GBASE®

GBase TD2GBA 迁移工具
说明书 V1.3

www.gbase.cn

GBase TD2GBA 迁移工具说明书，南大通用数据技术股份有限公司

通讯方式

南大通用数据技术股份有限公司

天津高新区华苑产业园区开华道 22 号普天创新产业园区东塔

电话：400-013-9696　　邮箱：info@gbase.cn

目 录

1 GBase TD2GBA 产品简介

1.1 产品简介

南大通用 Teradata 到 GBase 8a 的迁移工具，简称：GBase TD2GBA，采用 Python2 语言编写，该工具可以实现从 Teradata 到 GBase 8a 的 DDL、Perl、DSQL 脚本的迁移，迁移效率可以达到上千个脚本/分钟。

1.2 产品技术特点

GBase TD2GBA 迁移工具具有灵活性高、语法覆盖全面、转换效率高、使用简单等特点，可准确、快速完成从 Teradata 到 GBase 8a 的 DDL、Perl、DSQL 脚本的迁移，具体如下：

1) 灵活性高：基于 Python 语言编写的明文代码转换工具，只要有 Python 代码能力的使用者可以方便的对 Teradata 语法进行扩充，在发生转换语法覆盖不全情况下，可直接对源码进行修改适配；

2) 语法覆盖全面：TD2GBA 工具在多个 TD 迁移现场使用近上万个脚本进行打磨，现有工具基本覆盖到 Teradata 常用语法，基本可做到即转即用的效果；

3) 使用简单：迁移工具从用户使用角度考虑，用最简单的架构实现工具开发，使用者只需把需要转换的脚本按照 DDL、Perl、DSQL 不同类别分别放入到自定义目录，再新建转换后需要保存的文件目录，执行转换工具命令并制定原始文件目录、转换后文件目录即可自动实现转换迁移，无需繁琐的配置；

4) 高效转换：由于工具是基于 Teradata 语法搜索、匹配及替换，所以其转换效果非常高，可达到上千个脚本/分钟。

1.3 产品基本功能简介

功　能	描　述
DDL 转换	可转换从 Teradata 导出来的 DDL 语句文件，每个文件可以是单个 DDL 语句也可以是多个 DDL 的集合文件。
视图转换	可转换从 Teradata 导出来的视图语句文件，每个文件可以是单个视图语句也可以是多个视图语句的集合文件。
Perl 脚本	可转换基于 Teradata 语法编写的 Perl 脚本。工具转换后仍然是 Perl 语言的脚本文件，SQL 语法则转换成 GBase 8a 的语法。
DSQL 脚本	可转换基于 Teradata 语法编写的 DSQL 脚本。工具转换后可以是 Python、Perl 语言的脚本文件，SQL 语法则转换成 GBase 8a 的语法。

2 GBase TD2GBA 迁移工具产品架构

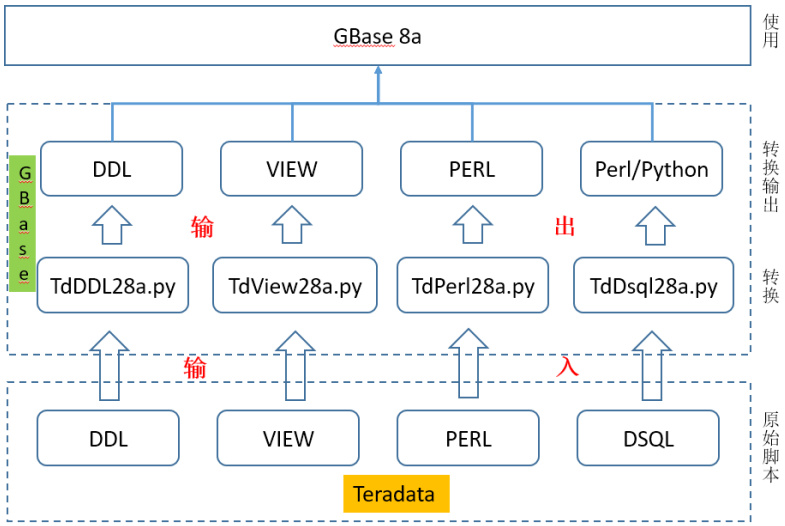

图 2-1 GBase TD2GBA 迁移工具产品架构图

GBase TD2GBA 对不同的转换内容采用不同的程序，共计四个执行程序，均有 Python 语言编写，每个程序的功能内容如下：

TdDDL28a.py：用于转换从 Teradata 导出来的 DDL 语句，DDL 语句可以存放到一个文件中，也可以每个 DDL 一个文件，转换后保持原有的文件数量不变，与原始文件一一对应。

TDView28a.py：转换从 Teradata 导出来了的 View 视图语句，View 视图语句可以存放到一个文件，也可以每个 View 语句一个文件，转换后保持原有的文件数量不变，与原始文件一一对应。

TdPer28a.py:转换基于 Teradata 编写的 Perl 批处理脚本，一般每个处理程序一个 Perl 脚本，转换仅仅对 Perl 中的 SQL 语法进行转换，Perl 结构、变量、返回值均保持不变。

TdDsql28a.py:转换 Teradata 特有的 DSQL 脚本，由于 DSQL 带有逻辑控制，而存储过程对逻辑控制的灵活度不高，选择转换成 Perl 或者 Pthon 脚本程序。

3 GBase TD2GBA 核心转换语法

3.1 TdDDL28a 语法

DDL 语法主要转换表类型、数据类型、分布键三大部分，通过程序可实现如下关系的转换对照表。

3.1.1 表类型迁移对照

TeraData 语法	GBase 语法
create set table t1(c1 int)	create table t1(c1 int)
create set table t1 , NO LOG(c1 int)	create table t1 (c1 int)
create volatile temporary table t1, NO LOG(c1 int)	create temporary table t1(c1 int)
create set global temporary table t1 , NO LOG(c1 int)	create table t1 (c1 int)
create set global temporary table t1 , NO LOG(c1 int) PRIMARY INDEX(c1)	create table t1 (c1 int,PRIMARY key(c1))
create set global temporary table t1 , NO LOG(c1 int,c2 int) PRIMARY INDEX(c1,c2)	create table t1 (c1 int,c2 int,PRIMARY key(c1,c2))
ct t1(c1 int)	CREATE TABLE t1(c1 int)
create volatile temporary table t1, NO LOG(c1 int) on commit preserve rows	create temporary table t1(c1 int)
create multiset table emp_data ,FALLBACK\n ,NO BEFORE JOURNAL ,NO AFTER JOURNAL ,FREESPACE=30 ,DATABLOCKSIZE=10000 BYTES(…	create table emp_data(…);

TeraData 语法	GBase 语法
);	
create table dept1 as department with no data	create table dept1 like department
create table dept1,FALLBACK as department with no data	create table dept1 like department
create table emp1 as (select employee_number,departmen_number,salary_amount from employee) with no data	create table emp1 as (select employee_number,departmen_number,salary_amount from employee) ;delete from emp1
create table emp1,FALLBACK as (select employee_number,departmen_number,salary_amount from employee) with no data	create table emp1 as (select employee_number,departmen_number,salary_amount from employee) ;delete from emp1
create global temporary table emp1,FALLBACK as (select employee_number,departmen_number,salary_amount from employee) with no data	create table emp1 as (select employee_number,departmen_number,salary_amount from employee) ;delete from emp1
create table emp1 as (select employee_number,departmen_number,salary_amount from employee) with data	create table emp1 as (select employee_number,departmen_number,salary_amount from employee)
create table emp1,FALLBACK as (select employee_number,departmen_number,salary_amount from employee) with data	create table emp1 as (select employee_number,departmen_number,salary_amount from employee)
create table emp1(a,b,c),FALLBACK as (select employee_number,departmen_number,salary_amount from employee) with data	create table emp1 as (select employee_number as a,departmen_number as b,salary_amount as c from employee)
create table emp1(a UNIQUE NOT NULL,b,c),FALLBACK as (select employee_number,departmen_number	create table emp1 as (select employee_number as a,departmen_number as

TeraData 语法	GBase 语法
,salary_amount from employee) with data	b,salary_amount as c from employee)
create table emp1(a UNIQUE NOT NULL,b NOT NULL DEFAULT 0,c),FALLBACK as (select employee_number,departmen_number ,salary_amount from employee) with data	create table emp1 as (select employee_number as a,departmen_number as b,salary_amount as c from employee)
create table emp1(a UNIQUE NOT NULL,b NOT NULL DEFAULT 0,c),FALLBACK as (select employee_number,departmen_number ,salary_amount from employee) with data primary index(a)	create table emp1 as (select employee_number as a,departmen_number as b,salary_amount as c from employee) DISTRIBUTED BY ('a')
create table emp1(a UNIQUE NOT NULL,b NOT NULL DEFAULT 0,c),FALLBACK as (select employee_number,departmen_number ,salary_amount from employee) with data primary index(a,b)	create table emp1 as (select employee_number as a,departmen_number as b,salary_amount as c from employee) DISTRIBUTED BY ('a','b')
create multiset volatile table tab1 as tab2 with no data on commit preserve rows	create table tab1 like tab2
create multiset volatile table tab1 as (select * from b) with data on commit preserve rows	create temporary table tab1 as (select * from b)
create multiset volatile table tab1 as (select * from b) with data on rollback preserve rows	create temporary table tab1 as (select * from b)
CREATE VOLATILE MULTISET TABLE VT_AUM , NO LOG (column1) PRIMARY INDEX (column2) ON COMMIT PRESERVE ROWS	CREATE temporary TABLE VT_AUM (column1) DISTRIBUTED BY ('column2')
CREATE VOLATILE MULTISET TABLE VT_AUM , NO LOG as(select column1 from t1) WITH NO DATA	CREATE temporary TABLE VT_AUM as(select column1 from t1) DISTRIBUTED BY

TeraData 语法	GBase 语法
PRIMARY INDEX (Core_Cust_Id) ON COMMIT PRESERVE ROW;	('Core_Cust_Id') ;delete from VT_AUM;
alter table emp_data,FALLBACK	空
alter table t1 add c1 format 'mmmBdd,Byyyy'	alter table t1 add c1 format 'mmmBdd,Byyyy'
alter table t1,FALLBACK drop c1,add c2 int	alter table t1 drop c1,add c2 int

3.1.2 数据类型迁移对照

数据类型分数值型、字符型、二进制类型进行迁移，日期类型完全兼容。

数值型对照表:

TeraData 语法	GBase 语法
NUMERIC(p,s)	DECIMAL(p,s)
REAL	DECIMAL
DEC	DECIMAL
DOUBLE PRECISION	DOUBLE
BYTEINT	TINYINT

字符类型对照表:

TeraData 语法	GBase 语法
CHARACTER VARYING(N)	N<=10922:VARCHAR;N>10922:LONGTEXT
VARCHAR(N)	N<=10922:VARCHAR;N>10922:LONGTEXT
LONG VARCHAR	N<=10922:TEXT;N>10922:LONGTEXT
CHARACTER	N<=255: CHAR;N>255: TEXT
CHAR(N)	N<=255: CHAR;N>255: TEXT

二进制类型对照表:

TeraData 语法	GBase 语法
BYTE(N)	N<=32K:BLOB;N>32K:LONGBLOB
VARBYTE(N)	N<=32K:BLOB;N>32K:LONGBLOB

字段注释、别名对照表:

TeraData 语法	GBase 语法
…col1 varchar TITLE ‘区号标识’ …	…col1 varchar comment ‘区号标识’ …
SELECT c1(named column1),c2(named column2) FROM t1	SELECT c1 AS column1,c2 AS column2 FROM t1
select 'a'(title 'b') from t1	select 'a' from t1
select c1,c2,c3/2(title 'abc//def') from tt1	select c1,c2,c3/2 from tt1

3.1.3 分布键迁移对照

Teradata 的 DDL 中部分表有 PRIMARY INDEX 语法，对其后面的 INDEX 值作为 GBase 8a 的分布键，但需要把 GBase 8a 不支持的分布键类型去掉。例如

Teradata 原始 Primary index 语句

PRIMARY INDEX (Orig_Airport_Cd ,Flt_Dpt_Dt ,Dest_Airport_Cd)

转换为 GBase 8a 的语法：

DISTRIBUTED BY(‘Orig_Airport_Cd’ ,’Dest_Airport_Cd’)，其中的 Flt_Dpt_Dt 是日期类型，不能用作 GBase 8a 的分布键。

3.2 TdPer128a 语法

3.2.1 操作符对照

操作符 Teradata 与 GBase 基本兼容，迁移主要体现在比较操作符上，迁移对照表如下：

TeraData 语法	GBase 语法
expr1 EQ expr2	expr1 = expr2
expr1 NE expr2	expr1 <> expr2
expr1 GT expr2	expr1 > expr2
expr1 LT expr2	expr1 < expr2
expr1 GE expr2	expr1 >= expr2
expr1 LE expr2	expr1 <= expr2
expr1 ^= expr2	expr1 <> expr2
expr1 ** expr2	power(expr1,expr2)
expr1 MOD expr2	MOD(expr1,expr2)
expr1 not= expr2	expr1 != expr2

3.2.2 函数

函数 Teradata 与 GBase 也基本兼容，但仍有部分需要转换，主要体现在如下的几个方面：

时间函数：

TeraData 语法	GBase 语法
date_trunc(current_date,'year')	trunc(current_date,'year')

字符串函数：

TeraData 语法	GBase 语法
chr(71)	char(71)
char('aa')	char_length('aa')
characters('bbb')	char_length('bbb')

TeraData 语法	GBase 语法
to_utf8('aa')	decode('aa')
from_utf8()	encode()
position('a' in 'Name')	instr('Name','a')
regexp_extract('Name','a')	regexp_substr('Name','a')
strpos('Name','a')	instr('Name','a')
SUBSTR('Quadratically',5)	substring('Quadratically',5)
substring('catalog' from 5 for 3)	substring('catalog' , 5 , 3)
substring('catalog' from 5)	substring('catalog' , 5)
substring('catalog' from 5 for 3)(title 'fi')	select substring('catalog' , 5 , 3)
substring(contact_name from index(contact_name,',')+2) || ' ' || substring(contact_name from 1 for index(contact_name,',')-1)	CONCAT(CONCAT(substring(contact_name , instr(contact_name,',')+2) , ' ') , substring(contact_name , 1 , instr(contact_name,',')-1))
LENGTH(trim(STRTOK('${JOB_NAME}','-',4)))	LENGTH(trim(SUBSTRING_INDEX(SUBSTRING_INDEX('${JOB_NAME}','-',4),'-',-1))))

比较函数:

TeraData 语法	GBase 语法
ZEROIFNULL(c1)	同名 UDF 函数
NULLIFZERO(c1)	同名 UDF 函数
c1 like all ('str1%','str2')	c1 LIKE 'str1%' AND c1 LIKE 'str2'
c1 not like all ('str1%','str2')	c1 NOT LIKE 'str1%' AND c1 NOT LIKE 'str2'

TeraData 语法	GBase 语法
c1 like any ('str1%','str2')	c1 LIKE 'str1%' OR c1 LIKE 'str2'
c1 not like any ('str1%','str2')	c1 NOT LIKE 'str1%' OR c1 NOT LIKE 'str2'

转换函数:

TeraData 语法	GBase 语法
cast('20150801' as date format 'yyyymmdd')	cast('20150801' as date)
cast((cast('20150801' as date format 'yyyymmdd')-1) as date format 'yyyymmdd')	cast((cast('20150801' as date)-1) as date)
cast('20150801' as date format 'yyyymmdd')(format 'yyyy-mm-dd')(title 'b')	to_char(cast('20150801' as date),'yyyy-mm-dd')
expr1 || cast(expr2 as date format 'yyyymmdd') || expr3	CONCAT(CONCAT(expr1 , cast(expr2 as date)) , expr3)
expr1(format '9(n)')	LPAD(expr1,n,'0')
expr1(format 'z(n)')	LPAD(expr1,n,'')
(expression1)(char(10))	CAST((expression1) AS char(10))
(expression1)(DEC(10,2))	CAST((expression1) AS decimal(10,2))
(expression1)(FORMAT 'yyyymmdd')(char(10))	CAST(to_char((expression1),'yyyymmdd') AS char(10))
expression1(FORMAT '9(m)')(char(10))	CAST(LPAD(expression1,m,'0') AS char(10))
cast(expression1 as int format '9(n)')	LPAD(cast(expression1 as int),n,'0')
cast(expression1 as decimal(10,2) format 'Z(n)')	LPAD(cast(expression1 as decimal(10,2)),n,'')
cast(cast(expression1 as decimal(10,2) format 'Z(n)') as varchar(10))	cast(LPAD(cast(expression1 as decimal(10,2)),n,'') as varchar(10))
expression1(decimal(10,2))(format '9(n)')(char(10))	CAST(LPAD(CAST(expression1 AS decimal(10,2)),n,'0') AS char(10))

TeraData 语法	GBase 语法
c1 as cc (format '9')	to_char(c1, '9') as cc

OLAP 函数:

TeraData 语法	GBase 语法
select c1,c2,csum(1,1) from tt1	select c1,c2,row_number() over(order by 1,1) from tt1
select c1,c2,c3,csum(1,1) from tt1 group by c1,c2	select c1,c2,c3,row_number() over(partition by c1,c2 order by 1,1) from tt1
select c1,c2,csum(c3,c4 desc,c5) from tt1	select c1,c2,sum(c3) over(order by c4 desc,c5) from tt1
select c1,c2,c3,csum(c4,c1 desc,c3) as csum1 from tt1 group by c1	select c1,c2,c3,sum(c4) over(partition by c1 order by c1 desc,c3) as csum1 from tt1
select c1,c2,c3,csum(c4,c3) from tt1 qualify row_number() over(order by c4)=1 group by 1,2	select * from (select c1,c2,c3,sum(c4) over(partition by 1,2 order by c3) ,row_number() over(partition by 1,2 order by c4) as qualify_colName1 from tt1) qualify_tbName1 where qualify_tbName1.qualify_colName1=1
select c1,c2,rank(c3,c4 desc,c5) from tt1	select c1,c2,rank() over(order by c3,c4 desc,c5) from tt1
select c1,c2,c3,rank(c4,c1 desc,c3) as rank1 from tt1 group by c1	select c1,c2,c3,rank() over(partition by c1 order by c4,c1 desc,c3) as rank1 from tt1
select c1,c2,c3,rank(c4,c3) from tt1 qualify row_number() over(order by c4)=1 group by 1,2	select * from (select c1,c2,c3,rank() over(partition by 1,2 order by c4,c3) ,row_number() over(partition by 1,2 order by c4) as qualify_colName1 from

TeraData 语法	GBase 语法
	tt1) qualify_tbName1 where qualify_tbName1.qualify_colName 1=1
select c1,c2,c3,rank(c3) from tt1 group by c1 qualify rank(c3)<=3	select * from (select c1,c2,c3,rank() over(partition by c1 order by c3) ,rank() over(partition by c1 order by c3) as qualify_colName1 from tt1) qualify_tbName1 where qualify_tbName1.qualify_colName 1<=3
select t.prodid,t.sumsales,rank(t.sumsales) from (select a.prodid,sum(a.sales) from salestb1 a group by 1) as t(prodid,sumsales) qualify rank(sumsales asc)<=3	select * from (select t.prodid,t.sumsales,rank() over(order by t.sumsales) ,rank() over(order by sumsales asc) as qualify_colName1 from (select a.prodid as prodid ,sum(a.sales) as sumsales from salestb1 a group by 1) as t) qualify_tbName1 where qualify_tbName1.qualify_colName 1<=3
select c1,c2,c3,row_number() over(order by c4,c3) from tt1 qualify rank(c4)=1 group by 1,2	select * from (select c1,c2,c3,row_number() over(order by c4,c3) ,rank() over(partition by 1,2 order by c4) as qualify_colName1 from tt1) qualify_tbName1 where qualify_tbName1.qualify_colName 1=1
select t2.bank_num as bank_num,t2.inn_org_id as inn_org_id,t2.up_inn_org_id as up_inn_org_id from ${pdm_view}.v04_inn_org_chg_h t2	select * from (select t2.bank_num as bank_num,t2.inn_org_id as inn_org_id,t2.up_inn_org_id as up_inn_org_id ,row_number()

TeraData 语法	GBase 语法
where t2.start_dt<='${v_f_tx_date}' and t2.end_dt>'${v_f_tx_date}' and t2.org_hrcy_typ_cd='ncrptt' and t2.efft_dt<='${v_f_tx_date}' and t2.bank_num='${edw_bank_no}' qualify row_number() over(partition by t2.bank_num,t2.inn_org_id order by t2.efft_dt desc)=1	over(partition by t2.bank_num,t2.inn_org_id order by t2.efft_dt desc) as qualify_colName1 from ${pdm_view}.v04_inn_org_chg_h t2 where t2.start_dt<='${v_f_tx_date}' and t2.end_dt>'${v_f_tx_date}' and t2.org_hrcy_typ_cd='ncrptt' and t2.efft_dt<='${v_f_tx_date}' and t2.bank_num='${edw_bank_no}') qualify_tbName1 where qualify_tbName1.qualify_colName1=1

3.2.3 SQL 语法

Teradata 与 GBase 8a 不同主要表现在如下的对照表内容:

TeraData 语法	GBase 语法
select c1,c2,max(c3) alias1 from tt1 group by 1,2 where c1>2 and c2>1	select c1,c2,max(c3) alias1 from tt1 where c1>2 and c2>1 group by 1,2
select c1,c2,max(c3) alias1 from tt1 where c1>2 having c2<>2 group by 1,2	select c1,c2,max(c3) alias1 from tt1 where c1>2 group by 1,2 having c2<>2
select c2 from tt2 where c2=any (select c2 from tt1)	select c2 from tt2 where c2 in (select c2 from tt1)
select c2 from tt2 where c2 not = all (select c2 from tt1)	select c2 from tt2 where c2 not in (select c2 from tt1)
select c2 from tt2 where c2=some (select c2 from tt1)	select c2 from tt2 where c2 in (select c2 from tt1)

TeraData 语法	GBase 语法
SEL date_trunc(logtbl.LogDate,'year'),logtbl.QueryId FROM PDCRINFO.DBQLogTbl_Hst logtbl WHERE logtbl.LogDate = CURRENT_DATE-1	select trunc(logtbl.LogDate,'year'),logtbl.QueryId FROM PDCRINFO.DBQLogTbl_Hst logtbl WHERE logtbl.LogDate = CURRENT_DATE-1
SELECT top 1 * FROM t1 order by c2	SELECT * FROM t1 order by c2 LIMIT 1
SELECT * FROM t1 order by c2 sample 1	SELECT * FROM t1 order by c2 LIMIT 1
SELECT last_name, salary_amount, department_number, avgsal FROM (SELECT AVG(salary_amount), department_number FROM employee GROUP BY department_number) my_temp(avgsal, deptno), employee ee WHERE salary_amount > avgsal AND department_number = deptno ORDER BY 2 DES	SELECT last_name, salary_amount, department_number, avgsal FROM (SELECT AVG(salary_amount) as avgsal , department_number as deptno FROM employee GROUP BY department_number) my_temp, employee ee WHERE salary_amount > avgsal AND department_number = deptno ORDER BY 2 DESC
SELECT c1(named column1),c2(named column2) FROM t1	SELECT c1 AS column1,c2 AS column2 FROM t1

部分 Teradata 中简写的语法，需要在 GBase 8a 中补全，语法对照表如下：

TeraData 语法	GBase 语法
SEL	SELECT
INS	INSERT
UPD	UPDATE
DEL	DELETE
CT	CREATE TABLE
CV	CREATE VIEW

3.3 TdView28a 语法

从 Teradata 到 GBase 8a 的视图转换比较简单，基本上兼容，所以只需对少量的语法进行替换即可，主要迁移内容对照表如下：

TeraData 语法	GBase 语法
cv v1 as select * from t1	create view v1 as select * from t1
replace view v1 as select * from t1	create or replace view v1 as select * from t1
create view v1 as select * from t1 with check option	create view v1 as select * from t1

3.4 TdDsql28a 语法

DSQL 的转换主要是两部分，一部分是逻辑控制的转换，一部分是 SQL 语法的转换，SQL 语法转换同 TdPerl28a 的语法一致，不再赘述。对于逻辑控制部分主要用 Perl/Python 自身的逻辑判断来实现。

SQL 执行结果的判断，ACTIVITYCOUNT 值对应 Perl 的 SQL 查询句柄 rows()函数值。也可用 FOUND_ROWS()及 ROW_COUNT()来获取前面 SQL 执行的结果来进行判断。

GOTO 语句 Teradata 与 GBase 8a 兼容。

LABEL 语句 Teradata 与 GBase 8a 稍微有点差异,Teradata 是:LABEL xxx，而 GBase 8a 则直接是 xxx:

4 运行环境和技术指标

TD2GBA 迁移工具是 Python 代码，所以跟硬件平台没有直接关系，只要具备 Python2 的环境即可运行。

技术指标:

技术指标	描　述
运行平台	Linux/Windows/MAC OS
DDL 转换性能	>=2000 个/Min
Perl 脚本转换性能	>=1000 个/Min。
View 转换性能	>=2000 个/Min
DSQL 脚本转换性能	>=1000 个/Min。
字符编码转换	UTF-8 到 GBK、GBK 到 UTF-8

5 工具操作

TD2GBA 工具操作极为简单，命令执行时候只需带两个参数即可，第一个参数是保存从 Teradata 中导出的原始语法文件，第二个参数则是存储转换后的对应的 GBase 8a 可执行的语法文件。

举例说明：

DDL 转换需要两个目录名称作为参数，比如 TDDDL 目录是保存原始 Teradata 的 DDL 语句，GBADDL 是保存转换后 GBase 8a 的语法格式 DDL 文件，在 linux 下的转换命令为：

```
[root@n234 ~]# python TdDDL28a.py TDDDL GBADDL
```

同理其他的三个转换工具命令类同，第一参数是原始的 Teradata 文件，第二个是转换后 GBase 8a 的语法格式文件：

```
[root@n234 ~]# python TDView28a.py TDVIEW GBAVIEW
```

```
[root@n234 ~]# python TdPer28a.py TDPERL GBAPERL
```

```
[root@n234 ~]# python TdDsql28a.py TDDSQL GBAPERL
```

GBASE®

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

官方微信

GBase 8a 技术社区

技术支持热线：400-013-9696