DBMaster

DCIMFユーザー参照編

SYSCOM

CASEMaker Inc./Corporate Headquarters

1680 Civic Center Drive
Santa Clara, CA 95050, U.S.A.

Contact Information:
CASEMaker US Division
E-mail : info@casemaker.com
Europe Division
E-mail : casemaker.europe@casemaker.com
Asia Division
E-mail : casemaker.asia@casemaker.com(Taiwan)
E-mail : info@casemaker.co.jp(Japan)

www.casemaker.com
www.casemaker.com/support

Document No. 645049-231888/DBM50J-M01312008-USG
発行日:2008-01-31

本文には記されていない新しい機能についての説明は、CASEMakerのDBMasterをインストールしてからREADME.TXTを読んでください。

登録商標
CASEMaker、CASEMakerのロゴは、CASEMaker社の商標または登録商標です。
DBMasterは、Syscom Computer Engineering社の商標または登録商標です。
Microsoft、MS-DOS、Windows、Windows NTは、Microsoft社の商標または登録商標です。
UNIXは、The Open Groupの商標または登録商標です。
ANSIは、American National Standards Institute, Incの商標または登録商標です。

ここで使用されているその他の製品名は、その所有者の商標または登録商標で、情報として記述しているだけです。SQLは、工業用語であって、いかなる企業、企業集団、組織、組織集団の所有物でもありません。

注意事項
本書で記述されるソフトウェアは、ソフトウェアと共に提供される使用許諾書に基づきます。

保証については、ご利用の販売店にお問い合わせ下さい。販売店は、特定用途への本コンピュータ製品の商品性や適合性について、代表または保証しません。販売店は、突然の衝撃、過度の熱、冷気、湿度等の外的な要因による本コンピュータ製品へ生じたいかなる損害に対しても責任を負いません。不正な電圧や不適合なハードウェアやソフトウェアによってもたらされた損失や損害も同様です。

本書の記載情報は、その内容について十分精査していますが、その誤りについて責任を負うものではありません。本書は、事前の通知無く変更することがあります。

コンテント

1 はじめに

本書は有効なデータベース管理(RDBMS)とCOBOL プログラムを総合したいソフトウェア開発者に提供します。MFCOBOL (MFCOBOLのDCI)またはDBMasterCOBOL インタフェース使用する方法を説明します。プログラムはDBMasterデータベースエンジンでCOBOLデータの総合と有効な管理をデザインします。

MFCOBOLのDCIはプログラムとDBMasterの間にコミュニケーションチャンネルを提供します. MFCOBOLの DBMasterCOBOLインタフェース(MFCOBOLのDCI) がDBMasterデータベースにCOBOLプログラム有効な情報をストアすることができます。データをストアするためCOBOL プログラムはB-TREE ファイルを使用します。B-TREEファイルに保存する情報はCOBOL I/O構文でアクセスします 。例えばREAD, WRITE とREWRITE です。

COBOL プログラムはDBMasterRDBMSに保存する情報もアクセスできます.COBOLプログラムは組込みSQLを使用してSQL構文をCOBOLソースコードに組む込みます。ソースコードを編集する前、特別なSQL構文"calls"がデータベースエンジンに表示ます。この"calls"はDBMaster RDBMSをアクセスするため実行します。

COBOL プログラムのデータベースにでこれは情報を保存する良い技術ですが,欠点もあります。まず 、COBOL プログラムはSQL言語の知識を持ちます。そして、プログラムの書き込みは携帯ではありません。これは

B-TREE ファイルとDBMasterRDBMS一緒に存在で実行できません。その上、SQL句は常にデータベースからデータベースに変更します。COBOLプラグラムはほかのデータベースに実行できないという意味です。最後、組込みSQLは既存のプログラムに実現できません。事実上、組込みSQLは重要なアプリケーションを要求します。ワークストレージ、データストレージ、I/O構文倫理を含みます。ここには組込みSQLに選択項目があります。供給者はシームレスインタフェースがCOBOLからデータベースに開発しました。このインタフェースはCOBOL I/Oコマンドを SQL構文に転換します。これを通じてCOBOLプログラムはSQLに知る必要がありませんが、性能は問題です。

事実上、SQLは違う目的があります、SQLはセットベースad hocクエリ言語で
説明書からデータの組合せが探せます。これを比べてCOBOL B-TREEコールはディレクトデータにデザインされます。だから、強制トランザクション、性能敏感のCOBOLアプリケーションは排他via SQL-based I/Oを実行するのは不適です。

CASEMaker's COBOLインタフェース製品、MFCOBOLのDCIはSQLを使用することができません。替わりにデータストレージアクセスを提供します。DCI がある MFCOBOLはCOBOL プログラムとDBMasterファイルシステムにシームレスインタフェースを提供します. アプリケーションとデータベースの変更情報はユーザに見えません。ほかの方面で、デスクトップサポートシステム(DSS), data 保管, または 4GLアプリケーションにDBMasterは全部SQL-based ファイル/ データストレージアクセスを提供します。

CASEMakerのデータベースとMFCOBOLのDCI製品は4GLs パワー、SQLベースの柔軟性を総合してデータベースをアクセスします。スタート性能も提供します。

1.1 その他のマニュアル

*DBMaster*はDBMSマニュアルに完全な設置を提供します、詳細は以下のようにをご参考ください。
DBMasterの権限と機能について:

- DBMasterに関する性能と特性は　“*DBMaster入門編*”　をご覧ください。
- DBMasterの設計、管理、保守についての詳細は、*「データベース管理者参照編」*をご覧下さい。
- DBMasterの管理についての詳細は、*「JServer Managerユーザーガイド」*を参照して下さい。
- DBMasterの環境設定についての詳細は、*「JConfiguration Tool参照編」*をご覧下さい。
- DBMasterの機能についての詳細は、*「JDBA Toolユーザーガイド」*を参照して下さい。
- dmSQLツールに関しては*″dmSQLユーザガイド″*をご覧ください。
- DCI COBOLインターフェースに関しては*″DCI ユーザガイド″*をご参考ください。
- ESQL/Cプログラムに関しては*″ESQL/Cプログラマー参照編″*をご参考ください。
- ODBC APIに関しては*″ ODBCプログラマー参照編 ″*をご覧ください。
- エラーと警告メッセージについて*″エラー・メッセージ参照編″* をご覧ください。

1.2 テクニカルサポート

CASEMakerでは評価期間内で、30日間の無償でのe-mail、電話によるサポートを提供しております。ソフトウェアが登録された際の追加の30日間のサポートも含まれます。延長により合計でソフトウェアのサポートを60日間受けることができます。しかしながらバグ報告については無償サポート期間終了後も引き続きe-mailでの無償サポートを提供いたします。
60日を超えた追加サポートはほとんどの製品で有効です。製品価格の

20%にてサポートを継続して受けることができます。
詳細や価格についてはsales@casemaker.com までお問い合わせください。郵便、電話、e-mailにてお問い合わせいただけるお近くのCASEMakerサポートをこちらからご確認いただけます。
www.casemaker.com/supportCASEMakerサポートスタッフにお問い合わせいただく前に、FAQデータベースをご覧頂くことを推奨いたします。
トラブルシューティングのお電話の際、郵送、e-mailでのお問い合わせの際に以下の情報をお伝えいただくようお願いいたします：

- 製品名とバージョン番号
- レジストレーション番号
- 登録顧客名と住所
- 購入した代理店、購入場所
- コンピュータのプラットフォーム、システム設定
- エラー発生前の操作詳細
- エラーメッセージがある場合は、メッセージとその番号
  その他関連があると思われる情報

1.3 字体規則

本書は、標準の字体規則を使用しているので、簡単かつ明確に読むことができます。手順、例、コマンドライン規則には別の設定があり、インデンテーションにて使用されます。

字体	解説
斜体	斜体は、ユーザー名や表名のような特定の情報を表します。斜体の文字そのものを入力せず、実際に使用する名前をそこに置き換えてください。斜体は、新しく登場した用語や文字を強調する場合にも使用します。
太字	太字は、ファイル名、データベース名、表名、カラム名、関数名やその他同様なケースに使用します。操作の手順においてメニューのコマンドを強調する場合にも、使用します。
キーワード	文中で使用するSQL言語のキーワードは、すべて英大文字で表現します。
小さい英大文字	小さい英大文字は、キーボードのキーを示します。2つのキー間のプラス記号 (+) は、最初のキーを押したまま次のキーを押すことを示します。キーの間のコンマ(,)は、最初のキーを放してから次のキーを押すことを示します。
ノート	重要な情報を意味します。
➲ プロシージャ	一連の手順や連続的な事項を表します。ほとんどの作業は、この書式で解説されます。ユーザーが行う論理的な処理の順序です。
➲ 例	解説をよりわかりやすくするために与えられる例です。一般的に画面に表示されるテキストと共に表示されます。
コマンドライン	画面に表示されるテキストを意味します。この書式は、一般的にdmSQLコマンドやdmconfig.iniファイルの内容の入/出力を表示します。

図 1-1 字体規則表

2 DCIの追加

この章はDCIの追加について説明します。DCI特性、DCI関数を含みます。DCI関数はMF COBOLプログラムに呼ばれます。この関数を有効になるため、ユーザはプロジェクトに**cobolcall.obj**をリンクする必要があります。

2.1 DCI追加の特性

ユーザに**cobolcall.obj**、Cソースコードのようなコボルコールを提供します。ユーザはcobolcall.objを使用してそれを作成します。コボルコールソースコードを編集することは簡単です。

cobolcall ソースコード (cobolcall.c):

```
/*
_____     _  _   Corso  Italia,  178
(_|__  . (_  |_|_ 56125        Pisa
(_|_) |)|(()_)()|| tel.  +39  050 46380
 | |           picosoft@picosoft.it

 (C) 2003

*/
static char rcsid[] = "$Id: cobolcall.c,v 1.1 2003/12/04 14:23:30 picoSoft Exp $";
```

```
extern int DCI_GETENV_MF (char *key, char *val);
extern int DCI_SETENV_MF (char *key, char *val);
extern int DCI_DISCONNECT_MF(char *cdb);
extern int DCI_SET_TABLE_CACHE_MF (char *cStart, char *cNext, char *cPrev);

int DCI_GETENV (char *key, char *val)
{
  return DCI_GETENV_MF (key, val);
}

int DCI_SETENV (char *key, char *val)
{
  return DCI_SETENV_MF (key, val);
}

int DCI_DISCONNECT (char *cdb)
{
  return DCI_DISCONNECT_MF (cdb);
}

int DCI_SET_TABLE_CACHE (char *cStart, char *cNext, char *cPrev)
{
  return DCI_SET_TABLE_CACHE_MF (cStart, cNext, cPrev);
}
```

2.2 DCI 追加の関数

目前、dmmfcbl.lib は以下の四つの DCI関数のみをサポートします。
ユーザはDCI 関数が呼べます:

```
CALL "dci_function_name" USING variable [, variable, …]
```

MFCOBOLプログラムにあります。

DCI_SETENV

この変数を使用する場合、DCI_SETENVを呼ぶ前、文字列にnullを追加することは重要です。
ここには無効な文字列を終了する方法があります：

➲ **例 1:**

```
CALL "DCI_SETENV" USING z"DCI_DATABASE" z"DBSAMPLE4".
CALL "DCI_SETENV" USING z"DCI_LOGIN" z"SYSADM".
```

➲ **例 2:**

```
....
01 command-str1 pic x(50).
01 command-str2 pic x(50).
....
MOVE "DCI_DATABASE"&x"00" TO command-str1.
MOVE "DBSAMPLE4"&x"00" TO command-str2.
CALL "DCI_SETENV" USING command-str1 command-str2.
MOVE "DCI_LOGIN"&x"00" TO command-str1.
MOVE "SYSADM"&x"00" TO command-str2.
CALL "DCI_SETENV" USING command-str1 command-str2.
```

➲ **例3:**

```
....
01 command-str1 pic x(50).
01 command-str2 pic x(50).
....
move spaces to command-str1 command-str2
string "DCI_DATABASE" delimited by size
low-values delimited by size into command-str1
```

```
string "DBSAMPLE4" delimited by size
low-values delimited by size into command-str2
CALL "DCI_SETENV" USING command-str1 command-str2
move spaces to command-str1 command-str2
string "DCI_LOGIN" delimited by size
low-values delimited by size into command-str1
string "SYSADM" delimited by size
low-values delimited by size into command-str2
CALL "DCI_SETENV" USING command-str1 command-str2
```

DCI_GETENV

この関数は環境変数を読むために使用されます。
構文:
CALL “DCI_GETENV” USING “environment variable”, variable

➲ **例:**

```
CALL “DCI_GETENV” USING “DCI_DATABASE”, mf_dci_database
```

DCI_DISCONNECT

この関数はデータベースから切断するために使用されます。

➲ **例:**
MF COBOL プログラムに一つだけの接続があると,以下のコードを使用してデータベースから切断.

```
CALL “DCI_DISCONNECT”.
```

➲ **例2:**
MF COBOL プログラムに一つ以上の接続があると,以下のコードを使用してデータベースから切断.

```
CALL “DCI_DISCONNECT” USING “DBSAMPLE4”
```

DCI_SET_TABLE_CACHE

DCI 前リードデータはクライアントデータバッファにあると、クライアント／サーバ側ネットワークトラフィックを影響します。初期最大前リードバッファは8kb/(レコードサイズ)以下、または5 レコードです。
ユーザのアプリケーションは小さい表を読むこと、8kb/(レコードサイズ)より少ないレコードを読む可能性もあります。例えば,平均レコードサイズは20バイト、総合1000レコードの表に対してDBMaster は400レコード(8kb/20)が読めますが、ユーザのアプリケーションは4 または5レーコードを読んでSTART構文を再呼びます。以下の変数を設置、キャッシュサイズを増加して性能を上げます。この変数を使用するときアプリケーションとデータの操作が性能を下げる可能性もあります。
DCI_CONFIG ファイルに三つのDCI_CACHE変数を設置することができます:

DCI_DEFAULT_CACHE_START –START またはREADに第一のキャッシュリードレコードを設置します。初期最大値は8kb/(レコードサイズ)或いは5レコードです。

DCI_DEFAULT_CACHE_NEXT –読まれた、捨てられたSTART または READに次のキャッシュリードレコードを設置します。初期最大値は8kb/(レコードサイズ)或いは5レコードです。

DCI_DEFAULT_CACHE_PREV –START または READが読まれた、捨てられた後前のレコードにリードレコードを設置します。

初期値は DCI_DEFAULT_CACHE_NEXT/2です.
DCI_CONFIG に変数を設置するとユーザのアプリケーションに全てのテーブルを影響します。

➲ **例:**

```
DCI_DEFAULT_CACHE_START 10
DCI_DEFAULT_CACHE_NEXT 10
DCI_DEFAULT_CACHE_PREV 5
```

テーブルのキャッシュを自動に変更するためSTART 、READ 構文の前変数を設置します.
COBOL コード fragment:

```
…
```

```
WORKING-STORAGE SECTION.
01 CACHE-START PIC 9(5) VALUE 10.
01 CACHE-NEXT PIC 9(5) VALUE 20.
01 CACHE-PREV PIC 9(5) VALUE 30.
…
PROCEDURE DIVISION.
OPEN INPUT IDX-1-FILE
MOVE SPACES TO IDX-1-KEY
CALL "DCI_SET_TABLE_CACHE" USING CACHE-START
CACHE-NEXT
CACHE-PREV
START IDX-1-FILE KEY IS NOT LESS IDX-1-KEY.
PERFORM VARYING IND FROM 1 BY 1 UNTIL IND = 10000
READ IDX-1-FILE NEXT AT END EXIT PERFORM END-READ
DISPLAY IND AT 0101
END-PERFORM
CLOSE IDX-1-FILE
```

3 MFCOBOLのDCI

この章はDBMaster環境でMFCOBOLのDCIの構成と設置情報を説明しますDCI（DBMasterインタフェースの基礎ライブラリDCI）とDCIBench（XMLファイルの生成基礎パーサー）基礎関数でデモンストレーションを実行する情報も提供します。
以下のトピックはこの章に含みます：

- ソフトウェアとハードウェアの必要条件。
- WindowsとUnixプラントフォームの設置命令。
- DBMasterのMFCOBOLのDCI 構成オプション。

3.1 MFCOBOLのDCI概要

MFCOBOLのDCIを使用する時、MFCOBO LにDCIライブラリを含むことが重要です。これをXMLファイルの生成に有効になります。
伝統なCOBOLファイルシステムとデータベースがデータを含むがこれは違います。データベースは伝統なファイルシステムより強力と信頼性を持ちます、その上、ソフトウェア、ハードウェアキャッシュからのデータを回復することに有効です。データの完全性を確保するため、DBMasterRDBMSはドメイン、カラム、強制テーブルなどのアクションを提供します。

ファイルシステムとデータベース

データベースとCOBOL索引ファイルでデータをストアする方法があります。以下の表が違うデータ配置、ほかのと一致性にする方を表示しま

す。

COBOL INDEXED FILE SYSTEM OBJECT	DATABASE OBJECT
Directory	Database
File	Table
Record	Row
Field	Column

図3-1 COBOL とデータベースオブジェクト構成

索引ファイル操作はCOBOLのレコードまたはデータベースのカラムに完成します。倫理的に、COBOL索引ファイルはデータベース表を表現して、COBOLのレコードは行を表現して、フィールドはテーブルカラムを表現するということです。テーブルカラムと整数、文字、データなどような特別なデータタイプを結合する時データはCOBOLに様々な定義型があります。

➲ **例**

以下のフォーマットでCOBOL レコードが定義されます:

```
terms-record.
        03      terms-code      PIC 999.
        03      terms-rate      PIC s9v999.
        03      terms-days      PIC 9(2).
        03      terms-descript  PIC x(15).
```

上の例示のCOBOLレコードは以下のようにデータベースに表示されます、各の行はCOBOL 01レコードの例です。

TERMS_CODE	TERMS_RATE	TERMS_DAYS	TERMS_DESCRIPT
234	1.500	10	net 10
235	1.750	10	net 10
245	2.000	30	net 30
255	1.500	15	net 15
236	2.125	10	net 10
237	2.500	10	net 10
256	2.000	15	net 15

図3-2 COBOL レコードがデータベース行に転換

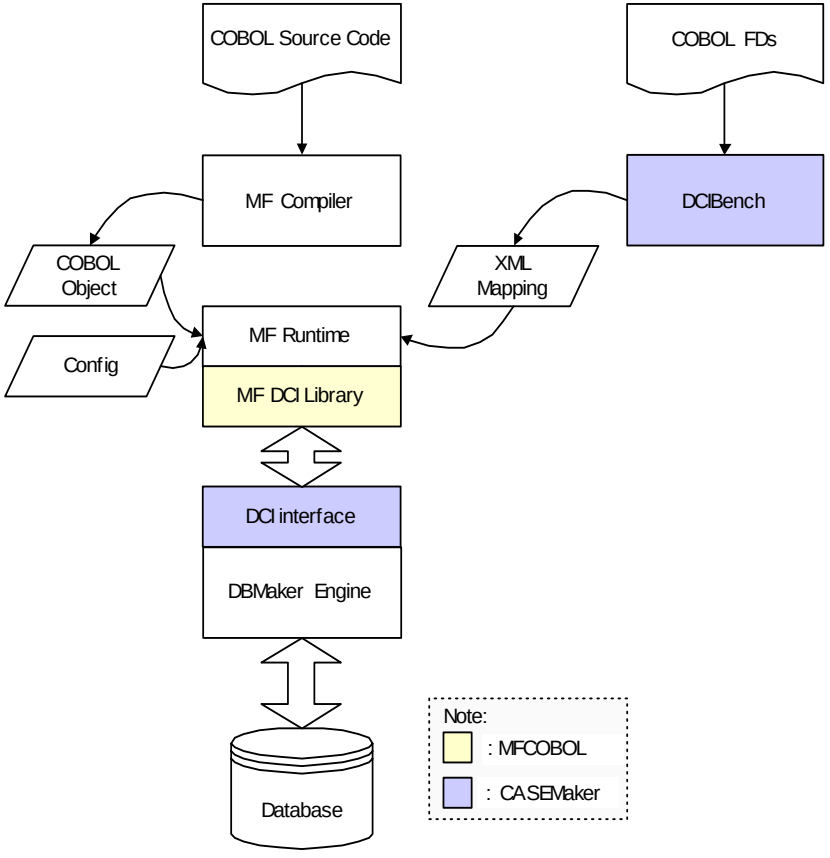

図3-3 データフローチャート

3.2 システムの必要条件

DBMasterのMFCOBOLのDCIはMicro Focus NetExpressをリンクしなければなりません。NetExpress 3.1以上のが使用されます。README.TXTファイルはMFCOBOLのDCIディレクトリリストにあって製品と一緒に送ります。
以下のプラントフォームをサポートします:

- Windows 98/ME/NT/2000/XP

- Linux 2.2

以下のソフトウェアをインストールしなければなりません:

- DBMasterversion 4.1
- Micro Focus NetExpress 3.1 or later
- IE 6.0 or later

3.3 セットアップガイド

DCIBenchを配置する前Net Express 3.1と DBMasterversion 4.1をインストールしなければなりません。DBMasterのインストールはQuick Startをご参考ください。

WindowsでDCIのセットアップ

Windows環境でDCIのセットアップの詳細は以下のプロシージャが説明します。

➲ **Windows**でMFCOBOL のDCI をセットアップします**:**

1. NetExpress 3.1をインストール。
2. DBMaster4.1をインストール。
3. MFCOBOL にDBMasterライブラリを受け取る、ワークディレクトリをコピー。

   ➲ 例

   ```
   copy dmdcic.lib d:\mfdcilib
   copy dmmfcbl.lib d:\mfdcilib
   copy dmapi41.lib d:\mfdcilib
   ```

4. COBOL プログラムの初めに以下の構文を追加してください

   ```
   $SET CALLFH "DBMASTERINTF"
   ```

5. MFCOBOLプロジェクトにMFCOBOLのDCIで生成します。
   a) MicroFocus NetExpressを実行。
   b) 新規空きプロジェクトを作成。

c) **Project -> Add file to project**を選択して 、COBOLソースコードを追加。

d) **Project-> Create Package file**、**Executable File**を選択**(EXE)**。

e) **tempate.int**に右のボタンをクッリクして, **Remove file build type**を選択 。

f) **Project-> Build Setting** 、**Link**を選択してカテゴリー**Advanced**に変更 。

g) **Link with these LIBs**にMFDCIライブラリを追加:

```
d:\mfdcilib\*.lib
```

h) **Project**を選択->コンパイルを**Rebuild** 、プロジェクトを生成。

6. **Type of Build**の設置によってdebug\ or release\ directoryにtemplate.exeがあります。
7. COBOL FD 定義からDCIBenchを使用してXMLファイルを生成します。
8. XMLファイルをMFCOBOLが実行されたディレクトリにコピー。
9. NetExpressまたは DOS コマンドボックスにcommand box, DCI_CONFIG=d:\mfdci.cfgをセットします。
10. mfdci.cfg ファイルをエディット。

➲ **例:**

```
edit d:\mfdci.cfg
DCI_DATABASE    DBSAMPLE4
DCI_LOGIN       KSYSADM
DCI_PASSWD
```

11. MFCOBOL のDCIが実行できます。
12. COBOLプログラムにDCI_SETENV, DCI_GETENV を使用すると:

a) Cobolcall.obj をd:\mfdcilibにコピー。

b) **Project -> Build Setting** 、**Link**を選択してカテゴリー**Advanced**に変更。

c) **Link with these OBJs**にMFDCIライブラリを追加**:**

```
d:\mfdcilib\cobolcall.obj
```

d) **Project** -> **Rebuild**を選択->プロジェクトをコンパイル、生成。

UNIXでDCIをセットアップ

UNIX環境でDCIのセットアップの詳細は以下のプロシージャが説明します。

➲ **Linux** でMFCOBOL のDCI をセットアップします**:**

1. MFCOBOL のDCIを Linuxにインストール:

   a) $COBDIR と LD_LIBRARY_PATH to MF ファイルパスを設置。

   ➲ **例:**

```
setenv COBDIR /usr/local/mfcobol
setenv LD_LIBRARY_PATH /usr/local/mfcobol/coblib:$LD_LIBRARY_PATH
```

   b) $COBDIR/coblib/liblist を編集してgcclib/OS/version ディレクトリのパスを修正。

   例: linux システムはmandrakeなら 、gcc バージョンは 2.95.3
   change i/usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/....
   i/usr/lib/gcc-lib/i586-mandrake-linux/2.95.3/

2. DBMaster4.1をインストール。
3. 以下のファイルパス(ex: DCIDIRLIB)に**Makefile**を編集します。

   ➲ **例:**

```
vi Makefile
DCIDIRLIB=/home/DBMaster/4.1/lib
CFLAGS=-Wall -DPROTOTYPE -I$(SRCDIR) $(INCL) -I$(COBDIR)/src
CC=cc
OOPS=-I OOPS
INCFLAGS1=-I SESSION -I COBWIN1 -I fhutil -I DSRTNS -I CICS -I
CCITCP -I CCINAMPU $(OOPS)
LDFLAGS=
all:rtsdbmmf
rtsdbmmf:
cob -vgxo rtsdbmmf -e "" +F DBMASTERINTF +F DCI_SETENV +F
DCI_GETENV +F DCI_DISCONNECT $(INCFLAGS1) $(INCFLAGS2) \
```

```
$(LDFLAGS) $(DCIDIRLIB)/libdmmfcbl.a $(DCIDIRLIB)/libdmdcic.a
$(DCIDIRLIB)/libdmapic.a
clean:
rm rtsdbmmf
```

4. '**make**'を入力して実行タイムrtsdbmmfを生成します。
5. COBOLプログラムの初めに以下の構文を追加してください。

```
$SET CALLFH "DBMASTERINTF"
```

6. COBOLファイルをコンパイルして.int ファイルを生成します。

➲ **例:**

```
cob template.cbl
```

7. COBOL FD 定義からDCIBenchを使用してXMLファイルを生成します。
8. DCI_CONFIG 環境変数を設置します。

➲ **例:**

```
setenv DCI_CONFIG $HOME/mfdci.cfg
```

9. DCIのMFCOBOLユーザガイドに説明するようにmfdci.cfgファイルを編集します。

➲ **例:**

```
vi $HOME/mfdci.cfg
DCI_DATABASE    DBSAMPLE4
DCI_LOGIN       SYSADM
DCI_PASSWD
```

10. rtsdbmmf を使用してCOBOL .int プログラムを実行します。

➲ **例:**

```
rtsdbmmf template.int
```

3.4 DCI の設置

環境変数設置

環境変数に値の設置を通じて構成ファイル、違うアドレスを行う必要があ

ります。 DCI_CONFIGと呼ばれます.

```
DCI_CONFIG = Work Directory\*.cfg

PATH = $NetExpress_Install_Dir\Micro Focus\Net Express 5.0\Base\Bin
```

*.cfg は構成ファイルです。データベース名、ユーザ名、パスワードを含みます. ユーザはDCI_CONFIG 環境変数に構成ファイルの全パス名を設置しなければなりません。

DCI構成の基礎

DCI_CONFIGファイルは環境変量によって決定するディレクトリにあります。(詳細は“**配置ファイル変数**”をご参考ください)。DCIを正確にスタートするため、DCI_CONFIGファイルに重要な設置があります。データベースにデータを表示するためDCI_CONFIGファイルはDCIに引数を設置します。以下の配置変数は設置する必要があります。

- DCI_DATABASE
- DCI_LOGIN
- DCI_PASSWD

➲ **例：**

基礎 DCI_CONFIG ファイルは以下のように表示します。

```
DCI_LOGIN SYSADM
DCI_PASSWD
DCI_DATABASE DBMaster_Test
DCI_XFDPATH /usr/DBMaster/Dictionaries
```

DCI_DATABASE

DCIからのトランザクションがDCI_DATABASEに定義されます。データベースは DBMasterセットアップにあることを確保します。データベース名は大小文字を区別します。長さは8文字より少ないです。

DBMaster_Testというデータベースがあると：

➲ **構文**

以下のエントリは配置ファイルに含まれます。

```
DCI-DATABASE DBMaster_Test
```

NOTE 詳細は“*DCI_DATABASE*をご参考ください。

DCI_LOGIN

COBOLアプリケーションがデータベースにオブジェクトのアクセスを許すことを確保するため、ユーザ名が必要です。DCI.を使ってDCI_LOGINはCOBOLアプリケーションにユーザ名を設置します、データベースのアクセスを確保するために変数はSYSADMに設置されます。この値はほかの名が設置できます。詳細は"DCI_LOGIN"参考ください。

➲ 構文

データベースはSYSADMを通じて指定のユーザ名を接続するため, 以下のをDCI配置ファイルに設置しなければなりません:

```
DCI_LOGIN SYSADM
```

DCI_PASSWD

DCI_LOGIN 変数を通じてユーザ名を指定されると, データベースがこれを結合します。

SYSADMにパスワードがありません。これはDBMasterの初期値です。しかし、変更することができます。説明情報を正確に理解するためデータベース管理者をご参考ください、詳細は"DCI_PASSWD"をご覧ください。

➲ **構文**

データベース情報はSYSADMに設置すると配置ファイルが以下のように表示します。

```
DCI_PASSWD
```

DCI ライブラリー

プロジェクトにDBMaster DCIライブラリーを追加する時、ユーザはこのプロジェクトはファイルが探せることを確保します。ユーザはファイルのフールパスまたはプロジェクトディレクトリにこのファイルをコピーすることができます。

Lib Name	Lib From
dmdcic.lib	DBMaster はDCIにLibを提供.

dmmfcbl.lib	DBMaster はDCIにLibを提供.
dmapi43.lib	DBMaster directory\lib ディレクトリに,例えば C:\DBMaster\4.3\lib.

3.5 MFCobol プログラムの注意点

1. $SET CALLFH "DBMASTERINTF"
To run your MFCOBOL program with MFDCIでMFCOBOLを実行する時,プログラムの前以上のラインを追加しなければなりません。

➲ **例:**

```
$SET CALLFH "DBMASTERINTF"
IDENTIFICATION DIVISION.
PROGRAM-ID.  template.
...
```

2. DCI_SETENVを呼ぶ前、文字列にnullを追加します。
There are several ways to null terminate the stringここには方法があります；

➲ **例 1:**

```
CALL "DCI_SETENV" USING z"DCI_DATABASE" z"DBSAMPLE4".
CALL "DCI_SETENV" USING z"DCI_LOGIN"  z"SYSADM".
```

➲ **例2:**

```
....
    01 command-str1 pic x(50).
    01 command-str2 pic x(50).
....
    MOVE "DCI_DATABASE"&x"00" TO command-str1.
    MOVE "DBSAMPLE4"&x"00" TO command-str2.
```

```
    CALL "DCI_SETENV" USING command-str1 command-str2.
    MOVE "DCI_LOGIN"&x"00" TO command-str1.
    MOVE "SYSADM"&x"00" TO command-str2.
    CALL "DCI_SETENV" USING command-str1 command-str2.
```

➲ **例3:**

```
....
    01 command-str1 pic x(50).
    01 command-str2 pic x(50).
....

    move spaces to command-str1 command-str2
    string "DCI_DATABASE" delimited by size
    low-values delimited by size into command-str1

    string "DBSAMPLE4" delimited by size
    low-values delimited by size into command-str2

    CALL "DCI_SETENV" USING command-str1 command-str2

    move spaces to command-str1 command-str2
    string "DCI_LOGIN" delimited by size
    low-values delimited by size into command-str1

    string "SYSADM" delimited by size
    low-values delimited by size into command-str2

    CALL "DCI_SETENV" USING command-str1 command-str2
```

4 コンパイルと実行タイムオプション

この章はMFCOBOLのDCIの配置設定を説明します。この配置設定はどんなファイルシステムを使用するかを指定します。

4.1 MFCOBOLのDCIシステムを使用

COBOLアプリケーションで既存ファイルはDCIBench配置ファイルに定義される各のファイルと結合します。COBOLアプリケーションのDCIで新規ファイルを作成する時、ユーザはCOBOLプログラム前に以下の構文を挿入しなければなりません。

➲ **構文**

```
$SET CALLFH "DBMASTERINTF
```

4.2 MFCOBOL ディフォルトシステムを使用

COBOLプログラムの第1ラインに"$SET CALLPFH "DBMASTERINTF"を追加した後、COBOLに開けるファイルはDCIファイルに転換します。ユーザはファイルを一般なMFCOBOLファイルように開くと、以下のラインを書いてください。

➲ **構文**

```
DCI_STANDARD_FILE
```

➲ **例**

If file1 and file2 are MFCOBOL files

```
DCI_STANDARD_FILE  file1
DCI_STANDARD_FILE  file2
```

4.3 ビューを使用

DCI は表を替わってDBMasterビューの使用を許せます。DCIユーザは手動でビューを作成して以下の制限も熟知しなければなりません。

- シングルテーブルのみに提供される。
- 原テーブルにプロジェクトカラムだけ (式, 総合,, UDF, ...などがありません)。
- group by, distinct, union, join…がありません。
- 簡単な"Where" 述語 (sub-query 不許可)。

4.4 DCI_WHERE_CONSTRAINTを使用

START オプションを成功するためDCI_WHERE_CONSTRAINT は追加の WHERE 条件句を指定する時使用されます。

➲ **例**

Aから都市の名をクエリすると, 以下のコードを追加します：

```
WORKING-STORAGE SECTION.
01  dci_where_constraint pic x(4095) is external.
...

PROCEDURE DIVISION.
...

* to pecify dci_where_constraint
move low-values to dci_where_constraint
  open i-o idx-1-file
  move "city_name = 'a%'" to dci_where_constraint
  inspect dci_where_constraint replacing trailing spaces by low-values.
```

```
move spaces to idx-1-key
start idx-1-file key is not less idx-1-key
....

 * to remove dci_where_constraint

   move low-values to dci_where_constraint
   move spaces to idx-1-key
   start idx-1-file key is not less idx-1-key
   ...
```

5 配置ファイル変数

この章はDCIデータの範囲をリストします、COBOLデータタイプはDBMasterデータタイプにどうマップするかの表もあります。配置ファイル変数はDCIの状態の変更が使用されます、そして、DCI_CONFIGというファイルにストアされます。

5.1 DCI_CONFIG 変数の設置

DCI_CONFIGという環境変数の設置を通じて配置ファイルに違うアドレスを設定することは可能になります。この値は配置変数のフールパス名、ディレクトリを指定します。DCIはDCI_CONFIGというファイルを探します。このファイルは配置変数にストアされると表示できません、DCIはエラーが出ませんが、配置変数が表示しないことと仮定します。この変数はCOBOL実行タイム配置ファイルに設置されます。

➲ **構文**

DCI 実行タイムに, DCIBenchはディレクトリc:\etc\testにDCI_CONFIGという配置ファイルを読みます。

```
set DCI_CONFIG=c:\etc\test
```

DCI_CASE

COBOL ファイル名は大小文字を区別しませんがテーブル名は大小文字を区別します。
この設定変数はファイル名がテーブル名に転換することを決定します。
この変数は*lower* を設定すると、ファイル名が小文字でテーブル名に転換

するという意味です。*upper* を設定すると、ファイル名が大文字でテーブル名に転換するという意味です、*ignore*を設定すると、ファイル名が大文字または小文字でテーブル名に転換します。DCI_CASE の初期値は*lower* です.。ファイル名はe DBCSワードならDCI_CASEは*ignore*を設置します。

➲ **例**

```
DCI_CASE IGNORE
```

DCI_COMMIT_COUNT

The DCI_COMMIT_COUNT 設定変数はどんな条件でCOMMIT WORK 操作が実行できることを説明します。ここには二つの可能値があります。 0 と*n*>.

DCI_COMMIT_COUNT=0

自動コミットできません (初期値)。

DCI_COMMIT_COUNT=<*N*>

DCIがWRITE, REWRITE, ANDDELETE操作が<*n*>になるとCOMMIT WORK構文が実行始めます。ファイルが"output" または"exclusive"に開かれるだけ条件でこのルールが適用します。

DCI_DATABASE

DCI_DATABASE はDBMasterのセットアップで確立したデータベース名を指定します。

➲ **例 1**

データベース名はDBMaster_Testなら以下のエントリは配置ファイルに含みます。

```
DCI_DATABASE DBMaster_Test
```

➲ **例 2**

データベース名は知らない時、実行タイムに動的にこれを設定する必要があります。COBOLプログラムに以下のように類似なコードを書く可能

にします。以下のコードはOPEN構文の実行を終わった前実行します。

```
CALL "DCI_SETENV" USING "DCI_DATABASE" , "DBMaster_Test"
```

➲ **例 3**

違ったデータベースにテーブルをアクセスすると、DCI_DATABASEを使用して複数のデータベースが接続できてデータベースの中に動的に転換することもできます。

```
* connect to DBSAMPLE4 to access idx-1-file
CALL "DCI_SETENV" USING "DCI_DATABASE" "DBSAMPLE4"
....
open output idx-1-file
....
* connect to DCIDB to access idx-2-file
CALL "DCI_SETENV" USING "DCI_DATABASE" "DCIDB"
....
open output idx-2-file

* to switch dynamically to DBSAMPLE4 connection
CALL "DCI_SETENV" USING "DCI_DATABASE" "DBSAMPLE4"
close idx-1-file
...
```

DCI_DATE_CUTOFF

この変数は二桁の値で二桁の年を設置します。DCI_DATE_CUTOFFに初期値は20.です。そうすると、2000は二桁値で20より小さいです。1900は二桁値で20より大きいです。COBOL日付99/10/10は1999/10/10に訳されます。COBOL日付00/02/12は2000/02/12に解釈されます。.

DCI_DEFAULT_RULES

WHENのディフォルト管理方法は複数定義ファイルにあります。BEFORE構文は$WHEN条件がマッチされる時テーブルが開けることを表示します。POST構文はCOBOLアプリケーションがマルチ定義ファイルを開く時、全てのテーブルが開くことを表示します。
可能値は:
POST or COBOL

BEFORE or DBMS

DCI_DEFAULT_TABLESPACE

この変数はディフォルト表領域を設置します。この表領域はデータベースに存在します。この変数で指定される表領域がないと、新規テーブルはユーザの表領域に作成します。

DCI_DISCONNECT

DCI_DISCONNECT はデータベース接続から切断する時使用されます。

➲ **例 1**

cobolプログラムに一つだけの接続があれば、以下のコードを使用してデータベースから切断。

```
CALL "DCI_DISCONNECT".
```

➲ **例 2**

cobolプログラムに複数の接続があれば、以下のコードを使用してデータベースから切断

```
CALL "DISCONNECT" USING "DBSAMPLE4"
```

DCI_DUPLICATE_CONNECTION

同じCOBOLプロセス、違ったデータベース接続で同じ表を開く DCI_DUPLICATE_CONNECTIONはロックを受け取ります。同じCOBOLアプリケーションに同じレコードを二回ロックします。
初期値はoff です(0)。

➲ **例**

違ったデータベース接続でCOBOLアプリケーションは表にロックを受け取ります:

```
DCI_DUPLICATION_CONNECTION 1
```

DCI_GET_EDGE_DATES

ユーザはDATEフィールドにlow/high　値を入力すると

DCI_SET_EDGE_DATEはこの値を指定します。ユーザはCOBOLプログラムにDATEフィールドにlow/highを入力する時、例えば、00010101/99991231を入力して、データはCOBOLの low/high 値 00000000/99999999が表示されます。この変数は使用される時、DATEフィールドのlow/high　値は00010101/99991231を表示します。このルールはDATEフィールドがキーの部分である時も適用できます。初期値はoffです。

➲ **例**

以下のラインはdci.cfgファイルに追加しなければなりません:

```
DCI_GET_EDGE_DATES 1
```

DCI_INV_DATE

不適なデータフォーマットがデータベースに書かれた時、問題が避けるためこの変数を使用して有効な日付(2000/02/31のように)を設定します。初期値は99991230（十二月30th, 9999）です。

DCI_LOGFILE

この変数はDCIログファイルのパス名を指定します。*/tmp* ディレクトリの *dci_trace.log* logファイルはデバッグすることに使用されます。ログファイルを使用してDCI.の性能を下げます、だから、配置ファイルにこの変数を追加しないほうがいいです。

➲ **例**

Config.ini ファイルにログファイルを入力:

```
DCI_LOGFILE /tmp/dci_trace.log
```

DCI_LOGIN

データベースシステムを接続するため、DCI_LOGINはユーザ名を指定する変数です。初期値がありませんが、ユーザ名は指定しないとログインが使用されません。

DCI_LOGINでユーザ名を指定すると、データベースRESOURCE権限があるべきです。その上、ユーザは既存のデータテーブルの許可を取る必要

があります。新規ユーザはJDBA Tool, dmSQLを使用して作成されます。

NOTE *新規ユーザの作成についての詳細は, JDBA ツールユーザガイドまたはデータベース管理者をご参考ください。*

➲ **例**

Config.cfgファイルにユーザ名JOHNDOEを入力します:

```
DCI_LOGIN JOHNDOE
```

DCI_JULIAN_BASE_DATE

この変数はJulianデータ計算に基礎データを設置します。フォーマットはYYYYMMDDで初期値はJanuary 1st, 1 ADです。

DCI_LOGTRACE

この変数はトレースログに違ったレベルを指定します。

- 0: トレースがない
- 1: トレースに接続
- 2: i/oトレースをレコード
- 3: フールトレース
- 4: 内部デバッグトレース

DCI_MAPPING

この変数はDCIシステムにXMLディレクトリでファイル名を結合します。その上、一つのXMLディレクトリは複数のファイルと結合することができます。"*pattern*"は有効なファイル名から組みます。ウィルドカード"*"、疑問符"?"を含んでマルチタイムに使用されます。

➲ **構文**

```
DCI_MAPPING [pattern = base-xml-name]  ...
```

➲ **例 1**

"CUST*1"と base-XML-name "CUSTOMER" 生成するファイル名：
"CUST01", "CUST001", "CUST0001" と "CUST00001" がXMLディレクトリ

ファイル“customer.XML”と結合します。

```
DCI_MAPPING CUST*1=CUSTOMER ORD*=ORDER “ord cli*=ordcli”
```

➲ **例 2**

“CUST????”と base-XML-name “CUST”が生成するファイル名：“CUSTOMER”, “CUST0001” がXMLディレクトリファイル“cust.XML”と結合します。

```
DCI_MAPPING  CUST????=CUST
```

DCI_MAX_ATTRS_PER_TABLE

DBMasterテーブルは252カラムがありました。COBOLファイルは252以上のフィールドがありますので全部のフィールドを表のカラムにマップできません。DCIはDCI_MAX_ATTRS_PER_TABLE配置変数を提供して複数のテーブルに分割されるテーブルのフィールド数を定義します。マルチ結果テーブルはユニコードがないとだめです。だから、DCIは連続オーダー(A, B, C,など.)によって下線を引く文字(_)を使ってテーブル名を追加します。

➲ **例 1**

A COBOL ファイルは300フィールドがありました、以下の構文:

```
SELECT FILENAME ASSIGN TO “customer”
```

➲ **構文**

dci.**cfg** ファイルに以下のラインを追加しなければなりません:

```
DCI_MAX_ATTRS_PER_TABLE = 100.
```

➲ **例 2**

以下の名で三つの表を作成します:

```
customer_a
customer_b
customer_c
```

DCI_MAX_BUFFER_LENGTH

DCI_MAX_BUFFER_LENGTHはcobolデータをマルチデータベーステーブルに分割します。カットオフ値はバッファ長さを通じてどこのテーブル

が分割するかを決定します。初期値は4096です。

➲ **例 1**

A COBOL レコードサイズは9000バイトです、以下の構文：

```
SELECT FILENAME ASSIGN TO "customer"
```

➲ **構文**

ci.**cfg** ファイルに以下のラインを追加しなければなりません:

```
DCI_MAX_BUFFER_LENGTH 3000
```

➲ **例 2**

以下の名で三つの表を作成します:

```
customer_a
customer_b
customer_c
```

DCI_MAX_DATE

不適なデータフォーマットがデータベースに書かれた時、問題が避けるためこの変数を使用してhigh-value、日付を設定します。初期値は99991231（十二月31th, 9999）です。

DCI_MIN_DATE

不適なデータフォーマットがデータベースに書かれた時、問題が避けるためこの変数を使用してlow-value、0、日付を設定します。初期値は00010101（一月1th, 1AD）です。

DCI_NULL_ON_ILLEGAL_DATA

DCI_NULL_ON_ILLEGAL_DATA は非法COBOL データが保存される前データベースに転換されることを決定します。値1になると全部の非法データを転換します。初期値0は以下の情報が起こします：

- 非法な LOW-VALUES:最小値 (0 または - 99999...) 或いは DCI_MIN_DATE の初期値をストア.
- 非法な HIGH-VALUES:最大値 (99999...) 或いはDCI_MAX_DATE の初期値をストア.

- 非法な SPACES: 0値としてストア(または DCI_MIN_DATE,データフィールド).
- 非法な DATE 値: DCI_INV_DATE初期値としてストア.
- 非法な TIME: DCI_INV_DATE 初期値としてストア.
- キーフィールドの非法なデータがいつも転換されます.

DCI_PASSWD

DCI_LOGIN変数でユーザ名が指定されると、データベースユーザはあります。このデータベースユーザにパスワードを設置する必要があります。DCI_PASSWDを使用して設置します。

➲ **例 1**

このデータベースユーザにパスワードがSUPERVISORを指定すると、配置ファイルに以下のを指定しなければなりません：

```
DCI_PASSWD SUPERVISOR
```

➲ **例 2**

ユーザはプログラムを実行するパスワードが認められます。以下の応対によってDCI_PASSWD変数を設置しなければなりません：

```
ACCEPT RESPONSE NO-ECHO.
CALL “DCI_SETENV” USING  "DCI_PASSWD" , RESPONSE.
```

環境変数を書く、読むため、ローカルAPIを提供すべきです。

➲ **構文 1**

COBOL プログラムに環境変数を書く、更新する時この構文を使用します。

```
 CALL “DCI_SETENV” USING “environment variable”, value.
```

➲ **構文2**

この構文はCOBOLプログラムに使用して環境変数を読みます。

```
 CALL “DCI_GETENV” USING “environment variable”, value.
```

DCI_Standard_File

この変数はユーザが一般MFCOBOLファイルシステムフォーマットにファ

イルを開くことを許します。.

➲ **構文**

```
DCI_STANDARD_FILE
```

➲ **例**

If file1とfile2はMFCOBOLファイルなら：

```
DCI_STANDARD_FILE  file1
DCI_STANDARD_FILE  file2
```

DCI_SETENV

この変数を使用する時、ユーザはDCI_SETENVをコールする前文字列にnull値を追加することを注意してください。
文字列にnull値を追加するためここには方法があります：

➲ **例 1**

```
CALL "DCI_SETENV" USING z"DCI_DATABASE" z"DBSAMPLE4".
CALL "DCI_SETENV" USING z"DCI_LOGIN"  z"SYSADM".
```

➲ **例 2**

```
....
01 command-str1 pic x(50).
01 command-str2 pic x(50).
....
MOVE "DCI_DATABASE"&x"00" TO command-str1.
MOVE "DBSAMPLE4"&x"00" TO command-str2.
CALL "DCI_SETENV" USING command-str1 command-str2.
MOVE "DCI_LOGIN"&x"00" TO command-str1.
MOVE "SYSADM"&x"00" TO command-str2.
CALL "DCI_SETENV" USING command-str1 command-str2.
```

➲ **例 3**

```
....
01 command-str1 pic x(50).
01 command-str2 pic x(50).
....
move spaces to command-str1 command-str2
string "DCI_DATABASE" delimited by size
```

```
low-values delimited by size into command-str1

string "DBSAMPLE4" delimited by size
low-values delimited by size into command-str2

CALL "DCI_SETENV" USING command-str1 command-str2

move spaces to command-str1 command-str2
string "DCI_LOGIN" delimited by size
low-values delimited by size into command-str1

string "SYSADM" delimited by size
low-values delimited by size into command-str2

CALL "DCI_SETENV" USING command-str1 command-str2
```

DCI_USEDIR_LEVEL

この変数> 0を設定すると、ディレクトリを使用して表名を追加します。

➲ **例1**

以下のラインはイコール： /usr/test/01/clients 01clients

```
DCI_USEDIR_LEVEL 1
```

➲ **例2**

以下のラインはイコール： /usr/test/01/clients test01clients

```
DCI_USEDIR_LEVEL 2
```

➲ **例3**

以下のラインはイコール： /usr/test/01/clients usrtest01clients

```
DCI_USEDIR_LEVEL 3
```

DCI_USER_PATH

DCIはファイルを探す時、DCI_USER_PATHはユーザ名が指定できます。
The user argument can be a period (.) with regard to the files, or the name of a user on the system.

➲ **構文**

```
DCI_USER_PATH user1 [user2] [user3] .
```

OPEN構文のタイプはこのセットの結果を決定します。

OPEN STATEMENT	DCI_USER_PATH	DCI SEARCH SEQUENCE	RESULT
OPEN INPUT or OPEN I/O	Yes	1-list of users in USER_PATH 2-the current user	The first valid file will be opened.
OPEN INPUT or OPEN I/O	No	The user associated with DCI_LOGIN.	The first file with a valid user/file-name will be opened.
OPEN OUTPUT	Yes or no	Doesn't search for a user.	A new table will be made for the name associated with DCI_LOGIN.

Figure 5-1 Types of OPEN Statements

DCI_XMLPATH

DCI_XMLPATHはデータ辞書を保存するディレクトリ名を指定します。初期値はカレントディレクトリです。

➲ **例 1**

*/usr/DBMaster/Dictionaries*にデータ辞書をストアするため、配置ファイルに以下のエントリを含みます。

```
DCI_XMLPATH /usr/DBMaster/Dictionaries
```

➲ **例 2**

複数のパスを指定する必要があると、違ったディレクトリがスペースで区切ります。

```
DCI_XMLPATH /usr/DBMaster/Dictionaries /usr/DBMaster/Dictionaries1
```

➲ **例 3**

WIN-32 環境で,引用符を使って"embedded spaces"が指定されます。

```
DCI_XMLPATH c:\tmp\xmllist "c:\my folder with space\xmllist"
```

*<filename>*_RULES

マルチ定義ファイルのディフォルト管理です、実際のファイル名は *<filename>*を替わります

➲ **例**

以下のコマンドが使用されるときCLIENTファイルを除いて全てのファイルはPOSTルールを使用します。

```
DCI_DEFAULT_RULES        POST
CLIENT_RULES             BEFORE
```

DCI TABLE CACHE 変数

クライアントデータバッファのDCI予めリードデータはクライアント/サーバトラフィックを減少します。初期値は8kb/(レコードサイズ)または5レコードより小さいです。

ユーザのアプリケーションが8kb/(レコードサイズ)より小さいレコードと小さい表を読む可能性があります。例えば,平均レコードサイズは20バイト、総合は1000レコードの表にDBMasterは400レコード(8kb/20)を読めますがユーザのアプリケーションは4、5レコードだけ読みます。この場合で、以下の変数が性能を上げることができます。これを使用する時、ネットワークの通行を増加しますので、性能を下げる可能性があります。

DCI_CONFIGファイルに三つのDCI_CACHE 変数があります:

- DCI_DEFAULT_CACHE_START –START またはREADに第一のリードレコードがキャッシュを設置します、初期最大値は8kb/(レコードサイズ) 或いは5レコードです。
- DCI_DEFAULT_CACHE_NEXT – 次のリードレコードを設置します。初期最大値は8kb/(レコードサイズ) 或いは5レコードです
- DCI_DEFAULT_CACHE_PREV –キャッシュした前のレコードにリードレコードを設置します。

初期値は DCI_DEFAULT_CACHE_NEXT/2です。

DCI_CONFIGを設置するとユーザアプリケーションの表を影響します。

➲ **例**

```
DCI_DEFAULT_CACHE_START        10
DCI_DEFAULT_CACHE_NEXT         10
```

```
DCI_DEFAULT_CACHE_PREV          5
```

自動にテーブルにキャッシュを変更するためSTART 、READ構文の前に変数を設置します。

COBOL コードフラグメント:

```
…
WORKING-STORAGE SECTION.
       01 CACHE-START PIC 9(5) VALUE 10.
       01 CACHE-NEXT  PIC 9(5) VALUE 20.
       01 CACHE-PREV  PIC 9(5) VALUE 30.
…
PROCEDURE DIVISION.
      OPEN INPUT IDX-1-FILE
           MOVE SPACES TO IDX-1-KEY
           CALL "DCI_SET_TABLE_CACHE" USING CACHE-START
                                            CACHE-NEXT
                                            CACHE-PREV
           START IDX-1-FILE KEY IS NOT LESS IDX-1-KEY.
           PERFORM VARYING IND FROM 1 BY 1 UNTIL IND = 10000
              READ IDX-1-FILE NEXT AT END EXIT PERFORM END-READ
              DISPLAY IND AT 0101
           END-PERFORM
           CLOSE IDX-1-FILE
```

5.2 マルチデータベースをマップ

違ったファイルの指定またはCOBOL ファイル接頭辞によるDCI_MAPのデータベースの表を参考する可能性があります。

➲ **例**

DBSAMPLE4 (初期値), DBCED, DBMULTIの表**idx1**を参考して, 以下のをDCI_CONFIG 配置ファイルに設定してください：

```
DCI_DB_MAP      /usr1/CED       DBCED
DCI_DB_MAP      /usr1/MULTI     DBMULTI
```

違ったファイルの指定を通じて表**idx-1**を読みます:

```
...
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
```

```
        SELECT IDX-1-FILE
        ASSIGN TO DISK "/usr/CED/IDX1"
        ORGANIZATION IS INDEXED
        ACCESS IS DYNAMIC
        RECORD KEY IS IDX-1-KEY.

        SELECT IDX-2-FILE
        ASSIGN TO DISK "/usr/MULTI/IDX1"
        ORGANIZATION IS INDEXED
        ACCESS IS DYNAMIC
        RECORD KEY IS IDX-2-KEY.

        SELECT IDX-3-FILE
        ASSIGN TO DISK "IDX1"
        ORGANIZATION IS INDEXED
        ACCESS IS DYNAMIC
        RECORD KEY IS IDX-3-KEY.

    DATA DIVISION.
     FILE SECTION.
     FD  IDX-1-FILE.
     01  IDX-1-RECORD.
         03  IDX-1-KEY                    PIC X(10).
         03  IDX-1-ALT-KEY.
             05  IDX-1-ALT-KEY-A          PIC X(30).
             05  IDX-1-ALT-KEY-B          PIC X(10).
         03  IDX-1-BODY                   PIC X(50).

     FD  IDX-2-FILE.
     01  IDX-2-RECORD.
         03  IDX-2-KEY                    PIC X(10).
         03  IDX-2-ALT-KEY.
             05  IDX-2-ALT-KEY-A          PIC X(30).
             05  IDX-2-ALT-KEY-B          PIC X(10).
         03  IDX-2-BODY                   PIC X(50).

     FD  IDX-3-FILE.
     01  IDX-3-RECORD.
```

```
           03  IDX-3-KEY                      PIC X(10).
           03  IDX-3-ALT-KEY.
               05  IDX-3-ALT-KEY-A            PIC X(30).
               05  IDX-3-ALT-KEY-B            PIC X(10).
           03  IDX-3-BODY                     PIC X(50).
       WORKING-STORAGE SECTION.


       PROCEDURE DIVISION.
       LEVEL-1 SECTION.
       MAIN-LOGIC.
           set environment "default-host" to "dci"

      *    make IDX1 table on DBCED

           OPEN OUTPUT IDX-1-FILE
           MOVE "IDX IN DBCED" TO IDX-1-BODY
           MOVE "A" TO IDX-1-KEY
           WRITE IDX-1-RECORD
           MOVE "B" TO IDX-1-KEY
           WRITE IDX-1-RECORD
           MOVE "C" TO IDX-1-KEY
           WRITE IDX-1-RECORD
           CLOSE IDX-1-FILE

      *    make IDX1 table on DBMULTI
           OPEN INPUT IDX-1-FILE
           OPEN OUTPUT IDX-2-FILE
           PERFORM UNTIL 1 = 2
             READ IDX-1-FILE NEXT AT END EXIT PERFORM END-READ
             MOVE IDX-1-RECORD TO IDX-2-RECORD
             MOVE "IDX IN DBMULTI" TO IDX-2-BODY
             WRITE IDX-2-RECORD
           END-PERFORM
           CLOSE IDX-1-FILE IDX-2-FILE

      *    make IDX1 table on DBSAMPLE4
           OPEN INPUT IDX-1-FILE
           OPEN OUTPUT IDX-3-FILE
```

```
          PERFORM UNTIL 1 = 2
            READ IDX-1-FILE NEXT AT END EXIT PERFORM END-READ
            MOVE IDX-1-RECORD TO IDX-3-RECORD
            MOVE "IDX IN DBSAMPLE4" TO IDX-3-BODY
            WRITE IDX-3-RECORD
          END-PERFORM

          CLOSE IDX-1-FILE IDX-3-FILE
```

ファイル接頭辞によってデータベースの表idx-1を読みます:

```
....
      INPUT-OUTPUT SECTION.
       FILE-CONTROL.

           SELECT IDX-1-FILE
           ASSIGN TO DISK "IDX1"
           ORGANIZATION IS INDEXED
           ACCESS IS DYNAMIC
           RECORD KEY IS IDX-1-KEY.


       DATA DIVISION.
       FILE SECTION.
       FD  IDX-1-FILE.
       01  IDX-1-RECORD.
           03  IDX-1-KEY                       PIC X(10).
           03  IDX-1-ALT-KEY.
               05  IDX-1-ALT-KEY-A             PIC X(30).
               05  IDX-1-ALT-KEY-B             PIC X(10).
           03  IDX-1-BODY                      PIC X(50).

       WORKING-STORAGE SECTION.

       PROCEDURE DIVISION.
       LEVEL-1 SECTION.
       MAIN-LOGIC.
           set environment "default-host" to "dci"

           set environment "file-prefix" to "/usr/MULTI:/usr/CED".
           OPEN INPUT IDX-1-FILE
```

```
           READ IDX-1-FILE NEXT
           DISPLAY IDX-1-BODY
           ACCEPT OMITTED
           CLOSE IDX-1-FILE

           set environment "file-prefix" to "/usr/CED:/usr/MULTI".
           OPEN INPUT IDX-1-FILE
           READ IDX-1-FILE NEXT
           DISPLAY IDX-1-BODY
           ACCEPT OMITTED
           CLOSE IDX-1-FILE

           set environment "file-prefix" to ".:/usr/CED:/usr/MULTI".
           OPEN INPUT IDX-1-FILE
           READ IDX-1-FILE NEXT
           DISPLAY IDX-1-BODY
           ACCEPT OMITTED
           CLOSE IDX-1-FILE
```

6 COBOL転換

転換中、トランザクションがDCIに強制実行されます。 全てのI/O 操作がトランザクションを通じて完成されます。DCI はAUTOCOMMITをoffに設定して、レコードを変更するため、DBMasterトランザクションを管理します。DCIはSTART TRANSACTION, COMMIT/ROLLBACK TRANSACTIONようなCOBOLトランザクション構文をサポートします。DCIはレコード暗号化、レコード圧縮、変更比較手順をサポートすることができません。コードにこのオプションを含んでも無効になります。DCIは“P” PICtureエディット関数もサポートできます、全てのファイル名は小文字に転換されます。.

DBMASTERDATABASE SETTINGS	RANGE LIMIT
Indexed key size.	1024
Number of columns per key.	16
Length for a CHAR field.	3992 bytes
Simultaneous RDBMS connections.	1024
Character for column names.	32
Database tables simultaneously open by a single process.	256

図 6-1 DBMasterデータベースが範囲を設定

6.1 マップCOBOLデータ型

データベース表のカラムを作成する時、DCIはCOBOL データタイプに最

適なマッチを計算します確定します。Any data the COBOL データ型のデータはデータカラムに含みます。

COBOL	DBMASTER		COBOL	DBMASTER
9(1-4)	SMALLINT		9(5-9) comp-4	INTEGER
9(5-9)	INTEGER		9(10-18) comp-4	DECIMAL(10-18)
9(10-18)	DECIMAL(10-18)		9(1-4) comp-5	SMALLINT
s9(1-4)	SMALLINT		9(5-10) comp-5	DECIMAL(10)
s9(5-9)	INTEGER		s9(1-4) comp-5	SMALLINT
s9(10-18	DECIMAL(10-18)		s9(5-10) comp-5	DECIMAL(10)
9(n) comp-1 n (1-17)	INTEGER		9(1-4) comp-6	SMALLINT
s9(n) comp-1 n (1-17)	INTEGER		9(5-9) comp-6	INTEGER
9(1-4) comp-2	SMALLINT		9(10-18) comp-6	DECIMAL(10-18)
9(5-9) comp-2	INTEGER		s9(1-4) comp-6	SMALLINT
9(10-18) comp-2	DECIMAL(10-18)		s9(5-9) comp-6	INTEGER
s9(1-4) comp-2	SMALLINT		s9(10-18) comp-6	DECIMAL(10-18)
s9(5-9) comp-2	INTEGER		signed-short	SMALLINT
s9(10-18) comp-2	DECIMAL(10-18)		unsigned-short	SMALLINT
9(1-4) comp-3	SMALLINT		signed-int	CHAR(10)
9(5-9) comp-3	INTEGER		unsigned-int	CHAR(10)
9(10-18) comp-3	DECIMAL(10-18)		signed-long	CHAR(18)
s9(1-4) comp-3	SMALLINT		unsigned-long	CHAR(18)
s9(5-9) comp-3	INTEGER		float	FLOAT
s9(10-18) comp-3	DECIMAL(10-18)		Double	DOUBLE
9(1-4) comp-4	SMALLINT		PIC x(n)	CHAR(n) n 1-max column length

図 6-2 COBOLから DBMaster データ型に転換表

6.2 マップDBMasterデータ型

DCIはローカルデータ型がCOBOLデータ型に移動するデータベースのデータを読みます。（大部分はCHAR句がありますので、dmSQLで表示することができます）。
ICOBOLデータ型とデータベースデータ型をマッチすることを心配する必要がありません。PIC X(nn) はCHAR句があるデータベースタイプのカラムに使用されます。PIC 9(9) はINTEGER 型があるデータベースに近付くCOBOL マッチです。多くのデータベース型を知ると多くのマッチCOBOL 型が探せます。例、DBMasterデータベースにのカラムは0－99の値を含むと、PIC 99は十分なCOBOL データマッチです。
BINARY データ型は変更しなくて再書きます、COBOL と関係がないからです。しかし、BINARYカラムと近づく分析は違った解決方法を探します。DECIMAL,NUMERIC, DATE とTIMESTAMP タイプは正確なCOBOLマッチがありません。文字フォームでデータベースから返します、だから、最適なCOBOL データ型環境はUSAGE DISPLAYです。
以下の表はデータベースデータ型とCOBOLデータ型の最適なマッチを表示します：

DBMASTER	COBOL	DBMASTER	COBOL
SMALLINT	9(1-4)	INTEGER	9(5-9) comp-4
INTEGER	9(5-9)	DECIMAL(10-18)	9(10-18) comp-4
DECIMAL(10-18)	9(10-18)	SMALLINT	9(1-4) comp-5
SMALLINT	s9(1-4)	DECIMAL(10)	9(5-10) comp-5
INTEGER	s9(5-9)	SMALLINT	s9(1-4) comp-5
DECIMAL(10-18)	s9(10-18	DECIMAL(10)	s9(5-10) comp-5
INTEGER	9(n) comp-1 n (1-17)	SMALLINT	9(1-4) comp-6
INTEGER	s9(n) comp-1 n (1-17)	INTEGER	9(5-9) comp-6
SMALLINT	9(1-4) comp-2	DECIMAL(10-18)	9(10-18) comp-6
INTEGER	9(5-9) comp-2	SMALLINT	s9(1-4) comp-6
DECIMAL(10-18)	9(10-18) comp-2	INTEGER	s9(5-9) comp-6
SMALLINT	s9(1-4) comp-2	DECIMAL(10-18)	s9(10-18) comp-6
INTEGER	s9(5-9) comp-2	SMALLINT	signed-short
DECIMAL(10-18)	s9(10-18) comp-2	SMALLINT	unsigned-short
SMALLINT	9(1-4) comp-3	CHAR(10)	signed-int
INTEGER	9(5-9) comp-3	CHAR(10)	unsigned-int
DECIMAL(10-18)	9(10-18) comp-3	CHAR(18)	signed-long
SMALLINT	s9(1-4) comp-3	CHAR(18)	unsigned-long
INTEGER	s9(5-9) comp-3	FLOAT	float
DECIMAL(10-18)	s9(10-18) comp-3	DOUBLE	Double
SMALLINT	9(1-4) comp-4	CHAR(n) n 1-max column length	PIC x(n)

Figure 6-3 DBMasterから COBOLデータ型に転換表

7 DCIのMFCobolアプリケーション

7.1 DLL

DCI_CONFIG 変数の設置と構成ファイルの実行を完成した後、ユーザは MF DCIプログラムを DLLに生成します.
Micro Focus NetExpressに, 空きプロジェクトを作成して、プロジェクトに COBOL ファイルを追加、DLL 型を選択して、DBMaster DCI Lib(**dmdcic.lib,dmapi43.lib,dmmfcbl.lib**) を追加、必要によって **cobolcall.obj** とリンクします。今プロジェクトを再生成します。一つの DLLはdebug\ or release\ directory下にあります。
今, DCI DLLがあります, MF COBOL プログラムに呼べます。 呼ぶ方法を以下のセクションを説明します。

7.2 EXE

ユーザはEXEを生成したいと, 手順は DLLの生成と同じですが, 一つの違うところは: w Micro Focus NetExpress プロジェクトにEXE を生成する時、ユーザは"Executable file (EXE)”を選択すべきです, "Dynamic link Library (DLL)"ではありません.

7.3 DCI DLLを呼ぶ

EXEが直接に実行できますが、 DLLに対してユーザはコールプログラムでDLLを呼びます。普通のMF COBOL DLL.のコールと同じです。

8 DCIのGNT/INT

GNT/INTはbytecodeと似合う実行モードですので、DCIはwindowsで直接に使用できません。必要なDCIをほかのcobolプロセスに転換して、dllモードをエディットします、そして、cobolはそれを呼びます。毎度使用すると呼ばなければなりません。LinuxでDCI libraryと連結のruntimeを作成してDCIが直接に使用できます。

8.1 GNT/INTを使用して間接的にDCIを呼ぶ

DCIをリンクするcobolプロセスをDLL/EXEに生成します。GNT/INTを使用してDLL/EXEを呼びます、そして間接的にDCIを呼ぶ目的を達します。

➲ **例(dll sample): custchar.cbl**

以下のcobolプロセスはDLLにエディックされます、GNT/INTが間接的にD使用します。

```
$SET CALLFH "DBMASTERINTF"
 IDENTIFICATION DIVISION.                                        00003000
 PROGRAM-ID. CUSTCHAR.                                           00004000
 INPUT-OUTPUT SECTION.                                           00014000
 FILE-CONTROL.                                                   00015000
     SELECT CUSTOMER-FILE ASSIGN TO "customer"                   00016000
     ORGANIZATION IS INDEXED                                     00017000
     RECORD KEY F-C-CODE
     ACCESS IS DYNAMIC                                           00019000
```

```
       LOCK MODE IS AUTOMATIC.                                  00020000
     DATA DIVISION.                                             00023000
     FILE SECTION.                                              00026000
     FD  CUSTOMER-FILE.                                         00027000
     01  CUSTOMER-RECORD.                                       00028000
        03 F-C-CODE        PIC X(5).                            00029000
        03 F-C-NAME        PIC X(15).                           00030000
        03 F-C-EMAILID     PIC X(25).                           00031000
        03 F-C-TEL         PIC X(20).                           00032000
        03 F-C-ADDRESS1    PIC X(58).                           00033000
        03 F-C-ADDRESS2    PIC X(58).                           00034000
        03 F-C-LIMIT       PIC 9(8).                            00035000
        03 F-C-AREA        PIC X.                               00036000

     WORKING-STORAGE SECTION.                                   00038000
     PROCEDURE DIVISION.                                        00059000
     PROCEDURE-BODY.
         OPEN OUTPUT CUSTOMER-FILE.
         CLOSE CUSTOMER-FILE
         OPEN I-O CUSTOMER-FILE.                                00068000
                 MOVE "12" TO F-C-CODE
                 MOVE "OK" TO F-C-NAME
                 MOVE "OK" TO F-C-EMAILID
                 MOVE "OK" TO F-C-TEL
                 MOVE "OK" TO F-C-ADDRESS1
                 MOVE "OK" TO F-C-ADDRESS2
                 MOVE 123 TO F-C-LIMIT
                 MOVE "OK" TO F-C-AREA
        write  CUSTOMER-RECORD.
```

```
           CLOSE CUSTOMER-FILE
           OPEN I-O CUSTOMER-FILE.
           READ CUSTOMER-FILE                                    00099000
               INVALID KEY                                       00100000
                   DISPLAY "顧客コードが無効です"                00102000
               NOT INVALID KEY                                   00103000
                   DISPLAY "OK"                                  00104000
           END-READ.                                             00105000
           CLOSE CUSTOMER-FILE.
           EXIT PROGRAM
           STOP RUN.
```

用語表

API
アプリケーションプログラムインタフェース: API はアプリケーションから操作システムまでのインタフェースです。

Binary Large Object (BLOB)
データベースにラージブロックデータは表にはっきりなレコードとしてストアされません。BLOB は一般なレコードとして同じ方法でデータベースを通じてアクセスされません。データベースはBLOB 名と場所がアクセスできます。ほかのアプリケーションはデータを読むます。

Buffer
バッファは操作を入力、出力する時データを臨時にストアするスペースです。

Client
コンピューターはサーバ側のデータをアクセス、操作することができます。

Column
データベース表にのデータセットです。このデータセットは同じデータ型を持つデータレコードから組みます。

Data dictionaries
拡張ファイル記述子で、データベーススキーマとCOBOLアプリケーションファイル記述にマップ（リンク）です。

Directive
これはCOBOL コードに選択できるコマンドです。実行フィールドにデータタイプを設定しますがディフォルトDCI設定ではありません。

Field
データベースカラムと相応しているCOBOLファイル記述です。COBOLレコードに非連続なデータです。

File Descriptor
ファイル記述子は整数です。この整数はプロセスでファイルを操作することを判断します。ファイル記述子を入力引数としてファイルに読む、書く、閉めるなどの操作をします。

Indexed file
全てのレコードを識別するキーを含むファイルです。

Key
データベースにレコードを識別するユニコード値です。（詳細は**Primary Key**をご参考ください）

Primary key
カラムのユニコード値（キー）から構成します。表に独立なレコードを識別します。

Query
DBMasterに, SQLコマンドはデータクエリ要求を実行してユーザに詳細な情報を取らせます。

Record
COBOLでこれはデータ区の関係フィールドです。DBMasterでレコードは行で表カラムに関係データ項の集合です。

Relational Database
関係データベースはデータベースシステムです。このデータベースシステムはキーまたはユニコード索引を通じて違ったデータベースに関係がある内部データベース表です。

Schema
データベース表の構成です。データ型、サイズ、カラム数、キーは表ス

キーマを制御します。

Server
サーバは中心コンピュータです。ネットワーク構成ファイルをストアします。これはデータベースシステムから構成してネットワーク接続を通じてデータをストアします。

SQL
構造クエリ言語: DBMasterと ODBCプログラムがデータをアクセスするため使用する言語です。

Table
表はロジックストレージユニットです。カラムと行から構成してレコードをストアします。

XFD file
拡張ファイル記述とデータ辞書に頭字語です。データ辞書にファイル拡張を形成します。