（Hash）分片。

轮转分片语法示例：

```
CREATE TABLE dept
(
    deptno SMALLINT, dname VARCHAR(14), loc VARCHAR(13)
)
FRAGMENT BY round robin
     IN dbspace1, dbspace2, dbspace3, dbspace4;
```

表达式分片语法对比：

```
CREATE TABLE dept
(
    deptno SMALLINT, dname VARCHAR(14), loc VARCHAR(13)
)
FRAGMENT BY EXPRESSION
```

```
deptno < 11 IN dbspace1,
deptno >= 11 AND deptno < 21 IN dbspace2,
deptno >= 21 AND deptno < 31 IN dbspace3,
REMAINDER IN dbspace4;
```

GBase 8s 可以使用 partition 方式来指定分片表的存储位置，这样可以把多个分区放置在同一个 dbspace 中，避免了太多 dbspace，简化维护管理。

Partition 语法示例：

```
CREATE TABLE dept
(
    deptno SMALLINT,
    dname VARCHAR(14),
    loc VARCHAR(13)
)
PARTITION BY EXPRESSION
partition p1 (deptno < 11) IN dbspace1,
partition p2 (deptno >= 11 AND deptno < 21) IN dbspace2,
partition p3 (deptno >= 21 AND deptno < 31) IN dbspace2;
```

List 方式，List 方式实际上是对 or 和 in 运算的改进，在表达式计算上更具有灵活性和效率。List 方式示例如下：

```
create table customer(cust_id integer,
  name varchar(128),street varchar(255),
  state char(2),zipcode char(5),phone char(15))
 Fragment by list(state)
 PARTITION p0 values('RS','IL') in datadbs01,
 PARTITION p1 values('CA','OR') in datadbs02,
 PARTITION p2 values('NY','MN') in datadbs03,
 PARTITION p3 values(NULL) in datadbs04 ;
```

间隔分片，是指分片数据是基于一个间隔值，比如，一个分片是基于一个月，一年或几百万顾客的记录。数据表有一个最初的分片，它是基于一个 range 语句来定义的。当一条记录不能满足最初的分片时，IDS 系统将会自动创建一个分片来存储这条记录，数据库表和索引都可以用这种策略来进行分片。这个过程是不需要 DBA 参与的。

间隔分片语法示例：

```
create table order
  (
    order_num integer not null,
    order_date DATE
  )fragment by RANGE(order_date)
```

```
    interval(NUMTOYMINTERVAL(3,'MONTH')) store in (dbs1, dbs2,dbs3, dbs4)
        partition Q0 VALUES<'01/01/2014' in dbs1;
```

哈希分片，是按分区列的 HASH 计算结果来决定其分区的，而特定的分区列其 HASH 值是固定的，也就是说 HASH 分区表的数据是按分区列值来聚集的，同样的分区列肯定在同一分区。例如在证券行业，我们经常查询某一只股票的 K 线，建成 HASH 分区表，则数据按 HASH 分区列聚集，就更适合 K 线数据的查询，因为同样 id 的记录必定在同一分区，同时，同样 id 值的记录落在同一数据块的几率也增大了，从而一定程度上减少 IO，提高查询效率。

哈希分片语法示例：

```
CREATE TABLE USER(
    name varchar(20),
    departmentID int,
    age int,
    address varchar(50)
)PARTITION BY HASH(departmentID)
(
partition p1 DBSPACE dbs1,
partition p2 DBSPACE dbs2,
partition p3 DBSPACE dbs3
);
```

GBase 8s 的每个分片是一个独立的 tablespace，按照 tablespace 的存储特性，每个分片使用的页上限为 16777215，如果 dbspace 使用 16K 数据页，那么这个最大值应是 256G，超过这个限制将无法向该分片插入数据。

3.3 索引

3.3.1 语法

```
CREATE [UNIQUE|DISTINCT] [CLUSTER] INDEX index-name
ON table-name (column-name [ASC|DESC],...) storage option [online]
```

语法示例：

```
create index zip_ix on customer (zipcode) ;
create index zip_ix on customer (zipcode)  in idxdbs;
```

默认使用 btree 索引，默认 dbspace 和表所使用的 dbspace 一致。

在线创建索引：

```
create index idx_name on table_name(col_name1,colname2,…) ONLINE;
```

在创建索引时，使用 online 关键字，可以在用户访问这张表的同时，进行创建索引操作，而不会造成锁表。online 关键字在删除索引时同样有效。

3.3.2 索引限制

GBase 8s 的索引长度限制于使用的数据页大小相关

常用页大小	索引长度限制
2K	387 字节
4K	796 字节
8K	1615 字节
12K	2435 字节
16K	3254 字节

每个索引分片的最大页数 2,147,483,647。

3.3.3 复合索引

GBase 8s 中一个复合索引最多可以使用 16 个字段做键值，或最多 341 个键值作为 SPL/JARA UDR，102 个键值作为 C UDR 的返回值。这在不同的语言中有不同的上限限制。

3.3.4 索引分片

GBase 8s 支持索引分片。

创建分片索引的语法与创建分片表的语法非常相似：

```
    CREATE [UNIQUE|DISTINCT] [CLUSTER] INDEX index-definition indexname
    on table-name (column-name[,...]) [FILLFACTOR percent] [IN dbspace]
[fragment-clause]
    FRAGMENT|PARTITION  BY
    EXPRESSION Expression Fragment Clause |
    RANGE (fragment_key) Interval Fragment Clause |
    LIST (fragment_key) List Fragment Clause
```

3.3.5 其它

GBase 8s 可以根据应用需要，用 CLUSTER 选项来建立聚集索引，这个选项会对表中的数据按索引的顺序排序。每个表只能建立一个聚集索引。DML 语句的数量是判断能否可以建立聚集索引的重要因素，聚集索引上的大量的 DML 语句会导致性能问题。

GBase 8s 的索引 FILLFACTOR 指定了一个百分比，用来设定索引页的填充度。

3.4 约束

语法

```
    CREATE [TEMP] TABLE table-name
            (
            column-name
                {  datatype  }
                [table-constraint-definition][,...]
                [column-constraint-definition][,...]
                [NOT NULL] [UNIQUE [CONSTRAINT constr-name]][,...]
                [UNIQUE (unique-column-list) [CONSTRAINT constr-name]][,...]
            )
            [WITH NO LOG][IN DBspace-name] [EXTENT SIZE extent-size]
            [NEXT SIZE next-size] [LOCK MODE {PAGE | ROW }]

    ALTER TABLE table-name ADD CONSTRAINT UNIQUE
(old-column-name[,...])[table-constraint-definition]
```

GBase 8s 建立约束的语法：

```
    PRIMARY KEY(CUST_NUM) CONSTRAINT PK_NUMBER
    CHECK (EMP_CODE > 100) CONSTRAINT CK_EMPCD
```

3.5 视图

创建视图语法：

```
    CREATE VIEW view-name [(column-list)] AS SELECT-statement [WITH CHECK OPTION]
```

创建视图示例：

```
    create view v_dept (deptno, dname) as
```

```
select x0.deptno ,x0.dname from dept x0 ;
```

由于视图并不是一张实体表，不能对其使用 ALTER 操作，如果有视图名或字段需要修改，需要将其重建。

GBase 8s 目前不支持物化视图。

3.6 序列

GBase 8s 支持 DML 语句(CREATE SEQUENCE, ALTER SEQUENCE, RENAME SEQUENCE, DROP SEQUENCE)来操纵序列，多用户可以并发访问同一个序列，每个序列占用 8 个字节，GRANT 和 REVOKE 语句可以用来更改序列的访问权限。可以为本地数据库中的序列建立同义词。用于调整序列值的 CURRVAL 和 NEXTVAL 在同义词中同样适用。

对于 SQL Server 中的数值类型的自增功能，可使用 GBase 8s 相应的序列数据类型来实现。特殊的自增功能，如 int identity(1,5)，步长为 5，则需要使用序列（SEQUENCE）来实现。

3.7 存储过程和函数

3.7.1 定义格式

GBase 8s	SQL Server

drop procedure if exists proc_name();
 	
create procedure proc_name()
define in_modify_type char(1);
define in_application_msg char(10);
let in_application_msg = 'E';
……
end procedure;
	IF EXISTS (SELECT 1 FROM sysobjects WHERE type='P' AND name =' proc_name')
	DROP PROCEDURE proc_name
	go
	create procedure proc_name
	AS
	declare @in_modify_type char(1)
	declare @in_application_msg char(10)
	set @in_application_msg ='E'
	……
	go

删除存储过程时的判断有区别，参数定义有区别。

3.7.2 入参出参

GBase 8s	SQL Server
drop procedure if exists proc_name(); create procedure proc_name(v_1 varchar(255), v_2 varchar(255), out v_3 varchar(255)) …… end procedure;	 create procedure proc_name @v_1 varchar(255), @v_2 varchar(255), @v_3 varchar(255) output AS …… go

in/out/inout 关键字在位置上有所区别且名称有所不同，并且对于变长字段类型，SQL Server 不指定长度默认为最大长度，而 GBase 8s 默认长度为 1，所以转换时需要指定最大支持字符长度。

3.7.3 变量定义和赋值

GBase 8s	SQL Server
define v_var1 varchar(255); let v_var1 = 'GBase 8s' ;	declare @v_var1 varchar(255) set @v_var1='GBase8s'
define v_var2 varchar(255); select name into v_var2 from tb1;	decalre @v_var2 varchar(255) select @v_var2 = name from tb1;
define v_var3 rt_tb1; select * into v_var3 from tb1;	declare @v_var3 table(col1 int, col2 int); select * into @v_var3 from tb1;

在存储过程中，可以预先自定义好 row type 类型，然后再在存储过程中使用该自定义 row type 类型。如上表中的 rt_tb1 类型可以替换 SQL Server 中的 table 类型。上例中的表及 row type 定义格式如下：

```
create table tb1(
  id int,
  name varchar(20),
  primary key(id)
);
create row type rt_tb1(
  id int,
  name varchar(20)
);
```

3.7.4 IF 条件判断

GBase 8s	SQL Server

if 条件 1 then
语句 1 ;
elif 条件 2 then
语句 2 ;
else
语句 3 ;
end if ;
	if 条件 1
	begin
	语句 1
	end
	else if 条件 2
	begin
	语句 2
	end
	else
	begin
	语句 3
	end

语法区别就是 else if 的写法 ,SQL Server 为 ELSE IF ,GBase 8s 为 ELIF ;SQL Server 中子句中需要使用语句块（begin ... end），GBase 8s 不强制使用语句块。

3.7.5 WHILE 循环

GBase 8s	SQL Server

while 条件成立
begin
执行语句;
end;
end while;
	while 条件成立
	begin
	执行语句;
	end

语法区别就是 SQLServer 不需要 end while; 这样的结束语句，但是必须使用语句块 (begin ... end)，GBase 8s 不强制使用语句块。

3.7.6 WHEN THEN 条件分支

GBase 8s	SQL Server
case when 条件 1 then 语句 1 when 条件 2 then 语句 2 ... else 语句 N end	set @week=case when @today=1 then '星期一' when @today=2 then '星期二' when @today=3 then '星期三' when @today=4 then '星期四' when @today=5 then '星期五'

	when @today=6 then '星期六' when @today=7 then '星期日' else '值错误' end

SQL Server 的 WHEN THEN 条件分支，与 GBase 8s 的条件表达式类似，可直接替代。

3.7.7 异常处理

GBase 8s	SQL Server
define sql_err int; define isam_err int; define err_info varchar(255); on exception set sql_err,isam_err,err_info rollback; end exception;	BEGIN TRY { sql_statement | statement_block } END TRY BEGIN CATCH [{ sql_statement | statement_block }] END CATCH

on exception in (10000)
rollback;
return 0;
end exception;
 	
if v_flag = 0 then
raise exception 10000;
end if;	

值得注意的是，GBase 8s 的 exception 的错误号必须是一个 smallint 的数字；异常处理语句必须写在变量定义之后，事务语句开始之前。

3.7.8 动态 SQL 与绑定变量

GBase 8s	SQL Server
-- 代入语句法 let v_sql = 'select id,name into ' || v_id || ',' || v_name || ' from tb1 where id = 1'; execute immediate v_sql;	set @v_sql = 'select id,name from tb1 where id=1'; EXECUTE (@v_sql)
-- 使用游标方式 define v_cursor sys_refcursor; let v_sql = 'select id,name from tb1 where id = 1'; open v_cursor for v_sql; loop fetch v_cursor into v_id, v_name;	

exit when SQLCODE = 100; …… end loop;	
let v_sql = 'delete from tb1 where id = :id'; let v_sql = replace(v_sql,':id',v_id); execute immediate v_sql;	set @v_sql= N'delete from tb1 where id = @id' EXEC SP_EXECUTESQL @v_ql, N'@id NVARCHAR(50)', @id

GBase 8s 不支持动态 SQL 插入和绑定变量执行 转换的原则就是将变量代入动态 SQL 中既定的位置组成新的动态 SQL 字符串执行，或者使用游标方式。

3.7.9 游标

GBase 8s	SQL Server
define v_id integer; define v_oid integer; define v_login varchar(255); foreach select id,oid,login into v_id,v_ oid v_login from ST_User …… end foreach;	--定义一个游标 declare user_cur cursor for select ID,Oid,[Login] from ST_User --打开游标 open user_cur while @@fetch_status=0 begin --读取游标 fetch next from user_cur into @ID,@Oid,@Login print @ID --print @Login end close user_cur --摧毁游标 deallocate user_cur
define v_id integer; define v_oid integer; define v_login varchar(255); prepare mycur from 'select id,oid,login from ST_User'; declare cur cursor for mycur; open cur; loop fetch cur into v_id,v_oid,v_loing; …… end loop; close cur;	

define v_id integer;
define v_oid integer;
define v_login varchar(255);
define v_cursor sys_refcursor;
define v_sql varchar(1024);
let v_sql = 'select id,oid,login from ST_User';
open v_cursor for v_sql;
loop
fetch v_cursor into v_id, v_oid,v_login;	

exit when SQLCODE = 100; …… end loop; close v_cursor;	

针对 SQL Server 的游标，GBase 8s 可以有显示和隐式游标两种迁移方法，一般常用隐式 foreach 写法，但是显示游标支持绑定变量，如：

```
prepare mycur from 'select id,name from tb1 where id = ?';
declare cur cursor for mycur;
open cur using nid;
loop
fetch cur into v_id,v_name;

...

end loop;
close cur;
```

3.7.10使用其他数据库中的表

GBase 8s	SQL Server
select * from DBNAME@DBSERVER:TABNAME;	select * from [DBSERVER].[helpdesk].[dbo].[Messages];

GBase 8s 使用分号作为分隔符，而 SQL Server 使用点号作为分隔符。

3.8 触发器

触发器的迁移相比存储过程的迁移相对比较简单，主体语句上相似。相对而言，GBase 8s 在触发器的语言上更简洁一些。

GBase 8s	SQL Server
create tirgger tri_tab1 delete on tb1 REFERENCING OLD AS OLD for each row(insert into tb1_del(id,name) values(old.id, old.name));	create trigger tri_tb1 on tb1 after delete as begin declare @v_id int declare @v_name varchar(40) select @v_id=id from deleted select @v_name=name from delete insert into tb1_del(id,name) values(@v_id,@v_name) end

	go

4 数据迁移

4.1 迁移过程

数据的迁移过程由两部分组成。

第一部分是迁移数据库的结构，主要包括数据库对象的迁移；第二部分是将数据迁移到目标数据库中。

业务数据迁移是在第二部分完成的。对于数据的迁移，通常情况下是要暂停应用程序，这样可以保证迁移前后的数据一致性和完整性。使用常规方法，如果数据量很大的话，则需要较长的应用停机时间窗口。

4.2 文件格式

把数据从 SQL Server 导出时，要将格式尽量调整为匹配 GBase 8s 的导入格式，这样可以最大程度上减小数据在转换过程中的工作量。这里以 GBase 8s 的 load 工具格式为例，来说明 一下导入时对文件格式的要求。

GBase 8s 的 load 工具是一个常用的文本导入工具，它的字段间默认分隔符是“|”管道符。由于“|”在文本中是很少出现的字符，所以也推荐在导出时使用“|”来作为分隔符。每个字段后都应由“|”来作为结束标识，换行符作为行与行之间的分隔符。每个表的数据单独存储在一个文件中。

示例：

```
create table customer_log
  (  id char(14),
    update_date datetime year to second,
    tablename varchar(20),
    update_count float,
updated float  );
```

表 customer_log 的导出/导入格式如下：

```
20101013114153|2010-10-13 11:41:53|2|53.0|53.0|
```

```
20101013114153|2010-10-13 11:41:53|3|0.0|0.0|
20101015094917|2010-10-15 09:49:17|2|15.0|15.0|
20101015094917|2010-10-15 09:49:17|3|0.0|0.0|
20101015094918|2010-10-15 09:49:18|4|1.0|1.0|
20101015102622|2010-10-15 10:26:22|2|2.0|2.0|
20101015102622|2010-10-15 10:26:22|4|0.0|0.0|
20101015111103|2010-10-15 11:11:03|1|1.0|1.0|
```

4.3 数据导出

SQL Server 的数据可以通过其自带的导入和导出工具，导出符合 GBase 8s 要求的文本格式。

4.4 数据导入

4.4.1 注意事项

建立数据库时，以无日志模式建立，这样数据在导入时无需记录逻辑日志，导入效率会大幅提高。当数据导入完毕后，使用 ontape 命令将数据库改为需要的日志模式。

例如将某数据库日志模式改为 UNBUFF：

```
ontape  -U database_name  -s  -L 0  -t /dev/null
```

在数据量大的迁移中，可以将建立索引、约束等脚本单独提取出来，在所有数据成功导入后，再打开 PDQ 建立。

4.4.2 load 工具

使用 load 工具进行导入：

load 是 GBase 8s 最基础和最常用的文本数据导入工具，支持多表并发导入，操作简单。

示例：

```
load from '/opt/gbase8s/data/order.unl' insert into order;
```

4.4.3 外部表导入

使用 GBase 8s 外部表(external table)进行导入：

对于数据量大的表，传统的 load 导入方式会在迁移过程中占用大量的时间窗口，成为迁移效率的瓶颈。针对这个问题，对大表的导入，可以采用 GBase 8s 外部表的方式进行，

创建 External table：

语法

```
CREATE EXTERNAL TABLE table-name
(
column-name { datatype [DEFAULT default_opts] | <UDTs> } [
<external-column-defn> ] [,...]
)
USING  (DATAFILES("{DISK | PIPE} : file-path" [,...] )
[, <table-option> [...] ])
<external-column-defn>: EXTERNAL CHAR( size ) [ NULL 'null-string' [ NOT NULL ] ]
<table-options>:
FORMAT format-type
DEFAULT | DELUXE | EXPRESS
ESCAPE 'escape-character'
DELIMITER 'field-delimiter'
RECORDEND 'record-delimiter'
MAXERRORS num-errors
REJECTFILE 'filename'
NUMROWS num-rows
```

示例：

```
create external table orders_ext
  ( order_num serial, order_date date, customer_num integer,ship_instruct
char(40),
  backlog char(1), po_num char(10),ship_date date, ship_weight decimal(8,2),
  ship_charge money(6,2), paid_date date )
  using
    (
      datafiles  ("DISK:/opt/gbase/test/external_table/orders1.unl",
                  "DISK:/opt/gbase/test/external_table/orders2.unl" ),
      format "delimited",
      DELIMITER "|",
      rejectfile "/opt/gbase/test/external_table/orders_rejfile.err",
      maxerrors 100
    );
```

也可根据已有表结构建立相同结构的外部表：

```
create external table orders_ext SAMEAS orders
using
  (
      datafiles  ("DISK:/opt/gbse/test/external_table/orders1.unl",
                  "DISK:/opt/gbase/test/external_table/orders2.unl" ),
    format "delimited",
    DELIMITER "|",
    rejectfile "/opt/gbase/test/external_table/orders_rejfile.err",
    maxerrors 100
  );
```

导入数据：

```
insert into orders select * from orders_ext;
```

外部表使用技巧：PDQ & 分片表 & Light append

打开 PDQ 功能，并行处理。

目的表是分片表能进行并行的 insert 和 select。

当导入表为 RAW TABLE 时，利用 Light append 进行快速数据导入。

4.5 GBase Migration Toolkit 迁移工具

GBase Migration Toolkit 迁移工具是 GBase 提供的一款实现异构数据库进行数据迁移的工具。可以实现将源数据库（支持的源数据库有：ACCESS、Oracle、SQL Server、DB2、MySQL、ShenTong、GBase8sV8.3、GBase8s、PostgreSQL 和 Teradata）中数据库对象和数据迁移到目标数据库（支持的目标数据库有：GBase8a、GBase8t、GBase8s 和 GBase8s_M）。

迁移工具是一个 C/S 结构的软件，安装简便，只需要获取安装包解压后即可使用。迁移工具具有简单易操作的图形化界面，根据数据迁移需求创建相应的任务，并且可以对迁移任务进行相应的设置，实现多线程进行并发数据迁移。

4.5.1 操作步骤

下面以 SQLServer 数据库迁移到 GBase 8s 数据库为例进行介绍，具体操作步骤如下：

1）、完成迁移参数设置和驱动设置：

迁移参数设置：设置列表刷新时间间隔、读取表结构线程数、写入表结构线程数、读取数据线程数、写入数据线程数、数据提交条数、一次读取数据条数、队列大小、任务文件存放路径。

驱动配置：配置主机、端口、用户名、密码、数据库、参数串、实例名/服务名、驱动文件参数。

2）、创建迁移任务：

新建迁移任务，填写任务名称，选择源数据库的类型和目标数据库类型，配置源数据库，工具自动记忆最近一次使用该类型数据库的配置，填写 SQLServer 数据和 GBase 8s 数据库的相关参数，设置好要迁移的对象以及迁移方式。

3）、确定迁移范围：

迁移工具支持迁移数据库表结构和数据，可以同时进行也可分开进行。针对 SQLServer 数据库支持迁移表对象，对表对象可以选择是否重建表和迁移外键，或者设置只迁移表结构和只迁移数据，在设置迁移表结构时可以选择相应的约束迁移，可以根据自己的需要去掉不迁移的表，同时可以对其新表名进行重命名，指定在目标库中的表名称，指定表对象中的迁移列信息，不设置是默认迁移表中所有的列。

4）、迁移设置：

设置 SQLServer 数据库和 GBase 8s 数据库中的数据类型对应，以及迁入到 GBase 8s 库时各数据类型的长度和精度，如果对有长度和精度数值的数据库类型，设置后迁移工具将会采用用户设置的长度和精度，如果采取默认，迁移工具将会自动获取 SQLServer 数据库的精度和长度。

5）、迁移执行结果分析：

设置任务启动时间后按照任务计划启动任务，也可立即启动任务，任务执行完成后生成任务报告，可以查看任务的详细信息。其中：未结束的任务，可以查看任务详情，详情中展现任务的基本配置信息和迁移的基本信息；已完成的任务（已完成是指：状态为已完成或者已停止，其它状态均为未完成）将会显示任务报告，其中会给出相应任务的耗时以及结果信息等报告内容。对于有错误信息的迁移对象可以查看错误信息详情，可以对失败的任务信息进行导出，重建失败任务。

4.5.2 常用问题及注意事项

使用迁移工具完成 SQLServer 数据库迁移到 GBase 8s 数据库迁移常见问题以及注意事项：

1）、创建任务失败：

一般原因为对 persistence_file_path 目录没有操作权限，无法写入迁移任务。其次考虑对 conf 文件夹内文件是否有修改权限，创建任务同时会对 conf 下部分配置文件进行自动修改，如无权限也可能导致任务保存失败。

2）、启动任务失败：

任务启动失败优先查看迁移工具日志文件，目录为“安装路径/migration/logs/migration.log”。可能原因是工具对操作任务目录以及文件没有足够的权限，会导致启动失败。

3）、迁移任务失败：

任务启动失败优先查看迁移任务报告，报告中会展示常见的错误原因。具体错误原因可以查看工具日志文件，目录为“安装路径/migration/logs/migration.log”。

失败原因有很多，常见的有：

（1）源数据库和目标数据的权限问题；

（2）目标库中已存在重名的表名称；

（3）建表语句有误问题，造成建表语句有误的原因一般为 UserDataTypeMapping.xml 中的数据类型对应有问题，源数据库读取不到该数据类型，或者目标数据库不支持创建类型。

（4）网络中断问题，造成无法连接数据库。

（5）内存溢出问题，虚拟内存开辟空间不足，造成程序卡住，需要修改内存设置，配置文件为“安装目录/migration/Migration.ini”，设置-vmargs -Xmx40960m，设置参考值迁移数据量为 2 亿数据量，所需内存在 30G 左右。

注：任务迁移失败后，查看任务报告以及日志文件，可以判断任务是否已经在目标数据库内建表，若已经建表，排除问题原因后，清理环境重新启动任务，或者在任务中选择“重建表”，直接再次执行任务即可。

4）、Check 约束迁移问题

（1）迁移 check 约束的时候建议不要修改列名称，否则会出现建表失败的问题。

（2）SQLServer 迁移 check 约束的时候，check 约束的表达式中不要使用“中括号[]”。

4.6 校验

对数据行数进行校验

GBase 8s 数据库中各表的行数统计：

```
update statistics;
select tabname,nrows from systables
where tabid>99 and tabtpye='T' order by 2 desc;
```

5 应用迁移

5.1 SQL

5.1.1 关键字

与 DDL 语句类似，GBase 8s 的 DML 语句与 SQL Server 在绝大多数结构上是一致的。一些细节用法上的不同和对应功能的语法差异，是需要进行转换的。例如 YEAR、MONTH、DAY、HOUR 这些 GBase 8s 中的关键字，是必须进行转换的。

更多的关键字请参考附录 B。

5.1.2 不等于

GBase 8s 中不等于判定支持以下几种表示：

“not equal to”

“<>”

“!=”

5.1.3 SELECT

查询语句在被执行时，由数据库优化器来决定最佳的查询计划。GBase 8s 与 SQL Server 使用着各自开发的不同的查询优化器，有着各自的方法和规则。同一个查询语句在不同的优化器下可能会选择不同的查询路径，语句的执行时间也会有快有慢。因此，迁移后的每个查询语句的性能都要经过测试，来确定是否需要进行必要的转换来达到预期的性能。

5.1.4 查询优化器（伪指令）

GBase 8s 支持在查询语句中指定它的执行计划，指定的格式有多种，例如：

```
select {+ ORDERED AVOID_FULL(e)}
empname, deptno
from employee e, department d
where e.dept_no = d.dept_no;
```

通过对 SELECT 后的注释部分加入“+”来植入指定的执行计划。

5.1.5 INSERT

GBase 8s 在 INSERT…SELECT…语句结构中，与 SQL Server 语法结构不同的是，需要去掉 VALUES，并且不支持 INSERT…SELECT…UNION 的写法。

5.1.6 临时表

GBase 8s 中将临时表几乎当做一张常规表使用，当一个会话结束时，它遗留的临时表也自动被删除。

GBase 8s 中允许不同会话中创建同名临时表，任何时候同一用户的表名都唯一。

GBase 8s 中可以采用如下两种方式创建临时表：

使用 SELECT INTO TEMP 语句隐式的创建临时表；
例：Select * from customer into tmp_customer with no log;
使用 CREATE TEMP TABLE 语句显式的创建临时表；
例：Create temp table tmp_customer (c1 int) with no log;

5.1.7 排序

GBase 8s 和 SQL Server 的 NULL 在排序中默认都被作为最小值来对待。

5.1.8 别名

GBase 8s 在修改和删除语句中不支持对主表指定别名，例如：

```
UPDATE customer
SET zip_code = '92612'
WHERE zip_code IN
(SELECT zip_code FROM zip_table Z
WHERE customer.create_date <= Z.effective_date)

DELETE customer
WHERE order_date <=
(SELECT control_value FROM control_table T
WHERE T.control_field = 'archive_date'
AND customer.state = T.state )
```

GBase 8s 中，如果在 FROM 子句中使用的别名，在 SELECT 和 WHERE 中也必须使用这个别名，原表名必须被别名完全取代。

5.1.9 级联查询(Hierarchical queries)

GBase 8s 支持级联查询。

语法：

```
SELECT [ALL | DISTINCT | UNIQUE] select-list
        FROM [OUTER] table-name [table-alias] [,...]
        [WHERE condition]
        [START WITH condition] [CONNECT BY [NOCYCLE] condition]
        [GROUP BY column-list] [HAVING condition]
        [ORDER [SIBLINGS] BY column-name [ASC | DESC],...]
        [INTO TEMP table-name]
```

START WITH：告诉系统以哪个节点作为根结点开始查找并构造结果集，该节点即为返回记录中的最高节点。 当分层查询中存在上下层互为父子节点的情况时，会返回 26079 错误。此时，需要在 connect by 后面加上 NOCYCLE 关键字。同时，可用 connect_by_iscycle

伪列定位出存在互为父子循环的具体节点。 connect_by_iscycle 必须要跟关键字 NOCYCLE 结合起来使用。

5.1.10TRUNCATE

GBase 8s 支持 TRUNCATE 语句来删除整个表中的数据行及相关索引，并可选择性的释放原有数据 extent 占用的存储空间，默认为释放存储空间。语法如下：

```
TRUNCATE [TABLE] table [ [ DROP | REUSE ] STORAGE ];
```

对于有 DELETE 触发器的表进行 TRUNCATE 时 ,需要至少库级 RESOURCE,表级 ALTER 权限，因为数据库需要自动忽略触发器。对普通表进行 TRUNCATE 操作时只需要库级 CONNECT 和表级 DELETE 权限即可。

GBase 8s 允许截断操作在事务中执行，当回滚发生后，所有在截断中删除的数据将被恢复。在事务中，该操作是有一定限制的，TRUNCATE 执行后，只允许提交或回滚操作，执行其它语句将会返回错误。

TRUNCATE 操作不会重置 SERIAL 和 SERIAL8 类型字段的当前值，想在截断数据后重置这个值，需要使用 ALTER TABLE...MODIFY...

在执行 TRUNCATE 操作后，GBase 8s 会自动执行统计更新操作，不需要再手动执行这个操作。

5.1.11系统表

涉及到 SQL Server 系统表的应用，必须进行移植，用 GBase 8s 对应的系统表将其替换，对于数据库系统表中无法对应的部分，需要对这部分程序进行删除。

GBase 8s 的绝大多数系统表名都以 sys 开头，若 SQL Server 应用中的表与 GBase 8s 系统表重名，则必须在迁移时更换表名。

5.2 SPL(Stored Procedure Language)

5.2.1 SPL 概览

GBase 8s 的存储过程和方法都可以看做是 UDR(user-defined routines)，一个 UDR 可以使用 SPL 或是其他外部语言，例如 C 或 JAVA。存储过程就是一个不返回值的程序。SPL 语句只能在存储过程中使用，SPL 程序以可执行的格式被解析，优化，存储在系统目录表中。

GBase 8s 的存储过程支持输入，输出和输入输出参数，并且可以用在 SQL 语句中任何可以使用表达式的地方。

变量定义和赋值

DEFINE， LET

流程控制

分支控制 IF

循环控制 FOR，FOREACH，WHILE

EXIT， CONTINUE

函数调用与返回

CALL， SYSTEM ，RETURN

错误处理和调试

TRACE， ON EXCEPTION， RAISE EXCEPTION

5.2.2 大小限制

GBase 8s 的存储过程大小限制在 64K 左右。SQL Server 大于 64K 的存储过程，在移植时，可以改写成多个小的、互相关联的存储过程，通过传递和返回参数值来交互信息。

5.2.3 参数限制

GBase 8s 的存储过程参数上限是 341 个，使用 JAVA 编写的 UDR 也适用这个限制,C 语言编写的 UDR 最多支持 102 个参数。Java 编写的 UDR 中 DECIMAL 数据类型的参数不能超过 9 个。C 语言编写的 UDR 中如果返回不透明数据类型的值，必须在 C 宿主变量的定义中指定 opaque_type。

5.2.4 宿主变量

在移植过程中，宿主变量(Host variables)的格式是不需要主动转换的，它值需要跟随数据类型做相应的转换。在 SQL Server 中的一些类型，比如 DATE 变量，在 GBase 8s 中没有类型与其直接对应，所以在 SQL 执行的前后需要对数据进行类型转换。

5.2.5 游标

GBase 8s 中，SQL Server 游标可以被临时表替代。临时表的创建相当于游标的打开，临时表可以和游标一样使用，使用后，临时表可以被删除，就相当于关闭游标。

在 SQL Server 存储过程中，游标是显式地声明、打开和获取，SQL Server 存储过程游标可以使用 GBase 8s 的 FOREACH 结构替换。SQL Server 游标名应当在 FOREACH 语句中使用,例如:FOREACH cursor_name SELECT...END FOREACH。

在 GBase 8s 中,如果在 FOREACH 结构的 SELECT 语句不是多表关联,可以使用 UPDATE <table_name> WHERE CURRENT OF <cursor_name> 语句来更新每一行。GBase 8s 的存储过程中不支持 SELECT FOR UPDATE 语句。

5.2.6 Routine

SQL Server 的存储函数和存储过程，在概念上与 GBase 8s 的存储过程很相似。SQL Server 用户定义的数据库级别的存储过程与函数都必须用 SPL、C、或 Java 来转换成 GBase 8s 的存储过程。SPL 写的程序在维护上有很大优势，同时也可以被数据库备份软件备份。C 语言的程序的优点是执行速度快，但备份相对麻烦，它无法被自动备份，需要在操作系统单

独编译。Java 程序同样也不随数据库自动备份，它的优势在于跨平台。

5.2.7 动态 SQL

GBase 8s 在存储过程中支持动态 SQL。

动态 SQL 是指可以在运行期间根据用户提供的信息动态地构建和执行的 SQL 语句。许多数据库应用程序在设计和验证阶段需要动态 SQL 功能来验证不完全确定的 SQL。

EXECUTE IMMEDIATE 语句

EXECUTE IMMEDIATE 使程序在执行过程中动态执行单个 SQL 语句变得更加简单。

动态 SQL 示例

```
CREATE PROCEDURE create_tab (table_name CHAR(128), column_list CHAR(512))
   DEFINE l_crtstmt CHAR(1024);
   LET l_crtstmt = "CREATE TABLE " || table_name ||"("|| column_list || " )";
   EXECUTE IMMEDIATE l_crtstmt;
END PROCEDURE;
EXECUTE PROCEDURE create_tab ("tmp_cust","cust_num INTEGER,cust_fname CHAR(30)");
```

5.2.8 异常捕获

在移植过程中，SQL Server 的异常捕获必须按照 GBase 8s 的格式进行转换。

语法对比示例：示例

```
let v_returncode=1;
begin
    on exception in (100)
          let i=i;
    end exception;
             if (substr(i_dealdate,7,2)='01') then
                let v_sql='insert into bi_to_tsp_callnumber_bak
(tsp_code,area_code,brand_code,ca
ll_code,call_head,first_date,last_date)
      select
tsp_code,area_code,brand_code,
call_code,call_head,first_date,last_date
      from bi_to_tsp_callnumber t' ;
         execute immediate v_sql;
        if dbinfo("sqlca.sqlerrd2")=0
then
```

```
          raise exception 100;
       end if;
      end if;
end ;
let v_returncode=0;

   define sql_err int;
   define isqm_err int;
   define err_info varchar(255)
   on exception set sql_err,isam_err,err_info
        let o_return_code=-1
        let = substr('[01]' || 'p_bi_control_call_code alert' ||err_info,1,255);
        rollback;
        return o_returncode,o_returnmsg;
   end exception;
e_error exception;
   if v_flag not in('a','b') then
      raise e_error;
   end if;
   exception
   when e_error then
     rollback;
     vo_errmsg := '[p_expdb_acc_d_sms]error:'||
vo_errmsg;vo_return := '1161';

After the statement

     select … into/ insert into <tabname) select <column_list> ,

    if dbinfo( 'sqlca.sqlerrd2' )= 0  then

         raise exception 100;
      end if;

on EXCEPTION in (-268)
-- -268 means duplicate-record exception
       ROLLBACK;

       let v_errmsg = SUBSTR( 'rule:' ||v_val_name|| 'exists already!',1,500);

      let v_errcode = 1161;
           return v_errcode;
end exception;
```

5.3 特殊函数迁移

GBase 8s	SQL Server

sqlserver	GBase 8t
UNIQUEIDENTIFIER 36 位标识符	**36 位标识符**
字段使用的放括号[] 例如： [state]	去掉放括号[] 例如： state
isnull	nvl
newid() **36 位标识符**	newid() ： create dba function informix.sys_guid() returning char(32) with (PARALLELIZABLE) external name

	'c:\gbase\lib\liboracle.udr(sys_guid)'
	language C;
	grant execute on function informix.sys_guid() to public as informix;
	
	create function newid()
	returning char(36)
	define v_guid varchar(32);
	let v_guid=sys_guid();
	return lower(substr(v_guid,1,8)||'-'||substr(v_guid,9,4)||'-'||substr(v_guid,13,4)||'-'||substr(v_guid,17,4)||'-'||substr(v_guid,21,12));
	END FUNCTION

getdate	sysdate
CONVERT(VARCHAR(10),getdate(),120)	extend(sysdate,year to day)
CONVERT(VARCHAR(16),getdate(),120)	extend(sysdate,year to minute)
CONVERT(varchar(19),getdate(),120)	extend(sysdate,year to second)
CONVERT(VARCHAR(20),getdate(),120)	extend(sysdate,year to second)
CONVERT(XML 例如： select CONVERT(XML, '<v>'+REPLACE('111,222,333', ',', '</v><v>')+	select '<v>'||REPLACE('111,222,333', ',', '</v><v>')||'</v>' from sysdual;

'</v>')	
TOP 20000	first 20000
连接字符串使用 +	连接字符串使用 ||
FOR XML PATH('') 例如： CREATE VIEW [dbo].[v_SAP_Positions] AS SELECT a.*,b.[name] AS namedepartment,z.[name] AS nameroles FROM SAP_Positions a LEFT JOIN AA_Department b ON a.iddepartment=b.[id] LEFT JOIN (SELECT b1.id,LEFT([name],LEN([name])-1) AS [name] FROM (SELECT distinct id, (SELECT [name]+'、' FROM (SELECT aaa.idposition AS [id],bbb.[name] FROM SAP_Roles_Positions aaa LEFT JOIN SAP_Roles bbb ON aaa.idrole=bbb.[id]	聚集函数 strsum CREATE VIEW v_SAP_Positions AS SELECT a.*,b.name AS namedepartment,z.name AS nameroles FROM SAP_Positions a LEFT JOIN AA_Department b ON a.iddepartment=b.id LEFT JOIN (select id,rtrim(strsum(name),',') as name from(SELECT aaa.idposition as id ,bbb.name FROM SAP_Roles_Positions aaa LEFT JOIN SAP_Roles bbb ON aaa.idrole=bbb.id)a group by id order by id

) aa WHERE id=a1.id FOR XML PATH('')) AS [name] FROM (SELECT aaa.idposition AS [id],bbb.[name] FROM SAP_Roles_Positions aaa LEFT JOIN SAP_Roles bbb ON aaa.idrole=bbb.[id])a1)b1)z ON a.[id]=z.[id]	)z ON a.id=z.id;
With 递归查询: 例如： WITH CTE AS(SELECT * FROM NK_ProcurementCatalogue WHERE id=@id UNION ALL SELECT a.* FROM NK_ProcurementCatalogue a INNER JOIN CTE b ON a.idmaster=b.id) UPDATE NK_ProcurementCatalogue SET [state]='是' WHERE id IN (SELECT id FROM CTE)	**Start with** **connect by prior ：** select id from NK_ProcurementCatalogue start with id=v_id connect by prior id=idmaster into temp temp_id_idmaster2; UPDATE NK_ProcurementCatalogue SET state='是' WHERE id IN(select id from temp_id_idmaster2);

5.4 开发环境

5.4.1 语言环境

GBase 8s 可以支持许多语言、文化和代码集。所有特定于文化的信息汇集于单个环境中，称为 Global Language Support（GLS）语言环境。除了 ASCII 美国英语之外，GLS 允许您在其他语言环境中工作并在 SQL 数据和标识中使用非 ASCII 字符。可以使用 GLS 功能来与特定语言环境定制保持一致。语言环境文件包括特定于文化的信息，如货币和日期格式以及整理顺序。

GBase 8s 通过 DB_LOCALE 和 CLIENT_LOCALE 来设置数据库的语言本地化支持设置。正确设置 GLS 语言环境相关变量(DB_LOCALE, CLIENT_LOCALE)，保证 GBase 8s 数据库服务器、客户端能正确的支持中文字符和支持使用中文的对象名。DB_LOCALE 和 CLIENT_LOCALE 的值由四部分组成 (第 4 部分为可选)，字符集不区分大小写：

1	2	3	4
< 语言 > _ < 国家和地区 > . < 字符集名 / 字符集编码 > [@modifier]			

举例说明：

```
CLIENT_LOCALE=en_us.8859-1
CLIENT_LOCALE=en_us.819
# 以上两个为同一字符集：819 为 8859-1 的编码
DB_LOCALE=zh_cn.gb
```

数据库服务端

在创建数据库时（为了统一系统数据库与应用数据库的字符集，在创建数据库实例时），请按如下步骤设置数据库的 DB_LOCALE 值。

1）、设置环境变量 DB_LOCALE

set DB_LOCALE=zh_cn.gb

2）、创建数据库

create database dbname;

3）、验证当前数据库字符集

```
SELECT dbs_collate FROM sysmaster:sysdbslocale
WHERE dbs_dbsname = 'database_name'
```

客户端

1)、当我们使用 ODBC/JDBC 连接数据库时，我们需要在连接信息中正确设置语言环境变量：

DB_LOCALE 和 CLIENT_LOCALE。

2）、设置语言环境变量

DB_LOCALE=zh_cn.gb

CLIENT_LOCALE=zh_cn.gb

特别注意，在创建数据库前请设置好环境变量 DB_LOCALE 为规划的字符集，因为数据库一旦创建就不能修改其字符集，除非重新创建。

在默认情况下 GBase 8s 将使用 en_us.8859-1 字符集。

支持的中文字符集有

字符集名称
utf8
gb
gb18030-2000
big5(ILS)
Shift-Big-5(ILS)
ccdc(ILS)

5.4.2 JDBC

安装配置 JDBC

JDBC Driver 安装软件集成在 CSDK 软件安装包中，也可以下载单独的 JDBC 安装包。安装完 JDBC Driver 后，需要将文件 gbasedbtjdbc.jar 添加到环境变量 CLASSPATH 中。

设置 JDBC Driver 环境变量

```
CLASSPATH=${CLASSPATH}:${GBASEDBTDIR}/jdbc/lib/gbasedbtjdbc.jar
export CLASSPATH
```

设置 JDBC 连接字符串

在 Java 程序中使用 JDBC 连接数据库，首先应该加载使用的 JDBC 类，JDBC Driver 的类名为 com.gbasedbt.jdbc.Driver。

建立 Java 程序与 GBase 8s 数据库的连接需要使用 DriverManager.getConnection() 方法，该方法中的 URL 参数为一个数据库的连接字符串，指定数据库的连接信息。使用不同的 JDBC 所需的 URL 参数也不相同。

使用 JDBC Driver 连接数据库，连接字符串的格式如下：

```
jdbc:gbasedbt-sqli://[{ip-address|host-name}:{port-number|service-name}][/dbname
]:GBASEDBTSERVER=servername[{;user=user;password=password]
|CSM=(SSO=database_server@realm,ENC=true)} [;name=value[;name=value]...]
```

其中，“jdbc:gbasedbt-sqli”指定使用的 JDBC 为 JDBC Driver；

“{ip-address|host-name}”为数据库服务器的 IP 地址主机名；

“{port-number|service-name}”为数据库服务器监听客户端连接的端口号或服务名；

“dbname”为数据库名；

“servername”为数据库实例名。

URL 示例：

```
jdbc:gbasedbt-sqli://10.13.147.9:9088/sysmaster:gbasedbtSERVER=demoserver
jdbc:gbasedbt-sqli://9.125.66.130:6346/db18030:gbasedbtSERVER=instance_name;
NEWCODESET=gb18030,gb18030-2000,5488;DB_LOCALE=zh_cn.gb18030-2000;
CLIENT_LOCALE=zh_cn.gb18030-2000;
```

环境变量

GBase 8s 常用环境变量：

参数名	说明
GBASEDBTDIR	数据库安装路径
GBASEDBTSERVER	实例名
ONCONFIG	配置文件名称

DB_LOCALE	数据库字符集
CLIENT_LOCALE	客户端字符集
PATH	路径(必须包含$GBASEDBTDIR/bin)

附录

附录 A SQL Server 各版本功能对比

各版本新增加功能

SQL Server 2000	
日志传送	索引视图
SQL Server 2005	
分区	数据库快照
数据库镜像	复制
联机索引	故障转移群集
SQL Server 2008	
数据压缩	备份压缩
资源调控器	
SQL Server 2008 R2	
新增数据中心版，最大支持 256 核.	Unicode 压缩
SQL Server 2012	
AlwaysOn	增强的审计功能
Columnstore 索引	大数据支持
SQL Server 2014	
内存优化表	列存储索引
备份加密	缓冲池扩展 就是使用 SSD 扩展缓冲池
针对基数估计的新设计	增量统计信息
AlwaysOn 增强功能	资源调控器增强功能
延迟持续性	DBCC CHECK 支持 maxdop 提示
分区切换和索引生成	
SQL Server 2016	
全程加密技术(Always Encrypted)	Query Store
JSON 支持	支持 R 语言
多 TempDB 数据库文件	Live Query Statistics
SQL Server 2017	
可恢复的在线索引重建	图表数据库功能。用于多对多关系建模。

IDENTITY_CACHE option	Read-scale availability groups without cluster
CLR 在.NET Framework 中使用代码访问安全性（CAS）	R/PYTHON 机器学习方面的功能
SQL Server 2019	
大数据群集	数据库引擎更多功能

附录 B GBase 8s ANSI 保留字

保留关键字会随版本变化，请参考《GBase 8s SQL 指南：语法》的附录部分

A		
ABSOLUTE	ALLOCATE	ATTACH
ACCESS	ALTER	AUDIT
ACCESS_METHOD	AND	AUTHORIZATION
ADD	ANSI	AUTO
AFTER	ANY	AUTOFREE
AGGREGATE	APPEND	AVG
ALLIGNMENT	AS	AVOID_EXECUTE
ALL	ASC	AVOID_SUBQF
ALL_ROWS	AT	
B		
BEFORE	BOOLEAN	BY
BEGIN	BOTH	BYTE
BETWEEN	BUFFERED	
BINARY	BUILTIN	
C		
CACHE	CLOSE	CONNECTION
CALL	CLUSTER	CONST
CANNOTHASH	CLUSTERSIZE	CONSTRAINT
CARDINALITY	COARSE	CONSTRAINTS
CASCADE	COBOL	CONSTRUCTOR
CASE	CODESET	CONTINUE
CAST	COLLATION	COPY

CHAR	COLLECTION	COSTFUNC
CHAR_LENGTH	COLUMN	COUNT
CHARACTER	COMMIT	CRCOLS
CHARACTER_LENGTH	COMMITTED	CREATE
CHECK	COMMUTATOR	CURRENT
CLASS	CONCURRENT	CURSOR
CLIENT	CONNECT	CYCLE
D		
DATABASE	DECLARE	DIAGNOSTICS
DATAFILES	DECODE	DIRTY
DATASKIP	DEFAULT	DISABLED
DATE	DEFERRED	DISCONNECT
DATETIME	DEFERRED_PREPARE	DISTINCT
DAY	DEFINE	DISTRIBUTEBINARY
DBA	DELAY	DISTRIBUTEREFERENCES
DBDATE	DELETE	DISTRIBUTIONS
DBMONEY	DELIMITER	DOCUMENT
DBPASSWORD	DELUXE	DOMAIN
DEALLOCATE	DEREF	DONOTDISTRIBUTE
DEBUG	DESC	DORMANT
DEC	DESCRIBE	DOUBLE
DEC_T	DESCRIPTON	DROP
DECIMAL	DETACH	DTIME_T
E		
EACH	ESCAPE	EXPLAIN
ELIF	EXCEPTION	EXPLICIT
ELSE	EXCLUSIVE	EXPRESS
ENABLE	EXEC	EXPRESSION
END	EXECUTE	EXTEND
ENUM	EXECUTEANYWHERE	EXTENT
ENVIRONMENT	EXISTS	EXTERNEXTERNAL
ERROR	EXIT	

F		
FAR	FIXED	FOUND
FETCH	FLOAT	FRACTION
FILE	FLUSH	FRAGMENT
FILLFACTOR	FOR	FREE
FILTERING	FOREACH	FROM
FIRST	FOREIGN	FUNCTION
FIRST_ROWS	FORMAT	
FIXCHAR	FORTRAN	
G-H		
GENERAL	GOTO	HAVING
GET	GRANT	HIGH
GK	GROUP	HOLD
GLOBAL	HANDLESNULLS	HOUR
GO	HASH	HYBRID
I		
IF	INDICATOR	INTERNAL
IFX_INT8_T	GBASEDBT	INTERNALLENGTH
IFX_LO_CREATE_SPEC_T	INIT	INTERVAL
IFX_LO_STAT_T	INNER	INTO
IMMEDIATE	INSERT	INTERVL_T
IMPLICIT	INSTEAD	IS
IN	INT	ISCANNONICAL
INCREMENT	INT8	ISOLATION
INDEX	INTEG	ITEM
INDEXES	INTEGER	ITERATOR
J-K		
JOIN	KEEP	KEY
L		
LABELEQ	LAST	LOCALLOCATOR
LABELGE	LEADING	LOCK
LABELGLB	LEFT	LOCKS

LABELGT	LET	LOG
LABELLE	LEVEL	LONG
LABELLT	LIKE	LOW
LABELUB	LIST	LOWER
LABELTOSTRING	LISTING	VARCHAR
LANGUAGE	LOC_T	
M		
MATCHES	MEDIUM	MODERATE
MAX	MEMORY_RESIDENT	MODIFY
MAXERRORS	MIDDLE	MODULE
MAXLEN	MIN	MONEY
MAXVALUE	MINUTE	MONTH
MDY	MINVALUE	MOUNTING
MEDIAN	MODE	MULTISET
N		
NAME	NOCYCLE	NORMAL
NCHAR	NOMAXVALUE	NOT
NEGATOR	NOMIGRATE	NOTEMPLATEARG
NEW	NOMINVALUE	NULL
NEXT	NON_RESIDENT	NUMERIC
NO	NONE	NVARCHAR
NOCACHE	NOORDER	NVL
O		
OCTET_LENGTH	OPAQUE	OPTION
OF	OPCLASS	OR
OFF	OPEN	ORDER
OLD	OPERATIONAL	OUT
ON	OPTICAL	OUTER
ONLY	OPTIMIZATION	
P		
PAGE	PLI	PRIVATE
PARALLELIZABLE	PLOAD	PRIVILEGES

PARAMETER	PRECISION	PROCEDURE
PASCAL	PREPARE	PUBLIC
PASSEDBYVALUE	PREVIOUS	PUT
PDQPRIORITY	PRIMARY	
PERCALL_COST	PRIOR	
R		
RAISE	REMAINDER	RETURNS
RANGE	RENAME	REUSE
RAW	REOPTIMIZATION	REVOKE
READ	REPEATABLE	ROBIN
REAL	REPLICATION	ROLE
RECORDEND	RESERVE	ROLLBACK
REF	RESOLUTION	ROLLFORWARD
REFERENCES	RESOURCE	ROUND
REFERENCING	RESTART	ROUTINE
REGISTER	RESTRICT	ROW
REJECTFILE	RESUME	ROWID
RELATIVE	RETAIN	ROWIDS
RELEASE	RETURN	ROWS
RETURNING	ROWNUM	
S		
SAMEAS	SHARE	START
SAMPLES	SHORT	STATIC
SCHEDULE	SIGNED	STATISTICS
SCHEMA	SIZE	STDEV
SCRATCH	SKALL	STEP
SCROLL	SKINHIBIT	STOP
SECOND	SKSHOW	STORAGE
SECONDARY	SMALLFLOAT	STRATEGIES
SECTION	SMALLINT	STRING
SELCONST	SOME	STRINGTOLABEL
SELECT	SPECIFIC	STRUCT

SELFUNC	SQL	STYLE
SEQUENCE	SQLCODE	SUBSTR
SERIAL	SQLCONTEXT	SUBSTRING
SERIAL8	SQLERROR	SUM
SERIALIZABLE	SQLWARNING	SUPPORT
SERVERUUID	STABILITY	SYNC
SESSION	STACK	SYNONYM
SET	STANDARD	SYSTEM
T		
TABLE	TO	TOP
TEMP	TODAY	TRIGGERS
TEXT	TRACE	TRIM
THEN	TRAILING	TRUNCATE
TIME	TRANSACTION	TYPE
TIMEOUT	TRIGGER	TYPEDEF
U		
UNCOMMITTED	UNITS	USAGE
UNDER	UNLOCK	USE_SUBQF
UNION	UNSIGNED	USER
UNIQUE	UPDATE	USING
V		
VALUE	VARIABLE	VIEW
VALUES	VARIANCE	VIOLATIONS
VAR	VARIANT	VOID
VARCHAR	VARYING	VOLATILE
W		
WAIT	WHERE	WORK
WARNING	WHILE	WRITE
WHEN	WITH	WHENEVER
WITHOUT		
X-Y		
XLOAD	XUNLOAD	YEAR

附录 C SQL Server 到 GBase 8s 的数据类型映射

SQL Server 数据类型	GBase 8s 数据类型	精度/类型	存储长度
TINYINT	SMALLINT	-32,767 到 32,767	2
SMALLINT	SMALLINT	-32,767 到 32,767	2
INT	INT	-2,147,483,647 到 2,147,483,647	4
BIGINT	BIGINT INT8	-9,223,372,036,854,775,807 到 9,223,372,036,854,775,807	8 10
BIT	BOOLEAN	取值为't'、'f'或者 null	1
DECIMAL(p,s) NUMERIC(p,s)	DECIMAL(MIN(p,32), MIN(s,26)) NUMERIC(MIN(p,32),MIN(s,26))	在小数部分中最多 32 位有效数字，或在小数点的左边最多 32 位有效数字。	ceil((p+4)/2)
SMALLMONEY	MONEY(p,s)	在小数部分中最多 32 位有效数字，或在小数点的左边最多 32 位有效数字。	ceil((p+4)/2)
MONEY	MONEY(p,s)	在小数部分中最多 32 位有效数字，或在小数点的左边最多 32 位有效数字。	ceil((p+4)/2)
FLOAT(n)	FLOAT	14 位精度浮点数	8
REAL	SMALLFLOAT	6 位精度浮点数	4
INT IDENTITY	SERIAL	-2,147,483,647 到 2,147,483,647	4
BIGINT IDENTITY	BIGSERIAL SERIAL8	-9,223,372,036,854,775,807 到 9,223,372,036,854,775,807	8 10
CHAR(n)	CHAR(n)	定长的字符串。最大 32767 字节。	n
VARCHAR(n)	VARCHAR(n)	可变长度的字符串。最大 32765 字节。	
VARCHAR(max)	JSON	可变长度的字符串。最多 2GB 字节数据。	
TEXT	TEXT	可变长度的字符串。最多 2GB 字节数据。	
NCHAR(n)	NCHAR(n)	定长的字符串。最大 32767 字节。	
NVARCHAR(n)	NVARCHAR(n)	可变长度的字符串。最大 32765 字节。	

NVARCHAR(max)	JSON	可变长度的字符串。最多 2GB 字节数据。	
NTEXT	TEXT	可变长度的字符串。最多 2GB 字节数据。	
BINARY(n)	BSON BYTE	可变长度的二进制数据。最多 2GB 字节。	
VARBINARY(n)	BSON BYTE	可变长度的二进制数据。最多 2GB 字节。	
VARBINARY(max)	BSON BYTE	可变长度的二进制数据。最多 2GB 字节。	
IMAGE	BSON BYTE	可变长度的二进制数据。最多 2GB 字节。	
DATETIME	DATETIME YEAR TO FRACTION(5)	从 0001-01-01 至 9999-12-31 ,精度为 10 微秒	总位数/2+1
DATETIME2	DATETIME YEAR TO FRACTION(5)	从 0001-01-01 至 9999-12-31 ,精度为 10 微秒	总位数/2+1
SMALLDATETIME	DATETIME YEAR TO SECOND	从 0001-01-01 至 9999-12-31	总位数/2+1
DATE	DATE	仅存储日期。从 0001 年 1 月 1 日 到 9999 年 12 月 31 日。	4
TIME	DATETIME HOUR TO FRACTION(5)	仅存储时间。精度为 10 微秒。	
DATETIMEOFFSET	DATETIME YEAR TO FRACTION(5)	无对应，需要改造	
RWVERSION TIMESTAMP	VARCHAR(20) INT8	无对应，需要改造	
SQL_VARIANT	VARCHAR(8000)	无对应，需要改造	
UNIQUEIDENTIFIER	CHAR(36)	加 default(sys_guid())	
XML	JSON TEXT	可变长度的字符串。最多 2GB 字节数据。	

GBASE

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

微博二维码　微信二维码

技术支持热线：400-817-9696