在Oracle8i的SQL*Plus中如何利用LOB字段存取操作系统二进制文件 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/254/2021_2022__E5_9C_A8Oracle8_c102_254614.htm Oracle 8i数据库系统功能比前面版本更加完善，尤其是出现了BLOB,CLOB,NCLOB,BFILE这些LOB(大型对象)类型来取代功能有限的LONG、LONGRAW类型。BLOB字段最大长度为4G(4,294,967,295)字节,而且不再象LONGRAW那样每个表中只是限制有一个字段是LONGRAW（最长2G）型的。BLOB,CLOB,NCLOB为内部BLOB(数据通常在数据库中存放)，BFILE为外部LOB（所存储的只是指向外部操作系统文件的指针）,用户可以使用PL/SQL的包DBMS_LOB来处理LOB数据，但是遗憾的是，DBMS_LOB包只能将二进制操作系统文件写入到BLOB字段中，却无法将BLOB字段中的二进制操作系统文件取回到操作系统中，估计将来会有所改善。本文将就如何在SQL*Plus将WORD文件存入取出ORACLE中作详细解释说明，供各位同行参考。 实验的软件环境如下：Windows 2000 Advanced Server，Oracle 8.1.7，VC 6.0 SP5 硬件环境如下：双PIII866 CPU，768M内存 在internal这个用户下给scott用户授权如下：SQL>grant create any directory to scott. SQL>grant create any library to scott. 在scott这个用户下执行下述语句： SQL>create table bfile_tab (bfile_column BFILE). SQL>create table utl_lob_test (blob_column BLOB). SQL>create or replace directory utllobdir as C:\DDS\EXTPROC. SQL>set serveroutput on 然后执行下面语句就将C:\DDS\EXTPROC目录下的word文件COM.doc存入到utl_lob_test 表中的blob_column字段中了。 declare a_blob

BLOB. a_bfile BFILE := BFILENAME(UTLLOBDIR,COM.doc). --用来指向外部操作系统 文件 begin insert into bfile_tab values (a_bfile) returning bfile_column into a_bfile. insert into utl_lob_test values (empty_blob()) returning blob_column into a_blob. dbms_lob.fileopen(a_bfile). dbms_lob.loadfromfile(a_blob, a_bfile, dbms_lob.getlength(a_bfile)). dbms_lob.fileclose(a_bfile). commit. end. / SQL>show errors 此时可以使用DBMS_LOB包的getlength这个procedure来检测是否已经将该word文件存入 到blob字段中了。如： SQL> 0select dbms_lob.getlength(blob_column) from UTL_LOB_TEST. 结果如下： DBMS_LOB.GETLENGTH(BLOB_COLUMN) ---------- 83968 说明该word文件已经存入到blob字段中去了。 下面将就如何取出该word文件到操作系统下作详细解释： Oracle8.1.7只能用pro*c与OCI来实现该任务，所以Oracle服务器端必须支持pro*c 以及外部library ，Oracle8.1.7数据库默认安装为支持pro*c以及外部Procedure ， 用户可以自己检查一下listener.ora 和 tnsnames.ora这两个文件。 listener.ora中包含如下语句： SID_LIST_LISTENER = (SID_LIST = (SID_DESC = (SID_NAME = PLSExtProc) (ORACLE_HOME = D:\oracle\ora81) (PROGRAM = extproc)) (SID_DESC = (GLOBAL_DBNAME = hft) (ORACLE_HOME = D:\oracle\ora81) (SID_NAME = hft))) tnsnames.ora中包含如下语句： EXTPROC_CONNECTION_DATA = (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))) (CONNECT_DATA = (SID = PLSExtProc) (PRESENTATION = RO))) 下面这个文件为lob2file.c,具体作用

是将BLOB中的二进制文件倒出到操作系统中。 /*begin of lob2file.c*/ #include #include #include #include #include #define DEFAULT_CHUNK_SIZE 1024 static int logging. static char logfile[512]. static FILE *logfilep = NULL. int lob2file (OCILobLocator *a_lob, /* the LOB */ short lbind, /* LOB indicator */ char *path, /* file to write */ short pind, /* file indicator */ int plen, /* filename length */ char *lpath, /* logfile name */ short lpind, /* logfile indicator */ int lplen, /* logfile name length */ int logit, /* logging enabled? */ OCIExtProcContext *ctxt /* OCI Context */) { sword errnum = 0. OCIEnv *envhp = NULL. OCISvcCtx *svchp = NULL. OCIError *errhp = NULL. char lobfile[512]. FILE *lobfilep = NULL. /* * If required, open the log file for writing * Use the user provided logfile name if possible * Otherwise, default the logfile to lob2file.log */ logging = logit. if (logging) { if (lpind == -1 || lplen == 0 || lplen >= 512) { strcpy(logfile, "lob2file.log"). } else { strncpy(logfile, lpath, lplen). logfile[lplen] = \0. } logfilep = fopen(logfile, "w"). if (logfilep == NULL) { if ((logfilep = fopen("lob2file.log", "w")) != NULL) { fprintf(logfilep, "Error: Unable to open logfile %s\n", logfile). fprintf(logfilep, "Error: errno = %d\n", errno). } } } /* * Retrieve the environment, service context, and error handles */ if ((errnum = OCIExtProcGetEnv(ctxt, amp.svchp, amp.amp.amp.amp.amp.amp.totsz)) != 0) return -1. /* * For 8.0.X the OCILogGetChunkSize will not have been called. * IN this case, reset the chunk size to 1K. */ if (cksz == 0) cksz = DEFAULT_CHUNK_SIZE. if (logging amp. logfilep != NULL) fprintf(logfilep, "Allocating %d bytes of memory for LOB chunks\n",

(int) cksz). /* * Dynamically allocate enough memory to hold a single chunk */ if ((chunk = OCIExtProcAllocCallMemory(ctxt, (size_t) cksz)) != NULL) { int cnt = 1. ub4 amt = cksz, offset = 1. /* * Read data from the LOB and write it to the file while * more data remains. */ while (offset { if (logging amp. logfilep != NULL) fprintf(logfilep, "Reading chunk %d starting at %d for max %d bytes\n", cnt, (int) offset, (int) amt). errnum = OCILobRead(svchp, errhp, a_lob, amp.amp.amp.amp.amp.amp.amp.errnum, (text *) errbuf, (ub4) sizeof(errbuf), OCI_HTYPE_ERROR). break. default: break. } if (errnum != 0) { if (logging amp. logfilep != NULL) { fprintf(logfilep, "Error: %d %s\n", errnum, errbuf). fclose(logfilep). } (void)OCIExtProcRaiseExcpWithMsg(ctxt, errnum, errbuf, strlen(errbuf)). } return errnum. } /*end of file lob2file.c*/ 将文件lob2file.c放到D:\oracle\ora81\plsql\demo目录下：然后在dos下执行下述编 译 语句将该文件编译成lob2file.dll文件，make.bat文件包含如下： @echo off cl -ID:\oracle\ora81\oci\include -D_DLL -D_MT /LD -Zi lob2file.c /link D:\oracle\ora81\oci\lib\msvc\oci.lib msvcrt.lib /nod:libcmt /DLL /EXPORT:lob2file /EXPORT:checkerr 进入D:\oracle\ora81\plsql\demo目录(DOS状态)执行make就可以将lob2file.c编 译成lob2file.dll文件了。 然后用scott连到sql*plus中，执行 SQL>CREATE OR REPLACE LIBRARY UTLLOBLIB AS D:\oracle\ora81\plsql\demo\lob2file.dll / SQL>GRANT EXECUTE ON UTLLOBLIB TO PUBLIC 然后执行下述代码： create or replace package utl_lob is procedure SetLogging(which BOOLEAN, a_log VARCHAR2). procedure UnloadToFile(a_lob

BLOB, a_file VARCHAR2, status OUT NUMBER). end utl_lob. / show errors create or replace package body utl_lob is logSetting BOOLEAN := FALSE. logFileName VARCHAR2(512) := NULL. procedure SetLogging(which BOOLEAN, a_log VARCHAR2) is begin logSetting := which. if (logSetting = TRUE) then logFileName := a_log. else logFileName := NULL. end if. end. function LobToFile(a_lob BLOB, a_file VARCHAR2,a_log VARCHAR2, logging BOOLEAN) return BINARY_INTEGER as external name "lob2file" library utlloblib LANGUAGE C with context parameters (a_lob OCILOBLOCATOR, a_lob INDICATOR SHORT, a_file STRING, a_file INDICATOR SHORT, a_file LENGTH INT, a_log STRING, a_log INDICATOR SHORT, a_log LENGTH INT, logging INT, CONTEXT, RETURN). procedure UnloadToFile(a_lob BLOB, a_file VARCHAR2, status OUT NUMBER) is begin status := LobToFile(a_lob, a_file, logFileName, logSetting). end. end utl_lob. / show errors grant execute on utl_lob to public. 该代码创建package utl_lob，而utl_lob调用library utlloblib，我们的测试程序调用package utl_lob中的procedure SetLogging和UnloadToFile。在scott用户下执行如下脚本，就可以将先前保存的COM.doc取出放到C:\DDS\EXTPROC\test.doc这个文件中，当然C:\DDS\EXTPROC这个目录必须存在。脚本执行完毕后生成两个文件test.log与test.doc，test.log纪录了取出的详细信息，test.doc是COM.doc的复制品，取出82K大小的文件大约用了4秒。 --以下为测试脚本 set serveroutput on declare a_blob BLOB. status NUMBER. begin 0select blob_column into a_blob

from utl_lob_test. utl_lob.SetLogging(TRUE, C:\DDS\EXTPROC\test.log). utl_lob.UnloadToFile(a_blob, C:\DDS\EXTPROC\test.doc, status). dbms_output.put_line(Exit status = || status). end. / 大家对上面测试脚本稍微改动一下，形成一个带参数的Procedure供应用程序调用就可以了。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com