COBOL SQL アクセス
プログラミングの手引

本書について

本書は，埋込み SQL COBOL ソースを記述して，プログラミングを行うために，SQL 文の具体的な使い方を説明しています。また，プログラミングを行う埋込み SQL COBOL ソースのファイル形式について説明しています。

本書の構成

本書の構成について説明します。

本書は，3 つの章と付録で構成しています。それぞれの章の内容は，次のとおりです。

「第 1 章　SQL プリコンパイラで扱うファイル（1 ページ）」

SQL プリコンパイラで扱うファイル形式について説明します。

「第 2 章　埋込み SQL プログラミング（5 ページ）」

埋込み SQL 文の言語要素および COBOL プログラムから SQL 文を使用する場合にどこへ埋め込むかについて説明します。

「第 3 章　SQL 文（23 ページ）」

埋込み SQL 文の用途および使用方法を具体例を挙げて説明します。

「付録 A.　注意／制限事項（44 ページ）」

SQL プリコンパイラの注意／制限事項について説明します。

「付録 B.　エラーメッセージ（45 ページ）」

SQL プリコンパイラが出力するエラーメッセージについて説明します。

説明書の構成

COBOL SQL アクセスをご使用していただくために各種の説明書を用意しています。

説明書名	記述している内容
COBOL SQL アクセス 言語説明書	COBOL ソースに埋め込む形式でサポートする SQL データ操作言語の言語仕様（書き方や規則）について説明しています。
COBOL SQL アクセス プログラミングの手引	埋込み SQL COBOL ソースを記述して，プログラミングを行うために，SQL 文の具体的な使い方を説明しています。
COBOL SQL アクセス ユーザーズガイド	作成した埋込み SQL COBOL ソースをプリコンパイルして SQL 展開済み COBOL ソースを生成する方法や生成した実行モジュールを実行する際に，埋込み SQL COBOL ソースの CONNECT 文で指定したサーバ名と ODBC ドライバで設定するデータソース名を関連付ける方法について説明しています。

関連製品の説明書

関連製品の説明書として次のものがあります。

説明書名	記述している内容
COBOL 言語説明書	COBOL の言語仕様について説明しています。
COBOL プログラミングの手引	COBOL のプログラミング技法を説明しています。
COBOL ユーザーズガイド	COBOL を使ったアプリケーションの開発方法について説明しています。
COBOL 開発環境 操作ガイド	COBOL プログラムプロダクトの機能と操作方法について説明しています。

ご注意

1. 本書の内容の一部または全部を無断転載することは禁止されています。
2. 本書の内容に関しては将来予告なしに変更することがあります。
3. 本書は内容について万全を期して作成いたしましたが，万一ご不審な点や誤り，記載もれなどお気付きのことがありましたら，ご連絡ください。
4. 運用した結果の影響については，(3)項にかかわらず責任を負いかねますのでご了承ください。

商標情報

- Microsoft，Windows は，米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です
- Oracle と Java は，Oracle Corporation およびその子会社，関連会社の米国およびその他の国における登録商標です。文中の社名，商品名等は各社の商標または登録商標である場合があります。
- そのほかの会社名および商標名は各社の商標または登録商標です。なお，本文中では TM や®は明記しておりません。

輸出する際の注意事項

本製品 (ソフトウェア)は日本国内仕様であり，外国の規格等には準拠しておりません。

本製品は日本国外で使用された場合，当社は一切責任を負いかねます。また，当社は本製品に関して海外での保守サービスおよび技術サポート等は行っておりません。

著作権

本書の内容は，日本電気株式会社が開示している情報のすべてが掲載されていない場合，またはほかの方法で開示された情報とは異なった表現の仕方をしている場合があります。また，予告なしに内容が変更または廃止される場合がありますので，あらかじめご承知おきください。

本書の制作に際し，正確さを期するために万全の注意を払っております。しかしながら，日本電気株式会社はこれらの情報の内容が正確であるかどうか，有用なものであるかどうか，確実なものであるかどうか等につきましては保証いたしません。 また，当社は皆様がこれらの情報をご使用されたこと，またはご使用になれなかったことにより生じるいかなる損害についても責任を負うものではありません。本書のいかなる部分も，日本電気株式会社の書面による許可なく，いかなる形式または電子的，機械的，記録，その他のいかなる方法によってもコピー再現，または翻訳することはできません。

本文中の記号／略称

本書で使用する記号や略称について説明します。

形式で用いている記法

1. 英字の語と日本語の語

   英字の語は予約語を表しています。

   日本語の語は，その項または他の項で記述されている形式を表しています。

2. 角かっこと中かっこ

   a. 角かっこ[]

      角かっこ[]で囲んである部分は書くか省くかを利用者が選択します。

      角かっこ[]内に縦線|で分割した複数の形式がある場合，それらのうちの１個を指定するか，またはすべて省くかを選択できます。

      角かっこ[]内で下線がついている形式は，[]内を省いたときに暗黙的に指定される形式です。

   b. 中かっこ{}

      中かっこ{}に縦線|で分割した複数の形式がある場合，複数の形式のうち，必ず１個の形式を利用者が選択します。

3. 反復記号

   反復記号“…”の意味は以下のとおりです。

   []…は角かっこ[]内における形式の 0 回以上の繰り返しです。

{}…は中かっこ{}内における形式の１回以上の繰り返しです。

本書の中で使用する略称

略称	意味
SQL 機能	COBOL SQL アクセスを表します
SQL プリコンパイラ	COBOL SQL アクセスのプリコンパイラを表します
SQL ランタイム	COBOL SQL アクセスのランタイムを表します
埋込み SQL COBOL ソース	SQL プリコンパイラの入力となる埋込み SQL 文を含む COBOL ソースを表します
SQL 展開済み COBOL ソース	SQL プリコンパイラの出力となる，埋込み SQL 文を COBOL コンパイラで解釈可能な記述に変換した後の COBOL ソースを表します

目次

第 1 章
SQL プリコンパイラで扱うファイル

SQL プリコンパイラは，入力として埋込み SQL COBOL ソースおよび INCLUDE ファイルを扱います。また，出力として SQL 展開済み COBOL ソースを扱います。

1.1　ファイルの形式

SQL プリコンパイラの入力および出力として扱うことが可能なファイルの形式を説明します。

表 1-1　SQL プリコンパイラが扱うファイル形式

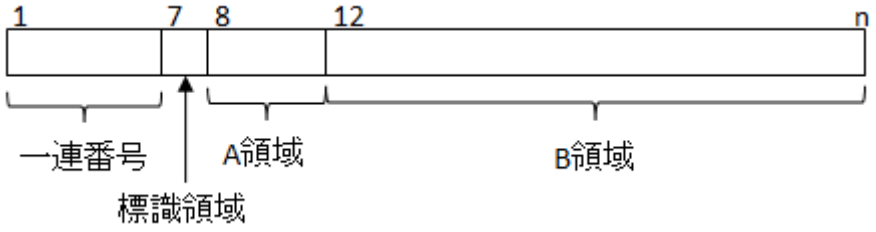
形式	名前	概要	拡張子
形式 1	固定形式 (標準：識別領域なし)	COBOL の「固定形式正書法」に対応した形式です。一連番号，標識領域，A 領域，B 領域からなる，最大 255 桁の形式	.qcob .cob
形式 2	固定形式 (識別領域あり)	COBOL の「固定形式正書法」に対応した形式です。一連番号，標識領域，A 領域，B 領域，識別領域からなる，最大 80 桁の形式	.qcbl .cbl

[形式 1]

下図のような書式で記述する識別領域なしの固定形式です。(1 行の最大長 n≦255 桁 (バイト数))

埋込み SQL COBOL ソースファイルの拡張子は，.qcob です。

SQL 展開済み COBOL ソースファイルおよび INCLUDE ファイルの拡張子は，.cob です。

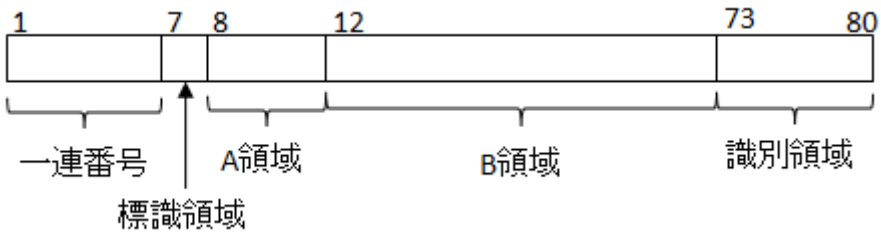


[形式 2]

下図のような書式で記述する識別領域ありの固定形式です。(1 行の最大長 n≦80 桁(バイト数))

埋込み SQL COBOL ソースファイルの拡張子は，.qcbl です。

SQL 展開済み COBOL ソースファイルおよび INCLUDE ファイルの拡張子は，.cbl です。



形式(書式)は，ファイルの拡張子によって指定する必要があります。

埋込み SQL COBOL ソースファイル， SQL 展開済み COBOL ソースファイルおよび INCLUDE ファイルの文字コードおよびファイル名はシフト JIS として作成する必要があります。

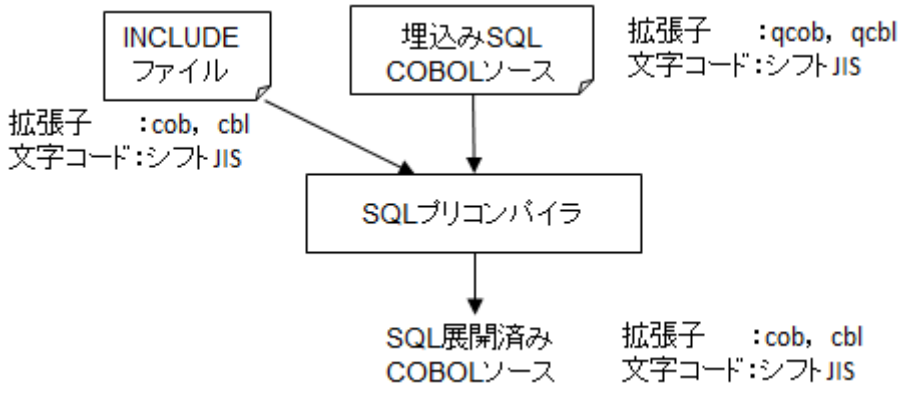
COBOL の自由形式正書法での記述は使用できません。

SQL プリコンパイラは，標識領域に記述されている文字によって，次のような解釈を行い，SQL 展開済み COBOL ソースを生成する。

表 1-2　SQL プリコンパイラが解釈する標識領域の文字

標識領域の文字	解釈	SQL 展開済み COBOL ソース出力
数字(0-9)	選択行指定とみなす -N オプションによって選択された行は翻訳を行う。 選択されなかった行はコメントとみなす。	選択された行およびその行から展開された行の識別領域には，元の文字(数字)は設定される。 コメントとみなされた行は，元のイメージのまま出力される。
上記以外 *，D，/，...	コメント行とみなす。	元のイメージのまま出力される

SQL 文および EXEC SQL， END-EXEC は， A 領域， B 領域のいずれかから書き始めることができます。



SQL プリコンパイラは，COBOL ソースの形式を拡張子で判断します。

埋込み SQL COBOL ソースが.qcob の場合は形式 1， .qcbl の場合は形式 2 を指定したものとみなします。

SQL 展開済み COBOL ソースまたは INCLUDE ファイルが.cob の場合は形式 1，　.cbl の場合は形式 2 を指定したものとみなします。

注

埋込み SQL COBOL ソースを形式 1(.qcob)，SQL 展開済み COBOL ソースを形式 2(.cbl)と指定した場合は，72 桁を超えた文は折り返して出力します。

埋込み SQL COBOL ソースを形式 2(.qcbl)，SQL 展開済み COBOL ソースを形式 1(.cob)と指定した場合は，識別領域に記述した文字列は削除します。

1.2　INCLUDE ファイル

INCLUDE 文は，指定したファイルの内容を INCLUDE 文を記述した場所に展開することを指示します。

[形式]

```
EXEC SQL INCLUDE ファイル名 END-EXEC.
```

INCLUDE 文で指定するファイルを INCLUDE ファイルといいます。

利用者は，　この機能を使うことにより，　次のような利点を得ることができます。

1. 複数のプログラムから参照するホスト変数をファイルとして登録しておくことにより，　ホスト変数の仕様変更に対する修正をそのファイルだけに留めることができ，プログラム間のインタフェースミスを防ぐことができます。
2. 共通処理(たとえば，エラー処理ルーチンなど，ある一定の SQL 処理ルーチン)をファイルとして登録しておくことにより，プログラムの標準化を容易に行うことができます。

利用者はあらかじめ INCLUDE ファイルを「1.1　ファイルの形式（1 ページ）」の形式で作成しておく必要があります。

SQL プリコンパイラは，次の順序で INCLUDE ファイルを検索します。

1. -I オプションを指定している場合，最初に-I オプションで指定しているパス配下を次の順序で探します。

   ファイル名.cob⇒**ファイル名**.cbl

2. 次に 2 番目の-I オプションで指定しているパス配下を探します。

   検索するファイル名の順序は「1.（3 ページ）」と同じです。

3. 次に 3 番目の-I オプションで指定しているパス配下を探します。

   検索するファイル名の順序は「1.（3 ページ）」と同じです。

4. 以下，最後の-I オプションで指定しているパスまで繰り返し探します。
5. カレントディレクトリ配下を探します。

検索するファイル名の順序は「1.（3 ページ）」と同じです。

第 2 章
埋込み SQL プログラミング

埋込み SQL 文とは，COBOL プログラム内に記述する SQL 文のことです。

COBOL プログラムから SQL 文を使用する場合，利用者は SQL の言語要素を COBOL プログラム中の適切な位置に埋め込む必要があります。

本章では，SQL の言語要素にはどのようなものがあるか，それを COBOL プログラム内のどこへ埋め込むかについて説明します。

2.1　言語要素

SQL の言語要素には次のものがあります。

データ部作業場所節に記載するもの

ホスト変数定義

標識変数定義

SQLCA

手続き部に記載するもの

埋込み例外処理

SQL 文

プログラム中のどの部に記載してもよいもの

カーソル宣言文

2.1.1　ホスト変数

ホスト変数は COBOL プログラムとデータベース間のデータの引き渡しに使用します。

データベースにデータを入力する場合， COBOL プログラムでホスト変数にデータを代入し， SQL 文を用いてデータベースにデータを格納します。

また COBOL プログラムでデータベースのデータを参照したい場合， SQL 文を用いてホスト変数にデータを格納することによりデータを参照することができます。

ホスト変数を SQL 文中で使用する場合， SQL 文識別子などと区別するためにホスト変数の前にコロン(:)を付けなければなりません。

ホスト変数として集団項目を指定することもできます。

(例)ホスト変数として集団項目を指定する

```
01 ホスト変数 1.
  02 ホスト変数 2.
    03 ホスト変数 3 ～
  02 ホスト変数 4 ～
                :
```

集団項目に従属する項目を SQL 文中で指定する場合，従属する項目だけでは一意とならないときには，上位の項目を指定して修飾します。

(例)上位の項目を指定して修飾する

「(例)ホスト変数として集団項目を指定する（6 ページ）」の集団項目でホスト変数 3 を参照する場合，次のように指定します。

```
:ホスト変数 1.ホスト変数 2.ホスト変数 3
```

注

ホスト変数 1 とホスト変数 2，ホスト変数 2 とホスト変数 3 の間にピリオド(.)を指定することで修飾できます

ホスト変数の属性とデータベースのデータ型の対応は次のようになります。

属性	データベースのデータ型	COBOL 記述[*1]	説明
集団項目		レベル番号(01～47) ホスト変数名 .	
真数	SMALLINT	レベル番号(01～48,77) ホスト変数名 COMP-1.	
	INTEGER	レベル番号(01～48,77) ホスト変数名 COMP-2.	
	NUMERIC	レベル番号(01～48,77) ホスト変数名 PIC {9(n) | 9(n)V9(m) | V9(m) | S9(n) | S9(n)V9(m) | SV9(m) } [[USAGE [IS]] DISPLAY] [[SIGN [IS]] { LEADING | TRAILING} [SEPARATE [CHARACTER]]].	ｎとｍは正の整数で n+m が 18 を越えてはならない。
	DECIMAL	レベル番号(01～48,77) ホスト変数名 PIC {9(n) | 9(n)V9(m) | V9(m) | S9(n) | S9(n)V9(m) | SV9(m) }	ｎとｍは正の整数で n+m が 18 を越えてはならない

属性	データベースのデータ型	COBOL 記述[*1]	説明
		[USAGE [IS]] {COMP-3 | PACKED-DECIMAL}.	
文字列型	CHAR	レベル番号(01～48,77) ホスト変数名 PIC X(n) .	n は正の整数でその値は 1 ～ 65535 を越えてはならない。
日本語文字列型	CHAR	レベル番号(01～48,77) ホスト変数名 PIC N(n).	n は正の整数でその値は 1 ～ 32767 を越えてはならない。
可変長文字列型	VARCHAR	**レベル番号**(01～48) **ホスト変数名**. 49 **ホスト変数名－１** COMP-2. 49 **ホスト変数名－２** PIC X(n).	n は正の整数でその値は 1 ～ 65535 を越えてはならない。
可変長日本語文字列型	VARCHAR	**レベル番号**(01～48) **ホスト変数名**. 49 **ホスト変数名－１** COMP-2. 49 **ホスト変数名－２** PIC N(n).	n は正の整数でその値は 1 ～ 32767 を越えてはならない。

*1 正確な記述方法は,『COBOL SQL アクセス 言語説明書』の「2.4.1 データ属性の詳細」を参照のこと

各データの内部表現形式は次のようになります。

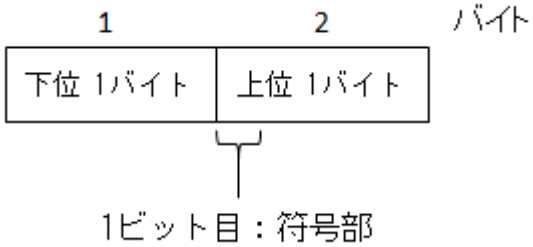
1. 集団項目

   集団項目に従属するデータの内部表現形式に依存します。

2. 真数 — 単精度 2 進数

```
77  A  COMP-1.
```

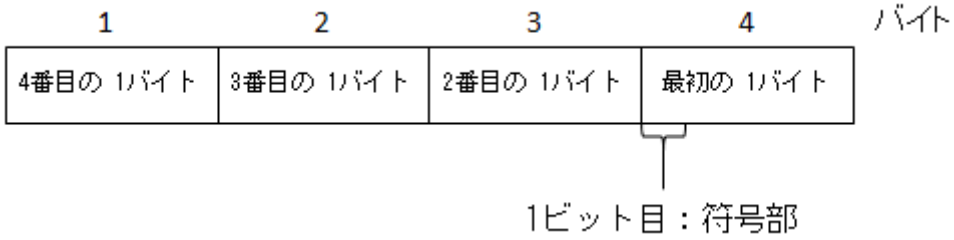
   2 バイトのリトルエンディアン形式バイナリで表現します。



3. 真数 — 倍精度 2 進数

```
77 A  COMP-2.
```

   4 バイトのリトルエンディアン形式バイナリで表現します。



4. 真数 ― 外部 10 進数

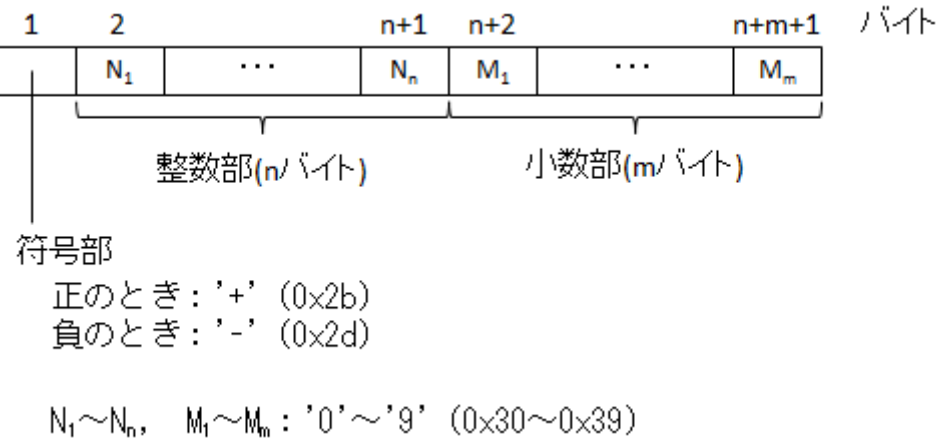
```
77  A  PIC S9(n)V9(m) DISPLAY ～.
```

a. SIGN 句あり

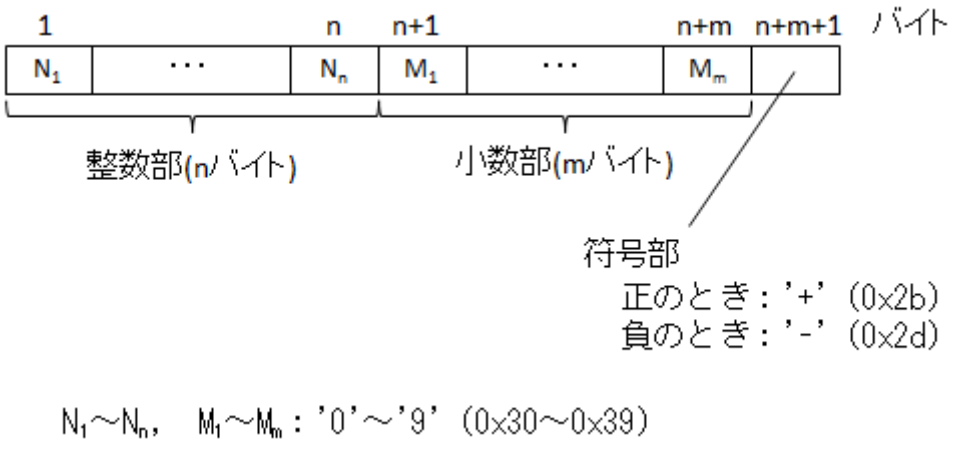
i. SEPARATE 句あり

符号部は前方または後方に 1 バイトの独立した領域として確保します。

A. LEADING 指定の場合



B. TRAILING 指定の場合



ii. SEPARATE 句なし

符号部は独立したバイトとしては確保されず，最初または最後のバイトの数値に符号(正負)の情報を重ね合わせて表現します。

A.　LEADING 指定の場合

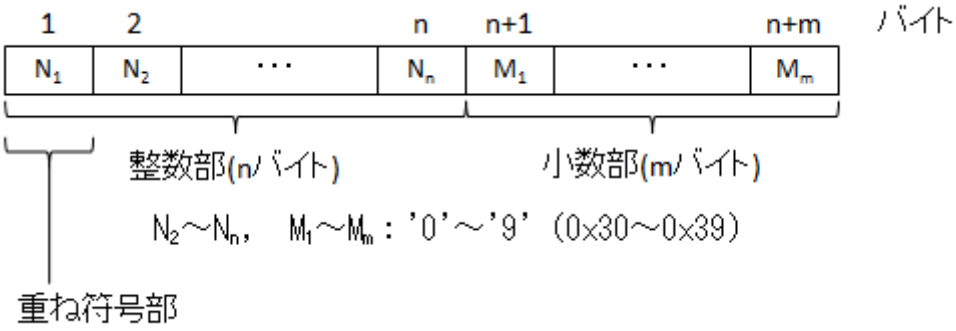


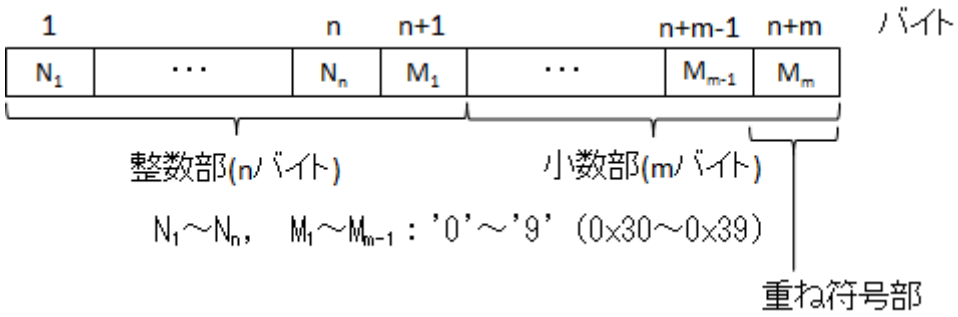
表 2-2　重ね符号部(NEC 独自形式)

最右端(最左端)バイトの値	重ね符号部		
	符号なし	正符号付き	負符号付き
0	0x30	0x7B	0x7D
1	0x31	0x41	0x4A
2	0x32	0x42	0x4B
3	0x33	0x43	0x4C
4	0x34	0x44	0x4D
5	0x35	0x45	0x4E
6	0x36	0x46	0x4F
7	0x37	0x47	0x50
8	0x38	0x48	0x51
9	0x39	0x49	0x52

表 2-3　重ね符号部(Pro*COBOL 形式)

最右端(最左端)バイトの値	重ね符号部		
	符号なし	正符号付き	負符号付き
0	0x30	0x30	0x70
1	0x31	0x31	0x71
2	0x32	0x32	0x72
3	0x33	0x33	0x73
4	0x34	0x34	0x74
5	0x35	0x35	0x75
6	0x36	0x36	0x76
7	0x37	0x37	0x77
8	0x38	0x38	0x78
9	0x39	0x39	0x79

B.　TRAILING 指定の場合



重ね符号部の内部表現は「SIGN IS LEADING（9 ページ）」と同じ

b. SIGN 句なし

データの内部表現は「SIGN IS TRAILING（9 ページ）」と同じ

注

ホスト変数として数字と符号の重ね合わせが行われた外部 10 進数の内部表現は，SQL 展開済み COBOL ソースを COBOL コンパイラでコンパイルする場合に指定するオプションにより異なります。

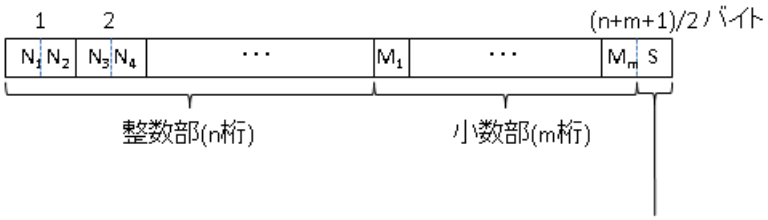
a. オプション-CS を指定した場合は，数字と符号の重ね合わせは Pro*COBOL 表現形式となります。

b. オプション-CS を指定しない場合は，数字と符号の重ね合わせは NEC 独自形式となります。

5. 真数 ― 内部 10 進数

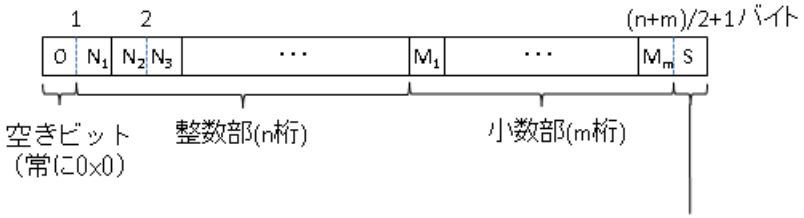
```
77  A  PIC S9(n)V9(m) COMP-3.
```

a. n+m が奇数の場合



N1～Nn，M1～Mmの値	表現形式
0	0x0
1	0x1
2	0x2
3	0x3
4	0x4
5	0x5
6	0x6
7	0x7
8	0x8
9	0x9

b.　n+m が偶数の場合

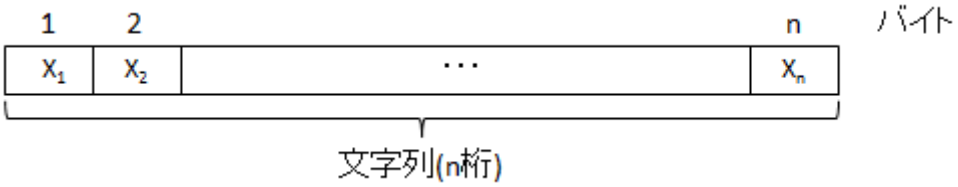


N_1～N_n, M_1～M_mの値	表現形式
0	0x0
1	0x1
2	0x2
3	0x3
4	0x4
5	0x5
6	0x6
7	0x7
8	0x8
9	0x9

符号部(4ビット)
正：0xC
負：0xD

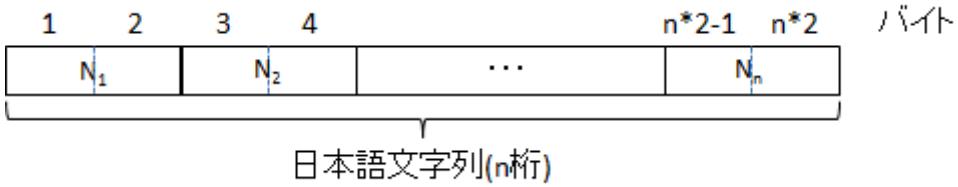
6.　文字列型

```
77  A  PIC X(n).
```



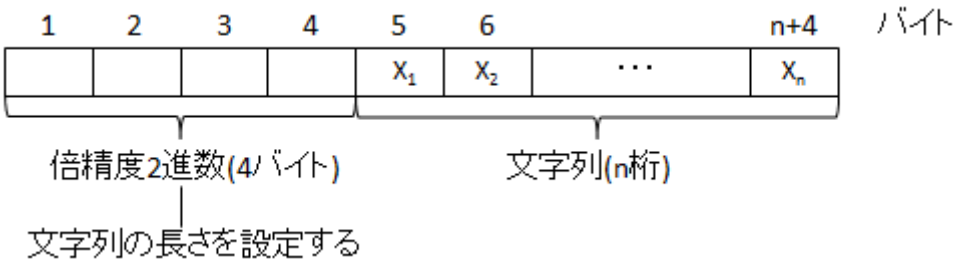
7.　日本語文字列型

```
77  A  PIC N(n).
```



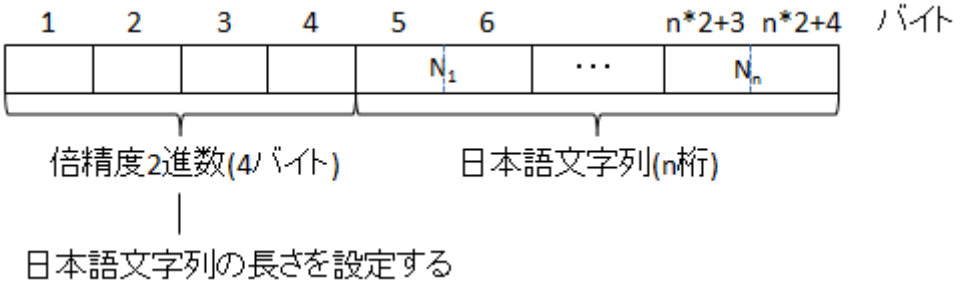
8.　可変長文字列型

```
01 A.
  49 B  COMP-2.
  49 C  PIC X(n).
```



9.　可変長日本語文字列型

```
01 A.
  49 B  COMP-2.
  49 C  PIC N(n).
```



2.1.2　標識変数

標識変数とはホスト変数の値または状態を示す真数の変数です。

標識変数を使用することによりデータベースに NULL 値を設定したり, データベースから入力した値が NULL 値か判断することができます。

SQL 文中で標識変数を指定する場合，対応付けるホスト変数の直後に記述するか，キーワード INDICATOR の直後に記述します。

2.1.3　SQLCA

SQLCA は SQL 文を実行した情報を COBOL プログラムに引き渡すための領域に使用します。

SQLCA は"INCLUDE SQLCA"文を記述することで，記述位置に展開します。

以下 SQLCA の各項目について説明します。

```
01     SQLCA.
  02   SQLCAID   COMP-2    VALUE 100.
  02   SQLCODE   COMP-2    VALUE 0.          … 1
  02   SQLERRM.
    03 SQLERRML  COMP-2    VALUE 0.
```

```
          03 SQLERRMC  PIC X(80).
        02   SQLRCNT   COMP-2.                    … 2
        02   FILLER    COMP-2.
        02   FILLER    PIC X(5).
        02   FILLER    PIC X(1).
        02   FILLER    PIC X(6).
        02   SQLSTATE  PIC X(5).                  … 3
        02   FILLER    PIC X(1).
        02   SQLMSG    PIC X(256).                … 4
        02   FILLER    PIC X(4).
```

	項目名	説明
1	SQLCODE	SQL 診断コードが返ります。
2	SQLRCNT	INSERT， UPDATE(探索)， DELETE(探索)の処理件数が返ります。
3	SQLSTATE	ODBC ドライバから 5 桁の文字列が返ります。
4	SQLMSG	ODBC ドライバからエラーメッセージが返ります。

注

SQLSTATE， SQLMSG の値は， ODBC ドライバが直接返却する値であり， その値は ODBC ドライバに依存します。

本項目の設定値に関する説明は， 使用している ODBC ドライバに関する説明書を参照してください。

SQLCA の実態ファイル(SQLCA.cob)は SQL プリコンパイラの管理するディレクトリ内に格納されており，利用者は通常，意識する必要はありません。

2.1.4　埋込み例外処理

WHENEVER 文を使用すると埋込み SQL 文処理中にエラーが発生した場合や FETCH 文等でデータの終わりを検出した際， 指定した手続きに制御を移行することができます。

移行先は，次の文から(CONTINUE)か， ラベル付き文への分岐(GO TO)になります。

2.1.5　SQL 文

SQL 文は，COBOL プログラム中で"EXEC SQL"から"END-EXEC"の間に記述し，データベース操作要求を記述するものです。

COBOL プログラムから使用できる SQL 文には次のものがあります。

データ操作

DELETE

INSERT

SELECT

UPDATE

トランザクション

COMMIT

ROLLBACK

コネクション

CONNECT

SET CONNECTION

DISCONNECT

2.2　プログラミング

SQL 文を使用するプログラムを作成する場合，利用者はデータ部の作業場所節にホスト変数定義と，SQL との連絡領域である SQLCA の宣言を行う必要があります。

また，手続き部に SQL データ操作文の記述やエラー時の処理を制御する埋込み例外宣言を記述します。

カーソルを使用してデータ操作を行う場合は，カーソル宣言を行う必要があります。カーソル宣言文は，プログラム中のどこに記述しても構いません。

2.2.1　変数の宣言

SQL 文で使用するホスト変数および，データベースと COBOL プログラムとの連絡領域である SQLCA の宣言は，COBOL プログラムのデータ部(DATA DIVISION)内の作業場所節(WORKING-STORAGE SECTION)で記述します。

[形式]

ホスト変数の宣言

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC.
  { ホスト変数の宣言 } …
EXEC SQL END DECLARE SECTION END-EXEC.
```

SQLCA の宣言

```
EXEC SQL INCLUDE SQLCA END-EXEC.
```

COBOL プログラム例

```
       IDENTIFICATION  DIVISION.
                  :
       ENVIRONMENT     DIVISION.
                  :
       DATA            DIVISION.
                  :
       WORKING-STORAGE SECTION.

       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01 HNAME   PIC X(20).
       01 HIND    COMP-2.
       EXEC SQL END   DECLARE SECTION END-EXEC.
      *    … ホスト変数の宣言

       EXEC SQL INCLUDE SQLCA END-EXEC.
      *    … SQLCA の宣言

       PROCEDURE       DIVISION.
                  :
       EXEC SQL
         INSERT INTO TBL1 (NAME) VALUES (:HNAME)
       END-EXEC.
      *    … テーブル TBL1 にホスト変数"HNAME"の値を挿入する

       EXEC SQL
         FETCH  CUR1 INTO :HNAME INDICATOR :HIND
       END-EXEC.
      *    … カーソル CUR1 からデータをホスト変数"HNAME"に読み込み
      *    … 読み込んだ値が NULL 値の場合，標識変数"HIND"に-1 を
      *    … 設定します
```

2.2.2　データベース操作

SQL 文は，データベース操作要求を記述するものです。

データベース操作を行うには，SQL 文を，COBOL プログラムの手続き部(PROCEDURE DIVISION)に記述します。

埋込み SQL 文では"EXEC SQL"と"END-EXEC"の間に SQL 文を 1 文記述することができます。

[形式]

```
EXEC SQL
  SQL 文
```

```
END-EXEC.
```

データベース操作の大まかな流れは次のようになります。

1. データベース接続
2. データベース操作要求
3. データベース切断

2.2.3　埋込み例外宣言

SQL 文を実行した結果， 例外条件が発生した場合の処理を COBOL プログラム中で宣言することができます。

埋込み例外の宣言は，SQL 文を，COBOL プログラムの手続き部(PROCEDURE DIVISION)に記述します。

埋込み例外の宣言は， "EXEC SQL"から"END-EXEC"の間に， 1 文のみ記述できます。

[形式]

```
EXEC SQL
  WHENEVER { NOT FOUND | SQLERROR }
           { CONTINUE | GOTO :手続き名 | GO TO :手続き名 }
END-EXEC.
```

WHENEVER の有効範囲の例 1

```
        :
PROCEDURE DIVISION.
EXEC-PROC.
  EXEC SQL WHENEVER SQLERROR GOTO :ERROR-1 END-EXEC.  …①
  EXEC SQL WHENEVER NOT FOUND GOTO :EXIT-1 END-EXEC.  …②

  EXEC SQL
   DECLARE C1 CURSOR FOR SELECT NAME FROM JINJI
  END-EXEC.

  EXEC SQL OPEN C1 END-EXEC.
  EXEC SQL FETCH C1 INTO :H-NAME END-EXEC.
  EXEC SQL CLOSE C1 END-EXEC.

  EXEC SQL WHENEVER SQLERROR GOTO :ERROR-2 END-EXEC.  …③

  EXEC SQL
   DECLARE C2 CURSOR FOR SELECT * FROM JINJI
  END-EXEC.
       :
```

①の"SQLERROR"例外宣言は，③の"SQLERROR"例外宣言を指定する前の SQL 文まで有効です。

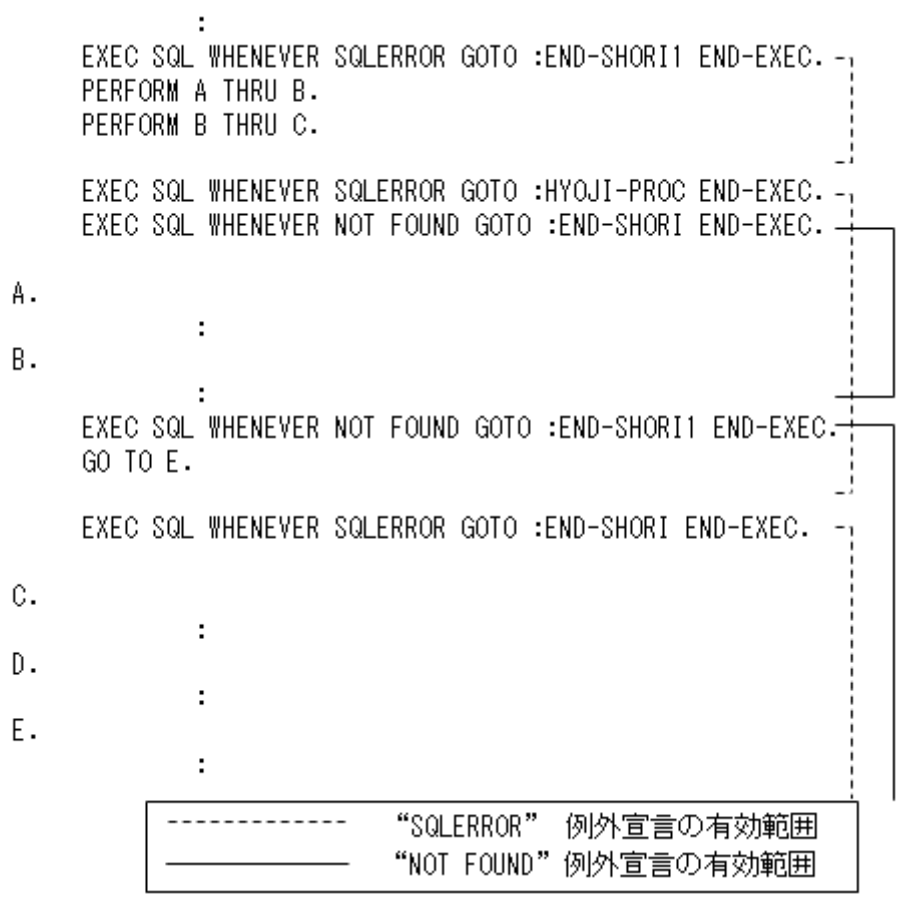
WHENEVER の有効範囲の例 2

```
       :
PROCEDURE DIVISION.
EXEC-PROC.
     EXEC SQL WHENEVER SQLERROR GOTO :ERROR-1 END-EXEC. …①
     EXEC SQL WHENEVER NOT FOUND CONTINUE END-EXEC.     …②

     EXEC SQL
      DECLARE C1 CURSOR FOR SELECT NAME FROM JINJI
     END-EXEC.

     EXEC SQL OPEN C1 END-EXEC.
     PERFORM FETCH-P THRU FETCH-E.
     EXEC SQL CLOSE C1 END-EXEC.

     EXEC SQL WHENEVER NOT FOUND GOTO :ETFH-E END-EXEC. …③

FETCH-P.
     EXEC SQL
      FETCH C1 INTO :H-NAME
     END-EXEC.
FETCH-E.
     EXIT.
       :
```

---------- “SQLERROR” 例外宣言の有効範囲
———————— “NOT FOUND” 例外宣言の有効範囲

②の"NOT FOUND"例外宣言は，③の"NOT FOUND"例外宣言を指定する前の SQL 文まで有効です。

WHENEVER の有効範囲の例 3



例外宣言の有効範囲は，プログラムの処理の流れとは無関係です。

2.2.4　プログラム例

以下に簡単な埋込み SQL 文を用いた COBOL プログラム例を示します。

埋込み SQL 文を用いた COBOL プログラム例

処理概要

- DEFAULT で宣言したデータベースへ接続を行います。
- C1 で宣言したカーソルをオープンします。

  宣言したカーソルはテーブル "JINJI" の "SYOZOKU" が "10" である行の，"BUCODE" と "NAME" の列を取り出すものです。
- オープンしたカーソルからレコードを読み込み，列名 "BUCODE" と "NAME" を表示する処理を最後のレコードまで繰り返し行います。
- カーソル C1 をクローズします。

- データベースの接続を解除します。

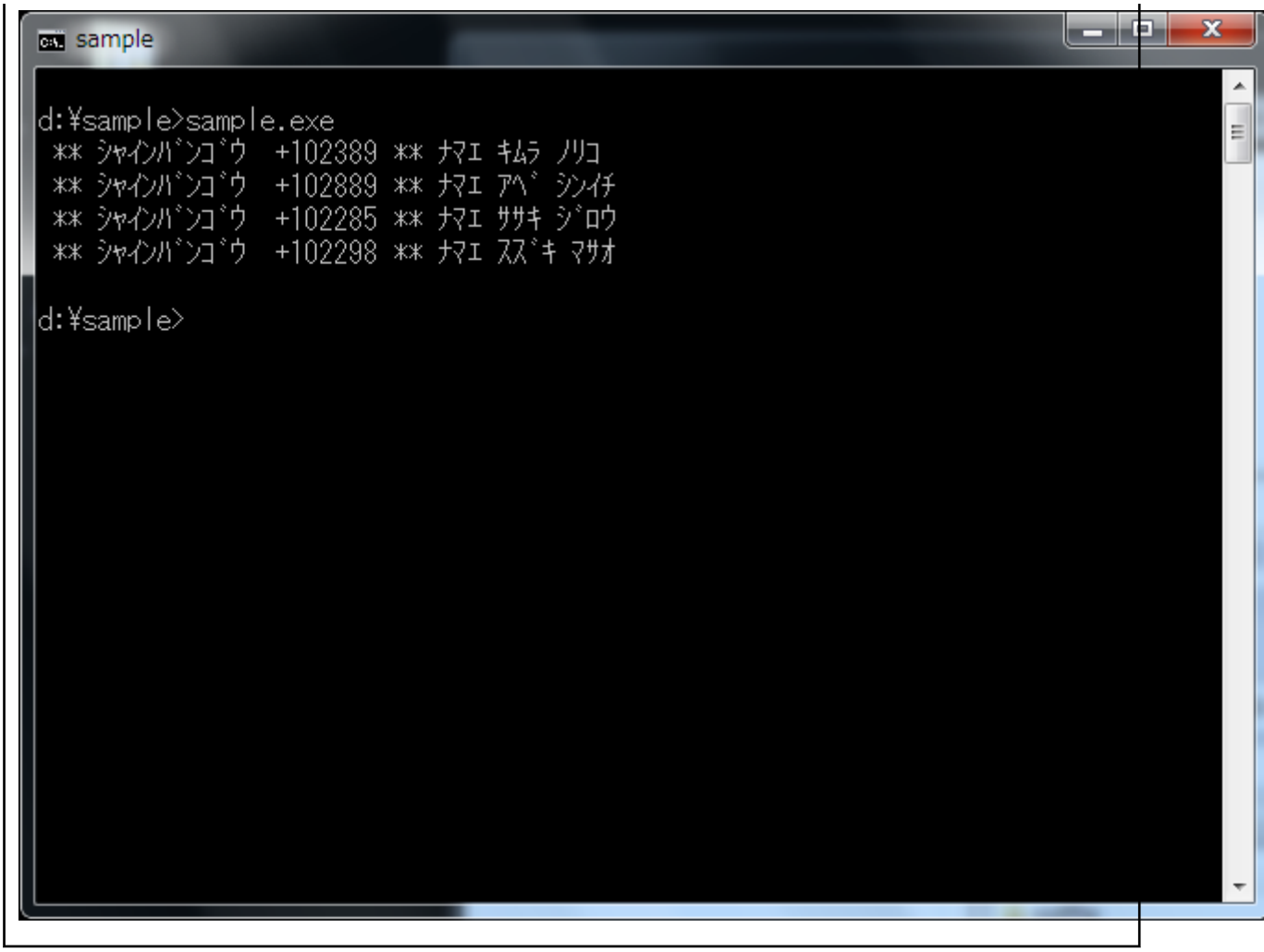
```
 IDENTIFICATION  DIVISION.
 PROGRAM-ID.     SAMPLE1.
 DATA            DIVISION.
 WORKING-STORAGE SECTION.
************************************************
*  SQL 文で使用するホスト変数定義を行う
************************************************
 EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 77 HNAME    PIC X(20).
 77 HBUCODE  PIC S9(06) USAGE IS DISPLAY
                 SIGN IS LEADING SEPARATE.
 EXEC SQL END   DECLARE SECTION END-EXEC.
************************************************
*  SQL との連絡領域である SQLCA の宣言を行う
************************************************
 EXEC SQL INCLUDE SQLCA END-EXEC.
*
 PROCEDURE DIVISION.
 WHENEVER-PROC.
************************************************
*  埋込み例外条件の宣言を行う
*  このプログラムの埋込み SQL 文で
*     ・例外条件が発生した場合，SQL-ERROR という手続きに
*      制御を渡す
*     ・最後までファイルを読み込んだら，F-END に制御を渡す
************************************************
 EXEC SQL
  WHENEVER SQLERROR GOTO :SQL-ERROR
 END-EXEC.
 EXEC SQL
  WHENEVER NOT FOUND GOTO :F-END
 END-EXEC.
 START-PROC.
************************************************
*  このプログラムで使用するカーソルを定義する
************************************************
 EXEC SQL
  DECLARE C1 CURSOR FOR
     SELECT BUCODE , NAME FROM JINJI
            WHERE SYOZOKU = 10
 END-EXEC.
************************************************
*  DEFAULT で定義したデータベースと接続を行う
************************************************
 EXEC SQL
  CONNECT TO DEFAULT
 END-EXEC.
************************************************
*  定義したカーソルの OPEN を行う
************************************************
 EXEC SQL
  OPEN C1
 END-EXEC.
************************************************
*  BUCODE と NAME を読み込み表示する
```

```
      ************************************************
       FLOOP.
       EXEC SQL
        FETCH C1 INTO :HBUCODE , :HNAME
       END-EXEC.
            DISPLAY "** ｼｬｲﾝﾊﾞﾝｺﾞｳ " HBUCODE
                    " **ﾅﾏｴ " HNAME.
            GO TO FLOOP.
       F-END.
      ************************************************
      *   終了処理　カーソルを CLOSE する
      ************************************************
       EXEC SQL
        CLOSE C1
       END-EXEC.
      ************************************************
      *   データベースの接続を解除する
      ************************************************
       EXEC SQL
        DISCONNECT ALL
       END-EXEC.
       PROC-END.
            STOP RUN.
      ************************************************
      *   SQL で例外条件が発生した場合，この手続きに制御が渡される
      *   この例では，例外が発生した場合，SQLCODE の内容を表示し
      *   終了する
      ************************************************
       SQL-ERROR.
            DISPLAY ">>>>>>>> SQLCODE  = " SQLCODE.
            DISPLAY ">>>>>>>> SQLSTATE = " SQLSTATE.
            GO TO PROC-END.
```

本 COBOL プログラム例で，下記に示すテーブル"JINJI"を使用し実行すると，次のような結果となります。

表 2-3　JINJI

SYOZOKU	BUCODE	NAME
10	102389	ｷﾑﾗ ﾉﾘｺ
10	102889	ｱﾍﾞ ｼﾝｲﾁ
10	102285	ｻｻｷ ｼﾞﾛｳ
30	308724	ﾔﾏﾀﾞ ﾀﾛｳ
50	509675	ﾅｶﾞｼﾏ ﾖｳｺ
10	102298	ｽｽﾞｷ ﾏｻｵ
30	308321	ﾏﾂﾓﾄ ﾏｻｵ
50	505942	ﾅｶﾀ ｺｳｲﾁ


sample
d:¥sample>sample.exe
** ｼｬｲﾝﾊﾞﾝｺﾞｳ　+102389 ** ﾅﾏｴ ｷﾑﾗ ﾉﾘｺ
** ｼｬｲﾝﾊﾞﾝｺﾞｳ　+102889 ** ﾅﾏｴ ｱﾍﾞ ｼﾝｲﾁ
** ｼｬｲﾝﾊﾞﾝｺﾞｳ　+102285 ** ﾅﾏｴ ｻｻｷ ｼﾞﾛｳ
** ｼｬｲﾝﾊﾞﾝｺﾞｳ　+102298 ** ﾅﾏｴ ｽｽﾞｷ ﾏｻｵ
d:¥sample>

第 3 章
SQL 文

以下の SQL 文について，用途，および使用方法を具体例を挙げて説明します。

3.1　コネクション

データベースとのコネクションを扱う SQL 文には，以下の用途をもつものがあります。

3.1.1　コネクションの確立

コネクションを確立するには，CONNECT 文を使用し，データベースへ接続する必要があります。

CONNECT 文を使用してデータベースへ接続する方法は 2 種類あります。

- サーバ名を指定して接続する

```
CONNECT TO 'DB1' AS 'CN1' USER 'sqlext/sqlext'
           ①      ②          ③     ④
```

①

サーバ名を指定します。

サーバ名の指定は必須です。

CONNECT 文を実行すると，実行環境設定ツールでサーバ名と関連付けたデータソース名を検索し，データベースへ接続を行います。

②

コネクション名を指定します。

コネクション名はデータベースのコネクション変更や切断を行うときに使用します。

"AS コネクション名"指定を省略した場合, コネクション名はサーバ名と同じになります。

③, ④

ユーザ名およびパスワードを指定します。

ユーザ名とパスワードは定数指定で'ユーザ名/パスワード'と指定します。

"USER 'ユーザ名/パスワード'"指定を省略する場合は実行環境設定ツールで設定したサーバ名に対するユーザ名とパスワードの指定が有効になります。

- DEFAULT を指定して接続する

```
CONNECT TO DEFAULT
```

DEFAULT を指定した場合は，実行環境設定ツールで"デフォルトサーバ情報"に指定したサーバ名と関連付けたデータソース名を検索し，データベースへ接続を行います。

注

実行環境設定ツールでサーバ名の指定は, "ユーザ単位"と"マシン単位"に指定することができます。

"ユーザ単位"と"マシン単位"に同じサーバ名を指定した場合，"ユーザ単位"の指定を優先します。

3.1.2　コネクションの切断

コネクションを切断するには DISCONNECT 文を使用します。

指定方法は 4 種類あります。

- コネクション名を指定する

```
DISCONNECT コネクション名
```

CONNECT 文で指定したコネクション名に対するデータベースを切断します。

- DEFAULT を指定する

```
DISCONNECT DEFAULT
```

実行環境設定ツールで"デフォルトサーバ情報"に指定したサーバ名と関連付けたデータソース名を検索し，対応するデータベースを切断します。

- CURRENT を指定する

```
DISCONNECT CURRENT
```

現在使用しているコネクションのデータベースを切断します。

- ALL を指定する

```
DISCONNECT ALL
```

確立しているすべてのコネクションを切断します。

3.1.3 コネクションの変更

SQL 機能では複数の CONNECT 文を実行することにより，複数のデータベースに接続することができます。

接続したデータベースのうち、操作要求を行うことができるデータベースは最後に接続したコネクション(カレントのコネクション)に対応するデータベースだけです。

カレントのコネクションを変更するには，SET CONNECTION 文を使用します。

指定方法は 2 種類あります。

- コネクション名を指定する

```
SET CONNECTION コネクション名
```

SET CONNECTION 文で指定したコネクション名をカレントのコネクションに設定します。

- DEFAULT を指定する

```
SET CONNECTION DEFAULT
```

実行環境設定ツールで"デフォルトサーバ情報"に指定したサーバ名と関連付けたコネクションをカレントのコネクションに設定します。

注

コネクションを切断した場合は，カレントのコネクションが不定となるため，SET CONNECTION 文を使用してカレントのコネクションを設定し直す必要があります。

カレントのコネクションを設定せずに SQL 文を実行すると実行時エラーとなります。

3.2　静的 SQL 文(データ操作)

データ操作に関する SQL 文の用途とその SQL 文を使用することによって，どのような結果が得られるかを，下記の"人事"表を基に説明します。

表 3-1　人事

氏名	給料	性別	所属	出身地
斎藤明	185000	男	経理	東京
高木良夫	161000	男	経理	神奈川
青山信子	184000	女	経理	千葉
山田春男	159000	男	購買	東京
吉永一夫	215000	男	購買	東京
高山昌夫	147000	男	製造	大阪
長島明子	176500	女	製造	大阪
小林悌二	234900	男	製造	東京
鈴木紀子	180000	女	勤労	神奈川
佐藤弘	202000	男	勤労	埼玉
秋山信	138000	男	勤労	大阪

3.2.1　集合関数

集合関数指定は，引数への関数の適用によって導出した値を指定します。

(例)"人事"表を所属でグループ化し，所属ごとの給料の合計を問い合わせる。

- SQL 文

```
SELECT 所属 , SUM(給料) FROM 人事 GROUP BY 所属
```

- 結果

表 3-2　問い合わせ結果

所属	SUM(給料)
勤労	520000
経理	530000
購買	374000
製造	558400

(例)"人事"表の中で，給料の最大値とレコード件数を問い合わせる。

- SQL 文

```
SELECT MAX(給料) , COUNT(*) FROM 人事
```

- 結果

表 3-3　問い合わせ結果

MAX(給料)	レコード件数
234900	11

3.2.2　比較述語

比較述語は２つの値の比較を指定します。

(例)"人事"表の中から所属が'経理'であるレコードの氏名と出身地を問い合わせる。

- SQL 文

```
SELECT 氏名 , 出身地 FROM 人事 WHERE 所属 = '経理'
```

- 結果

表 3-4　問い合わせ結果

氏名	出身地
斎藤明	東京
高木良夫	神奈川
青山信子	千葉

(例)"人事"表の中から性別が'男'である人の給料の平均より給料の多いレコードの氏名と給料を問い合わせる。

- SQL 文

```
SELECT 氏名 , 給料 FROM 人事 WHERE 給料 >
       (SELECT AVG(給料) FROM 人事 WHERE 性別 = '男')
```

- 結果

表 3-5　問い合わせ結果

氏名	給料
斎藤明	185000
青山信子	184000

氏名	給料
吉永一夫	215000
小林悌二	234900
佐藤弘	202000

3.2.3　BETWEEN 述語

BETWEEN 述語は範囲比較を指定します。

"人事"表の中から給料が 150000 以上で 180000 以下のレコードの氏名と給料を問い合わせる。

SQL 文

```
SELECT 氏名 , 給料 FROM 人事 WHERE 給料
       BETWEEN 150000 AND 180000
```

表 3-6　問い合わせ結果

氏名	給料
高木良夫	161000
山田春男	159000
長島明子	176500
鈴木紀子	180000

(例) "人事"表の中から給料が 150000 未満または 200000 を超えるレコードの氏名と給料を問い合わせる。

- SQL 文

```
SELECT 氏名 , 給料 FROM 人事 WHERE 給料
       NOT BETWEEN 150000 AND 200000
```

- 結果

表 3-7　問い合わせ結果

氏名	給料
吉永一夫	215000
高山昌夫	147000
小林悌二	234900
佐藤弘	202000

氏名	給料
秋山信	138000

3.2.4　IN 述語

IN 述語は値の集合との比較を指定します。

(例)"人事"表の中から出身地が'東京'でも'大阪'でもないレコードの氏名と出身地を問い合わせる。

- SQL 文

```
SELECT 氏名 , 出身地 FROM 人事
       WHERE 出身地 NOT IN ('東京','大阪')
```

- 結果

表 3-8　問い合わせ結果

氏名	出身地
高木良夫	神奈川
青山信子	千葉
鈴木紀子	神奈川
佐藤弘	埼玉

(例)"人事"表の中から出身地が'神奈川'である人の所属と同じ所属を持つレコードの氏名と所属を問い合わせる。

- SQL 文

```
SELECT 氏名 , 所属 FROM 人事 WHERE 所属 IN
       (SELECT 所属 FROM 人事 WHERE 出身地 = '神奈川')
```

- 結果

表 3-9　問い合わせ結果

氏名	所属
斎藤明	経理
高木良夫	経理
青山信子	経理
鈴木紀子	勤労
佐藤弘	勤労

氏名	所属
秋山信	勤労

3.2.5　LIKE 述語

LIKE 述語はパターン照合比較を指定します。

(例) "人事"表の中から氏名が'%山%'とパターンが一致する(氏名に'山'という文字を含む) レコードの氏名を問い合わせる。

- SQL 文

```
SELECT 氏名 FROM 人事 WHERE 氏名 LIKE '%山%'
```

- 結果

表 3-10　問い合わせ結果

氏名
青山信子
山田春男
高山昌夫
秋山信

(例)"WKTBL"表から各'LIKE'述語を実行したときの問い合わせ結果。

- 前提

  WKTBL は下記のようになっているものとします。

表 3-11　WKTBL

COL
ABCDE
CABED
BA%DE
_BADC
#ED#B
A%BED
##EAB

- SQL 文と結果

表 3-12　SQL 文と問い合わせ結果

LIKE 述語の形式	LIKE 述語の問い合わせ結果
COL LIKE 'A%'	ABCDE, A%BED
COL LIKE '%E'	ABCDE, BA%DE
COL LIKE '%C%'	ABCDE, CABED, _BADC
COL LIKE 'A%E'	ABCDE
COL LIKE '_B%'	ABCDE, _BADC
COL LIKE 'A%#%%' ESCAPE '#'	A%BED
COL LIKE '#_%C' ESCAPE '#'	_BADC
COL LIKE '##E%' ESCAPE '#'	#ED#B

3.2.6　NULL 述語

NULL 述語は NULL 値のテストを指定します。

（例）"人事"表の中から所属が NULL 値であるレコードの氏名を問い合わせる。

- SQL 文

```
SELECT 氏名 FROM 人事 WHERE 所属 IS NULL
```

- 結果

表 3-13　問い合わせ結果

氏名

3.2.7　限定述語

限定述語は，限定した値の集合との比較を指定します。

（例）"人事"表の中から所属が'勤労'であるすべての人より給料が多いレコードの氏名と給料を問い合わせる。

- SQL 文

```
SELECT 氏名 , 給料 FROM 人事 WHERE 給料 > ALL
       (SELECT 給料 FROM 人事 WHERE 所属 = '勤労')
```

- 結果

表 3-14　問い合わせ結果

氏名	給料
吉永一夫	215000
小林悌二	234900

（例）"人事"表の中から所属が'勤労'である人の誰かより給料の少ないレコードの氏名と給料を問い合わせる。

- SQL 文

```
SELECT 氏名 , 給料 FROM 人事 WHERE 給料 < ANY
       (SELECT 給料 FROM 人事 WHERE 所属 = '勤労')
```

- 結果

表 3-15　問い合わせ結果

氏名	給料
斎藤明	185000
高木良夫	161000
青山信子	184000
山田春男	159000
高山昌夫	147000
長島明子	176500
鈴木紀子	180000
秋山信	138000

3.2.8　探索条件

探索条件は, 指定する条件に論理演算子を適用して"真"または"偽"となる条件を指定します。

（例）"人事"表の中から性別が'男'で出身地が'大阪'であるレコードの氏名と所属を問い合わせる。

- SQL 文

```
SELECT 氏名 , 所属 FROM 人事
       WHERE 性別 = '男' AND 出身地 = '大阪'
```

- 結果

表 3-16　問い合わせ結果

氏名	所属
高山昌夫	製造
秋山信	勤労

(例)"人事"表の中から性別が'女'か，所属が'経理'でないレコードの氏名と所属を問い合わせる。

- SQL 文

```
SELECT 氏名 , 所属 FROM 人事
       WHERE 性別 = '女' OR NOT 所属 = '経理'
```

- 結果

表 3-17　問い合わせ結果

氏名	所属
青山信子	経理
山田春男	購買
吉永一夫	購買
高山昌夫	製造
長島明子	製造
小林悌二	製造
鈴木紀子	勤労
佐藤弘	勤労
秋山信	勤労

3.2.9　FROM 句

FROM 句は，１つ以上の名前付きの表から導出する表を指定します。

(例)FROM 句に"T1"表，"T2"表を適用し導出表を作成する。

- SQL 文

```
SELECT * FROM T1 , T2
```

- 結果

表 3-18　T1

A	B	C
1	2	3
1	1	1

表 3-19　T2

D	E	F
1	2	3
2	2	2

表 3-20　導出表

A	B	C	D	E	F
1	2	3	1	2	3
1	1	1	1	2	3
1	2	3	2	2	2
1	1	1	2	2	2

3.2.10　GROUP BY 句

GROUP BY 句は，前に指定した句の結果に GROUP BY 句を適用することによって導出するグループ表を指定します。

(例)"T1"表に GROUP BY 句を適用することにより導出表を作成する。

- SQL 文

```
GROUP BY A , B
```

- 結果

表 3-21　T1

A	B	C
1	2	3
2	1	1
1	1	2
2	2	1
1	1	3
1	2	2

表 3-22　T1 より生成したグループ表

A	B	C
1 1	1 1	2 3
1 1	2 2	3 2
2	1	1
2	2	1

3.2.11　HAVING 句

HAVING 句は，直前で指定した句の結果であるグループ表に対し，探索条件に合うグループだけを選び出す制限を指定します。

(例)"T1"表に列 A，B グループ化を行い，作成したグループ表から列 C の合計が 3 以上のレコードをもつグループ表を作成する。

- SQL 文

```
HAVING SUM(C) > 3
```

- 結果

表 3-23　T1

A	B	C
1	2	3
2	1	1
1	1	2
2	2	1
1	1	3
1	2	2

表 3-24　T1 より生成したグループ表

A	B	C
1 1	1 1	2 3
1 1	2 2	3 2
2	1	1
2	2	1

表 3-25　グループ表に HAVING 句を適用したグループ表

A	B	C
1 1	1 1	2 3
1 1	2 2	3 2

3.2.12　カーソル

カーソルとは，ホストプログラムでは一度に表中の 1 つの行しか扱うことができないため，検査結果が複数行におよぶ場合の対処として導入している概念です。

検査結果をいったん作業用の領域に保存し，カーソルによって指します。ホストプログラムではそこから一行ずつ取り出して処理することができます。

（例）"人事"表から性別が"男"である人を所属ごとにその所属と給料の最大値を 2 番目の列の昇順，すなわち給料の最大値が少ない順に並べた表をカーソル CUR1 として定義する。

- SQL 文

```
DECLARE CUR1 CURSOR FOR SELECT 所属 , MAX(給料)
        FROM 人事 WHERE 性別 = '男'
        GROUP BY 所属 ORDER BY 2 ASC
```

- 結果

表 3-26　検査結果

所属	MAX(給料)
経理	185000
勤労	202000
購買	215000
製造	234900

（例）"人事"表から給料が最大の人の氏名と給料からなる問い合わせと，"人事"表から給料が最小の人の氏名と給料からなる問い合わせを併合した表をカーソル CUR2 として定義する。

- SQL 文

```
DECLARE CUR2 CURSOR FOR
        SELECT 氏名 , 給料 FROM 人事
```

```
                WHERE 給料 = (SELECT MIN(給料) FROM 人事)
        UNION ALL
        SELECT 氏名 , 給料 FROM 人事
                WHERE 給料 = (SELECT MAX(給料) FROM 人事)
```

- 結果

表 3-27　検査結果

氏名	給料
小林悌二	234900
秋山信	138000

3.2.13　DELETE 文(位置づけ)

カーソルで位置づけた行を削除します。

(例)"人事"表からカーソル CUR1 に関連付けている行を削除する。

- SQL 文

```
DECLARE CUR1 CURSOR FOR
        SELECT 氏名 FROM 人事 WHERE 所属 = '購買'
               :
OPEN CUR1
               :
FETCH CUR1 INTO :SHIMEI
               :
DELETE FROM 人事 WHERE CURRENT OF CUR1
```

- 結果

:SHIMEI に'吉永一夫'が代入されたところで，本 DELETE 文を行うと，氏名が'吉永一夫'のレコードを削除します。

人事表は以下のようになります。

表 3-28　人事

氏名	給料	性別	所属	出身地
斎藤明	185000	男	経理	東京
高木良夫	161000	男	経理	神奈川
青山信子	184000	女	経理	千葉
山田春男	159000	男	購買	東京
高山昌夫	147000	男	製造	大阪
長島明子	176500	女	製造	大阪
小林悌二	234900	男	製造	東京
鈴木紀子	180000	女	勤労	神奈川

氏名	給料	性別	所属	出身地
佐藤弘	202000	男	勤労	埼玉
秋山信	138000	男	勤労	大阪

3.2.14　DELETE 文(探索)

表から複数のレコードを削除します。

(例)"人事"表から出身地が'東京'である人のデータをすべて削除する。

- SQL 文

```
DELETE FROM 人事 WHERE 出身地 = '東京'
```

- 結果

表 3-29　人事

氏名	給料	性別	所属	出身地
高木良夫	161000	男	経理	神奈川
青山信子	184000	女	経理	千葉
高山昌夫	147000	男	製造	大阪
長島明子	176500	女	製造	大阪
鈴木紀子	180000	女	勤労	神奈川
佐藤弘	202000	男	勤労	埼玉
秋山信	138000	男	勤労	大阪

3.2.15　FETCH 文

カーソルを表の次の行に位置づけ，その行の値を取り出します。

(例)"人事"表から給料の昇順に並び替えたカーソル CUR1 を定義し，最初の行の氏名と給料を取り出す。

- SQL 文

```
DECLARE CUR1 CURSOR FOR
        SELECT 氏名 , 給料 FROM 人事 ORDER BY 給料
        :
OPEN CUR1
        :
```

```
FETCH CUR1 INTO :SHIMEI , :KYUURYOU
```

- 結果

  カーソル CUR1 が位置づいた行(先頭行)の氏名，給料の値を取り出し，それぞれホスト変数 SHIMEI，KYUURYOU に代入します。

  ホスト変数 SHIMEI には'秋山信'，KYUURYOU には'138000'が入ります。

3.2.16　INSERT 文

表中に新しい行を追加します。

(例)"新入社員"表に氏名がホスト変数　:HOST1 の値で性別がホスト変数　:HOST2 の値で出身地がホスト変数　:HOST3 の値である行を挿入する。

- SQL 文

```
MOVE "橋本豊明" TO HOST1.     … COBOL 記述
MOVE "男"       TO HOST2.     … COBOL 記述
MOVE "神奈川"    TO HOST3.     … COBOL 記述
        … ホスト変数への値設定はホストプログラムで設定します。

INSERT INTO 新入社員 (氏名 , 性別 , 出身地)
       VALUES (:HOST1 , :HOST2 , :HOST3)
```

- 結果

表 3-30　元の新入社員表

氏名	性別	希望	出身地
青山睦子	女	経理	大阪
岩間敏男	男	勤労	東京
西山峰男	男	製造	千葉
河村豊	男	購買	東京

表 3-31　INSERT 文実行後の新入社員表

氏名	性別	希望	出身地
青山睦子	女	経理	大阪
岩間敏男	男	勤労	東京
西山峰男	男	製造	千葉
河村豊	男	購買	東京
橋本豊明	男		神奈川

注

下線で示したデータが挿入データです。

(例) 新入社員で希望が'経理'である人すべてを"人事"表に所属を'経理'として挿入する。

- SQL 文

```
INSERT INTO 人事 (氏名 , 性別 , 所属 , 出身地)
       SELECT 氏名 , 性別 , '経理' , 出身地 FROM 新入社員
              WHERE 希望 = '経理'
```

- 結果

表 3-32　人事

氏名	給料	性別	所属	出身地
斎藤明	185000	男	経理	東京
高木良夫	161000	男	経理	神奈川
青山信子	184000	女	経理	千葉
山田春男	159000	男	購買	東京
吉永一夫	215000	男	購買	東京
高山昌夫	147000	男	製造	大阪
長島明子	176500	女	製造	大阪
小林悌二	234900	男	製造	東京
鈴木紀子	180000	女	勤労	神奈川
佐藤弘	202000	男	勤労	埼玉
秋山信	138000	男	勤労	大阪
青山睦子		女	経理	大阪

"人事"表への挿入時，給料を指定していないため，給料は NULL 値となります。

注

下線で示したデータが挿入データです。

3.2.17　UPDATE 文（位置づけ）

カーソルが位置づいた表の１つの行を更新します。

（例）“新入社員”表中でカーソルが位置づいている行の、出身地の列の値を‘東京’に更新する。

- SQL 文

```
UPDATE 新入社員 SET 出身地 = '東京' WHERE CURRENT OF CUR1
```

- 結果

'西山峰男'の行に位置づいていたとすると"新入社員"表は次のようになります。

氏名	性別	希望	出身地
青山睦子	女	経理	大阪
岩間敏男	男	勤労	東京
西山峰男	男	製造	東京
川村豊	男	購買	東京

注

下線で示したデータが更新か所になります。

3.2.18　UPDATE 文（探索）

表の複数の行を更新します。

（例）“人事”表で‘長島明子’と‘鈴木紀子’の所属を‘経理’に変更する。

- SQL 文

```
UPDATE 人事 SET 所属 = '経理'
       WHERE 氏名 = '長島明子' OR 氏名 = '鈴木紀子'
```

- 結果

氏名	給料	性別	所属	出身地
斉藤明	185000	男	経理	東京
高木良夫	161000	男	経理	神奈川
青山信子	184000	女	経理	千葉
山田春男	159000	男	購買	東京
吉永一夫	215000	男	購買	東京
高山昌夫	147000	男	製造	大阪
長島明子	176500	女	経理	大阪

氏名	給料	性別	所属	出身地
小林悌二	234900	男	製造	東京
鈴木紀子	180000	女	経理	神奈川
佐藤弘	202000	男	勤労	埼玉
秋山信	138000	男	勤労	大阪

注

下線で示したデータが更新か所になります。

3.3　動的 SQL 文

動的 SQL 文を使用すると，アプリケーションプログラム実行時に ACCEPT 文などによって入力した SQL 文や，プログラムによって生成した SQL 文を実行することができます。

また，SQL 文の一部をパラメータとして実行時に値を設定することができます。

3.3.1　動的 SQL 文を実行する

ここでは ACCEPT 文で入力した SQL 文を実行する例を示します。

(例)ACCEPT 文で入力した SQL 文を実行する。

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC.
77 STM1 PIC X(200).
       …ホスト変数を宣言する
EXEC SQL END DECLARE SECTION END-EXEC.
      :
     ACCEPT STM1 FROM CONSOLE.
        …ホスト変数 STM1 にユーザが入力した SQL 文を格納する

      :
        …入力した SQL 文を一度のみ実行する場合
EXEC SQL
  EXECUTE IMMEDIATE  :STM1
        …“EXECUTE IMMEDIATE”文で実行する
END-EXEC.

      :
        …入力した SQL 文を複数回実行する場合
EXEC SQL
  PREPARE SQLH1 FROM :STM1
        …“PREPARE”文で SQL 文を準備する
END-EXEC.
      :
EXEC SQL
  EXECUTE SQLH1
        …“EXECUTE”文で準備した SQL 文を実行する
END-EXEC.
      :
EXEC SQL
  EXECUTE SQLH1
        …もう一度同じ SQL 文を実行したい場合は“PREPARE”文で準備した
         同じ SQL 文識別子(SQLH1)を使用し“EXECUTE”文を実行する。
         ただし再実行する前に同じ SQL 文識別子を使用し“PREPARE”文で
```

```
                SQL 文を準備した場合，新たに準備した SQL 文を実行する。
       END-EXEC.
```

例で示した ACCEPT 命令実行時に

```
INSERT INTO 人事 VALUES ('川村豊',128000,'男','購買','東京')
```

と入力すると"EXECUTE IMMEDIATE"文もしくは“PREPARE”文，"EXECUTE"文実行時に上記 SQL 文を実行します。

また ACCEPT 命令の記述を

```
MOVE "INSERT INTO 人事 VALUES ('川村豊',128000,'男','購買','東京')"
     TO STM1.
```

と記述しても同じ結果となります。

3.3.2　動的パラメータを指定する

動的 SQL 文で SQL 文にパラメータを指定する方法を示します。

（例）“人事”表から ACCEPT 命令で入力したデータ以上の給料である人の導出表を作成する。

```
       EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       77 STM1    PIC X(200).
       77 HKYURYOU PIC S9(6) LEADING SEPARATE.
             …ホスト変数を宣言する

       EXEC SQL END DECLARE SECTION END-EXEC.
                :
       EXEC SQL
         DECLARE C1 CURSOR FOR SQLCU1
             …動的カーソル宣言を行う。
       END-EXEC.
                :
            MOVE "SELECT * FROM 人事 WHERE 給料 >= ?" to stm1.
              …SQL 文で動的パラメータを渡す部分については'?'を記述する
                :
            ACCEPT HKYURYOU FROM CONSOLE.
              …動的パラメータで渡す値を入力する。
                :
       EXEC SQL
         PREPARE SQLCU1 FROM :STM1
             …SQL 文を準備する。
       END-EXEC.
                :
       EXEC SQL
         OPEN C1 USING :HKYURYOU
             …ホスト変数'HKYURYOU'を動的パラメータとしカーソル'C1'を
               オープンする。
       END-EXEC.
```

付録 A. 注意／制限事項

1. 選択文字はプリコンパイル時に指定します。
2. COBOL プログラム中で使用出来ない利用者語は以下のとおりです。
   - SP ではじまる CALL 命令の定数－1
   - SP ではじまる変数の宣言

     ただしオプションのプリフィックス変更で SP 以外に変更することは可能です。
   - SQLCA の宣言

付録 B. エラーメッセージ

SQL プリコンパイラがプリコンパイル中にエラーを検出した場合の処理について説明します。

B.1 診断メッセージ一覧

SQL プリコンパイラが文法 エラー等を検出した場合に出力する診断メッセージは、標準出力およびリストファイルに出力します。

出力時の形式は、以下のとおりです。

出力形式

```
"ファイル名"(行番号)△X△Ynnn:メッセージ本文
```

ファイル名

原始プログラム，または COPY 原文登録集のファイル名を出力します。

行番号

エラーが発生した行の，ファイル内行番号を整数で出力します。

X

エラー種別を，以下のとおり出力します 。

エラー種別	説明
F	FATAL エラー
W	WARNING エラー

Y

エラーのカテゴリを現す文字を，以下のとおり出力します。

エラーカテゴリ	説明
A	COBOL 字句解析エラー
B	EXEC SQL，END-EXEC の記述に関するエラー
C	SQL 構文解析エラー
D	ホスト変数に関するエラー

nnn

エラー番号を整数 3 桁で出力します。

メッセージ本文

診断メッセージを出力します。文中の‘@’は，ある語などを埋め込むことを示しています。

診断メッセージ本文の一覧

COBOL 字句エラー解析エラーメッセージ

エラー番号	種別	メッセージ
A001	F	許されない文字がある
A010	W	標識領域に許されない文字がある。その文字は無視される
A011	W	継続行は B 領域から始まらなければならない。B 領域から始まっているものとみなす
A012	F	語の最大長である 30 文字を超えている
A013	F	ユーザ語として正しくない文字がある
A015	F	日本語による利用者語の長さが最大文字数を超えている
A016	F	文字定数の長さが最大文字数 256 を超えている
A017	F	継続の始まりが正しくない
A018	F	文字列の終りが正しくない
A019	F	日本語定数の長さが最大文字数 128 を超えている
A020	F	0～9、A～F 以外の文字が含まれている
A021	F	定数長は偶数でなければならない
A022	F	定数の桁数がゼロである
A023	F	一意名の修飾が正しくない
A024	F	定数長は 4 の倍数でなければならない
A025	F	PICTURE 文字列として正しくない文字がある
A026	F	ユーザ語の最後にハイフンを記述してはならない
A027	F	添字付けが正しくない
A028	F	部分参照付けが正しくない
A029	F	BY 句がない
A032	F	@DIVISION の記述が誤っている
		（@には記述が誤っている DIVISION 名が入る）
A033	F	@SECTION の記述が誤っている
		（@には記述が誤っている SECTION 名が入る）

EXEC SQL，END-EXEC の記述に関するエラーメッセージ

エラー番号	種別	メッセージ
B001	F	予約語 EXEC の次が SQL でない

エラー番号	種別	メッセージ
B002	F	END-EXEC がない
B003	W	ピリオドがない。このエラーを無視する
B004	F	予約語 SQL の前に EXEC がない
B005	F	この END-EXEC に対する EXEC SQL が記述されていない
B007	F	この SQL 文は、作業場所節に記述しなければならない
B008	F	この SQL 文は、手続き部に記述しなければならない
B009	F	END DECLARE SECTION が現れなかった
B010	F	BEGIN DECLARE SECTION が記述されていない
B012	F	ホスト変数@は、定義されていない
		(@には、定義されていないホスト変数名が入る)
B013	F	カーソル名@は、定義されていない
		(@には、定義されていないカーソル名が入る)
B014	F	カーソル名@は、二重に定義されている
		(@には、二重に定義されているカーソル名が入る)
B015	F	標識変数@の属性が、数値型でない
		(@には、数値型ではない標識変数名が入る)
B016	F	標識変数@に、小数部がある
		(@には、小数部がある標識変数名が入る)
B017	F	ホスト変数@の属性が、文字型、または可変長文字型でない
		(@には、文字型、または可変長文字型でないホスト変数名が入る)
B019	F	ホスト変数@の属性が、可変長文字型でない
		(@には、可変長文字型ではないホスト変数名が入る)
B020	F	入力パラメータは静的カーソルの OPEN 文には指定できない
B021	F	ホスト変数 @ は一意ではない
		(@には、ホスト変数名が入る)
B022	F	ホスト変数 @ の修飾が制限を超えている
		(@には、ホスト変数名が入る)
B023	F	INCLUDE ファイルに INCLUDE 文が含まれていてはならない
B024	F	標識変数 @ が、符号付きでない
		(@には、ホスト変数名が入る)

SQL 解析エラーメッセージ

エラー番号	種別	メッセージ
C001	F	@の記述が誤っている
		(@には、記述が誤っている SQL 文が入る)
C002	F	@の長さが 18 文字を超えている
		(@には、”識別子””、サーバ名”、”コネクション名”が入る)

エラー番号	種別	メッセージ
C003	F	@の長さが 30 文字を超えている
		(@には、”手続き名”、”埋込み変数名”が入る)
C006	F	ユーザ名／パスワードの長さが 256 文字を超えている
C007	F	日本語識別子の長さが 9 文字を超えている
C008	F	真数定数の長さが 18 桁を超えている

ホスト変数に関するエラーメッセージ

エラー番号	種別	メッセージ
D001	W	必要な終止符がない
D002	W	A 領域に書くべき語が B 領域にある
D003	F	レベル番号が誤っている
D004	F	@が誤っている
		(@には、誤っている文字列が入る)
D005	F	@句は既に指定されている
		(@には、２重に指定されている文字列が入る)
D006	F	@句の書き方が誤っている
		(@には、誤っている文字列が入る)
D007	F	@句はホスト変数に指定できない
		(@には、”ホスト変数”が入る)
D008	F	SIGN 句は内部 10 進項目に指定できない
D009	F	内部 10 進項目に対する PICTURE 句の指定が誤っている
D011	F	内部 10 進項目に P 項目は指定出来ない
D012	F	PICTURE 句が指定されていない
D013	F	SIGN 句に関連する基本項目が符号付きでない
D015	F	外部 10 進項目に P 項目は指定できない
D016	F	ホスト変数に指定できない項類である
D017	F	ホスト変数に指定できない USAGE である
D019	F	2 進項目に PICTURE 句は指定できない
D020	F	SIGN 句は 2 進項目に指定できない
D021	F	可変長文字列／可変長日本語文字列型のホスト変数の構成が誤っている
D023	F	上位レベルに VALUE 句の指定されている項目の USAGE が誤っている
D024	F	項類と VALUE 句の定数が矛盾している
D026	F	無名項目はホスト変数に指定できない
D027	F	@句はホスト変数に指定できない
		(@には、”ホスト変数”が入る)
D028	F	集団項目に@句は指定できない

エラー番号	種別	メッセージ
		(@には、”句の名前”が入る)
D029	F	ホスト変数@は既に定義されている
		(@には、”ホスト変数”が入る)
D030	F	PICTURE 文字列が誤っている

B.2 エラーメッセージ一覧

SQL プリコンパイラ 処理中に，致命的なエラーの検出などで，SQL プリコンパイラ が処理をアボートする場合は，標準エラー出力にエラーメッセージを出力します。

出力時の形式は，以下のとおりです。

出力形式

```
CBLSQLEX△:△nnnn:メッセージ本文[(追加情報)]
```

nnnn

エラー番号を整数 4 桁で出力します。

メッセージ本文

エラーメッセージを出力します。

追加情報

エラーに対して追加情報がある場合に，出力します。

エラーメッセージ本文の一覧

エラー番号	メッセージ本文
0001	オプション指定に誤りがあります
0002	ディレクトリ名の指定がありません
0003	ディレクトリ名の指定が誤っています
0004	複数の入力ファイルを指定した時は、-O でディレクトリ名を指定してください
0005	複数の入力ファイルを指定した時は、-H でディレクトリ名を指定してください
0006	指定した値が大きすぎます
0007	指定した文字列が長すぎます
0008	入力ファイル名と、出力ファイル名が同じです
0009	入力ファイル名と、リストファイル名が同じです
0010	テンポラリファイルのパス名が長すぎます
0011	ファイルアクセスエラーが発生しました

エラー番号	メッセージ本文
0012	動的メモリの確保に失敗しました
0013	入力レコード中の TAB 文字を編集した結果、最大レコード長を超えました
0014	致命的エラーの数が 100 を超えました。処理を中止します
0015	内部エラーが発生しました

COBOL SQL アクセス
プログラミングの手引

2018 年 10 月 3 版 発行

日本電気株式会社