GBASE

Oracle 到 GBase 8s
迁移技术指南

目 录

1. 概述

将数据库从 Oracle 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 Oracle 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 Oracle 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。

本文基于 GBase 8s v8.7 2.0.1a2_2 版本来介绍。

2. 概念及对比

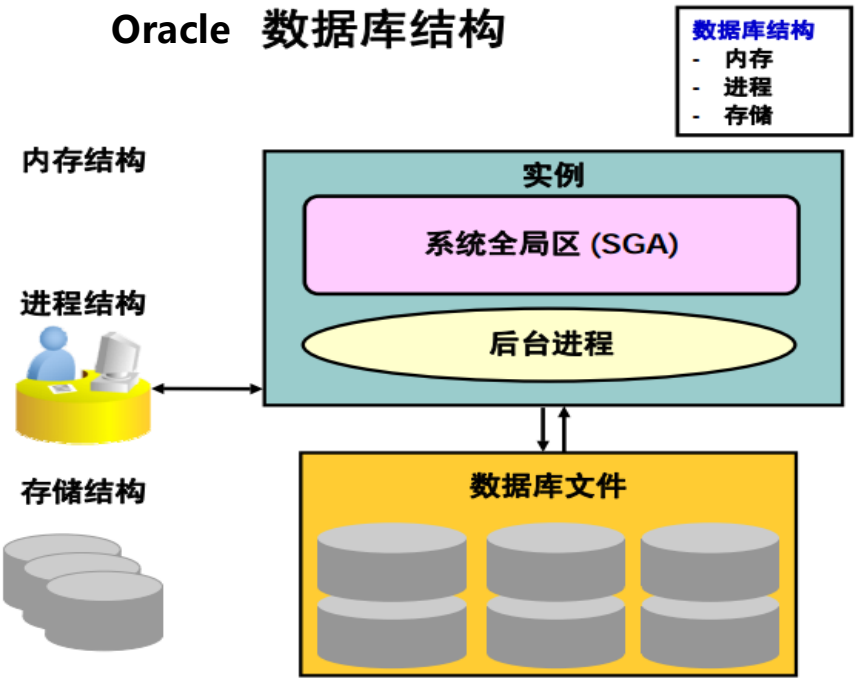

图 2-1-1

2.1 实例(Instance)

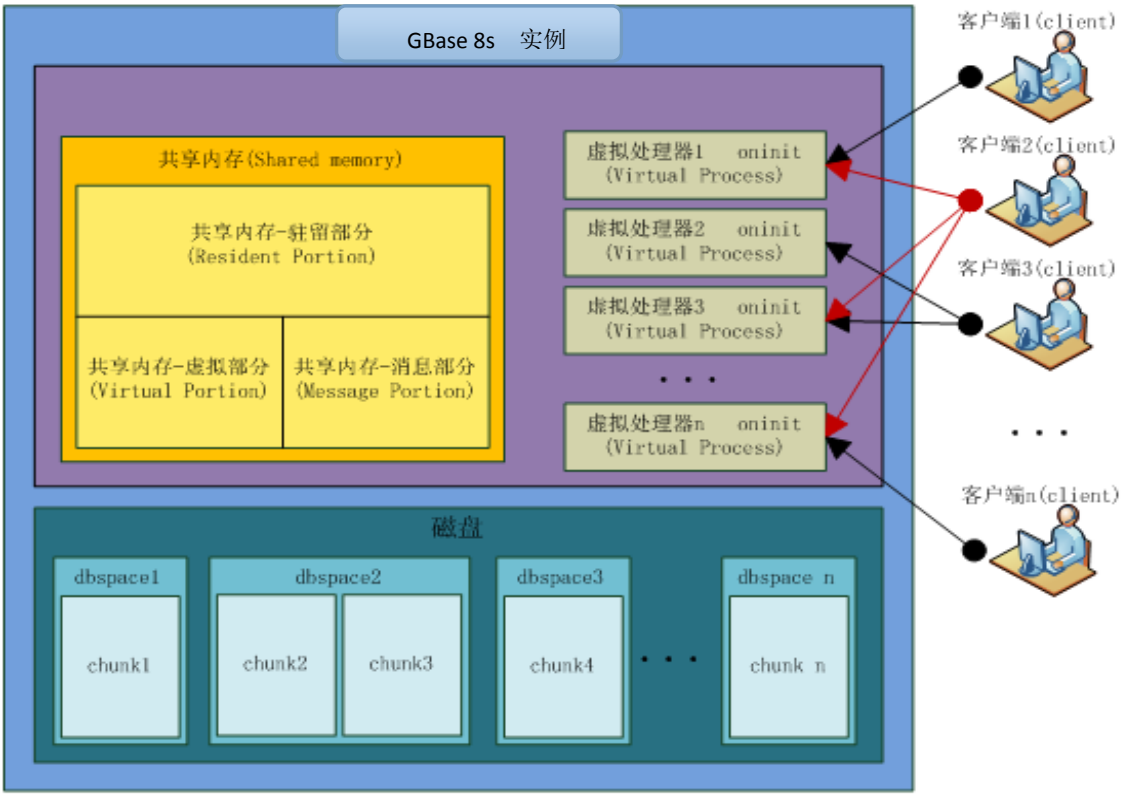

图 2-1-2

图 2-1-1,2-1-2 分别列出了 Oracle 与 GBase 8s 的数据库常规架构。

在 GBase 8s 中，实例(instance)包括共享内存，进程，存储。

Oracle 中的概念 ,数据库是一组数据文件 ,控制文件 ,日志文件的集合 ;实例(instance)是一组后台进程和内存结构，用来访问数据库。

由此可见，在 GBase 8s 中我们通常说的实例概念，相当于 Oracle 中实例和数据文件之和。

2.2 数据库(Database)

GBase 8s 中数据库(database)的用最简明的一句话来解释，是一系列包含特定用途对象的集合。在 Oracle 中没有与之完全对应的概念。如果非要类比的话，Schema 是与之最相近的概念。

2.3 存储(Storage)

GBase 8s 中存储相关的主要概念，从大到小分如下排列。 GBase 8s 与 Oracle 在存储方面的这 5 个概念高度相似，对应关系如下。

	GBase 8s	Oracle
1	数据空间(dbspace)	表空间(table space)
2	块(Chunk)	数据文件(data file)
3	表空间(table space)	段(segment)
4	区间(extent)	区间(extent)
5	页(page)	块(block)

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)，而 Oracle 中默认是行级锁。

GBase 8s 的锁结构存在共享内存中，而不是 dbspace 中。

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	Oracle	ANSI
DIRTY READ	READ ONLY	READ UNCOMMITTED
COMMITTED READ 默认	**READ COMMITTED 默认**	READ COMMITTED
REPEATABLE READ	SERIALIZABLE	REPEATABLE READ
REPEATABLE READ	SERIALIZABLE	SERIALIZABLE
CURSOR STABILITY		
COMMITTED READ LAST COMMITTED	READ COMMITTED	

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 Oracle 的 READ COMMITTED。

与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、字段明、索引名、视图名、存储过程名称等等。Oracle 最大表名长度 30 个字节，可以包括字符、数字、“_”、“$”、“#”。GBase 8s 最大表名长度 128 个字节，但不支持“#”，“$”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。

2.7 关键字(Keywords)

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

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；Oracle 默认则是在引号内外均不敏感。

例如： GBase 8s 中 gbase == GBASE == GBase=="gbase" != "GBASE" ,而在 Oracle 中，这些是完全等价的。

所以在使用 exp/imp,expdp/impdp 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。

GBase 8s 的建表语句在主体结构上是与 Oracle 相同的，但在一些特性上有不同的地方。

这里主要描述二者在使用中不同的地方。

2.9 用户(User)

Oracle 需要创建数据库用户来进行数据库权限的管理。而 GBase 8s 则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。

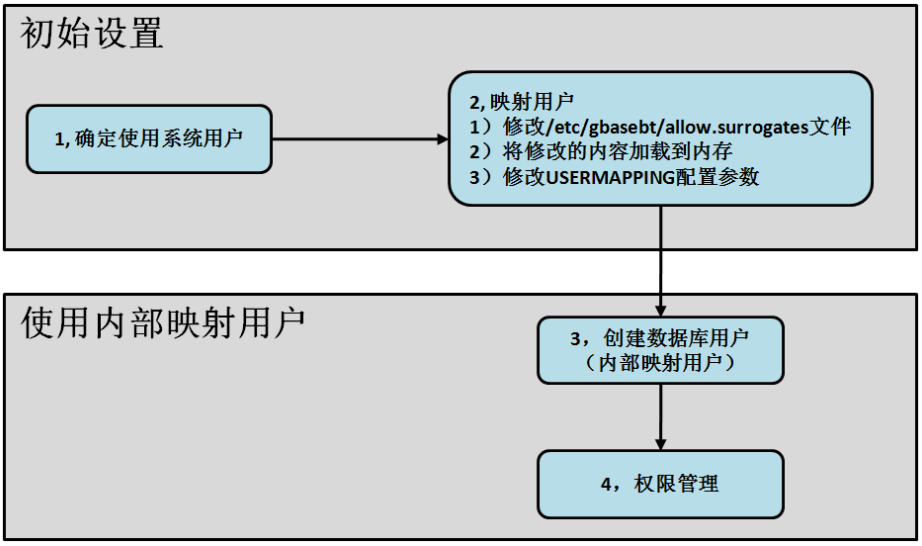

图 2-9 非系统用户访问数据库设置方法示意图

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		Oracle
SMALLINT		NUMBER/ DECIMAL/ FLOAT
size : 2 字节	范围： -32,767 to 32,767	
INTEGER/INT		
size : 4	-2,147,483,647 到 2,147,483,647	
BIGINT		
size : 8	-9,223,372,036,854,775,807 到 9,223,372,036,854,775,807	
DECIMAL (p,s)		
size :	最多 32 位有效位的浮点精度 在小数部分中最多 20 位有效数字	
NUMERIC(p,s)		
size : p/2+1	32 位精度浮点数 -10E-130 到 10E124	
FLOAT/DOUBLE PRECISION		
size : 8	与 C 语言中 double 数据类型的精度相同	

SMALLFLOAT/REAL		
size : 4	与 C 语言中 float 数据类型的精度相同	
SERIAL		
size : 2	1 到 2,147,483,647	
BIGSERIAL		
size : 8	1 到 9,223,372,036,854,775,807	
MONEY		
size : (精度+ 3/4) / 2	32 位精度浮点数 -10E-130 到 10E124	
DECIMAL(p)		
size:	最多 32 位有效位的浮点精度	

字符类型

CHAR		CHAR VARCHAR2 NCHAR NVARCHAR2
size :n	1-32765	
VARCHAR		
size :n	1-255	
LVARCHAR		
size :n	1-32739	
NCHAR		
size :n	1-32765	
NVARCHAR		
size :n	1-255	
CHARACTER VARYING		
size :n	1-255	

日期类型

DATE		DATE TIMESTAMP
size :	YEAR TO DAY	
DATETIME/TIMESTAMP		
size : 总位数/2 + 1	YEAR TO SECOND YEAR TO FRACTIONS(5)	
INTERVAL		INTERVAL
size : 总位数/2 + 1	YEAR TO FRACTIONS(5)	

大对象类型：

TEXT		LONG CLOB
size :n	1-2GB	
BYTE		LONG RAW BLOB
size :n	1-2GB	
CLOB		CLOB
size :n	1-4TB	
BLOB		BLOB

size :n	1-4TB	

其它类型：

BOOLEAN		
size :1	T 或 F，以及 null	
ROW		
size :n	自定义	

3.1.2 数据类型说明

数值类型：

Oracle NUMBER 类型兼容的数据类型

NUMERIC(p,s)：完全映射至 NUMBER(p,s)。如果 p 未指定，则默认为 38.

DECIMAL(p,s)或 DEC(p,s)：完全映射至 NUMBER(p,s)。如果 p 为指定，则默认为 38.

INTEGER 或 INT：完全映射至 NUMBER(38)类型。

SMALLINT：完全映射至 NUMBER(38)类型。

FLOAT(n)：映射至 NUMBER 类型。

DOUBLE PRECISION：映射至 NUMBER 类型。

REAL：映射至 NUMBER 类型。

字符类型

CHAR, VARCHAR 是 GBase 8s 常用的两个字符类型，CHAR 的长度都是 1-32765，VARCHAR 的长度是 1-255。

Oracle 中的 LONG VARCHAR 等超过 GBase 8s 长度上限的类型，可以使用智能大对象来代替，CLOB 或 TEXT。

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。DATE 只包含日期数据。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，最大精确到百分之一毫秒，常用的精度为 DATETIME YEAR TO SECOND。

Oracle 与之对应的数据类型是 DATE。

另外，TIMESTAMP(p)类型可直接转换成 TIMESTAMP(p)，也可以转换为 DATETIME YEAR TO FRACTIONS(min(5,p),如果 p=6 或者没有设置 数据精度会比 GBase 8s 高一位，数据迁移过程中会有微秒级的时间精度损失。

Oracle 到 GBase 8s 的具体的数据类型映射，请参照附录 B。

3.2 表

3.2.1 建表语句

GBase 8s 中建表语句与 Oracle 是非常相似的，只有一些细节需要修改。

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 }]
```

在语句中指定 DEFAULT 值时，Oracle 的字符型字段可以不加引号，但 GBase 8s 必须用引号标注。

GBase 8s	Oracle
CHAR(1) default **'0'**	CHAR(1) default 0

表的存储空间分配：

GBase 8s 中 EXTENT SIZE 和 NEXT SIZE 分别指定了表的初始大小和再次分配的大小。Oracle 与之对应的语法是 STORAGE INITIAL 和 STORAGE NEXT，而 MINEXTENTS, MAXEXTENTS 和 PCTINCREASE 在 GBase 8s 中没有对应项，需要去除。

示例：

GBase 8s	Oracle

CREATE TABLE dept (deptno SMALLINT, dname VARCHAR(14), loc VARCHAR(13)) EXTENT SIZE 200 NEXT SIZE 50;	CREATE TABLE dept(deptno NUMBER(2), dname VARCHAR2(14), loc VARCHAR2(13)) STORAGE (INITIAL 100K NEXT 50K MINEXTENTS 1 MAXEXTENTS 50 PCTINCREASE 5);

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;

Oracle 的表存储默认值在建立表空间时指定，指定的值即为存储分配的最小值。

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：轮转分片(Round-Robin)和表达式分片(Expression-Based)

轮转分片语法对比：

GBase 8s	Oracle
CREATE TABLE dept (deptno SMALLINT, dname VARCHAR(14), loc VARCHAR(13))	CREATE TABLE dept (deptno NUMBER(2), dname VARCHAR2(14), loc VARCHAR2(13)) partition by hash (deptno)

FRAGMENT BY round robin IN dbspace1, dbspace2, dbspace3, dbspace4;	partitions 4 store in (emp1,emp2,emp3,emp4);

表达式分片语法对比：

GBase 8s	Oracle
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;	CREATE TABLE dept (deptno NUMBER(2), dname VARCHAR2(14), loc VARCHAR2(13)) partition by range (deptno) (partition PART1 values less than (11) tablespace PART1_TS, partition PART2 values less than (21) tablespace PART2_TS, partition PART3 values less than (31) tablespace PART3_TS, partition PART4 values less than (MAXVALUE) tablespace PART4_TS);

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;
```

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，GBase 8s 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;
```

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 16,777,215，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：

GBase 8s	Oracle
create index zip_ix on customer (zipcode) ;	create index zip_ix on customer (zipcode) TABLESPACE idx;
create index zip_ix on customer (zipcode) in idxdbs;	create index zip_ix on customer (zipcode) TABLESPACE idx;

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 和 Oracle 同样支持索引分片。

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
```

按索引和数据相对位置的不同，索引分片可分为以下两类

索引和数据位置	GBase 8s	Oracle
相同	attached index	local index
不同	detached index	global index

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。

Oracle 的 INITRANS 和 PCTFREE 功能，与 GBase 8s 的索引 FILLFACTOR 选项非常相似。

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 建立约束的语法与 Oracle 有所不同。

GBase 8s	Oracle
PRIMARY KEY(CUST_NUM) CONSTRAINT PK_NUMBER CHECK (EMP_CODE > 100) CONSTRAINT CK_EMPCD	CONSTRAINT PK_NUMBER PRIMARY KEY(CUST_NUM) CONSTRAINT CK_EMPCD CHECK(EMP_CODE>100)

Oracle 的 PARALLEL 语句结构，在 GBase 8s 中没有与之对应的部分，需要去除。

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 ;
```

GBase 8s 建立视图的语句与 Oracle 类似，但 Oracle 中一些特有选项需要删除。包括 FORCE ,NO FORCE,READ ONLY.

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 触发器

GBase 8s 的触发器与 Oracle 的最大不同之处在于，GBase 8s 的一个触发器只能触发一个事件，而 Oracle 的一个触发器可以触发多个事件。所以在迁移过程中，Oracle 的一个触发器常会被拆分成多个 GBase 8s 触发器。

Oracle 允许 SQL 语句,逻辑语句和存储过程，在触发器中调用。GBase 8s 只允许 SQL 语句和存储过程在触发器内调用。所以：Oracle 触发器包含的逻辑语句必须转换为 GBase 8s 的存储过程，然后存储过程由触发执行，即触发存储过程。

GBase 8s 创建触发器常用语法

INSERT 触发器的语法：

```
CREATE TRIGGER trigger_name INSERT ON table_name
BEFORE [WHEN (condition)] (trig_action1,trig_action2,...)|
FOR EACH ROW [WHEN (condition)] (trig_action1,trig_action2,...)|
AFTER [WHEN (condition)] (trig_action1,trig_action2,...)|
[DISABLED|ENABLED]
```

DELET 触发器的语法：

```
CREATE TRIGGER trigger_name DELETE ON table_name
BEFORE [WHEN (condition)] (trig_action1,trig_action2,...)|
FOR EACH ROW [WHEN (condition)] (trig_action1,trig_action2,...)|
AFTER [WHEN (condition)] (trig_action1,trig_action2,...)|
[DISABLED|ENABLED]
```

UPDATE 触发器的语法:

```
CREATE TRIGGER trigger_name UPDATE [OF (column,column,...)] ON table_name
BEFORE [WHEN (condition)] (trig_action1,trig_action2,...)|
FOR EACH ROW [WHEN (condition)] (trig_action1,trig_action2,...)|
AFTER [WHEN (condition)] (trig_action1,trig_action2,...)|
[DISABLED|ENABLED]
```

3.7 序列

GBase 8s 中的序列与 Oracle 的序列在功能上是相同的。

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。

3.8 导出数据库结构

从 Oracle 数据库抽取表结构，可以使用 Oracle 原厂工具 exp/imp 或 expdp/impdp, 这里推荐使用后者，更新功能更强大。

示例：

```
SQL>connect / as sysdba;
SQL>CREATE OR REPLACE DIRECTORY dumpDir AS 'c: \temp';
SQL>GRANT read, write ON DIRECTORY dumpDir TO userID;
#卸载一个 SCHEMA：
   expdp ' userName/userPwd as sysdba' SCHEMAS=hr DIRECTORY=dumpDir
   DUMPFILE=hr.dmp LOGFILE=hr .log CONTENT=METADATA_ONLY
#从 DUMP 文件抽取生成 SQL 语句:
   impdp ' userName/userPwd as sysdba' DIRECTORY=dumpDir
   DUMPFILE=hr.dmp SQLFILE=dumpDir: hr.sql
```

生成的 SQL 文件是类似于下面这样的

CREATE TABLE "HR"."COUNTRIES"...

在上文中提到过，GBase 8s 在引号内的大小写是敏感的，所以这里需要使用文本处理工具将这些 SQL 语句按照 GBase 8s 支持的格式进行修改。

Oracle 以文本方式导出数据库结构很繁琐，GBase 8s 提供了非常简易的工具来完成此类操作。只需要执行命令 dbschema -d database_name 即可完成导出。

3.9 导入数据库结构

准备工作：

1. 导入数据库结构前，必须按照 GBase 8s 语法格式修改创建对象脚本。

2. 创建数据库 database_name

导入数据库结构：

执行命令 dbaccess -e -m -a database_name schema_file.sql 来将修改好的脚本文件执行，将数据库对象建立。

为了加快数据导入 ,建议索引和主外键约束等在这里先不要建立 ,等待数据全部导入后，再开 PDQ 一并建立。

4.数据迁移

4.1. 迁移过程示意

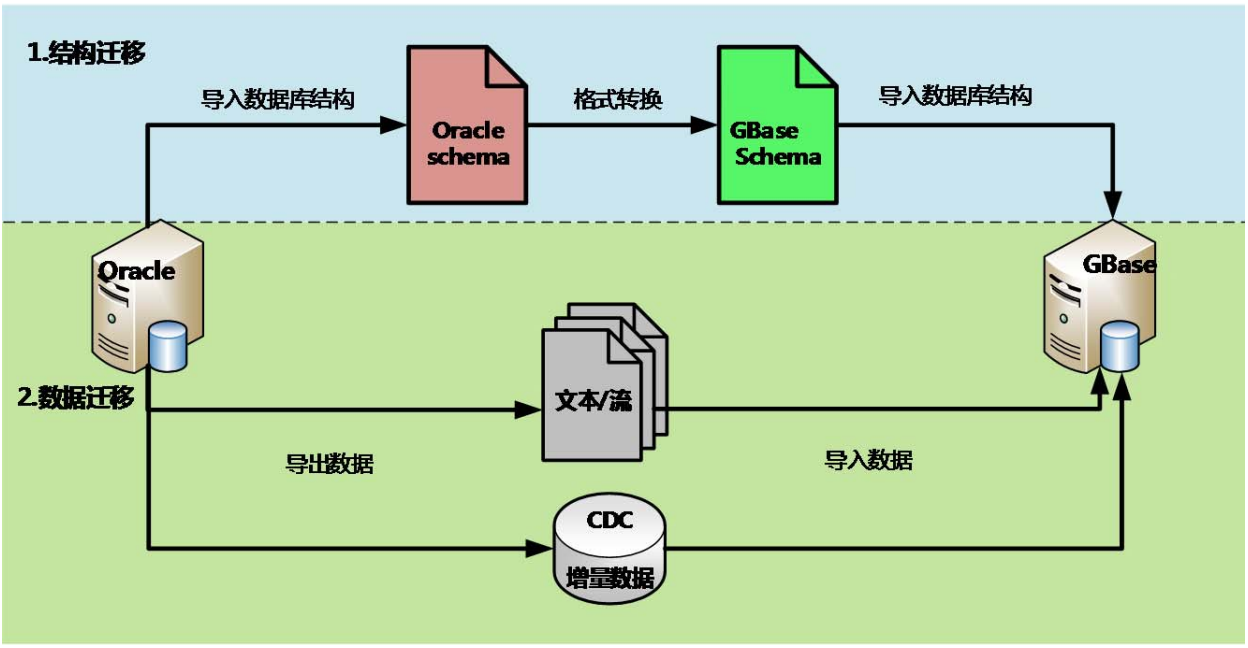

图 4-1

如图 4-1，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。

其中，第一部分我们在上一章节的 3.8、3.9 已经详细讲解。

真正的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。

在对时间窗口要求比较小的应用做迁移时 ,可以使用第三方工具(如 CDC)来进行增量数据的迁移，这样可以最大限度的减少停机时间，甚至可以实现 Oracle 到 GBase 8s 无缝迁移。

4.2. 文件格式

把数据从 Oracle 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

Oracle 的数据可以通过 SQLPlus 的功能，导出符合 GBase 8s 要求的文本格式。

以下是 SQLPlus 导出数据时的一个示例

```
   set colsep '|'
   set head off
   set echo off
   set headsep off
   set newp none
   set long 2000000000
```

```
set longchunksize 32767
set linesize 32767
set numw 32
set pagesize 0
set sqlblanklines off
set trimspool on
set termout off
set feedback off
set numw 32
set verify off
alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';
alter session set nls_timestamp_format='yyyy-mm-dd hh24:mi:ss.ff5';
spool c:\customer.unl
select customer_id,Birth_Date,Address||'|' from customer;
spool off
exit
```

可以编写脚本来并发执行以上操作来提升数据从 Oracle 导出的效率。

导出的数据以 GBase 8s 导入要求的格式存储在 c:\customer.unl 中。

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 校验

对数据行数进行校验

Oracle 数据库中各表的行数统计：

```
exec dbms_stats.gather_schema_stats (ownname=>'owner', options=>'gather auto' ,
estimate_percent=>dbms_stats.auto_sample_size);
select table_name,num_rows from dba_tables where owner='OWNER' order by 2 desc;
```

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 Oracle 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 UNITS,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 A。

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”

“<>”

“!=”

但 Oracle 中的“^=”在 GBase 8s 中并不支持，需要转换。

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 Oracle 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 Oracle 语法结构不同的是，需要去掉 VALUES; Oracle 中 INSERT…SELECT…UNION 在 GBase 8s 中也是不支持的。

5.1.6 临时表

GBase 8s 在使用临时表上与 Oracle 有着很大差异。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。Oracle 中的临时表则被用作批量导入数据时的工作区。

GBase 8s 中允许不同会话中创建同名临时表，而这在 Oracle 中是不允许的，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 的 NULL 在排序中被作为最小值来对待，而在 Oracle 中， NULL 在排序中被作为最大值。这个差异在多数应用中会造成影响，一般在 order by COLUMN_NAME nulls last 语句中指定 NULL 值在最后即可。但在个别应用中，则需要写一个存储过程来判断是否为 NULL,如果为空，则用一个很大的值来替换，以便排序后可以排在最大的位置。应用端也需要相应的调整，一旦需要显示这个字段，而要显示的这个值为先前设定的很大的值，则要再替换回 NULL 来显示。

5.1.8 别名

GBase 8s 在删除语句中不支持对主表指定别名，例如：

GBase 8s	Oracle
UPDATE customer SET zip_code = '92612' WHERE zip_code IN (SELECT zip_code FROM zip_table Z WHERE customer.create_date <= Z.effective_date)	UPDATE customer C SET zip_code = '92612' WHERE zip_code IN (SELECT zip_code FROM zip_table Z WHERE C.create_date <= Z.effective_date)

GBase 8s	Oracle
DELETE customer WHERE order_date <= (SELECT control_value FROM control_table T WHERE T.control_field = 'archive_date' AND customer.state = T.state)	DELETE customer C WHERE order_date <= (SELECT control_value FROM control_table T WHERE T.control_field = 'archive_date' AND C.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，原表名必须被别名完全取代。如果 Oracle 应用中的 SQL 是原名和别名混搭的，就必须按照 GBase 8s 的格式来替换。

5.1.9 级联查询(Hierarchical queries)

GBase 8s 与 Oracle 一样，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。

5.1.10 TRUNCATE

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];

与 Oracle 一个很小的差别是 TABLE 关键字是可以被省略的。

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。

Oracle 不支持 TRUNCATE 操作的回滚。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 Oracle 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 Oracle 应用中的表与 GBase 8s 系统表重名，则必须在迁移时更换表名。

DUAL（空表）：GBase 8s 中的 DUAL 是与 Oracle 的 DUAL 表作用相同的类表结构。

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, LOOP

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。Oracle 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 Packages

Oracle 中的 Package 是一系列方法、过程、变量的集合。GBase 8s 中没有与之直接对应的概念，但是有一些方法可以实现近似的功能。

在 Oracle 中,程序可以通过语法 package_name.procedure_name 访问一个包。如果在 GBase 8s 创建一个用户或 SCHEMA 与 Oracle 的 Package 名一样,那么这将是更容易迁移过程的调用。因为语法实际上是一样的,虽然意思有所改变。使用这种做法,调用一个迁移过程将是 owner.procedure_name（owner 将和 Package 的名字是一样的）。将 Package 名字转换为 owner 的名字是一个维持 Package 的方法。

5.2.5 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 Oracle 中的一些类型 ,比如 DATE 或 NUMBERIC 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。

5.2.6 游标

Oracle 中，游标可以在全局包中。GBase 8s 中，提供了兼容 sys_refcursor。也可以使用之前的方式：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 Oracle 存储过程中，游标是显式地声明、打开和获取，Oracle 存储过程游标可以使用 GBase 8s 的 sys_refcursor 代替。也可以使用之前的方式 :FOREACH 结构替换。Oracle 游标名应当在 FOREACH 语句中使用,例如:FOREACH cursor_name SELECT…END FOREACH。

为了更新 Oracle 游标行,游标必须声明 FOR UPDATE 子句。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。

5.2.7 Routine

Oracle 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。Oracle 用户定义的数据库级别的村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。

5.2.8 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)");
```

5.2.9 异常捕获

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

语法对比示例：

GBase 8s:	Oracle

let v_returncode=1;
begin
on exception in (100)
let i=i;
	o_returncode=:1;
	begin
	if (substr(i_dealdate,7,2)='21')
	then

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;
	v_sql:='insert into
	jtfx_to_tsp_callnumber_bak
	(tsp_code,area_code,brand_code)
	select
	tsp_code,area_code,brand_code
	from jtfx_to_tsp_callnumber t' ;
	execute immediate v_sql;
	commit;
	end if;
	exception when NO_DATA_FOUND
	o_returncode := 1;
	rollback;
	end ;
	o_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;	exception when others then o_returncode := -1; o_returnmsg := substr('[01]' || 'p_bi_control_call_code alert' ||sqlerrm,1,255); rollback;

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';
	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	EXCEPTION WHEN NO_DATA_FOUND THEN o_returncode := 1; o_returnmsg := SUBSTR('[01]' ||

raise exception 100; end if;	‘ no ’ || v_date || ‘ data ' || SQLERRM, 1, 255); ROLLBACK;

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;
	e_unique EXCEPTION;
	WHEN e_unique THEN
	RAISE DUP_VAL_ON_INDEX;
	END;
	
	WHEN DUP_VAL_ON_INDEX THEN
	ROLLBACK;
	v_errmsg :=
	SUBSTR(‘rule: ’ ||v_val_name|| ‘ exists already!',1,500);
	v_errcode := 1161;

5.2.10 语法对照示例

GOTO

GBase 8s	Oracle

define x integer;
Let x = 0;
BEGIN
<<increment_x>>
BEGIN
LET x = x + 1;
END;
 	
IF x < 10 THEN
GOTO increment_x;
END IF;
END;
	DECLARE
	x NUMBER := 0;
	BEGIN
	<<increment_x>>
	BEGIN
	x := x + 1;
	END;
	
	IF x < 10 THEN
	GOTO increment_x;
	END IF;
	END;

GOTO 语句和标签都不能用在 EXCEPTION 部分；

在一个存储过程中标签必须是唯一的。

循环

LOOP/END LOOP

GBase 8s	Oracle

LOOP
IF credit_rating IS NULL THEN
CONTINUE;
END IF
	LOOP
	IF credit_rating IS NULL THEN
	CONTINUE;
	END IF;

IF credit_rating < 3 THEN
EXIT; -- exit loop immediately
END IF
END LOOP;
	IF credit_rating < 3 THEN
	EXIT ; -- exit loop immediately
	END IF;
	END LOOP;

本例中，也可以使用”CONTINUE LOOP”和“EXIT LOOP” ；

可以在传统的循环语法中使用“CONTINUE WHILE”， “EXIT WHILE”， “CONTINUE FOR”， “EXIT FOR”来结束相应的循环。

GBase 8s	Oracle

WHILE (i < 10) LOOP
LET i = i + 1 ;
IF i < 2 THEN
CONTINUE;
END IF
IF i > 5 THEN
EXIT;
END IF
END LOOP;
	WHILE (i < 10)LOOP
	i := i + 1 ;
	IF i < 2 THEN
	CONTINUE;
	END IF;
	IF i > 5 THEN
	EXIT ;
	END IF;
	END LOOP;

旧版语法 WHILE/END WHILE ;

新版语法 WHILE … LOOP/END LOOP

FOR

GBase 8s	Oracle

FOR i IN (1 TO 5) LOOP
LET i = i +1 ;
IF i < 2 THEN
CONTINUE;
IF i > 5 THEN
EXIT;
END IF;
END LOOP;
	FOR i IN 1..5 LOOP
	i := i +1 ;
	IF i < 2 THEN
	CONTINUE;
	IF i > 5 THEN
	EXIT ;
	END IF;
	END LOOP;

旧版语法 FOR/END FOR ;

新版语法 FOR … LOOP/END LOOP

WHEN

GBase 8s	Oracle
<<outer>> LOOP LET x = x+1; <<inner>>	<<outer>> LOOP x:= x+1; <<inner>>

WHILE (i >10) LOOP
LET x = x+1;
EXIT inner WHEN x = 2;
EXIT outer WHEN x > 3;
END LOOP inner;
LET x = x+1;
END LOOP outer;
	WHILE (i >10) LOOP
	x:=x+1;
	EXIT inner WHEN x > 3;
	EXIT outer WHEN x > 3;
	END LOOP inner;
	x:= x+1;
	END LOOP outer;

5.3 函数

5.3.1 数值类

名称	描述
ceil(number)	查找大于等于 number 值的整数。
floor(number)	查找小于等于 number 值的整数。
power(number, power)	与 GBase 8s 的 pow()相同。

5.3.2 字符类

LENGTH，Oracle 的 LENGTH 函数 GBase 8s 的 LENGTH 函数在 CHAR 类型的输出上有不同之处。Oracle 函数返回的数值包括字符末尾的空格，但 GBase 8s 函数返回值不包括空格。

ltrim(string1 [, string2]) 从前端截断 string1。

示例：

```
SELECT LTRIM(' Hellohello world!', ' Hello') FROM mytab;
输出 hello world!
```

rtrim(string1 [, string2]) 从后端截断 string1。

示例：

```
SELECT RTRIM(' good night... *!#?theend ', ' theend!*#?') AS closing FROM mytab;
输出 good night...
```

Oracle 特有函数

一些 Oracle 特有的字符类函数，GBase 8s 并不支持，如 SOUNDEX, TRANSLATE.

5.3.3 日期类

Round / Trunc

round(date/datetime [, fmt])

trunc(date/datetime [,fmt])

Fmt:
YEAR
MONTH
DD (day of the month)
DAY (day of the week)
HH (hour)
MI (minute)
extent('2014-2' ,year to second)+12*interval(1) hour to hour

Round	Trunc
datetime year to fraction(5) column, col_dt: 2014-12-07 14:30:12.12300 > select round(col_dt, 'YEAR') from mytab (expression) 2015-01-01 00:00 > select round(col_dt, 'MI') from mytab (expression) 2014-12-07 14:30	**datetime year to fraction(5) column, col_dt: 2014-12-07 14:30:12.12300** > select trunc(col_dt, 'YEAR') from mytab (expression) 2014-01-01 00:00 > select trunc(col_dt, 'MI') from mytab (expression) 2014-12-07 14:30

add_months(date/datetime, integer) 返回 date/datetime 类型

为 date 或 datetime 的数据类型以月为单位增加。

示例

```
select dcol, add_months(dcol, 1)
as add_one_month from date_tab;
dcol     add_one_month
02/28/2015 03/28/2015
02/29/2016 03/29/2016
01/29/2015 02/28/2015
03/31/2015 04/30/2015
```

last_day(date/datetime) 返回 date/datetime

返回当月的最后一天相的应值。

```
select today as today, last_day(today) as last,
last_day(today) - today as days_left from systables where tabid = 1;
today          last           days_left
03/12/2015 03/31/2015              19
```

next_day(date/datetime, char(3)) 返回 date/datetime

返回从参数 1 开始，参数 2 指定的星期 x，date/datetime 类型。

示例

```
select ship_date, next_day(ship_date, 'SAT') as next_saturday,
next_day(ship_date, 'SAT') - ship_date as num_days from orders;
ship_date    next_saturday      num_days
06/01/2014 06/03/2014                2
02/12/2015 02/17/2015                5
```

months_between(date/datetime, date/datetime)　返回两个日期间的月数。

示例

```
select col_datetime, col_date, months_between(col_datetime, col_date) as months_between from mytab2;
col_datetime      2015-12-13 08:40:30.00000
col_date          11/13/2014
months_between    13.0000000000000
```

SYSDATE

GBase 8s 支持 SYSDATE 来返回一个系统当前时间，默认格式为 YEAR TO FRACTION(5),另一个时间常用关键字 CURRENT 的默认格式为 YEAR TO FRACTION(3),SYSDATE 可以看做是 CURRENT 的一个同义词。

5.3.4 位操作类

GBase 8s 支持以下二进制函数

bitand(arg1, arg2)

bitor(arg1, arg2)

bitxor(arg1, arg2)

bitnot(arg1)

bitandnot(arg1, arg2)

以上函数中 Oracle 仅支持 bitand。

arg1, arg2 可以使任意的数字类型值，使用时会被转化成 64 位整形值。

小数数值在转换时会被自动截断。

bitnot 的参数最大值为 9,223,372,036,854,775,806。

5.3.5 特殊函数

DBINFO()

DBINFO 实际上是一组函数,返回不同类型的数据库的相关信息。在参数位置指定一个特定的选项，就可以调用相应的函数功能。可以在 SQL 语句和 UDR 中使用 DBINFO 选项。

参数	返回值说明
('dbhostname')	应用程序连接的数据库服务器的主机名

('dbname')	应用程序连接的数据库名称
('dbspace' *tblspace_num*)	tblspace number 对应的 dbspace 名
('get_tz')	时区
('serial8')	插入表中的最后一个 SERIAL8 值
('bigserial')	插入表中的最后一个 BIGSERIAL 值
('sessionid')	当前会话的会话号
('cdrsession')	线程是否执行了企业级复制(ER)操作
('sqlca.sqlerrd1')	插入表中的最后一个 SERIAL 值
('sqlca.sqlerrd2')	已经被 SELECT, INSERT, DELETE, UPDATE, EXECUTE PROCEDURE, 和 EXECUTE FUNCTION 语句计算过的行数
('utc_current')	SQL 执行时刻的 UTC 时间值(从 1970-01-01 00:00:00+00:00 计算的整数秒)
('utc_to_datetime', *table.column*)	将指定字段存放的整数值当做 UTC 时间值转换成 DATETIM 值
('utc_to_datetime', *utc_value*)	将 UTC 时间值转换成 DATETIM 值
('version', '*parameter*')	应用程序所连接的数据库服务器类型和它的版本号

5.3.6 兼容的函数

时间日期类：

ROUND, TRUNC, ADD_MONTHS, LAST_DAY, NEXT_DAY,

MONTHS_BETWEEN, SYSDATE

转换类：

ASCII, TO_CHAR, TO_NUMBER

字符操纵类：

LTRIM, RTRIM

数字操作类：

ROUND, TRUNC, CEIL, FLOOR, POWER

位操纵类：

BITAND, BITOR, BITXOR, BITNOT, BITANDNOT

其它类：

NULLIF, FORMAT_UNITS

GBase 8s 支持的函数列表及映射关系，见附录 C。

5.4 嵌入式 SQL

5.4.1 Oracle Call Interface (OCI)

对应使用 OCI 接口开发的应用程序，在 GBase 8s 中没有与之对等的接口。以下信息列出了 GBase 8s 与 Oracle 在此类接口上的不同点。

	Oracle	GBase 8s
缓冲区	HAD, LDA, CDA	HENV, HDBC, HSTMT
连接和分配资源	OLOG, OOPEN	SQLConnect, SQLAllocEnv, SQLAllocConnect, SQLAllocStmt
Update,Delete, Insert, Select	OPARSE, OBNDRV, OEXEC	SQLPrepare, SQLBindParameter, SQLExecute
提交、回滚	OCOM, OROL	SQLTransact
断开连接	OLOGOF, OCLOSE	SQLDisconnect, SQLFreeStmt, SQLFreeConnect, SQLFreeEnv

数据库库调用

Oracle OCI 与 GBase 8s GCI 最主要的不同在于总体数据结构和处理数据库库调用机制上。Oracle 直接连接数据库服务器，而 GBase 8s 使用 ODBC 连接数据库。

Global area processing

另一个主要区别在于 GBase 8s 使用指针访问三个全局区域，Oracle 使用指针访问三个不同但相似的全局区域。因此，Oracle OCI 使用 HAD (Handle Data Area), LDA (Logon Data Area)和 CDA (Cursor Data Area)指针的调用，必须被转换成 GBase 8s GCI 使用 HENV(Environment Area), HDBC (Database Connection Area) and HSTMT (Statement Area)指针的调用。

Fetch 循环

Fetch 循环的语法不同，需要做相应的修改。在 Oracle 中，选取多行的 Fetch 循环使用包含 parsing, binding, defining, fetching 的游标执行到一个游标数据区中。这必须使用包含 preparing,binding, executing, binding 通过 For Each 循环来读取。

RAW 二进制数据类型的计算

Oracle 中的 RAW 数据类型需要转换为 GBase 8s 的 BLOB 数据类型

参数绑定

在 Oracle 中 ,多元输出绑定参数通过类似于 ODEFIN()调用中的 OUTPUT 类型来声明；在 GBase 8s 中，将其转换成 SQLBindCol()并且绑定输出参数到字段。

在输入输出绑定中也存在类似的不同。Oracle 中，这些绑定以 INPUT/OUTPUT 的类型在 ODEFIN()中声明 ;在 GBase 8s 中 ,在三个变量中使用轮询的方法 :A=B, B=C, C=A 。变量 A 在 SQL 使用绑定变量前被声明，方法被调用时则使用变量 B，方法以输出字段来返回变量 C。这样使变量 A 在其余程序中表现为一个输入输出变量。

5.4.2 C 语言的嵌入式 SQL

VARCHAR

Oracle Pro*C 的 VARCHAR 数据类型相关的用法有一些不同。使用 variable_name.len 的语句需要被移除；使用 variable_name.arr 的地方可以被 variable_name 替换。例如：

Oracle	GBase 8s
VARCHAR vName[40]; strcpy(vName.arr, “Company Name”); vName.len = strlen(vName.arr);	VARCHAR vName[40]; strcpy(vName, “Company Name”);

宿主变量

Oracle 的语句 EXEC SQL TYPE 与 C 语言的声明语法 typedef 是同义词，需要替换

成 GBase 8s 的声明语法 typedef。

5.5 开发环境

5.5.1 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

4.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。

5.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)

5.5.2 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 ifxjdbc.jar 添加到环境变量 CLASSPATH 中。

设置 JDBC Driver 环境变量

CLASSPATH=${CLASSPATH}:${GBASEDBTDIR}/jdbc/lib/ifxjdbc.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.IfxDriver。

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 GBase 8s ANSI 保留字

红色字体的同时也是 Oracle 的保留字

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		
XLOAD	XUNLOAD	YEAR

附录 B Oracle 到 GBase 8s 的数据类型映射

Oracle 数据类型	GBase 8s 数据类型	精度/范围	存储长度(字节)
BFILE LONG RAW BLOB	BLOB		

BINARY_DOUBLE	DOUBLE PRECISION	14 位精度浮点数	8
BINARY_FLOAT	DOUBLE PRECISION	14 位精度浮点数	8
BINARY_INTEGER PLS_INTEGER NATURAL NATURALN POSITIVE POSITIVEN	INTEGER	-2,147,483,647 到 2,147,483,647	4
BOOLEAN (PL/SQL)	BOOLEAN	TRUE 或 FALSE 或者 NULL	1
CHAR(n)	CHAR(n)	1≤n≤32,767	n
CHARACTER(n)	CHARACTER(n)	1≤n≤32,767	n
DATE	DATETIME YEAR TO SECOND	YEAR TO FRACTION(5)	总位数/2+1
DOUBLE PRECISION	DOUBLE PRECISION	14 位精度浮点数	4
FLOAT	FLOAT	14 位精度浮点数	8
INTEGER/INT	INTEGER	-2,147,483,647 到 2,147,483,647	4
INTERVAL DAY(p) TO SECOND(s)	INTERVAL DAY(p) TO FRACTION(min(5,s))		
INTERVAL YEAR(p) TO MONTH	INTERVAL YEAR(p) TO MONTH		
LONG CLOB	CLOB	最大 4T 字节	
LONG VARCHAR (xxx)	VARCHAR	1 到 32,767 字节	
NCHAR(n) NATIONAL CHAR(n) NATIONAL CHARACTER(n)	NCHAR(n)	1 到 32,767 字节	
NCLOB	CLOB	最大 4T 字节	
NUMBER NUMBER(*,0) NUMBER(p) NUMBER(p,s) (s<0)	INTEGER	-2,147,483,647 到 2,147,483,647	4
NUMBER(p) NUMBER (p,0) 10 <= p <= 18 NUMBER(p, s) 10 <= p-s < 19.	BIGINT	–9,223,372,036,854,775,807 到 9,223,372,036,854,775,807	
NUMBER(p, s) s > 0 and p >= s	DECIMAL (min(p,32), min(s,32))		p/2+1
NUMBER(p, s) (s > 0 且 p < s)	DECIMAL (min(s,32),min(s,32))		

NUMBER(p, s) (s<0 且 18<p-s<32)	DECIMAL (min(p-s,32),0)		
NUMBER(p, s) s>32, 或 p-s>32	DECIMAL(32) DECIMAL(32,32)		
DEC DECIMAL NUMERIC DEC(p) DECIMAL(p) NUMERIC(p) DEC(p, s) DECIMAL(p, s) NUMERIC(p,s)	与 Oracle 定义相同，无需更改		
NVARCHAR2(n) NCHAR VARYING(n) NATIONAL CHAR VARYING(n) NATIONAL CHARACTER VARYING(n)	VARCHAR(min(n, 32767))		
RAW(n)	BLOB	最大 4T 字节	
REAL	DOUBLE PRECISION	14 位精度浮点数	4
Record	自定义数据类型 CREATE ROWTYPE		
ROWID	INTEGER	-2,147,483,647 到 2,147,483,647	4
SMALLINT	SMALLINT	–32,767 到 32,767	2
TIMESTAMP(p) 参数 p 不指定时为默认值 6	TIMESTAMP(p) DATETIME YEAR TO FRACTION(min(5, p))		总位数/2+1
UROWID(n)	INTEGER	-2,147,483,647 到 2,147,483,647	4
VARCHAR2(n) VARCHAR(n) CHAR VARYING(n) CHARACTER VARYING(n)2 (n<=255)	VARCHAR(n)	1-255 字节	

VARCHAR2(n) VARCHAR(n) CHAR VARYING(n) CHARACTER VARYING(n)2 (n>255)	LVARCHAR(n)	255-32767 字节	
VARCHAR2(n char)	按照 VARCHAR2(2n)参考以上转换成对应类型		
XML	LVARCHAR(n) 1<=n<=32,767 CLOB	最大 4T 字节	

附录 C GBase 8s 函数列表及 Oracle 函数映射

ABS	DECRYPT_CHAR	NVL2
ACOS	ENCRYPT_AES	OCTET_LENGTH
ACOSH	ENCRYPT_TDES	POW
ADD_MONTHS	EXP	QUARTER
ASCII	EXTEND	RANGE
ASIN	FLOOR	REPLACE
ASINH	FORMAT_UNITS	ROOT
ATAN	GETHINT	ROUND
ATAN2	HEX	RPAD
ATANH	INITCAP	RTRIM
AVG	LAST_DAY	SIN
BITAND	LENGTH	SQLCODE(SPL)
BITANDNOT	LOCOPY	SQRT
BITNOT	LOG10	STDEV
BITOR	LOGN	SUM
BITXOR	LOTOFILE	TAN
CARDINALITY	LOWER	TANH
CEIL	LPAD	TO_CHAR
CHAR_LENGTH	LTRIM	TO_DATE
CHR	MAX	TO_NUMBER
CONCAT	MDY	TRIM

COS	MIN	TRUNC
COUNT	MOD	UPPER
DATE	MONTH	VARIANCE
DAY	NEXT_DAY	WEEKDAY
DBINFO	NULLIF	YEAR
DECODE	NVL	

Oracle 到 GBase 8s 的函数映射表

数值类

Oracle	GBase 8s
ABS	ABS
ACOS	ACOS
ASIN	ASIN
ATAN	ATAN
ATAN2	ATAN2
BITAND	BITAND
CEIL	CEIL
COS	COS
COSH	N/A
EXP	EXP
FLOOR	FLOOR
LN	LN
LOG	LOG
LOG(10,n1)	LOG10(n1)
MOD	MOD
NANVL	N/A
POWER	POWER
REMAINDER	N/A 可通过存储过程实现
ROUND(arg1,arg2)	ROUND(arg1,arg2)
SIN	SIN
SINH	N/A 可通过存储过程实现
SQRT	SQRT
TAN	TAN
TANH	N/A 可通过 C UDR 实现
TRUNC(n[,m])	TRUNC(n[,m])
TO_NUMBER	TO_NUMBER

字符类

Oracle	GBase 8s
ASCII	ASCII
CHR(n)	N/A 可通过 C UDR 实现
CONCAT	CONCAT

INITCAP	INITCAP
INSTR	INSTR
INSTRB INSTRC INSTR2 INSTR4	N/A 可通过 C UDR 实现
LENGTH	LENGTH
LENGTHB LENGTHC LENGTH2 LENGTH4	N/A
LOWER	LOWER
LPAD(arg1,arg2,arg3)	LPAD
LTRIM	LTRIM
REPLACE(arg1,arg2,arg3)	REPLACE(arg1,arg2,arg3)
RPAD	RPAD
RTRIM	RTRIM
SOUNDEX	N/A
SUBSTR	SUBSTR SUBSTRING
SUBSTRB SUBSTRC SUBSTR2 SUBSTR4	N/A
TRANSLATE	N/A
TREAT	通过 CAST 来实现
TRIM	TRIM
UPPER	UPPER
TO_CHAR	TO_CHAR

时间日期类

Oracle	GBase 8s
ADD_MONTHS	ADD_MONTHS
CURRENT_DATE	TODAY
CURRENT_TIME STAMP	CURRENT
DBTIMEZONE	N/A
EXTRACT(datetime)	YEAR() MONTH() DAY()
FROM_TZ	N/A
LAST_DAY	LAST_DAY
MONTHS_BETWEEN(arg1,arg2)	MONTHS_BETWEEN
NEW_TIME(date,arg1,arg2)	N/A 可通过 C-UDR 实现
NEXT_DAY(arg1,arg2)	NEXT_DAY

NUMTOSDINTERVAL NUMTOYMINTERVAL	通过 CAST 来实现
ROUND(date,fmt)	N/A 可通过 UDR 实现
SESSIONTIMEZONE	DBINFO（‘get_tz’）
SYS_EXTRACT_UTC	通过 CURRENT TIMESTAMP - CURRENT TIMEZONE 来实现
SYSDATE	SYSDATE
SYSTIMESTAMP	CURRENT
TRUNC(arg1,arg2)	N/A 可通过 UDF 实现
TZ_OFFSET	N/A

空值相关

Oracle	GBase 8s
COALESCE	N/A 可通过 C UDR 或存储过程实现
LNNVL	N/A
NULLIF	NULLIF
NVL	NVL
NVL2(arg1,arg2,arg3)	通过 NVL 实现

其它（加密解密）

Oracle	GBase 8s
DECODE(expr,srch_val1, result1,...default)	DECODE
N/A	DECRYPT_BIN DECRYPT_CHAR
N/A	ENCRYPT_AES ENCRYPT_TDES
N/A	GETHINT
MD5	MD5

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