Coal

言語仕様書編

第９章

第　５．７　版

２０１３年１２月７日

目　　次

9. 付録

9.1. ＳＱＬコマンドの処理番号

表１１．１－１に示す。

表１１．１－１　ＳＱＬコマンドの処理番号

項番	処理番号	処　　理　　内　　容
1	0	何もしない
2	1	START TRANSACTION（更新開始）　を発行する
3	2	COMMIT WORK　（実更新）　を発行する
4	3	ROLLBACK WORK　（更新の取消）　を発行する
5	4	何もしない
6	5	START TRANSACTION（更新開始）　を強制発行する
7	6	COMMIT WORK　（実更新）　を強制発行する
8	7	ROLLBACK WORK　（更新の取消）　を強制発行する

9.2. 実行時オプション

表１１．３－１　実行時オプション（１／２）

オプション番号	オプション値の意味(デフォルト値は、0x00)
1	変数値未設定時の処理 0x01 ： 配列要素のとき、NULL値とする。 0x02 ： 配列以外の変数のとき、NULL値とする。 これを設定すると、自動で0x01が設定される。 0x04 ： 連想配列要素を未設定にする。
2	名称に使用できる文字と文字列定数の引用符に使用する文字 ~~0x01 ： スペースと非表示文字以外を使用可能とする。~~ ~~0x02 ： 数字で始まる変数名を許す。~~ 0x04 ： １重引用符(')と２重引用符(")の機能を交換する。
3	テキスト出力時の改行コードを制御する 0x01 ： LFを出力しない 0x02 ： CRを出力する 0x04 ： テキスト中の改行コードも制御する
4	式がないときの処理 0x00 ： ワーニング終了(ret=100)、エラーメッセージあり 0x01 ： エラーメッセージなしビット 0x02 ： 正常終了ビット 0x04 ： エラー終了ビット
5	要素なしのリストの扱い 0x00 ： ＮＵＬＬ値とする 0x01 ： NULLリストとする
6	インポートするスクリプトの追加位置を指定する 0x00 ： 末尾に追加する 0x01 ： 先頭に挿入する
7	エラー時の処理 0x00 ： 当該手続きをエラー終了する 0x01 ： エラー行出力を行わない 0x02 ： エラーを無視して処理を継続する 0x04 ： 当該セッションをエラー終了する
8	ユーザ定義関数のサーチ順位 0x00 ： 定義済み関数より先にサーチする。 0x01 ： 定義済み関数の後にサーチする。 0x02 ： 手続き／関数の入れ子をサポートする。 クラスを使用するときは、自動で設定される。
9	入力構文の文字コード 0x00:引用符外側の英数記号の倍角文字は半角文字に変換する。 0x01:全ての英数記号の倍角文字は半角文字に変換する。 0x02:全て変換しない。
１０	デフォルトの変数種別 0x00 ： ローカル変数とする。 0x01 ： スクリプト変数とする。

表１１．３－１　実行時オプション（２／２）

オプション番号	オプション値の意味(デフォルト値は、0x00)
１１	１０進浮動(固定)小数点数の０表示 0x00 ： 少数点以下がないときに少数点以下を表示しない。 0x01 ： １未満のときに最初の０を表示しない。 0x02 ： 少数点以下がないときに少数点を表示する。 0x04 ： 少数点以下がないときに少数第１位の０を表示する。 0x20 ： ３桁毎にカンマを付ける。
１２	１０進固定小数点数の位取り未満の処理 0x00 ： 四捨五入 0x01 ： 切捨て 0x02 ： 切上げ
１３	ＰＲＩＮＴ文での配列、構造体出力形式 0x00 ： 詳細情報を出力しない 0x01 ： 詳細情報を出力する
１４	メッセージの言語種別 ０ ： 日本語 １ ： 英語 ２ ： その他
１５	配列の開始インデックスとデータの配置 0x00 ： 開始インデックスは、０ データの配置は、C言語形式 0x01 ： 開始インデックスは、１ 0x02 ： データの配置は、FORTRAN形式
１６	１０進浮動(固定)小数点数のオーバーフロー、アンダーフロー時の処理 0x00 ： オーバーフロー時、エラー終了。メッセージを出力する。 アンダーフロー時、続行。 0x01 ： オーバーフロー時、続行。 0x02 ： オーバーフロー時、メッセージを出力する。 0x04 ： アンダーフロー時、エラー終了。 0x08 ： アンダーフロー時、メッセージを出力する。
１７	１０進浮動小数点数になる場合の数字列の変換 0x00 ： １０進浮動小数点数に変換 0x01 ： ２進浮動小数点数に変換 0x10 ： 数字列の後ろに数字以外があってもエラーとしない。 0x20 ： カンマを無視する。
１８	サーチ開始位置指定時の返却値 0x00 ： INSTR,INLIKE,REPLIKEで、指定文字列の先頭を基準とする。 0x01 ： INSTRにおいて、サーチ開始位置を基準とする。 0x02 ： INLIKEにおいて、サーチ開始位置を基準とする。 0x04 ： REPLIKEにおいて、サーチ開始位置以降を返却する。

9.3. データ属性の値

データＩＤおよびデータ型の値を以下に示す。

表11.4-1　データＩＤ

項番	データＩＤ	値	データ長(バイト)
1	一般データ	' '	テータ型による
2	配列名	一般配列:'R'、MAPPED配列:'A'	３２
3	リスト	'L'	３２
4	構造体名	'T'	３２
5	構造体定義名	'P'	３２
6	関数名	'F'	３２
7	クラス名	'C'	３２
8	インスタンス名	'I'	３２
9	メソッド名	'M'	３２
１０	手続き名	'O'	３２

表11.4-2　一般データのデータ型とデータ長

項番	データ型	値	データ長(バイト)
1	文字	1	実データ長
2	整数	2	4
3	２進倍精度浮動小数点数	3	8
4	１０進小数点数	4	６４
5	ｂｕｌｋ	5	実データ長
6	日付	6	４０
7	バリアント(注)	7	0

(注)データ未設定時のみ存在し、データ設定後は、そのデータ型になる。

日付型は、14バイトから成る。各バイトのデータ内容を以下に示す。

表11.4-3　日付型のデータ内容

Index	意味	値
0	年の上２桁	0 - 99
1	年の下２桁	0 - 99
2	月	0 - 11
3	日	1 - 31
4	時	0 - 23
5	分	0 - 59
6	秒	0 - 59
7から９	マイクロ秒(intの下３バイト)	0 - 999999
１０	ＡＭ／ＰＭ	0 / 1
１１	曜日	0 - 6 (0:日曜)
１２，１３	年通算日数(short)	0 - 365 (0:１月１日)

9.4. 演算子の順位

表１１．５－１　演算子の順位

順位	演算子
１	．（　）　［　］　｛　｝　後置演算＋＋　－－
２	＊＊
３	単項演算子　！　～　－　＋　＋＋　－－　＊
４	キャスト
５	＊　／　％
６	＋　－
７	＜＜　＞＞
８	文字列演算子　CONCAT　SUBSTR　REP　CONDAS　TO　IS
９	＜　＞　＜＝　＜＝
１０	＝＝　！＝
１１	＆
１２	^
１３	｜
１４	＆＆
１５	｜｜
１６	？：
１７	＝ += -= *= /= %= <<= >>= &= ^= |= &+=
１８	,

（注）ピリオドはドット式で使われるとき。

9.5. 変数、手続きまたは関数のサーチ順

（1）変数のサーチ順

スコープ指定がないときは、以下の順にサーチする。

ローカル変数 → スクリプト変数 → 外部変数

手続きまたは関数の入れ子をサポートするときは、ローカル変数は、以下でサーチする。

実行済みの手続きまたは関数を実行の逆順に構造上の上位に向かってサーチする。、

（2）手続きまたは関数のサーチ順

（A）手続きまたは関数の入れ子をサポートしないとき

スクリプトの先頭から、第一レベルにある手続きまたは関数をサーチする。

（B）手続きまたは関数の入れ子をサポートするとき

手続きまたは関数を呼び出している手続きまたは関数の中 →上位の手続きまたは関数の中 → その上位の手続きまたは関数の中 ・・・

9.6. デバッグ形式での出力

（1）一般形式

（A）内容

ログヘッダ	出力元が設定したデータ	情報構造体の内容

データＩＤまたはデータ型による個別情報出力

（B）ログヘッダ

標準出力形式

```
[coal]
```

ファイル出力形式

```
yyyy/mm/dd hh:mm:ss coal/<ソースファイル名>(<ソース行数>):
```

(注)<>は、これで囲まれた情報が出力されることを示す。

（C）出力元が設定したデータ

出力対象となった最上位の変数のとき

```
式=
```

（2）個別情報

（A）式の並びまたはデータ並び式

並びの最後の式の値を一般形式で出力し、次に、並びの情報を" PARMINFO2:"を「出力元が設定したデータ」として一般形式で出力する。

（B）配列

情報構造体の内容

```
pInfo=%08x id=%c attr=%d scale=0x%02x dlen=%d len(pScCT)=%08x hlen(gid)=%d
 aux=0x%02x 0x%02x auxlen=%04x %08x data=%08x
```

配列の情報

```
(VarIndex=%08x Attr=%d %d %d %d size=%d index=%d[%d,%d,%d] xhp=%08x
  (_xhash Id=%c%c keylen=%d max=%d pre=%d ha=%08x next=%08x datlen=%d dreg=%08x))
 varnam=[配列名]
```

(注)"(_xhash ・・・ dreg=%08x)"は、連想配列のとき出力する。

（C）その他

「情報構造体の内容」は共通で、その後に個別情報が出力される。

情報構造体の内容

```
pInfo=%08x id=0x%02x[%c] attr=%d scale=0x%02x code=%d dlen=%d len(gid)=%d hlen=%d pos=%d(%08x) aux=0x%02x 0x%02x auxlen=%04x %08x data=%08x
```

（a）構造体または構造体定義

ヘッダ

```
size=%d data=0x%08x ntype=%d vname=0x%08x pType=0x%08x varnam=[<オブジェクト名>]
```

メンバー情報（メンバー数分出力する）

```
"vnlen=<メンバー名長> <メンバー名>: "を「出力元が設定したデータ」として、一般形式で出力
```

（b）リスト

```
 rb_bfsize=%d rb_max=%d rb_num=%d rb_used=%d rb_pos=%d rb_raddr=0x%08x rb_waddr=0x%08x rb_wpriv=0x%08x rb_cur=0x%08x
```

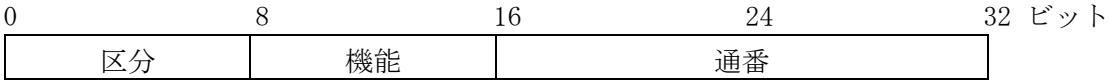
（c）その他

データ型	単純	範囲
データへのポインタがNULL	NULL	なし
CHAR	[%s]	[%s]..[%s]
BIN	%d	%d..%d
FLOAT	%e	%e..%e
DEC	10進表示	10進表示..10進表示
BULK または PARMINFO2の２要素目	先頭32バイトの16進表示	なし
その他	**INVALID**	なし

9.7. 例外の種類

9.7.1. 例外番号の構成

（1）値の構成



（2）名称の構成

区分名称__[機能名称__[個別名称__]]ＥＸＣＥＰＴＩＯＮ

9.7.2. 区分

No.	区分		内容
	名　　称	値	
1	ＩＳ	1	検査
2	ＴＯ	2	変換
3	ＣＭＰＲ	3	比較
4	ＳＴＲＩＮＧ	4	文字列操作
5	ＭＡＴＨ	5	数値演算
6	ＦＩＬＥ	6	ファイル
7	ＬＯＧ	7	ログ
8	ＳＱＬ	8	ＳＱＬ
9	ＣＯＭＭ	9	通信
10	ＵＳＥＲ	0x7c	ユーザ
11	ＳＹＳＴＥＭ	0x7d	システム
12	ＥＴＣ	0x7e	その他
13	ＡＬＬ	0x80	全ての例外を表す。 例外の値は、これとのＯＲを取った値となる。

9.7.3. 機能

（5）ＭＡＴＨ

No.	機能		個別		内容
	名　　称	値	名　　称	値	
	ＣＯＭＰ	1	［ ＥＲＲＯＲ ］	0	演算エラー全般
			ＤＥＶＩＤＥ	1	０割
	ＥＴＣ	254	［ ＥＲＲＯＲ ］	0	その他エラー全般

（6）ＦＩＬＥ

No.	機能		個別		内容
	名　　称	値	名　　称	値	
	ＯＰＥＮ	1	［ ＥＲＲＯＲ ］	0	オープン・エラー全般
			ＮＯＴＦＯＵＮＤ	2	file not found
	ＣＬＯＳＥ	2	［ ＥＲＲＯＲ ］	0	クローズ・エラー全般
	ＲＥＡＤ	3	［ ＥＲＲＯＲ ］	0	入力エラー全般
			ＥＮＤ	1	end of file
	ＷＲＩＴＥ	4	［ ＥＲＲＯＲ ］	0	出力エラー全般
	ＥＴＣ	254	［ ＥＲＲＯＲ ］	0	その他エラー全般

9.8. スクリプトの例

9.8.1. Hellow World

```
proc main;
        print 'Hellow World.';
        return 0;
end proc;
```

9.8.2. クロージャ

test_class3.cl

```
//
Class Counter;
        int i=0;
        f = _count;
        func Counter;
                return f;       // クラス変数を使って関数を返す。
        end func;
        func Counter(n);
                i = n;
                return f;
        end func;
        func Counter(x,y);
                i = x + y;
                return My._count;       // Myは、本クラスを示す。
        end func;
        func _count();
                return ++i;
        end func;
end class;

proc main;
        c = new(Counter);
        print c();
        print c();
        d = new(Counter,10);
        print d();
        print d();
        print c();
        x = new(Counter,100,1);
        print x();
        print x();
        return ERROR;
end proc;
```

実行結果

```
145:/home/coal/test>coal test_class3
c()=1
c()=2
d()=11
d()=12
c()=3
x()=102
x()=103
```

9.8.3. 構造体を使った擬似クラス

test_class3.cl

```
//
define type struct Greeter
         variant salute
        ,variant name           // variantは、データ型指定なしと同じ。
;
define var g as Greeter;        // define var g,h as Greeter;
define var h as Greeter;        // gとhは、まとめて定義可能。

func _new (g,name);             // gには、構造体への参照が渡される。
        g.name = name;
//      g.salute = _salute;             // 直接関数名を指定しても同じ。
        g.salute = (FUNC)'{func _salute;// {}で囲った文字列を関数化して指定可能。
        print My.name&'' World!!'';     // 文字定数中の引用符は２つ続けて指定する。
        return 0;
end func;}';
        return 0;
end func;
/*
func _salute;
        print My.name&' World!!';       // Myは、構造体で修飾したときのみその構造体を示す。
        return 0;
end func;
*/
proc main;
        _new(g,'Hellow');
        g.salute();
        _new(h,'Good night');
        h.salute();
        return ERROR;
end proc;
```

実行結果

```
146:/home/coal/test>coal test_struct_class
My.name&' World!!'="Hellow World!!"
My.name&' World!!'="Good night World!!"
```

(注)printは、対象となった式=結果の形式で出力する。(ただし、定数は値のみを出力する。)