Coal

言語仕様書編

第４～７章

第　６．４　版

２０２０年2月１３日

目　　次

4. 式

4.1. 一般形式

ここでの表記は、必ずしも厳密ではなく、一般的な形式を示している。括弧の挿入、左右の辺の交換等は適宜可能である。
なお、スクリプトの定義部分では、ユーザ定義関数は指定できない。

式は

［キャスト演算子］ { 代入式 / 論理式 / 比較式 / 算術式 / 文字列式 / ３項演算式 / 範囲式 / ドット式 / 関数式 / 名前付き引数式 / ｲﾝｽﾀﾝｽ生成式 }

（注）代入式については、ＬＥＴを参照。
（注）算術式には、ビット式を含むものとする。
（注）式を指定できる所には、式をカンマで区切って、複数個指定できる。このときには、最後の式が評価対象となる。（関数式の括弧内を除く）
（注）データ要素以外の式にキャスト演算子を付けるときは、式全体を括弧で囲む。

論理式は

{ ［論理単項演算子］（ 論理式 ） / 比較式 [論理演算子 （ 論理式 ）] }

比較式は

算術式 [比較演算子 （ 比較式 ）]

または

文字列式 [比較演算子 （ 比較式 ）]

算術式は

{ ［算術単項演算子］（ 算術式 ） / ［算術単項演算子］ データ要素 [算術演算子 （ 算術式 ）] }

文字列式は

データ要素 [文字列演算子 （ 文字列式 ）]

３項演算式は

```
論理式　？　式　：　式
```

範囲式は

```
式　．．式
```

(注)第一番目の式が下限値、第二番目の式が上限値。

ドット式は

```
式１　．式２
```

構造体メンバ、クラス変数にアクセスするため。または、クラス関数を実行するために使用する。

名前付き引数式は

```
{ 仮引数名 ＝＝＞ 式１
  式１ ＜＝＝ 仮引数名 }
```

本式は、手続き呼び出しまたは関数式の実引数部で使用する。
式１には、引数に設定する値を指定する。
仮引数名は、名称文字列または式で指定する。
式は、値が仮引数名となる。

インスタンス生成式は

```
ＮＥＷ　クラス名 [ （ 式〔，式，・・・〕） ]
```

データ要素は

```
{ [ アクセス修飾子 △ ] [*]変数
  定数
  関数式
  データ・リスト式
  [*](式) }
```

(注)式中の文字定数は、スクリプトが旧仕様でも、新仕様で処理される。
アクセス修飾子は、内部名称変数または外部変数が直接指定されているときにのみ指定できる。
'*'は、データ指定単項演算子である。

関数式は

```
[{手続き名|関数名}.・・・.] 関数名（ [ 式 ] , [ 式 ] , ・・・ ）
```

式には、名前付き引数式を指定できる。仮引数との対応は、手続き呼び出しと同じ。

関数名は

```
{ 名称文字列
  式(値は関数名 または ″{}″で囲った関数本体の文字列を関数化した値)
  関数化関数(FF)の戻り値(内容は関数名) }
```

変数は、配列変数

変数名 [[式] , [式] , ・・・]

(注)'[',']'は、そのまま指定する。

または、スカラー変数

{ [$] / % / # } { 整数定数 / 名称文字列 / 変数 / (式) }

(注) $,%,#がネストする場合は、$は省略できない。

データ・リスト式は

左中括弧({) [式] , [式] , ・・・ 右中括弧(})

論理演算子は

{ AND | OR }
または
{ && / || }

比較演算子は

{ EQ | NE | GT | LT | GE | LE }
または
{ == | != | NOT= | <> | >< | > | < | >= | => | <= | =< }

各記号は、以下を意味する

No	記号	意味
1	EQ、==	＝
2	NE、!=、NOT=、<>、><	≠
3	GT、>	＞
4	LT、<	＜
5	GE、>=、=>	≧
6	LE、<=、=<	≦

特殊な比較演算子は

```
{ ＬＩＫＥ｜ｉＬＩＫＥ｜ｉＥＱ｜ｉＮＥ｜
  ＩＮＳＴＲ｜ＩＮｉＳＴＲ｜ＩＮＲＳＴＲ｜ＩＮｉＲＳＴＲ
  ＩＮ｜ｉＩＮ }
```

各記号は、以下を意味する。詳細は、対応する関数を参照。

No	記号	意味
1	ＬＩＫＥ	パターンマッチング
2	ｉＬＩＫＥ	大文字、小文字を区別しない パターンマッチング
3	ｉＥＱ	大文字、小文字を区別しない＝
4	ｉＮＥ	大文字、小文字を区別しない≠
5	ＩＮＳＴＲ	文字列サーチ
6	ＩＮｉＳＴＲ	大文字、小文字を区別しない 文字列サーチ
7	ＩＮＲＳＴＲ	後ろからの文字列サーチ
8	ＩＮｉＲＳＴＲ	大文字、小文字を区別しない 後ろからの文字列サーチ
9	ＩＮ	複数文字列比較
10	ｉＩＮ	大文字、小文字を区別しない 複数文字列比較

算術演算子は

```
{ ＋ ｜ － ｜ ＊ ｜ ／ ｜ ％ ｜ ＭＯＤ ｜ ＊＊ }
```

または、以下のビット演算子

```
{ ＆ ｜ 縦棒(｜) ｜ ＾ ｜ ＜＜ ｜ ＞＞ }
```

文字列演算子は

```
{ ＣＯＮＣＡＴ ｜ ＆[＋] ｜ 縦棒(｜)[＋]
  ＳＵＢＳＴＲ
  ＲＥＰ
  ＩＳ
  ＣＯＮＤＡＳ
  ＴＯ }
```

算術単項演算子は

```
{ ＋ ｜ － ｜ ～ }
```

論理単項演算子は

{ ! | NOT }

データ指定
単項演算子は

*

配列名、構造体名または範囲値に付け、データ指定属性を付与する。

３項演算子は

? :

キャスト演算子は

(データ型)

キャストされたデータ型を以下の型に変換する。長さ省略時は、変換後のデータ長となる。

データ型名	データ型
CHAR[(長さ)] または **STR**ING[(長さ)]	文字
BIN または INT	整数
LONG または LNG	(倍精度)整数
FLOAT または **DOU**BLE または FLT または DBL	倍精度２進浮動小数点
DEC[(精度[, 位取り])]	１０進浮動小数点 または １０進固定小数点
BULK[(長さ)]	バルク
DATE	日付
VARIANT	バリアント
FUNC	関数

(注)長さ、精度、位取りは、式。倍精度整数は未サポート、短精度になる。

その他演算子は

一部の関数の 関数名

関数名を二項演算の演算子として指定したときは、第一項、第二項を関数の第一引数、第二引数としたときと同じになる。引数が１の関数のときは第一項が使われ、第二項は無視される。

以下の関数群が対応する。（関数一覧を参照）

- 文字列演算子と同じ関数
- TO_XXXX関連
- 文字列演算関連
- 比較・マッチング関連

4.2. 論理式

（1）一般規則

（A）論理演算は、論理値で演算する。演算の両辺は数値属性でなければならない。

（B）両辺の値は、０なら偽、０でないなら真とみなす。

（C）論理演算の結果は、真なら１、偽なら０となる。

4.3. 比較式

（1）一般規則

（A）比較の両辺のデータＩＤは同じ属性でなければならない。

一般データの属性が異なるときは、バルクを除き以下の順位で同じ属性に合わせられる。

（オプション２によりエラーにすることができる）

文字 --> 数値 --> 日付

両辺が数値属性の場合で、データの型が異なる場合は、順位の高い方に変換される。

変換の順位は以下となる。

整数 --> １０進浮動(固定)小数点数 --> ２進倍精度浮動小数点数（順位が高い）

（B）比較演算の結果は、真なら１、偽なら０となる。

（D）文字列の比較は、辞書順の大小で行う。比較は、１文字ごとのコードの大小比較となる。

［例］

（ i） ’ａ’ ＜ ’ｂ’

（ ii） ’ａｂｃ’ ＜ ’ａｂｃｄ’

(iii) ’ａｂｃ’ ＜ ’ｂ’

4.4. 算術式

（1）一般規則

（A）定数又は変数は、数値属性でも文字属性でも良い。文字属性のときは、数値に変換する。

（B）演算において、どちらか一方が配列名のときは、他方は、配列の先頭からの相対位置と解釈され、’+’または’-’の結果は、その位置を先頭とする１次元配列となる。

（C）演算において、どちらか一方が数値属性のときは、他方も数値属性に変換される。

変換の順位は以下となる。

整数 --> １０進浮動(固定)小数点数 --> ２進倍精度浮動小数点数（順位が高い）

（D）ビット演算は、ビット毎に演算する。演算の両辺は整数値に変換される。

（E）べき乗演算においては、演算の両辺は２進倍精度浮動小数点数に変換される。

（F）演算において、どちらか一方が日付属性のときは、日付演算となり、加算または減算のみが使用できる。

(a)両方が日付属性のときは、減算のみが使用でき、日数の差分が計算される。

(b)片方が日付属性のときは、他方は数値に変換され、日数として計算される。

4.5. 文字列式

（1）一般規則

（A）ＩＳ演算子の一部機能を除き、定数又は変数が数値属性のときは、数字文字列に変換する。

変換は、ゼロサプレスされた左詰めであり、負符号は先頭につく。

ＢＵＬＫ属性のときは、１６進表記文字列に変換する。変換はデータの１バイトを０～９、ａ～ｆの２バイトで表わす。

（2）文字列演算子

（A）文字列演算子の一覧

No.	文字列演算子	機能概要
1	ＣＯＮＣＡＴ ＆[＋]、｜[＋]	第一項の文字列と第二項の文字列を連結する。
2	ＳＵＢＳＴＲ	第一項の文字列から第二項で示される文字列を切り出す。
3	ＲＥＰ	第一項の文字列中の置換指定を第二項の文字列と置き換える。
4	ＩＳ	第一項または第一項を文字列に変換したものを第二項の指定で調べ、結果を返す。
5	ＣＯＮＤＡＳ	第一項の値に対して第二項で与えられる条件式を実行する。
6	ＴＯ	第一項の値を第二項で与えられる条件で変換する。

(注)文字列演算子は、関数名としても登録されているため、他の関数名と同じ扱いになる。

（B）文字列演算子の機能

（a）文字列の連結（演算子は、**ＣＯＮＣＡＴ　または、＆[＋]　または、｜[＋]**）

第一項の文字列と第二項の文字列を連結する。

第一項と第二項が文字列のときは、＆または｜でもよい。

（b）文字単位の文字列の切出し（演算子は、**ＳＵＢＳＴＲ**）

第一項の文字列から第二項で示される文字列を切り出す。

第二項は、以下の形式の文字列で指定する。

’切出し開始位置［，切出し文字数］’

（ｉ）　切出し開始位置は先頭を１として数える。
　　負のときは、末尾から数える。

（ii）　切出し文字数を省略したときは、切出し開始位置以降を切出す。

(iii)　切出し開始位置が範囲外のときはヌル文字列となる。

（iv）　切出し文字数が負または実際の文字数を越えるときは、省略時と同じ扱いとなる。

第二項が数値属性のときは、切出し開始位置のみが指定されたものとして扱う。

（c）文字列の置換（演算子は、**ＲＥＰ**）

第一項の文字列中の置換指定を第二項の文字列と置き換える。

置換指定は、以下の形式で指定する。

＠ｎ＠　　ｎは０以上の整数で第二項の置換文字列の順序を示す。（注１）

第二項は、置換指定の順序に対応する文字列を単価記号（@）２個（注１）で区切って指定する。

第一項文字列中に同じ置換指定があったときは、その全てが置き変わる。

また、第二項に対応するものがないとき、

（ i）［第１項］が＠で指定されている場合
　　そのまま指定を残す。

(ii)［第１項］が＆で指定されている場合
　　ヌル文字列と置き換わる

＜例１＞
［第１項］　＠０＠（＠１＠，＠２＠，＠３＠）――――　＄１
〔＆０＆（＆１＆，＆２＆，＆３＆）〕

［第２項］　0.5%以下＆＆1g＆＆105°＆＆1時間 ――――　＄２

＄３　＝　＄１　ＲＥＰ　＄２；
＄３　――――　0.5%以下（1g，105°，1時間）となる。
〔＄３　――――　0.5%以下（1g，105°，1時間）〕

＜例２＞
［第１項］　＠０＠（＠１＠，＠２＠，＠３＠）――――　＄１
〔＆０＆（＆１＆，＆２＆，＆３＆）〕

［第２項］　0.5%以下＆＆　――――　＄２

＄３　＝　＄１　ＲＥＰ　＄２；
＄３　――――　0.5%以下（，＠２＠，＠３＠）となる。
〔＄３　――――　0.5%以下（，，）　〕

＜例３＞
［第１項］　＠０＠（＠１＠，＠２＠，＠３＠）――――　＄１
〔＆０＆（＆１＆，＆２＆，＆３＆）〕

［第２項］　0.5%以下　――――　＄２

＄３　＝　＄１　ＲＥＰ　＄２；
＄３　――――　0.5%以下（＠１＠，＠２＠，＠３＠）となる。
〔＄３　――――　0.5%以下（，，）　〕

（注１）アンパサント（&）２個を使用してもよい。

（ｄ）文字列の調査（演算子は、**ＩＳ**）
第一項または第一項を文字列に変換したものを第二項の指定で調べ、結果を返す。
第二項は先頭の文字列で判定する。

No	調査項目	第二項の指定 (小文字も同じ)	結果と例
1	文字列長	L	旧仕様：文字列バイト長 新仕様：文字列文字数 ＜例＞ $1 = 'abcd' IS 'length'; $1-->4
	文字列バイト長	ＬＢ	文字列バイト長
2	数字列	Ｎ、Ｄ、Ｆ	整数属性または整数文字列のとき１、 ２進浮動少数点属性または２進浮動少数点文字列のとき２、 １０進少数点属性または１０進少数点文字列のとき３、 その他のときは０となる(数値定数を参照)。 文字列のときは、前後のブランクは無視される。 ＜例＞ $1 = '-12' IS 'numelic'; $1-->1 $2 = '-12.34' IS 'numelic'; $2-->2 $3 = '-12A' IS 'numelic'; $3-->0 $4 = '-12.34E+01' IS 'numelic'; $4-->2
3	半角ブランク	Ｂ、Ｓ	半角ブランクのとき、１、その他のとき、０となる。 ＜例＞ $1 = ″ ″ IS 'blank'; $1-->1
4	１６進文字列	Ｘ	ＢＵＬＫ属性のとき、または文字列が半角ブランク、小数点、符号、数字、'a'～'f'、'A'～'F'の組み合わせのとき、１となり、その他のときは０となる。
5	全角文字	Ｚ	先頭文字が全角文字のとき、１となり、その他のときは０となる。
6	半角文字	Ｈ	先頭文字が半角文字のとき、１となり、その他のときは０となる。
7	ＡＮＫ文字	Ａ	先頭文字がＡＮＫ文字のとき、１となり、その他のときは０となる。
8	文字のバイト数	Ｍ	先頭文字のバイト数を返す。
9	属性	Ｔ	属性値（第一項は変換しない）
10	データＩＤ	Ｉ	データＩＤ（第一項は変換しない）

（ｅ）条件式（演算子は、**ＣＯＮＤＡＳ**）

第一項の値に対して第二項で与えられる条件式を実行する。

条件式の形式は、

区切り文字　比較文字列　区切り文字　真時代入文字列　区切り文字　偽時代入文字列

(i)　区切り文字は、任意の半角文字を使用し条件式の先頭１文字が区切り文字となる。

(ii)　比較文字列は、第一項と比較される文字列であり等しいかどうか比較される。

(iii) 比較の結果が真のとき、真時代入文字列が代入され、偽のとき、偽時代入文字列が代入される。

(iv)　代入文字列は、Ｃ言語の書式指定と同じ形式で指定し、%sが第一項の文字列に置換される。

%qを指定したときは、第一項の文字列中の引用符（’）１個が２個に変換された後に、%sと同じ様に置換される。

(v)　文字列が省略されたときは、ヌル文字列とみなされる。

＜例＞

区切り文字にスラッシュ（/）を使用する場合

```
$3 = $1 CONDAS '//null/''%s''';
```

$1がヌルのときに、$3にはnullという文字列が入り、ヌルでないときには、$1の内容の前後に引用符（’）が付いた文字列が入る。

（ｆ）変換（演算子は、**ＴＯ**）

第一項の値を第二項で与えられる条件で変換する。

第二項が変数データ型を示す文字列のときは、対応する型に変換する。

このときには、データ型には、制度、桁数、長さ、VARIANTは指定できない。

第二項が変数データ型を示す文字列でないときは、以下で変換する。

文字列中を変換する処理では、第一項は、文字列に変換される。

項番	第二項の文字列の先頭	処理
1	Ｚ，ｚ	文字列中の半角英数字記号カナを全角に変換する
2	Ｈ，ｈ	文字列中の全角英数字記号カナを半角に変換する
3	Ｕ，ｕ	文字列中の全角と半角英字を大文字に変換する
4	Ｌ，ｌ	文字列中の全角と半角英字を小文字に変換する
5	Ｉ，ｉ	整数に変換する
6	Ｆ，ｆ，Ｄ，ｄ	２進倍精度浮動小数点に変換する
7	Ｘ，ｘ ［２文字目：Ｓ，ｓ］	１６進文字列に変換する。ａ～ｆは、ｘのときは小文字、Ｘのときは大文字になる。第一項が整数で、第二項２文字目がＳまたはｓのときは、ゼロサプレスする
8	Ｃ，ｃ，Ｐ，ｐ	文字列中の半角英単語の先頭文字を大文字に変換する Ｐ，ｐのときは、２文字目以降を小文字に変換する

4.6. 範囲式

式(第一番目が下限値)と式(第二番目が上限値)を範囲指定子(..)でつなげたものを範囲式という。
式の値には、文字と数値が指定できる。
下限値と上限値が共に文字のときは、先頭１文字が、範囲として使われる。
下限値と上限値のどちらかが数値のときは、以下の順位で変換され、同じ属性に合わされる。
　整数 --> １０進浮動(固定)小数点数 --> ２進倍精度浮動小数点数
以下の場所では、関数の引数および配列のインデックスを除き、増分値＝１で展開される。
~~・ＰＲＩＮＴのデータ~~
・ＦＯＲ　ＥＡＣＨの対象データ
・配列への初期値設定式および代入式
展開されない場合は、範囲値（範囲式の評価結果）が使われる。
バリアント型または可変長文字型の変数に範囲式を代入したときは、範囲値として保存される。
その他の型のときは、下限値がその型で保存される。
変数の範囲値は、式の中では展開されない。

4.7. 式の並び

4.7.1. 式の並びについて

式(代入式を含む)をカンマで区切って並べたものを"式の並び"と言う。
（関数名、クラス名の後ろ、データ型を除く）
式の並びを括弧(())で囲ったものは、通常の括弧と同様に優先的に、前から評価され、最後の要素が括弧内の値として採られる。
特に、式の並びを
①かぎ括弧([])で囲ったものを、"データ並び(式)"
②中括弧({})で囲ったものをデータ・リスト(式)
と言う。
一つのデータ・リスト(式)およびデータ並び(式)は、一つのデータ要素となり、一般の式として扱われる。

4.7.2. 式の並びの扱い

式を指定できるヶ所には、式の並びを指定できる。ただし、データ・リスト(式)を除き、そのか所によって扱いが異なる。
一般の式の途中にカンマがあるときは、カンマの前後は、別の式として扱われるため、一つの式の中で複数の式を処理させたいときは、括弧(())で囲った式の並びを使用するとよい。

（１）ＦＯＲ　ＥＡＣＨの対象データ
①式の並び　　：各式の値が、対象要素となる。
②データ並び(式)　：かぎ括弧の入れ子状態に係わらず、各式の値が、対象要素となる。
③データ・リスト(式)：第一レベルのリスト要素が、対象要素となる。

（２）ＦＯＲ　（；；）の各部分
（Ａ）初期設定部と増分部
①式の並び　　：各式が全て前から順に評価される。
②データ並び(式)　：同上。
③データ・リスト(式)：同上。
（Ｂ）条件部
①式の並び　　：各式が全て前から順に評価され、最後の式の値が条件に使われる。
②データ並び(式)　：各式が全て前から順に評価され、データ並びが条件に使われる。
③データ・リスト(式)：各式が全て前から順に評価され、データ・リストが条件に使われる。

（３）配列の初期値設定式および代入式
①式の並び　　：各式が全て前から順に評価され、式の値が前から順に配列要素に代入される。
②データ並び(式)　：各式が全て前から順に評価され、データ並びが最初の要素に代入される。
③データ・リスト(式)：各式が全て前から順に評価され、データ・リストが最初の要素に代入される。

（4）代入式（=）
①式の並び　　：カンマで区切られた各代入式が全て前から順に評価される。
②データ並び(式)　：左辺の場合：各式が全て前から順に評価され、各評価結果の全てに、右辺の値が代入される。
右辺の場合：各式が全て前から順に評価され、左辺に代入される
・左辺がデータ並び(式)の場合は、左右の先頭から順に対応する要素間で代入される。右辺の要素が足りないときは、最後の要素が代入される。
・左辺がデータ並び(式)でない場合は、データ並びが代入される。
③データ・リスト(式)：左辺の場合：指定できない。
右辺の場合：各式が全て前から順に評価され、左辺に代入される。

（5）ＲＥＴＵＲＮ
①式の並び　　：各式が全て前から順に評価され、最後の式の値が返される。
②データ並び(式)　：各式が全て前から順に評価され、データ並びが返される。
③データ・リスト(式)：各式が全て前から順に評価され、データ・リストが返される。

（5）その他
①式の並び　　：各式が全て前から順に評価され、最後の式の値が使われる。
②データ並び(式)　：各式が全て前から順に評価され、データ並びが使われる。
③データ・リスト(式)：各式が全て前から順に評価され、データ・リストが使われる。

4.8. ドット式

4.8.1. システム関数の実行

式に付けてシステム関数を実行する。
式には、データ値、配列名を結果とする式を指定可能。

```
式 . 関数名([引数リスト])
```

（注）式を第一パラメータ、引数リストを第二パラメータ以降とする関数式と同じ。
静的クラス名に"System"を指定する方法でも実行可能。

4.8.2. ユーザ定義関数の実行

関数へのパスを付けてユーザ定義関数を実行する。
手続きまたは関数をつなげて、実行する関数名へのパスを指定する。

```
関数へのパス . 関数名([引数リスト])
```

（注）手続きまたは関数の入れ子をサポートするオプションを指定したときのみ指定可能。

4.8.3. クラス関数の実行

インスタンス名または静的クラス名を付けて、実行する関数名へのパスを指定する。

```
{ インスタンス名 | 静的クラス名 } . 関数名([引数リスト])
```

（注）システム関数は、静的クラス名に"System"を指定する。

4.8.4. クラス変数へのアクセス

インスタンス名または静的クラス名を付けて、クラス変数へのパスを指定する。

```
{ インスタンス名 | 静的クラス名 } . クラス変数名
```

4.8.5. 構造体メンバ要素へのアクセス

構造体変数を参照。

4.8.6. 文字列の切り出し

指定された定数または変数値から切り出し指定の文字列を切り出す。

```
{ 一般データ変数 | 定数 } . 切り出し指定
```

(注)一般データ変数値は、文字列に変換される。
切り出し指定は、全体を定数、変数、または、式で指定し、内容は以下を指定できる。
変数に配列を使用するとき、または。式を指定するときは、全体を括弧で囲むこと。

```
{ 切り出し開始位置
  切り出し開始位置. . 切り出し終了位置
  '切り出し開始位置, 切り出し文字数' }
```

(注)切り出し規則は、ＳＵＢＳＴＲと同じ。
範囲指定を直接指定するときは、全体を括弧で囲むか、文字列で指定する。

4.9. 集合演算

配列、データ・リストまたはデータ並びに対して以下の集合演算を行うことができる。
配列は、配列名の前にアスタリスク(*)を付けて指定する。
データ・リスト、データ並びまたは一般変数は、変数名または定数をそのまま指定する。
第一項または第二項には、式を指定することができる。

表4.7-1 演算可能なデータ属性の組み合わせ

		第二項			
		一般配列	連想配列	データ・リスト / データ並び	一般変数
第一項	一般配列	一般配列	一般配列	×	一般配列
	連想配列	一般配列	連想配列(キー比較)	×	一般配列
	データ・リスト	×	×	データ・リスト	データ・リスト
	データ並び	×	×	データ並び	データ並び
	一般変数	一般配列	一般配列	データ・リスト / データ並び	×

(注) 各交点の×以外が可能な組み合わせであり、演算結果の属性を示す。

表4.7-2 演算の種類

No.	演算子	機能
1	＋	第一項のデータ要素と第二項のデータ要素を連結する
2	－	第一項のデータ要素から第二項にあるデータ要素のみを削除する
3	&	第一項と第二項の両方にあるデータ要素を連結する
4	|	第一項のデータ要素と第二項のデータ要素を重複がないように連結する
5	^	第一項と第二項の共通でないデータ要素を連結する

~~**(注)** データ・リストの場合は、第一レベルのデータが比較される。~~
~~第一レベルのデータがデータ・リストのときは、その定義情報のみが比較されるため、全て異なるデータとなる。~~

表4.7-3 演算結果

No.	結果のデータ型	演算	要素のインデックス	要素の順序	配列のサイズ
1	一般配列	＋	保存される	保存される	nm1 + nm2
		－	保存される	保存される	max(nm1, nm2)
		&	前詰め	保存される	max(nm1, nm2)
		|	前詰め	保存される	nm1 + nm2
		^	前詰め	保存される	nm1 + nm2
2	連想配列	全て	－	－	－
3	データ・リスト	全て	前詰め	保存される	－
4	データ並び	全て	前詰め	保存される	－

(注) nm1,nm2は、それぞれの項の配列定義サイズ。連想配列のときは、拡張済みエントリ数。
連想配列において同じキーのデータが残る場合は、第一項のデータが残される。

4.10. データ指定単項演算子

配列名、構造体名または範囲値に付け、データ指定属性を付与する。付与されたデータ要素は、それが使用される場所でデータ要素に対する操作がデータ要素属性により変わる。

No	操作	配列名	構造体名	範囲値
1	代入	データ実体がコピーされ、代入される。	データ実体がコピーされ、代入される。	データ指定属性付で、そのまま代入される。
2	PRINT	配列要素の内容が出力される。	構造体要素の内容が出力される。	下限値から上限値を超えない値まで１づつ増えながら出力される。
3	FOR EACH	配列要素の内容が対象データとなる。	使用できない。	下限値から上限値を超えない値まで１づつ増えながら要素データとなる。
4	集合演算	配列要素の内容が演算対象となる。	使用できない。	集合演算とならない。

4.11. インスタンス生成式

クラス名で指定したクラスのインスタンスを生成する。
クラス名の後ろには、実行したいコンストラクタと同じ数のパラメータを指定する。
機能は、ＮＥＷ関数と同じ。

4.12. 名前付き引数式

名前(第一項)と式(第二項)を名前付き引数子(==>)でつなげたもの、または、
式(第一項)と名前(第二項)を名前付引数子(<==)でつなげたものを名前付引数式と言う。
また、この値を名前付引数値と言う。
名前には、名称文字列または値が文字列のなる式で、手続きまたは関数の仮引数名を指定する。
バリアント型または可変長文字型の変数に名前付引数値を代入したときは、そのまま保存される。
名前付引数値は、演算式の中では、元の名前付引数式の値を示す式の値となる。

5. 変数

変数には、パラメータ変数、検索結果読み込み変数、内部変数、外部変数、システム変数がある。
システム変数を除く変数には、表７－１のデータを格納することができる。
外部変数とシステム変数を除く変数は、スクリプト内でのみ有効である。

5.1. パラメータ変数

・手続き、または、関数に引き渡されたデータを扱う。
・パーセント記号（%）で始まる整数で表す。
・呼出し側で指定されたパラメータの順に、％１から順に対応する。
・％０は、パラメータの個数を示す。
・未定義変数値は、ＮＵＬＬ値となる。
・マップド配列は、パーセント記号（%）で始まる名称文字列で表す。

5.2. 検索結果読み込み変数

・ＲＥＡＤコマンド実行時に検索結果が格納される。
・シャープ記号（#）で始まる整数で表す。
・タプル単位にｓｅｌｅｃｔ項目の順に、＃１から順に対応する。
・＃０は、検索カラム数を示す。（＄ＣＯＬＩＭＮと同じ）
・マップド配列は、シャープ記号（#）で始まる名称文字列で表す。

5.3. 内部変数

スクリプトの内部のデータを扱う。

（１）内部番号変数
・ドル記号（$）で始まる整数で表す。
・＄１から始まる。

（２）内部名称変数
・ドル記号（$）で始まる名称文字列で表す。ドル記号は省略できる。
・スカラー変数、配列変数、マップド配列変数がある。
・システム変数と同じ名称は使用できない。
・スクリプト内部(PRIVATE)変数と手続き内部(LOCAL)変数がある。
LOCAL変数は、手続き内でのみ有効である。
PRIVATE変数は、スクリプト内でのみ有効である。
デフォルトの変数種別は、LOCALである。デフォルトはオプションで変更できる。

5.4. 外部変数

・GLOBALは、セッション間で共有されるデータを扱う。
・PUBLICは、スクリプト間で共有されるデータを扱う。セッション内で有効である。
・ドル記号（$）で始まる名称文字列で表す。ドル記号は省略できる。
・スカラー変数と配列変数がある。
・システム変数と同じ名称は使用できない。

表５－２　定義済みPUBLIC外部変数一覧

項番	変　数　名	内　　容	データ型	初期値
1	$MAX_LOOP_WHILE	LOOP WHILEの最大ループ回数	整数	100000
2	$TRNSHID	Start Transactionを発行したホストのホストＩＤが規定バイト単位で連続して格納される。（重複はなし）	文字	ＮＵＬＬ値
3	$UNIX_DATE_FORMAT	Ｃ言語における日付フォーマット	文字	"%Y/%m/%d %H:%M:%S"
4	$SQL_DATE_FORMAT	ＳＱＬにおける日付フォーマット	文字	"YYYY/MM/DD HH24:MI:SS"

5.5. システム変数

ドル記号（$）で始まる英大文字で表す。ドル記号は省略できる。

表５－３　システム変数一覧（1/2）

項番	変　数　名	内　　容	データ型
１	$ＥＲＲＯＲ	直前に実行されたコマンドのリターン値	整数
２	$ＥＲＲＭＳＧ	直前に実行されたコマンドのエラーメッセージ	文字
３	$ＥＲＲＮＯ	システムエラー時のエラー番号	整数
４	$ＳＴＲＥＲＲＯＲ	上記のエラーメッセージ	文字
５	$ＴＵＰＬＥ	検索タプル数	整数
６	$ＣＯＬＵＭＮ	検索カラム数	整数
７	$ＵＳＥＲＩＤ	コマンドヘッダのユーザーコード	整数
８	$ＤＡＴＥ	ＹＹＹＹＭＭＤＤ	文字
９	$ＴＩＭＥ	ＨＨＭＩＳＳ	文字
１０	$ＤＡＴＥＴＩＭＥ	YYYY/MM/DD HH24:MI:SS	文字
１１	$ＨＩＤ	自ホストＩＤ	文字
１２	$ＵＳＥＲＩＮＦ	コマンドヘッダのユーザ情報 （内容はインタフェース仕様書「３．２ 端末－ＷＳ間インタフェース」を参照）	文字
１３	$ＶＥＲＳＩＯＮ	本モジュールのバージョン番号が格納される。 書式は以下の通り。 Ｖ／Ｒ＝バージョン番号 例）Ｖ／Ｒ＝３．１	文字
１４	$ＭＡＫＥＤＡＴＥ	本モジュールの作成日が格納される。 書式は以下の通り。 ＤＡＴＥ＝作成日 例）DATE=Wed Jan 18 14:27:32 JST 1995	文字
１５	$ＳＴＤＩＮ	標準入力へのファイルポインタ	整数
１６	$ＳＴＤＯＵＴ	標準出力へのファイルポインタ	整数
１７	$ＳＴＤＥＲＲ	標準エラー出力へのファイルポインタ	整数
１８	$ＳＣＲＩＰＴＮＡＭＥ	実行スクリプト名	文字
１９	$ＰＲＯＣＮＡＭＥ	実行手続き名または関数名	文字
２０	$ＳＣＲＩＰＴＬＩＮＥ	実行スクリプト行数	整数
２１	$**ＣＬＣＨＡ**Ｒ	文字属性値　（１）	整数
２２	$**ＣＬＢＩＮ**ＡＲＹ	整数属性値　（２）	整数
２３	$**ＣＬＦＬＯ**ＡＴ($CLFLT)	倍精度浮動小数点属性値　（３）	整数
２４	$**ＣＬＤＥＣ**ＩＭＡＬ	ＤＥＣＩＭＡＬ属性値　（４）	整数
２５	$**ＣＬＢＵＬ**Ｋ	ＢＵＬＫ属性値　（５）	整数
２６	$**ＣＬ**ＤＡＴＥ	日付属性値　（６）	整数
２７	$**ＣＬＶＡＲ**ＩＡＮＴ	バリアント属性値　（７）	整数
２８	$**ＣＬＩＮＴ**ＥＧＥＲ	整数属性値　（２）	整数
２９	$**ＣＬＤＯＵ**ＢＬＥ($CLDBL)	倍精度浮動小数点属性値　（３）	整数
３０	$ＮＵＬＬ	NULL	文字

表５－３　システム変数一覧（2/2）

項番	変　数　名	内　　容	データ型
３１	＄ＰＩ	円周率π	浮動小数
３２	＄ＰＩ２	円周率π（高精度 54桁）	10進浮動
３３	＄Ｍ＿Ｅ	ｅ	浮動小数
３４	＄Ｍ＿Ｅ２	ｅ（高精度 54桁）	10進浮動
３５	＄ＳＹＳＤＡＴＥ	現在日時	日付
３６	＄ＥＸＣＥＰＴＩＯＮ	例外番号	整数
３７	＄ＣＬＳＹＳＣＯＤＥ	システムコード値　　（０）	整数
３８	＄ＣＬＥＵＣ	ＥＵＣコード値　　（１）	整数
３９	＄ＣＬＳＪＩＳ	ＳＪＩＳコード値　　（２）	整数
４０	＄ＣＬＪＩＳ	ＪＩＳコード値　　（３）	整数
４１	＄ＣＬＥＢＣＤＩＣ	ＥＢＣＤＩＣコード値　　（４）	整数
４２	＄ＣＬＵＴＦ８	ＵＴＦ－８コード値　　（５）	整数
４３	＄ＣＬＵＮＩＣＯＤＥ	ＵＮＩＣＯＤＥコード値　（６）	整数
４４	＄ＣＬＥＢＣＤＩＫ	ＥＢＣＤＩＫコード値　　（８）	整数
４５	＄ＣＬＮＡＲＲＯＷ	半角コード値　　（２５７）	整数
４６	＄ＣＬＷＩＤＥ］	全角コード値　　（２５８）	整数
４７			
４８			

5.6. 変数の配列

（１）番号変数

パラメータ変数、検索結果読み込み変数、内部番号変数は、その種類毎に配列として定義されており、変数種類を指す文字（%、＃、$）の後に付ける整数文字列がインデックスを示している。

このインデックス部分を変数で指定することによって変数を間接的にアクセスすることができる。この指定はネストすることができる。

インデックス部分は、カッコ（()）で囲むことができる。これにより、ＳＱＬ文中の連続する文字列の一部に変数を指定することが可能となる。

変数種類を指す文字（%、＃、$）の後にカッコ（()）を付けたものは、それぞれの変数の先頭からマップしたマップド配列変数名として使うことができる。

~~/＊ 配列を指定するときは、カッコ（()）を使用してネストさせる。＊/~~

＜例１＞

```
$１　＝　２；
$２　＝　’ＡＢＣ’；
$３　＝　４；
$$３＝　$$１；
```

この結果$４には’ＡＢＣ’が入る。

＜例２＞

```
$１　＝　２；
$２　＝　$（１）；
```

$２には、２がはいる。

＜例３＞

例１の４行目は、次の様にも書ける。

```
$（$（３））　＝　$（$（１））；
```

＜例４＞

```
$１　＝　’Ａ’；
$２　＝　２；

ＳＱＬ　””　”””　”ｓｅｌｅｃｔ　＊　ｆｒｏｍ　$（１）ＢＣ
                  ｗｈｅｒｅ　ａ$（２）１　＝　’ｘｙ’””；
```

以下となる

```
ＳＱＬ　””　”””　”ｓｅｌｅｃｔ　＊　ｆｒｏｍ　ＡＢＣ
                  ｗｈｅｒｅ　ａ２１　＝　’ｘｙ’””；
```

＜例５＞

インデックス部分に配列を使用する。

```
$a[0] = 1;
$($a[0]) = 'A';
$2 = 2;

SQL ""  ""  ""select * from $($a[0])BC where a$(2)1 = 'xy';
```

（2）マップド配列変数
　パラメータ変数、検索結果読み込み変数、内部番号変数のあるインデックス以降にマップした変数である。定義方法は、「3.16.5 マップド配列変数定義」を参照。

（3）配列変数
　一般的な配列を持つ変数である。
　定義方法は、 「3.16.6 配列変数定義」を参照。

（4）データ・リストおよびデータ並び
　データ要素には、配列と同様にインデックス指定で参照/更新ができる。
　インデックスの開始値は、常に０である。

（5）インデックスの指定方法
　配列のインデックスは、かぎ括弧でくくって指定する。
　次元は、カンマ（, ）で区切って指定する。
　インデックスには、式を指定可能。
　（A）一般の配列のインデックス
　　数値で指定する。インデックスの開始値は、０である。
　　インデックス開始値は、オプション１５の指定により１となる。
　　定義した次元より大きい次元を指定できるが、指定できる値は、開始値である。
　　省略した次元は、インデックスに開始値が指定されたものと見なす。
　（B）連想配列のインデックス
　　文字で指定する。
　　省略した次元は、インデックスにnull値が指定されたものと見なす。

（6）連想配列
　インデックスに文字列を指定できる配列である。
　インデックスに数値が指定されたときは、文字列に変換される。
　次元数に制限はなく、任意の次元を使用可能である。
　指定されたインデックスは、全てがバッククォート＋カンマ(`,)で連結され、ハッシュで管理される。
　インデックス中に、バッククォート＋カンマ(`,)またはバッククォート＋バッククォート(``)以外でバッククォート(`)があるときは、その前にバッククォート(`)が付加される。
　インデックス中のバッククォート＋カンマ(`,)は、インデックスを区切るカンマ(,)と同値となる。
　未設定の配列要素を参照すると、ＮＵＬＬパラメータが設定される。(オプションで変わる)
［例］

```
define array $成績 hash;

$成績['小林明人','１学期','数学'] = 100;
```

5.7. 構造体変数

構造体型を指定して定義した変数を構造体と言う。
構造体のメンバ変数には、全てのデータ型を格納できる。
構造体のメンバ変数は、以下の形式でアクセスする。

```
構造体変数名.メンバのデータ要素[.メンバのデータ要素 ・・・]
```

メンバのデータ要素の内容が構造体変数名のときは、さらに、その構造体変数のメンバを指定できる。

6. 定数

定数には、文字列定数と数値定数とがある。

6.1. 文字列定数

１重引用符（’）で囲まれた文字列を文字列定数とする。
文字列が空の定数（’’）はＮＵＬＬ値を示す。

（１）旧仕様

文字列中に１重引用符を入れるときは、そのまま指定する。
文字列中に空白文字を入れるときは、１重引用符で囲まれた文字列を更に２重引用符で囲む。
＜例＞
① ’ａｂｃ’ｄ’は、ａｂｃ’ｄとなる。
② ”’ａｂ　ｄ’”は、ａｂ　ｄとなる

（２）新仕様

文字列中に１重引用符を入れるときは、２個続けて指定する。
文字列中に空白文字を入れるときは、そのまま指定する。
＜例＞
① ’ａｂｃ’’ｄ’は、ａｂｃ’ｄとなる。
② ’ａｂ　ｄ’は、ａｂ　ｄとなる。

（３）エスケープ処理

エスケープ文字の次の文字を定数文字とする。
ただし、文字列がエスケープ文字の１文字の場合、または、エスケープ処理の結果、末尾にエスケープ文字が残った場合は、エスケープ文字をそのまま定数文字とする。
また、特殊文字は、以下の形式で指定する。

No	指定形式	コード	名称
1	’¥n’	0x0a	ＬＦ
2	’¥t’	0x09	ＨＴ　タブ
3	’¥r’	0x0d	ＣＲ
4	’¥b’	0x08	ＢＳ　バックスペース
5	’¥f’	0x0c	ＦＦ　フォームフィード
6	’¥DDD’	Ｄは０～７	８進数
7	’¥xXX’	Ｘは０～９、Ａ～Ｆ、ａ～ｆ	１６進数

（４）空白文字

半角スペースまたはタブを空白文字と言う。

6.2. 数値定数

（１）整数値定数

２進、８進、１０進、１６進がある。

指定形式は、

［0　型指定文字］　数字列

型指定文字は、

Ｂ又はｂは	２進
Ｏ又はｏは	８進
~~Ｄ又はｄは~~	~~１０進~~
Ｘ又はｘは	１６進

１０進のときは、符号を含む数字列のみとする。

＜例＞

以下はすべて整数の１０６を表す。

２進	0b01101010
８進	0o152
~~１０進~~	~~0d106~~
１６進	0x6a

（２）２進倍精度浮動小数点定数（内部的に２進数で保持）

以下の形式で指定する。

少数点のみの指定はできない。

｛　［＋｜－］［整数］［．［整数］］｛Ｆ｜ｆ｜Ｄ｜ｄ｝
　［＋｜－］［整数］［．［整数］］｛｛Ｅ｜ｅ｜Ｄ｜ｄ｝［＋｜－］整数］　｝

（３）１０進浮動小数点定数（内部的に１０進数で保持）

以下の形式で指定する。

指数部の″＋－″は、数字列が文字列定数の時のみ有効となる。

少数点のみの指定はできない。

｛　［＋｜－］［整数］［．［整数］］
　［＋｜－］［整数］［．［整数］］｛｛＋｜－｝整数｝　｝

7. データ属性

7.1. データ属性一覧

使用できるデータの属性を表７－１に示す。

表７－１　データ属性

項番	データＩＤ	データ型	内　容
１	一般データ	文字	０ｘ０１～０ｘＦＦのコードを収容可能。ただし、スクリプト内においては非表示文字コードは設定できない。データ長は０～（４バイト整数の最大値）。データ長ゼロは、特別にＮＵＬＬ値として扱われる。
２	一般データ	整数	４バイトの整数値
３	一般データ	２進倍精度浮動小数点数	２進倍精度浮動小数点数値（内部的に２進保持） 最大値： 1.7976931348623157e+308 最小値：-1.7976931348623157e+308 正の最小値：9.99e-307
４	一般データ	１０進小数点数	１０進小数点数値（内部的に１０進保持）。 精度は５４桁。１０の－６５５３６乗～６５５３５乗。
５	一般データ	ｂｕｌｋ	０ｘ００～０ｘＦＦのコードを収容可能。データ長は１～（４バイト整数の最大値）。データ長０は文字型となる。
６	一般データ	日付	年月日時分秒
７	配列名	ｂｕｌｋ	配列定義した配列の名前
８	リスト	ｂｕｌｋ	データ・リスト
９	構造体名	ｂｕｌｋ	構造体の情報
１０	構造体定義名	ｂｕｌｋ	構造体の定義情報
１１	関数名	文字	関数の情報
１２	クラス名	文字	クラスの情報
１３	インスタンス名	文字	インスタンスの情報
１４	メソッド名	文字	メソッドの情報
１５	手続き名	文字	手続きの情報

データＩＤ、データ型およびデータ長の値は、付録を参照。

7.2. データ・リスト

（１）データ・リストには、全てのデータ型を任意の組み合わせで格納できる。
　　データ・リストも格納できる。

（２）以下の変数には、データ・リスト作成関数、または、データ・リスト式を使用して、
　　データ・リストを設定できる。
　　・内部変数
　　・外部変数
　　ただし、データ型指定の配列の要素を除く。