用C 产生SQL*Loader各类文件 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/167/2021_2022__E7_94_A8C___E4_BA_A7_E7_c102_167086.htm 1. 前 言 ---- 目 前， 我 国 许 多 单 位MIS 系 统 建 立 在 微 机PC 或 基 于 Novell NetWare 局 域 网 环 境 中， 数 据 库 和 开 发 工 具 采 用 Foxpro 2.5 for DOS 或 Foxpro 2.5 for Windows, 以dbf 文 件 为 数 据 组 织 管 理 手 段， 随 着 系 统 的 不 断 扩 大 和 实 际 应 用 的 需 要， 不 少 单 位 已 开 始 采 用 大 型 数 据 库Oracle。 在MIS 从Foxpro 升 级 到 大 型 数 据 库Oracle 过 程 中， 将Foxpro 的dbf 文 件 通 过Oracle 工 具SQL*Loader 加 载 到Oracle 数 据 库 中 是 一 项 非 常 重 要 的 工 作。 一 般 用SQL*Loader 加 载 的 具 体 实 施 步 骤 是 ： ---- * 运 行Foxpro, 打 开 数 据 库， 将dbf 文 件 拷 贝 为SDF 格 式 的 文 本 文 件 ---- * 根 据dbf 文 件 结 构， 产 生 建 立Oracle 表（CREATE TABLE） 的SQL 语 句 ---- * 登 录Oracle, 运 行 产 生Oracle 表（CREATE TABLE） 的SQL 语 句 ---- * 根 据dbf 文 件 结 构， 产 生SQL*Loader 的 控 制 文 件 ---- * 运 行SQL*Loader ， 加 载 数 据 ---- 用 手 工 方 法 产 生CREATE TABLE 的SQL 语 句， 特 别 是 产 生SQL*Loader 控 制 文 件 时， POSITION 起 始、 结 束 位 置 经 常 弄 错， 当 需 要 加 载 大 量 的 数 据 时， 不 但 烦 琐， 而 且 效 率 比 较 低。 我 们 在 实 践 中 利 用Borland C 5.0 编 制 了 一 个C 实 用 程 序load.cpp， 自 动 产 生SQL*Loader 的 数 据 文 件、 控 制 文 件 和 产 生CREATE TABLE 的SQL 语 句。 运 用load， 我 们 只 需 ： ---- * load ---- * 登 录Oracle, 运 行 产 生Oracle 表（CREATE TABLE） 的SQL 语 句 ---- * 运 行SQL*Loader ， 加

载数据----在实践中，我们通过这种方法，在实现从Foxpro到Oracle for Digital UNIX中文Windows 95的client/server平台的数据加载过程中提高了效率。----2.Foxpro中dbf文件结构----dbf文件由文件头和文件记录组成，其中文件头又由数据库说明和字段说明组成。数据库说明由32个字节组成，各字节含义见表一：字节含义0数据库文件标志有无备注型字段（03H无）1-3最后一次修改日期4-7文件记录数8-9文件头长度10-11记录长度12-31未用----表一----字段说明由若干个32字节组成，每32字节说明一个字段，各字节含义见表二：字节含义0-10字段名11字段类型12-15该字段在文件首记录中的地址16字段长度17小数位数18-31未用----表二----文件记录以ASCII形式存储，每条记录以空格（20H）开头，该空格用来作删除标志用。----3.建立对应Foxpro的Oracle表的SQL语句----Foxpro和Oracle对应的数据类型的描述见表三：Foxpro Oracle Character(n) char(n) varchar2(n) Number(n,m) number(n-1,m) m0 number(n,m) m=0 Float(n,m) Logical char(1) DATE DATE----三----【注】：----*不考虑完整性约束，同时对于TABLESPACE及STORAGE存储参数取缺省值。----*对于数字型字段，n表示数字的宽度，在Foxpro中包含小数点位置，而在Oracle中不包含。----*对于Foxpro logical型字段类型，由于Oracle中没有相应的逻辑型变量，故将其转换为字符类型。----*暂且不考虑memo、general、picture字段的转换。----4.SQL*Loader控制文件的建立----控制文件为SQL*Loader

的核心文件， 与Foxpro字段对应关系为表四： Foxpro数据类型---- 控制文件语句对应的格式 Character(n) CHAR Number(n,m) Float(n,m) DECIMAL EXTERNAL NULLIF = BLANKS （m0） INTEGER EXTERNAL NULLIF = BLANKS （m=0） Logical CHAR DATE DATE "YYYYMMDD" NULLIF = BLANKS ---- 四 ---- 以下是用Borland C 5.0 在中文Windows 95 下编制的产生CREATE TABLE SQL 语句和产生SQL*Loader 数据文件、 控制文件的源程序load.cpp 。 #include #include #include #include #include #include #define MAX_ROW_LENGTH 1200 #define MAX_FIELD_NUMBER 30 typedef struct head // dbf头文件结构 { unsigned char mask . unsigned char date[3] . unsigned long record_num. unsigned short int head_length. unsigned short int field_length . } HEAD . typedef struct field // dbf字段结构 { unsigned char name[11]. unsigned char type . unsigned long add. unsigned char length. unsigned char dec . } FIELD . int main(int argc,char **argv) { char buf[MAX_ROW_LENGTH],dbf[40],*sqlload. unsigned int i,field_num. HEAD *dbfhead . FIELD dbffield[MAX_FIELD_NUMBER]. FILE *fout, *fp. if (argc!=2) { cout return -1. } sqlload = new char(40). dbfhead = new HEAD. strcpy(buf,""). strcpy(dbf,argv[1]). strcat(dbf,".dbf"). if ((fp=fopen(dbf,"rb")) == NULL) { cout return -1 . } fseek(fp,0,SEEK_SET). fread(dbfhead,sizeof(HEAD),1,fp). // 读dbf头文件信息 field_num = (dbfhead- >head_length-1)/32 -1 . //字段个数 for(i=0. i { fseek(fp,32*(i 1),SEEK_SET). fread(&amp.dbffield[i],sizeof(FIELD),1,fp). // 读dbf结构信息 } //

产生SQL*Loader控制文件 strcpy(sqlload,argv[1]). strcat(sqlload,".ctl"). if ((fout=fopen(sqlload,"w")) == NULL) { cout return -1 . } fprintf(fout,"LOAD DATA\n"). fprintf(fout,"INFILE %s.txt\n", argv[1]). fprintf(fout,"INTO TABLE %s (\n", argv[1]). for(i=0.i { fprintf(fout, "s POSITION(%d:%d)", dbffield[i].name, dbffield[i].add, dbffield[i].add dbffield[i].length -1). switch (dbffield[i].type) { case C: case L: // 字符型/ 逻辑型 fprintf(fout, " CHAR"). break . case N: if (dbffield[i].dec == 0) //整数型 fprintf(fout, "INTEGER EXTERNAL NULLIF %s = BLANKS", dbffield[i].name). else //实数型 fprintf(fout, " DECIMAL EXTERNAL NULLIF %s =BLANKS", dbffield[i].name). break. case D: //日期型 fprintf(fout, " DATE YYYYMMDD NULLIF %s = BLANKS", dbffield[i].name). break. default: break. } if(i fprintf(fout, ",\n") . } fprintf(fout, ")\n"). fclose(fout). // 产生CREATE TABEL.的SQL语句 strcpy(sqlload,argv[1]). strcat(sqlload,".sql"). if ((fout=fopen(sqlload,"w")) == NULL) { cout return -1 . } fprintf(fout, "create table %s (\n", argv[1]). for(i=0.i { fprintf(fout,"s",dbffield[i].name). switch (dbffield[i].type) { case C: //字符型 fprintf(fout, " CHAR(%d)",dbffield[i].length). break. case L: //逻辑型 fprintf(fout, " CHAR(1)"). break. case N: //数字型 if (dbffield[i].dec==0) fprintf(fout," NUMBER(%d)", dbffield[i].length) . else fprintf(fout, " NUMBER(%d,%d)", 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