

ES/1 NEO

MF シリーズ *CS* シリーズ

クエリー文法解説書

第33版 2020年8月

©著作権所有者 株式会社 アイ・アイ・エム 2020年

© COPYRIGHT IIM CORPORATION, 2020

**ALL RIGHT RESERVED. NO PART OF THIS PUBLICATION MAY
REPRODUCED OR TRANSMITTED IN ANY FORM BY ANY MEANS,
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY RECORDING,
OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM WITHOUT
PERMISSION IN WRITING FROM THE PUBLISHER.**

“RESTRICTED MATERIAL OF IIM “LICENSED MATERIALS – PROPERTY OF IIM

目次

第 1 章 クエリーの基本動作.....	1
1.1. サイト／システムと表の関係	1
1.2. 表のデータ	2
1.3. 表からのデータ抽出	3
1.4. 複数の表からのデータ抽出	4
1.4.1. 日時による結合	4
1.4.2. 日時+フィールドによる結合	8
1.4.3. レコード欠落時の結合	11
1.4.4. 複数のシステムからのデータ抽出	13
1.4.5. グループピング	14
1.4.6. PIVOT 操作	16
第 2 章 クエリー定義の文法.....	18
2.1. 式の指定	18
2.1.1. データ列について	19
2.1.2. 変数について	19
2.1.3. 定数について	20
2.1.4. 数値演算子について	20
2.1.5. 文字列演算子について	20
2.1.6. 関係演算子について	21
2.1.7. 論理演算子について	21
2.1.8. グループ演算関数について	21
2.1.9. 数値操作関数について	33
2.1.10. 日時操作関数について	34
2.1.11. 文字列操作関数について	36
2.1.12. 書式化関数について	44
2.1.13. 条件判定関数について	48
2.1.14. サブクエリー指定について	49
2.1.15. 特殊変数(ES/1 NEO CS シリーズのみ有効)について	49
2.2. select 句の指定	51
2.2.1. AS 指定	51
2.2.2. PIVOTROW、PIVOTCOL 指定	51
2.2.3. PIVOT オプション指定	52
2.3. from 句の指定	53
2.4. join 句の指定	53
2.5. filter 句の指定 (ES/1 NEO CS シリーズのみ有効)	54
2.6. where 句の指定	56
2.7. when 句の指定	56
2.8. group by 句の指定	57
2.9. having 句の指定	57
2.10. order by 句の指定	58

2.11. sub query 句の指定	60
2.12. option 句の指定	62
2.13. runpy 句の指定	65
第 3 章 マクロ	71
3.1. マクロの使用方法	71
3.2. マクロ使用についての注意点	72

第1章 クエリの基本動作

1.1. サイト/システムと表の関係

各システム(ホスト/サーバ)は個別に複数の表を持っています。各表は表名によって識別されます(以降の説明で表を“テーブル”と表現することもあります)。

(ex. MF)

サイト	システム	表	表名
SITE_A	SYS0	プロセッサ	PROC
		主記憶	CS
		ボリュームグループ	VOL_GRP
	
	SYS1	プロセッサ	PROC
		主記憶	CS
		ボリュームグループ	VOL_GRP
	
...
SITE_B	SYS_0000	プロセッサ	PROC
	

...

(ex. CS)

サイト	システム	表	表名
SITE_A	SYS0	プロセッサ	ATCPU
		メモリ	ATPAGE
		デバイス	ATDEV
	
	SYS1	プロセッサ	ATCPU
		メモリ	ATPAGE
		デバイス	ATDEV
	
...
SITE_B	SYS_0000	プロセッサ	ATCPU
	

...

1.2. 表のデータ

各表にはパフォーマンスデータの実体である行が時系列に存在します。各表には列が存在します。列は列名によって識別されます。列中の各値は文字列か数値のいずれかです。

(以降の説明で表の行を“レコード”、レコード中の列に相当するものを“フィールド”、また時系列のことを“インターバル”と表現することがあります)

(ex.MF)

プロセッサ表(表名：PROC)とボリュームグループ表(表名：VOL_GRP)のイメージ

【プロセッサ(PROC)】

日付	時刻	平均CPU使用率 (CPU_AVG)	平均TBC使用率 (TCB_AVG)		最大CPU使用率 (CPU_MAX)	最大TBC使用率 (TCB_MAX)
20020101	0000	10	5	...	50	40
20020101	0100	20	15		60	30
20020102	0000	8	3		30	20
20020102	0100	10	5		50	30
...

【ボリュームグループ(VOL_GRP)】

日付	時刻	ボリュームグループ名 (GRP)	平均負荷率 (LOAD_AV)	平均アクセス回数 (ACC_SEC_AVG)	平均サービス時間 (SERVICE_AVG)	
20020101	0000	GRP_A	50	8	2	...
20020101	0000	GRP_B	40	7	1	
20020101	0100	GRP_A	80	19	1.5	
20020101	0100	GRP_B	30	10	1.2	
...	

(ex.CS)

メモリ表(表名：ATPAGE)とデバイス表(表名：ATDEV)のイメージ

【メモリ(ATPAGE)】

日付	時刻	ページスキャン (PAGESCAN)	ページイン (PAGEIN_VM)		スワップイン (PSWPIN)	スワップアウト (PSWPOUT)
20020101	0000	0	5	...	1	1
20020101	0100	0	15		2	3
20020102	0000	0	3		3	2
20020102	0100	0	5		5	3
...

【デバイス(ATDEV)】

日付	時刻	物理ディスク名 (DISKID)		ビジ率 (USE)	待ち時間 (WAITTM)	サービス時間 (SERVTM)
20020101	0000	0	...	50	8	2
20020101	0000	1		40	7	1
20020101	0100	0		80	19	1.5
20020101	0100	1		30	10	1.2
...

1 インターバルに 1 つのレコードしか存在しない表と複数のレコードが存在する表とがあります。

上記の例では、(ex.MF)のプロセッサ表と(ex.CS)のメモリ表は 1 インターバル 1 レコードの表であり、(ex.MF)のボリュームグループ表と(ex.CS)のデバイス表は 1 インターバル複数レコードの表です。

1.3. 表からのデータ抽出

表からデータを取り出したい場合(抽出)は、表名と列名をクエリー文中で指定します。
(日付・時刻はそれぞれ“DATE”・“TIME”で指定します)

(ex.MF)

プロセッサ表(表名 : PROC)からのデータ抽出

【プロセッサ(PROC)】

日付	時刻	平均CPU使用率 (CPU_AVG)	平均TBC使用率 (TCB_AVG)		最大CPU使用率 (CPU_MAX)	最大TBC使用率 (TCB_MAX)
20020101	0000	10	5	...	50	40
20020101	0100	20	15		60	30
20020102	0000	8	3		30	20
20020102	0100	10	5		50	30
...

select DATE,TIME,PROC.CPU_AVG,PROC.CPU_MAX from 'Site'. 'System'...

【結果】

DATE	TIME	PROC.CPU_AVG	PROC.CPU_MAX
20020101	0000	10	50
20020101	0100	20	60
20020102	0000	8	30
20020102	0100	10	50
...

(ex.CS)

デバイス表(表名 : ATDEV)からのデータ抽出

【デバイス(ATDEV)】

日付	時刻	物理ディスク名 (DISKID)		ビジー率 (USE)	待ち時間 (WAITTM)	サービス時間 (SERVTM)
20020101	0000	0	...	50	8	2
20020101	0000	1		40	7	1
20020101	0100	0		80	19	1.5
20020101	0100	1		30	10	1.2
...

select DATE,TIME,ATDEV.DISKID,ATDEV.USE from 'Site'. 'System'...

【結果】

DATE	TIME	ATDEV.DISKID	ATDEV.USE
20020101	0000	0	50
20020101	0000	1	40
20020101	0100	0	80
20020101	0100	1	30
...

どちらの例の結果でも日時順にレコードが並んでいますが、クエリーでソート順を指定しなかった場合のレコード抽出順序は不定となるのが原則です(以降の例も同様です)。

1.4. 複数の表からのデータ抽出

1.4.1. 日時による結合

複数の表からデータを抽出する場合、各表のレコードを結合して抽出します。結合は最低でも同一システム内のインターバル(日時)が一致するレコード間で行われます(システム・インターバルが一致しないレコード間で結合が行われることは絶対にありません)。

(ex.MF)

主記憶表(表名: CS)とページイン回数表(表名: PAGEIN)からのデータ抽出

【主記憶(CS)】

	日付	時刻	平均主記憶使用率 (CS_AVG)	最大主記憶使用率 (CS_MAX)	最小主記憶使用率 (CS_MIN)
(a)	20020101	0000	10	30	2
(b)	20020101	0100	20	50	4
(c)	20020102	0000	8	20	2

【ページイン回数(PAGEIN)】

	日付	時刻	平均ページイン回数 (PIN_SEC_AVG)	最大ページイン回数 (PIN_SEC_MAX)	最小ページイン回数 (PIN_SEC_MIN)
(1)	20020101	0000	8	15	3
(2)	20020101	0100	2	4	1
(3)	20020102	0000	5	9	1


```
select DATE,TIME,CS.CS_AVG,CS.CS_MAX,PAGEIN.PIN_SEC_AVG,
        PAGEIN.PIN_SEC_MAX from 'Site!'.'System' ...
```

【結果】

	DATE	TIME	CS.CS_AVG	CS.CS_MAX	PAGEIN.PIN_SEC_AVG	PAGEIN.PIN_SEC_MAX
(a)+(1)	20020101	0000	10	30	8	15
(b)+(2)	20020101	0100	20	50	2	4
(c)+(3)	20020102	0000	8	20	5	9

(ex.CS)

プロセッサ表(表名 : ATCPU)とメモリ表(表名 : ATPAGE)からのデータ抽出

【プロセッサ(ATCPU)】

	日付	時刻	ユーザモード使用率 (USRUSE)	カーネルモード使用率 (SYSUSE)	I/O Wait率 (IOWAIT)	
(a)	20020101	0000	50	30	10	...
(b)	20020101	0100	40	50	5	
(c)	20020102	0000	30	20	12	
	

【メモリ(ATPAGE)】

	日付	時刻	ページスキャン回数 (PAGESCAN)	追加フリーページ数 (PAGEFREE)	
(1)	20020101	0000	0.5	12	...
(2)	20020101	0100	0.8	8	
(3)	20020102	0000	1.1	4	
	

```
select DATE,TIME,ATCPU.USRUSE,ATCPU.SYSUSE,ATPAGE.PAGESCAN
from 'Site!'System' ...
```

【結果】

	DATE	TIME	ATCPU.USRUSE	ATCPU.SYSUSE	ATPAGE.PAGESCAN
(a)+(1)	20020101	0000	50	30	0.5
(b)+(2)	20020101	0100	40	50	0.8
(c)+(3)	20020102	0000	30	20	1.1

1 インターバルに複数のレコードが存在する表との結合ではすべての組み合わせが抽出されます。

(ex.MF)

プロセッサ表(表名 : PROC)と PR/SM 表(表名 : PR_SM)からのデータ抽出

【プロセッサ(PROC)】

	日付	時刻	平均CPU使用率 (CPU_AVG)	平均TBC使用率 (TCB_AVG)	
(a)	20020101	0000	50	40	...
(b)	20020101	0100	40	20	
	

【PR/SM(PR_SM)】

	日付	時刻	論理区画名 (NAME)	平均論理区画使用率 (PRSM_USE_AVG)	
(1)	20020101	0000	LPR0	12	...
(2)	20020101	0000	LPR1	4	
(3)	20020101	0000	LPR2	8	
(4)	20020101	0100	LPR0	7	
(5)	20020101	0100	LPR1	2	
	

```
select DATE,TIME,PROC.CPU_AVG,PR_SM.NAME,PR_SM.PRSM_USE_AVG from
'Site':'System'...
```

【結果】

	DATE	TIME	PROC.CPU_AVG	PR_SM.NAME	PR_SM.PRSM_USE_AVG
(a)+(1)	20020101	0000	50	LPR0	12
(a)+(2)	20020101	0000	50	LPR1	4
(a)+(3)	20020101	0000	50	LPR2	8
(b)+(4)	20020101	0100	40	LPR0	7
(b)+(5)	20020101	0100	40	LPR1	2

(ex.CS)

プロセッサ表(表名 : ATCPU)とユーザアカウント表(表名 : ATACCU)からのデータ抽出

【プロセッサ(ATCPU)】

	日付	時刻	ユーザモード使用率 (USRUSE)	カーネルモード使用率 (SYSUSE)	I/O Wait率 (IOWAIT)	
(a)	2002010	0000	50	30	10	...
(b)	2002010	0100	40	50	5	
	

【ユーザアカウント(ATACCU)】

	日付	時刻	ユーザ名 (USRNAME)	プロセッサ使用率 (CPUUSE)	
(1)	20020101	0000	root	10	...
(2)	20020101	0000	daemon	9	
(3)	20020101	0000	nobody	8	
(4)	20020101	0100	root	4	
(5)	20020101	0100	daemon	8	
	

```
select DATE,TIME,ATCPU.USRUSE,ATCPU.SYSUSE,ATACCU.USRNAME,
        ATACCU.CPUUSE from 'Site!'.'System'...
```

【結果】

	DATE	TIME	ATCPU.USRUSE	ATCPU.SYSUSE	ATACCU.USRNAME	ATACCU.CPUUSE
(a)+(1)	20020101	0000	50	30	root	10
(a)+(2)	20020101	0000	50	30	daemon	9
(a)+(3)	20020101	0000	50	30	nobody	8
(b)+(4)	20020101	0100	40	50	root	4
(b)+(5)	20020101	0100	40	50	daemon	8

1.4.2. 日時+フィールドによる結合

日時に加えて特定のフィールドの値が一致するレコード間のみで結合を行うことが可能です。

join 句の記述方法は文法解説を参照してください。

(ex.MF)

VSM VTV インターバル情報表(表名 : VTV)と VSM RTD インターバル情報表(表名 : RTD)からのデータ抽出

【VSM VTV インターバル情報(VTV)】

	日付	時刻	VTSS識別子 (VTSSID)	VTVマウント回数 (MNT)	VTVマウント時間 (秒) (MTM)	
(a)	20020101	0000	VSM1	31	1.868	...
(b)	20020101	0000	VSM2	57	1.379	
(c)	20020101	0000	VSM3	59	0.488	
(d)	20020101	0000	VSM4	18	0.145	
	

【VSM RTD インターバル情報(RTD)】

	日付	時刻	VTSS識別子 (VTSSID)	RTDマウント回数 (RMNT)	RTD平均マウント時間 (秒) (RMTM)	
(1)	20020101	0000	VSM1	3	15.03	...
(2)	20020101	0000	VSM2	5	14.261	
(3)	20020101	0000	VSM3	1	11.534	
(4)	20020101	0000	VSM4	1	11.534	
	

(例 1)

```
select DATE,TIME,VTV.VTSSID,VTV.MNT,VTV.MTM,RTD.VTSSID,RTD.RMNT,RTD.RMTM
from 'Site'.System'...
```

日時一致のみの結合では(a)~(d)と(1)~(4)のすべての日時が一致しているので、結果は以下のようになります。

	DATE	TIME	VTV.VTSSID	VTV.MNT	VTV.MTM	RTD.VTSSID	RTD.RMNT	RTD.RMTM
(a)+(1)	20020101	0000	VSM1	31	1.868	VSM1	3	15.03
(a)+(2)	20020101	0000	VSM1	31	1.868	VSM2	5	14.261
(a)+(3)	20020101	0000	VSM1	31	1.868	VSM3	1	11.534
(a)+(4)	20020101	0000	VSM1	31	1.868	VSM4	1	11.534
(b)+(1)	20020101	0000	VSM2	57	1.379	VSM1	3	15.03
(b)+(2)	20020101	0000	VSM2	57	1.379	VSM2	5	14.261
(b)+(3)	20020101	0000	VSM2	57	1.379	VSM3	1	11.534
(b)+(4)	20020101	0000	VSM2	57	1.379	VSM4	1	11.534
(c)+(1)	20020101	0000	VSM3	59	0.488	VSM1	3	15.03
(c)+(2)	20020101	0000	VSM3	59	0.488	VSM2	5	14.261
(c)+(3)	20020101	0000	VSM3	59	0.488	VSM3	1	11.534
(c)+(4)	20020101	0000	VSM3	59	0.488	VSM4	1	11.534
(d)+(1)	20020101	0000	VSM4	18	0.145	VSM1	3	15.03
(d)+(2)	20020101	0000	VSM4	18	0.145	VSM2	5	14.261
(d)+(3)	20020101	0000	VSM4	18	0.145	VSM3	1	11.534
(d)+(4)	20020101	0000	VSM4	18	0.145	VSM4	1	11.534

(例 2)

```
select DATE,TIME,VTV.VTSSID,VTV.MNT,VTV.MTM,RTD.VTSSID,RTD.RMNT,RTD.RMTM
       from 'Site'.System' JOIN VTV ON RTD (VTSSID=VTSSID) ...
```

日時に加え VSM VTV インターバル情報と VSM RTD インターバル情報のそれぞれの VTSS 識別子を結合の条件に含めると、結果は以下のようになります(前頁結果の網掛部分は抽出されません)。

	DATE	TIME	VTV.VTSSID	VTV.MNT	VTV.MTM	RTD.VTSSID	RTD.RMNT	RTD.RMTM
(a)+(1)	20020101	0000	VSM1	31	1.868	VSM1	3	15.03
(b)+(2)	20020101	0000	VSM2	57	1.379	VSM2	5	14.261
(c)+(3)	20020101	0000	VSM3	59	0.488	VSM3	1	11.534
(d)+(4)	20020101	0000	VSM4	18	0.145	VSM4	1	11.534

(ex.CS)

Oracle セッション情報表(表名 : ORSESS)と Oracle セッションサマリ表(表名 : ORSESSUM)からのデータ抽出

【Oracle セッション情報(ORSESS)】

	日付	時刻	ドメイン名 (DOMAIN)	DB名 (DBNAME)	セッションID (SESSID)	Oracleユーザ名 (ORACLEUSR)	
(a)	20020101	0000	world	ora1	0	SYSTEM	...
(b)	20020101	0000	world	ora1	1	MASTER	
(c)	20020101	0000	world	ora2	0	DBA	
(d)	20020101	0000	domx	dbx	9	SYS	
	

【Oracle セッションサマリ(ORSESSUM)】

	日付	時刻	ドメイン名 (DOMAIN)	DB名 (DBNAME)	セッションID (SESSID)	プロセッサ使用時間 (CPUSEC)	
(1)	20020101	0000	world	ora1	0	300	...
(2)	20020101	0000	world	ora1	1	150	
(3)	20020101	0000	world	ora2	0	600	
(4)	20020101	0000	domx	dbx	9	200	
	

(例 1)

```
select DATE,TIME,ORSESS.DOMAIN,ORSESS.DBNAME,
       ORSESS.SESSID,ORSESS.ORACLEUSR,ORSESSUM.DOMAIN,ORSESSUM.DBNAME,
       ORSESSUM.SESSID,ORSESSUM.CPUSEC from 'Site'.'System'...
```

日時一致のみの結合では(a)~(d)と(1)~(4)のすべての日時が一致しているので、結果は以下のようになります。

	DATE	TIME	ORSESS. DOMAIN	ORSESS. DBNAME	ORSESS. SESSID	ORSESS. ORACLEUSR	ORSESSUM. DOMAIN	ORSESSUM. DBNAME	ORSESSUM. SESSID	ORSESSUM. CPUSEC
(a)+(1)	2002010	0000	world	ora1	0	SYSTEM	world	ora1	0	300
(a)+(2)	2002010	0000	world	ora1	0	SYSTEM	world	ora1	1	150
(a)+(3)	2002010	0000	world	ora1	0	SYSTEM	world	ora2	0	600
(a)+(4)	2002010	0000	world	ora1	0	SYSTEM	domx	dbx	9	200
(b)+(1)	2002010	0000	world	ora1	1	MASTER	world	ora1	0	300
(b)+(2)	2002010	0000	world	ora1	1	MASTER	world	ora1	1	150
(b)+(3)	2002010	0000	world	ora1	1	MASTER	world	ora2	0	600
(b)+(4)	2002010	0000	world	ora1	1	MASTER	domx	dbx	9	200
(c)+(1)	2002010	0000	world	ora2	0	DBA	world	ora1	0	300
(c)+(2)	2002010	0000	world	ora2	0	DBA	world	ora1	1	150
(c)+(3)	2002010	0000	world	ora2	0	DBA	world	ora2	0	600
(c)+(4)	2002010	0000	world	ora2	0	DBA	domx	dbx	9	200
(d)+(1)	2002010	0000	domx	dbx	9	SYS	world	ora1	0	300
(d)+(2)	2002010	0000	domx	dbx	9	SYS	world	ora1	1	150
(d)+(3)	2002010	0000	domx	dbx	9	SYS	world	ora2	0	600
(d)+(4)	2002010	0000	domx	dbx	9	SYS	domx	dbx	9	200
...

(例 2)

```
select DATE,TIME,ORSESS.DOMAIN,ORSESS.DBNAME,
       ORSESS.SESSID,ORSESS.ORACLEUSR,ORSESSUM.DOMAIN,ORSESSUM.DBNAME,
       ORSESSUM.SESSID,ORSESSUM.CPUSECfrom 'Site'.'System'JOIN ORSESS ON ORSESSUM
(DOMAIN=DOMAIN,DBNAME=DBNAME,SESSID=SESSID) ...
```

日時に加え Oracle セッション情報と Oracle セッションサマリのそれぞれのドメイン名・DB 名・セッション ID を結合の条件に含めると、結果は以下のようになります(前頁結果の網掛部分は抽出されません)。

	DATE	TIME	ORSESS. DOMAIN	ORSESS. DBNAME	ORSESS. SESSID	ORSESS. ORACLEUSR	ORSESSUM. DOMAIN	ORSESSUM. DBNAME	ORSESSUM. SESSID	ORSESSUM. CPUSEC
(a)+(1)	2002010	0000	world	ora1	0	SYSTEM	world	ora1	0	300
(b)+(2)	2002010	0000	world	ora1	1	MASTER	world	ora1	1	150
(c)+(3)	2002010	0000	world	ora2	0	DBA	world	ora2	0	600
(d)+(4)	2002010	0000	domx	dbx	9	SYS	domx	dbx	9	200
...

1.4.3. レコード欠落時の結合

前述の規則による結合の対象となるレコードが一部の表に存在しない場合、2通りの結合方法があります。クエリー文中で特に指定しない限り結合対象が存在しない場合そのレコードは無視されます(抽出結果には含まれません)。

クエリー文中で OUTJOIN キーワード(記述方法は文法解説を参照してください)を使用すると結合対象が存在しなかったレコードの情報も抽出することができます(レコードが存在しなかった表の各列の値は、数値を持つ列については欠損値(-1)、文字列を持つ列については空文字列('')として扱われます)。

(ex.MF)

主記憶表(表名:CS)とページイン回数表(表名:PAGEIN)からのデータ抽出

【主記憶(CS)】

	日付	時刻	平均主記憶使用率 (CS_AVG)	最大主記憶使用率 (CS_MAX)	
(a)	20020101	0000	10	30	...
(b)	20020102	0000	8	20	
	

【ページイン回数(PAGEIN)】

	日付	時刻	平均ページイン回数 (PIN_SEC_AVG)	最大ページイン回数 (PIN_SEC_MAX)	
(1)	20020101	0100	2	4	...
(2)	20020102	0000	5	9	
	

(例 1)

```
select DATE,TIME,CS.CS_AVG,CS.CS_MAX,PAGEIN.PIN_SEC_AVG,
        PAGEIN.PIN_SEC_MAXfrom 'Site'.'System'...
```

日時との結合において(a)と(1)はそれぞれ結合対象が他方の表に存在しないため、結果は以下のようになります。

	DATE	TIME	CS.CS_AVG	CS.CS_MAX	PAGEIN.PIN_SEC_AVG	PAGEIN.PIN_SEC_MAX
(b)+(2)	20020102	0000	8	20	5	9

(例 2)

```
select DATE,TIME,CS.CS_AVG,CS.CS_MAX,PAGEIN.PIN_SEC_AVG,PAGEIN.PIN_SEC_MAX
        from 'Site'.'System' OUTJOIN( )...
```

(a)と(1)からの情報を含むため、抽出結果は以下のようになります。

	DATE	TIME	CS.CS_AVG	CS.CS_MAX	PAGEIN.PIN_SEC_AVG	PAGEIN.PIN_SEC_MAX
(a)のみ	20020101	0000	10	30	-1	-1
(1)のみ	20020101	0100	-1	-1	2	4
(b)+(2)	20020102	0000	8	20	5	9

(ex.CS)

プロセッサ表(表名 : ATCPU)とメモリ表(表名 : ATPAGE)からのデータ抽出

【プロセッサ(ATCPU)】

	日付	時刻	ユーザモード使用率 (USRUSE)	カーネルモード使用率 (SYSUSE)	
(a)	20020101	0000	50	30	...
(b)	20020102	0000	30	20	
	

【メモリ(ATPAGE)】

	日付	時刻	ページスキャン回数 (PAGESCAN)	
(1)	20020101	0100	0.8	...
(2)	20020102	0000	1.1	
	

(例 1)

```
select DATE,TIME,ATCPU.USRUSE,ATCPU.SYSUSE,ATPAGE.PAGESCAN
from 'Site!'System'...
```

日時の場合(a)と(1)はそれぞれ結合対象が他方の表に存在しないため、結果は以下のようになります。

(b)+(2)

	DATE	TIME	ATCPU.USRUSE	ATCPU.SYSUSE	ATPAGE.PAGESCAN
	20020102	0000	30	20	1.1

(例 2)

```
select DATE,TIME,ATCPU.USRUSE,ATCPU.SYSUSE,ATPAGE.PAGESCAN
from 'Site!'System' OUTJOIN( ) ...
```

(a)と(1)からの情報を含むため、抽出結果は以下のようになります。

	DATE	TIME	ATCPU.USRUSE	ATCPU.SYSUSE	ATPAGE.PAGESCAN
(a)のみ	20020101	0000	50	30	-1
(1)のみ	20020101	0100	-1	-1	0.8
(b)+(2)	20020102	0000	30	20	1.1

1.4.4. 複数のシステムからのデータ抽出

複数のシステムからデータを抽出する場合、各システムからの抽出結果がレコードの方向に展開されます。

(ex.MF)

複数システムのプロセッサ表(表名：PROC)からのデータ抽出

【Site A,System0 のプロセッサ(PROC)】

日付	時刻	平均 CPU 使用率 (CPU_AVG)	平均 TCB 使用率 (TCB_AVG)	...
20020101	0000	50	40	...
20020101	0100	40	20	
...	

【Site Z,System9 のプロセッサ(PROC)】

日付	時刻	平均 CPU 使用率 (CPU_AVG)	平均 TCB 使用率 (TCB_AVG)	...
20020101	0000	20	10	...
20020101	0100	18	10	
...	

```
select SITE,SYSTEM,DATE,TIME,PROC.CPU_AVG,PROC.TCB_AVG
      from 'SiteA':'System0','SiteZ':'System9' ...
```

【結果】

SITE	SYSTEM	DATE	TIME	PROC.CPU_AVG	PROC.TCB_AVG
SiteA	System0	20020101	0000	50	40
SiteA	System0	20020101	0100	40	20
SiteZ	System9	20020101	0000	20	10
SiteZ	System9	20020101	0100	18	10

(ex.CS)

複数システムのプロセッサ表(表名：ATCPU)からのデータ抽出

【Site XX, System00 のプロセッサ(ATCPU)】

日付	時刻	ユーザモード使用率 (USRUSE)	カーネルモード使用率 (SYSUSE)	...
20020101	0000	50	30	...
20020101	0100	40	50	

【Site XX, System01 のプロセッサ(ATCPU)】

日付	時刻	ユーザモード使用率 (USRUSE)	カーネルモード使用率 (SYSUSE)	...
20020101	0000	20	30	...
20020101	0100	18	40	

```
select SITE,SYSTEM,DATE,TIME,ATCPU.USRUSE,ATCPU.SYSUSE
      from 'SiteXX':'System00','SiteXX':'System01'...
```

【結果】

SITE	SYSTEM	DATE	TIME	ATCPU.USRUSE	ATCPU.SYSUSE
SiteXX	System00	20020101	0000	50	30
SiteXX	System00	20020101	0100	40	50
SiteXX	System01	20020101	0000	20	30
SiteXX	System01	20020101	0100	18	40

1.4.5. グループング

グループング操作を行う場合は、前述の規則に従って抽出された各レコードに対し、グループング単位（種類）の決定とそのグループング単位でのグループ演算を行います。

(ex.1)

時系列によるグループング

【表 X(TBLX)と表 Z(TBLZ)からの抽出レコード(非グループング時)】

	DATE	TIME	TBLX.FLD00	TBLZ.FLDAA
(1)	20020101	0000	10	0
(2)	20020101	0600	20	0
(3)	20020101	1200	30	4
(4)	20020101	1800	40	0
(5)	20020102	0000	50	-1
(6)	20020102	0600	-1	0
(7)	20020102	1200	60	0

```
select DATE,AVG(TBLX.FLD00),AVG(TBLZ.FLDAA) from 'Site'. 'System' group by DATE ...
```

(AVG は平均値を算出するグループ演算関数です。詳細は文法解説を参照してください)

【結果】

DATE	AVG(TBLX.FLD00)	AVG(TBLZ.FLDAA)
20020101	25	1
20020102	55	0

TBLX と TBLZ を結合して抽出された各レコードについてグループング(group by)で指定された式(DATE)の評価とグループの決定が行われます。(1)~(4)は日付 2002/01/01、(5)~(7)は日付 2002/01/02 のグループと判定されます。グループ演算(この例では平均値の算出)は各グループのレコード集合の値を基に行われます。

欠損値(-1)のインターバルは無効なインターバルと見なして平均値や最小値の計算では無視されます。
 (ex. 1)の 2002/01/02 の AVG(TBLX.FLD00) の計算において、06:00 の -1 は無視され、
 $AVG(TBLX.FLD00) = (50 + 60) \div 2 = 55$ となります。
 (CS シリーズでは平均値の算出で欠損値を含んだ計算をすることも可能となっています。
 その場合の計算式と結果は $(50 + 60) \div 3 = 36.666\dots$ となります。)

(ex.2)

特定のフィールドによるグルーピング

【デバイス表(DEVICEX)】

※これは仮想の表です。ES/1 NEO CS/MF シリーズにはそれぞれ独自のデバイス情報の表があります。

	DATE	TIME	DEVNAME	BUSY
(1)	20020101	0000	AAA	50
(2)	20020101	0000	BBB	40
(3)	20020101	0000	CCC	19
(4)	20020101	0000	DDD	80
(5)	20020101	0100	AAA	-1
(6)	20020101	0100	BBB	20
(7)	20020101	0100	CCC	13

```
select DEVICEX.DEVNAME,AVG(DEVICEX.BUSY) from  
      'Site!'System' group by DEVICEX.DEVNAME ...
```

【結果】

DEVICEX.DEVNAME	AVG(DEVICEX.BUSY)
AAA	50
BBB	30
CCC	16
DDD	80

任意の値を使用してグルーピングすることが可能です。

どちらの例もグルーピング(group by)で指定した値とグループ演算の結果を抽出しています。group by の評価結果により一意にならない値を抽出した場合、結果は不定となります。

例えば、上の(ex.1)(ex.2)どちらの例でも時刻(TIME)を抽出しようとすると、そのグループに含まれるいずれかのレコードの時刻が結果となります。このような場合にグループ内のどのレコードの値を使用するかについてはクエリーでは規定していません。

1.4.6. PIVOT 操作

PIVOT 操作は出力結果の行列の再配置を行います(クエリー文における PIVOT 操作の指定については「2.2.2 PIVOTROW、PIVOTCOL 指定」、「2.2.3 PIVOT オプション指定」を参照してください)。

以下に例を用いて説明します。

(ex.1)

日時を行に展開、名前要素を列に展開

【デバイス表(DEVICEX)】

※これは仮定の表です。CS/MF にはそれぞれ独自のデバイス情報の表があります。

DATE	TIME	DEVNAME	BUSY
20020101	0000	AAA	50
20020101	0000	BBB	40
20020101	0000	DDD	80
20020101	0100	AAA	-1
20020101	0100	BBB	20
20020101	0100	CCC	13

(例 1)

```
select DATE,TIME,DEVICEX.DEVNAME,DEVICEX.BUSY from 'Site'.'System' ...
```

【PIVOT 操作を行わない場合の結果】

DATE	TIME	DEVICEX.DEVNAME	DEVICEX.BUSY
20020101	0000	AAA	50
20020101	0000	BBB	40
20020101	0000	DDD	80
20020101	0100	AAA	-1
20020101	0100	BBB	20
20020101	0100	CCC	13

(例 2)

```
select PIVOTROW DATE,PIVOTROW TIME,PIVOTCOL DEVICEX.DEVNAME,
DEVICEX.BUSY as ' ' from 'Site'.'System' ...
```

(as は出力列名を指定するキーワードです。詳細は文法解説を参照してください)

【上記 PIVOT 操作を行った場合の結果】

DATE	TIME	AAA	BBB	CCC	DDD
20020101	0000	50	40	-1	80
20020101	0100	-1	20	13	-1

PIVOTCOL 指定された列(上の例では DEVICEX.DEVNAME)の値は列方向に展開され、PIVOTROW で指定された列(上の例では DATE と TIME)の値は一意になるように出力されます。PIVOTROW も PIVOTCOL も指定されなかった列の値は PIVOTROW と PIVOTCOL の値がマッチする位置に展開されます。

該当データが無い部分は、数値の場合は欠損値(-1)、文字列の場合は空文字列("")となります。
上記の例では 2002/01/01 00:00 の CCC と 2002/01/01 01:00 の DDD のレコードがありません。しかし CCC と DDD のレコードはいずれかのインターバルに存在する為、CCC と DDD の列が生成され、該当レコードが無いインターバルにおいては値が-1 となっています

PIVOT 操作においては PIVOTROW 指定された列の値は一意になります。PIVOTROW と PIVOTCOL の値が複数のレコードにおいて一致する場合、PIVOTROW/COL 指定されない列にどのレコードの値が使用されるかは不定となります。
以下に例を示します。

```
select PIVOTROW DATE,PIVOTCOL DEVICEX.DEVNAME,DEVICEX.BUSY from  
'Site'.System' ...
```

【結果】

DATE	AAA	BBB	CCC	DDD
20020101	不定	不定	13	80

DATE のみで一意にしようとすると AAA と BBB は 2002/01/01 のレコードが 2 件あるのでこのような結果になります(CCC と DDD はいずれも 1 件しかレコードがないため、そのレコードの値が使用されます)。

第2章 クエリー定義の文法

- ・以下の文中 [] で囲まれた部分は省略可能であることを意味します。
- ・繰り返し可能な部分は、1 [,2...] のように表記しています。
- ・A | B はまたはの意味です。([A | B]なら A または B または省略可能となります。)

全体の構文

select 句 from 句 [join 句][filter 句][where 句][when 句][group by 句][having 句]
[order by 句][sub query 句][option 句] [runpy 句]

クエリーの各句は式と特定のキーワードとの組み合わせによって記述します。

各句の式とキーワードの記述において英大小文字は区別されません。
(後述の文字列定数内においては区別されます)

2.1. 式の指定

【式に使用可能な基本要素】

1	データ列	XXX.XXX
2	変数	SITE,SYSTEM,OS,OSNAME,DATE,TIME,HOUR,MINUTE,WEEK,DAY, MONTH,YEAR,FROMDATE,TODATE,YEAR2,MONTH2,YEARBASE,MONTHBAS E,LOOPIDX
3	定数	数値定数, 文字列定数, 曜日定数, OS 定数
4	数値演算子	+, -, *, /, %
5	文字列演算子	&
6	関係演算子	=, <, >, <=, >=, IN, LIKE
7	論理演算子	AND, OR, NOT
8	グループ演算関数	AVG(..), SAV(..), MIN(..), MAX(..), SUM(..), CNT(..), MINOF(..), MAXOF(..), WAV(..), PTL(..), IBAVG(..),UNQCNT(..), MOA(..), MOAD(..), MOS(..), MOSD(..), ISUMMIN(..), ISUMMAX(..), BUCKET(..), ISUMMINOF(..), ISUMMAXOF(..), IRATIOMIN(..), IRATIOMAX(..), IRATIOMINOF(..), IRATIOMAXOF(..), IHITRATIOMIN(..), IHITRATIOMAX(..), IHITRATIOMINOF(..), IHITRATIOMAXOF(..), GMINOF(..), GMAXOF(..), FEDMAXOF(...)
9	数値操作関数	FIX(..), FLOOR(..), CEIL(..), ROUND(..)
10	日時操作関数	DATEADD(..), ROUNDT(..), WEEKNUM(..)
11	文字列操作関数	TOUPPER(..), TOLOWER(..), SUBSTR(..), STRIDX(..), REPSTR(..), NETMASK(..), DATEVAL(..), TOINT(..), TODBL(..), SVTOP(..), PTOSV(..), RXREPSTR(..), XXREPSTR(..)
12	書式化関数	FMTI(..), FMTF(..), FMTD(..), FMTT(..), WTOS(..)
13	条件判定関数	FIF(..), FELSE(..), FSW(..)
14	サブクエリー指定	(SUBQn)
15	特殊変数 (ES/1 NEO CS シリーズのみ有効)	# {TOPNUM}, # {SUMUNIT}, # {CPUNUM}, \$ {ORDB}, \$ {ORDBS}, \$ {SFDB}, \$ {SFDBS}, \$ {SQ8I}, \$ {SQ8IS}, \$ {R3ID}, \$ {R3IDS}, \$ {UDBDB}

2.1.1. データ列について

パフォーマンスデータを表名+'.(ピリオド)'+列名で指定します。使用可能な表名と列名については以下の資料を参照してください。

ES/1 NEO MF シリーズ	ES/1 NEO MF シリーズのクエリーで使用可能なデータ列名
ES/1 NEO CS シリーズ	ES/1 NEO CS シリーズのクエリーで使用可能なデータ列名
(ex.MF)	select PROC.CPU_AVG,PROC.TCB_AVG,...
(ex.CS)	select ATCPU.SYSUSE,ATCPU.USRUSE,...

2.1.2. 変数について

対象データに関する情報を以下のキーワードで指定します。

SITE	データのサイト名
SYSTEM	データのシステム名
OS	データのOS種別 (※1)
OSNAME	データのOS識別子 (※2)
DATE	データの日付 (YYYYMMDD) (※3)
TIME	データの時刻 (HHMM) (※4)
HOURL	データの時刻 (HHMM) の時間 (HH)
MINUTE	データの時刻 (HHMM) の分 (MM)
WEEK	データの日付 (YYYYMMDD) の曜日
DAY	データの日付 (YYYYMMDD) の日 (DD)
MONTH	データの日付 (YYYYMMDD) の月 (MM)
YEAR	データの日付 (YYYYMMDD) の年 (YYYY)
FROMDATE	読込対象の開始日 (YYYYMMDD)
TODATE	読込対象の終了日 (YYYYMMDD)
YEAR2	データの日付 (YYYYMMDD) の年 (YYYY) (※5)
MONTH2	データの日付 (YYYYMMDD) の月 (MM) (※5)
YEARBASE	年区切りの指定 (※6)
MONTHBASE	月区切りの指定 (※6)
LOOPIDX	option 句の loop (2.12 参照) で指定された繰り返しの何番目かを示す整数

- ※1:ES/1 NEO CS シリーズと Flatfile Maintenance でのみ有効です。Unix のデータ、または Windows のデータを返します。
- ※2:ES/1 NEO CS シリーズと Flatfile Maintenance でのみ有効です。OS を識別する文字列 (ex. "Solaris 2.7", "Microsoft Win 2000", ...)を返します。
- ※3:日付は 8 桁の十進整数として扱われます。加減算を行っても暦に基づいた繰り上げ(下げ)はしません。暦に基づいた日付の計算を行いたい場合は後述の DATEADD(..)関数を使用してください。
- ※4:時刻は 4 桁の十進整数として扱われます。加減算を行っても時刻に基づいた繰り上げ(下げ)はしません。
- ※5:ES/1 NEO CS シリーズの CS-MAGIC でのみ有効です。カレンダープロファイルにて設定した年月の区切りで調整された値を返します。
- ※6:ES/1 NEO CS シリーズの CS-MAGIC でのみ有効です。カレンダープロファイルにて設定した年月の区切りの値を返します。カレンダープロファイルを使用していない場合は 1 を返します。

2.1.3. 定数について

数値定数は符号付き(省略時は正)の整数あるいは浮動小数点が指定可能です。

(ex.)123,-1,12.5

加減算の演算子(後述)と定数値を組み合わせる場合には演算子が定数値の符号と誤って認識される場合があるので、演算子と定数値をスペースで区切ってください。

(ex.)123-456 → 123 - 456

文字列定数はシングルクォートで囲まれた任意の文字列です。

(ex.)'Use(%)'

曜日定数は SUN、MON、TUE、WED、THU、FRI、SAT が指定可能です。

(ex.)where WEEK = SUN or WEEK = SAT

OS 定数は OS_UNX(Unix)、OS_WNT(Windows)、OS_I5(i5 OS)、OS_ZVM(z/VM)、OS_VMWARE (VMware)、OS_HYPV(Hyper-V)、OS-VIRTG(Virtage)が指定可能です(ES/1 NEO CS シリーズと Flatfile Maintenance で有効です)。

(ex.)where OS = OS_WNT

2.1.4. 数値演算子について

数値演算子として '+'(加算)、『-』(減算)、『*』(乗算)、『/』(除算)、『%』(剰余)の各演算子が使用可能です。

【注意事項】

クエリーでは負の数値を欠損値(無効な値)として扱っており、以下の規則で演算を行います。

【加算】	$a + b$	<ul style="list-style-type: none"> • $a \geq 0 \wedge b \geq 0$ → $a + b$ • $a \geq 0 \wedge b < 0$ → a • $a < 0 \wedge b \geq 0$ → b • $a < 0 \wedge b < 0$ → -1
【減算】	$a - b$	<ul style="list-style-type: none"> • $a \geq 0 \wedge b \geq 0$ → $a - b$ • $a \geq 0 \wedge b < 0$ → a • $a < 0$ → -1
【乗算】	$a * b$	<ul style="list-style-type: none"> • $a \geq 0 \wedge b \geq 0$ → $a * b$ • $a < 0 \vee b < 0$ → -1
【除算】	a / b	<ul style="list-style-type: none"> • $a \geq 0 \wedge b > 0$ → a / b • $a \geq 0 \wedge b = 0$ → 0 • $a < 0 \vee b < 0$ → -1
【剰算】	$a \% b$	<ul style="list-style-type: none"> • $a \geq 0 \wedge b > 0$ → $a \% b$ • $a \geq 0 \wedge b = 0$ → 0 • $a < 0 \vee b < 0$ → -1

また、小数部が2進数で表すことができない数値を扱う場合は、誤差が発生する場合があります。

2.1.5. 文字列演算子について

文字列演算子として '&' が使用可能です。2つの文字列の連結を行います。

2.1.6. 関係演算子について

これらの演算子は真偽値を返します(条件判定に使用)。

関係演算子として '=' (等しい)、 '<>' (等しくない)、 '<' (小さい)、 '>' (大きい)、 '<=' (以下)、 '>=' (以上) の各演算子が使用可能です。この演算子は左辺と右辺を比較します。文字コードに基づいた文字列同士の大小関係の比較も可能です。

IN 演算子は以下の構文で使用します。

```
expr [NOT] IN (expr1[,expr2...])
```

expr が括弧内の exprn の何れかと一致するかを判定します。

また、ES/1 NEO MF シリーズのみ有効な構文として、

```
TIME [NOT] IN (PNTIME)
```

があります。この場合、レコードの時刻を ES/1 NEO Performance Navigator で設定したサイト/システム毎の対象時刻と比較します。

LIKE 演算子は以下の構文で使用します。

```
expr [NOT] LIKE 'string'
```

expr が 'string' とマッチするかを判定します。'string' 内にはワイルドカードとして '*' (0 文字以上の任意の文字にマッチ) と '?' (任意の一文字にマッチ) が使用可能です。

2.1.7. 論理演算子について

真偽値の演算を行います。AND(論理積)、OR(論理和)、NOT(論理否定)の各論理演算子が使用可能です。

2.1.8. グループ演算関数について

group by 句(後述)と共に使用した場合は各グループのグループ演算値を、group by 句を指定せずに実行した場合は、すべてのデータを 1 つのグループとした場合のグループ演算値を返します。

【注意事項】

小数部が 2 進数で表すことができない数値を扱う場合は、誤差が発生する場合があります。

```
SAV(..),MIN(..),MAX(..),SUM(..)
```

【構文】

SAV(expr)

MIN(expr)

MAX(expr)

SUM(expr)

expr には数値を返す式を指定します。

(但しこの式中にはグループ演算関数は使用できません)

【説明】

それぞれ expr の平均値・最小値・最大値・合計値を求めます。

AVG(..)

【構文】

AVG(expr)

expr には数値を返す式を指定します。

(但しこの式中にはグループ演算関数は使用できません)

【MF シリーズの場合】

expr の平均値を求めます(前述の SAV と同じです)。

【CS シリーズの場合】

CS 動作環境設定の“平均値の算出方法”で“欠損値を含まない”とした場合は expr の平均値を求めます。欠損値(-1)は計算対象外とします(前述の SAV と同じです)。

“欠損値を含む”とした場合は平均算出の際に欠損値(-1)を 0 として扱います。

(ex.)

15 分間隔でサンプルリングした CPU の使用率

時刻	CPU使用率
10:00	60
10:15	-1
10:30	40
10:45	20

“欠損値を含まない”とした場合は、

$$(60 + 40 + 20) \div 3 = 40(\%)$$

となり、

“欠損値を含む”とした場合は、

$$(60 + 0 + 40 + 20) \div 4 = 30(\%)$$

となります。

CNT(..)

【構文】

CNT([expr])

expr には任意の式を指定します。

(但しこの式中にはグループ演算関数は使用できません)

【説明】

グループ中のレコード件数を求めます。

MINOF(..),MAXOF(..)

【構文】

MINOF(base-expr,expr)

MAXOF(base-expr,expr)

base-expr には数値を返す式を、expr には任意の数値を返す式を指定します。

(但しこの式中にはグループ演算関数は使用できません)

【説明】

それぞれ base-expr が最小・最大であった時の expr の値を求めます。

【例】

欠損値の場合は、以下の動作仕様となります。

以下のようなデータを対象にした場合

	expr1	expr2
10:00	90	-1
10:15	80	2

...

expr1 の 10:00 の 90 が最大値で、10:15 の 80 がその次に大きい値の場合

expr2 は 10:15 の 2 を返します。

WAV(..)

【構文】

WAV(exprA,exprB)

exprA、exprB には数値を返す式を指定します。

(但しこの式中にはグループ演算関数は使用できません)

【説明】

exprA を exprB で加重した平均値を求めます。WAV(A,B)は $SUM(A*B)/SUM(B)$ と等価です。

PTL(..)

【構文】

PTL(expr,position)

expr には数値を返す式を指定します。

(但しこの式中にはグループ演算関数は使用できません)

position は 0~100 までの数値が指定可能です。

【説明】

expr の position パーセンタイル値を求めます(position が 0 の時は最小値、100 の時は最大値になります)。position ÷ 100 が 1 ÷ (expr の要素数 -1)の倍数でない場合、position に相当する値を算出する為に値の補間を行います。

【例】

(0,10,20,40)という要素から 50 パーセンタイル値を求める場合、10 と 20 の中間の値として 15 を出力します。

IBAVG(..)

【構文】

IBAVG(expr)

expr には数値を返す式を指定します。

(但しこの式中にはグループ演算関数は使用できません)

【説明】

expr の合計値をインターバル数で割った値を求めます。

【例】

TIME	User	Command	CPU使用率 (%)
1000	Sys	sh	10
1000	Sys	coma	20
1000	Sys	comb	6
1030	Sys	sh	10
1030	Sys	coma	4

User-Sys の 10 時台の平均 CPU 使用率を求めようとする時に、IBAVG(..)を用いると CPU 使用率の合計値 (10+20+6+10+4)をインターバル数(10:00 と 10:30 の 2 インターバル)で割った結果の 25(%)を返します。

UNQCNT(..)

【構文】

UNQCNT(expr)

expr には任意の式を指定します。

(但しこの式中にはグループ演算関数は使用できません)。

【説明】

expr の結果を一意にした時の要素数を求めます。

【例】

(A,B,C,A,A,B,B,B,D,C)という集合を一意にすると(A,B,C,D)の 4 つなので UNQCNT(..)は 4 を返します。

MOA(..),MOAD(..),MOS(..),MOSD(..)

【構文】

MOA(expr)
MOAD(expr)
MOS(expr)
MOSD(expr)

expr には数値を返す式を指定します(但しこの式中にはグループ演算関数は使用できません)。

【説明】

MOA は expr の日毎の平均値を求め、その内の最大値を返します。
MOAD は expr の日毎の平均値を求め、それが最大であった日付を返します。
MOS は expr の日毎の合計値を求め、その内の最大値を返します。
MOSD は expr の日毎の合計値を求め、それが最大であった日付を返します。

- ・「日」より大きい単位(週/月/年など)でグルーピングするクエリーで有効です。
"group by DATE" や"group by DATE, TIME" のようにグルーピングするクエリーで使用しても意味はありません。
- ・複数システムデータにこの関数は使用できません。

ISUMMIN(..),ISUMMAX(..)

【構文】

ISUMMIN(expr)
ISUMMAX(expr)

expr には数値を返す式を指定します(但しこの式中にはグループ演算関数は使用できません)。

【説明】

expr のインターバル毎の合計値の最小、最大を返します。

【例 1】

表 X(TBL_X)

時刻	FLD_0
01:00	10
01:15	3
01:15	4
01:30	9
01:30	8
01:45	12

“group by HOUR”とした時、ISUMMIN(TBL_X.FLD_0)は 7(01:15 の合計値)、ISUMMAX(TBL_X.FLD_0)は 17(01:30 の合計値)を返します。

1 つのグループに複数システムのデータが集約される場合は、システム毎のインターバル毎の合計値の最小、最大を返します。

【例 2】

表 X(TBL_X)システム A のデータ

時刻	FLD_0
01:00	18
01:15	3
01:15	4
01:30	9
01:30	8
01:45	12

システム B のデータ

時刻	FLD_0
01:00	10
01:15	3
01:15	4
01:30	9
01:30	8
01:45	2

“group by HOUR”とした時、ISUMMIN(TBL_X.FLD_0)は 2(システム B の 01:45 の合計値)、ISUMMAX(TBL_X.FLD_0)は 18(システム A の 01:00 の合計値)を返します。

BUCKET(..)

【構文】

BUCKET(expr, a, b)

BUCKET(expr, a)

expr、a、b には数値を表す式を指定します(但しこの式中にはグループ演算関数は使用できません)。

【説明】

引数を 3 つ指定した場合、 $a \leq \text{expr} < b$ となった回数をカウントします。

引数を 2 つ指定した場合、 $a \leq \text{expr}$ となった回数をカウントします。

なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

ISUMMINOF(..), ISUMMAXOF(..)

【構文】

ISUMMINOF(a, b)

ISUMMAXOF(a, b)

a、b には数値を表す式を指定します(但しこの式中にはグループ演算関数は使用できません)。

【説明】

a のインターバル合計値が最小もしくは最大となったインターバルの b の値を返します。

最小もしくは最大となるインターバルが複数あった場合、最も古いインターバルの b の値を返します。b がインターバル内で一意にならない場合、どの値が返るかは不定です。

なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

IRATIOMIN(..), IRATIOMAX(..)

【構文】

IRATIOMIN(a, b)

IRATIOMAX(a, b)

【説明】

a のインターバル合計値 ÷ b のインターバル合計値で算出した比率のうち、最小もしくは最大の値を返します。a のインターバル合計値が 0 未満、もしくは、b のインターバル合計値が 0 以下となったインターバルは無視します。なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

IRATIOMINOF(..), IRATIOMAXOF(..)

【構文】

IRATIOMINOF(a, b, c)

IRATIOMAXOF(a, b, c)

【説明】

a のインターバル合計値 ÷ b のインターバル合計値で算出した比率が最小もしくは最大となったインターバルの、c の値を返します。最小もしくは最大となるインターバルが複数あった場合、最も古いインターバルの c の値を返します。

a のインターバル合計値が 0 未満、もしくは、b のインターバル合計値が 0 以下、もしくは、a のインターバル合計値 > b のインターバル合計値となったインターバルは無視します。

c がインターバル内で一意にならない場合、どの値が返るかは不定です。

なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

IHITRATIOMIN(..), IHITRATIOMAX(..)

【構文】

IHITRATIOMIN(a, b)

IHITRATIOMAX(a, b)

【説明】

1 - a のインターバル合計値 ÷ b のインターバル合計値で算出したヒット率(0~1)、最小もしくは最大の値を返します。

a のインターバル合計値が 0 未満、もしくは、b のインターバル合計値が 0 以下、もしくは a のインターバル合計値 > b のインターバル合計値となったインターバルは無視します。

なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

IHITRATIOMINOF(..), IHITRATIOMAXOF(..)

【構文】

IHITRATIOMINOF(a, b, c)

IHITRATIOMAXOF(a, b, c)

【説明】

1 - a のインターバル合計値 ÷ b のインターバル合計値で算出したヒット率(0~1)が最小もしくは最大となったインターバルの、c の値を返します。a のインターバル合計値が 0 未満、もしくは、b のインターバル合計値が 0 以下、もしくは、a のインターバル合計値 > b のインターバル合計値となったインターバルは無視します。

最小もしくは最大となるインターバルが複数あった場合、最も古いインターバルの c の値を返します。

c がインターバル内で一意にならない場合、どの値が返るかは不定です。

なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

GMINOF(..), GMAXOF(..)

【構文】

GMINOF(集計方法 1, 式 1)

GMINOF(集計方法 1, 式 1, 集計方法 2, 式 2)

GMAXOF(集計方法 1, 式 1)

GMAXOF(集計方法 1, 式 1, 集計方法 2, 式 2)

【説明】

集計方法 2、式 2 が省略されている場合、インターバル毎に集計方法 1(式 1)を算出し、その集計結果が最小/最大となったインターバルの値を取得します。

集計方法 2、式 2 が指定されている場合、インターバル毎に集計方法 1(式 1)を算出し、その集計結果が最小/最大となったインターバルの集計方法 2(式 2)の集計結果を取得します。

集計方法 1, 2 には以下を指定可能です。

'MIN' (最小値)

'MAX' (最大値)

'SUM' (合計値)

'AVG' (平均値)

'UNQCNT' (ユニークなカウント)

なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

【例 1】

ユニーククライアント数の最大値

gmaxof('unqcnt', WLOGDET.PEER)

インターバル	UNQCNT(WLOGDET.PEER)
2016/05/10 13:00	16
2016/05/10 13:15	42
2016/05/10 13:30	46
2016/05/10 13:45	32

UNQCNT(WLOGDET.PEER)が最大となったインターバル 2016/05/10 13:30 の値 46 を取得します。

【例 2】

アクセス件数 (SUM(1)) が最大となったインターバルの平均レスポンス時間 (AVG(WLOGDET.RSPMS))

gmaxof('SUM', 1, 'AVG', WLOGDET.RSPMS)

インターバル	SUM(1)	AVG(WLOGDET.RSPMS)
2016/05/10 13:00	501	1.259
2016/05/10 13:15	1290	1.564
2016/05/10 13:30	1200	1.259
2016/05/10 13:45	1115	2.102

アクセス件数 SUM(1)が最大となったインターバル 2016/05/10 13:15 の AVG(WLOGDET.RSPMS) 1.564 を取得します。

FEDMAXOF(..)

【構文】

FEDMAXOF(式 1,式 2)

※式 1 に文字列型を指定しないでください。

【説明】

group by 句を指定している場合、group by 指定を無視して集約した式 1 の合計値が最大となるインターバルの、式 2 の合計値を取得します。これを利用することで、例えばグループ化項目毎(LPAR 毎、ユーザ毎など)のプロセッサ使用率(式 1)を積み上げた合計値が最大となったインターバルの、グループ化項目毎の任意の項目の合計値を取得することが可能です。

例) select

```
ATACCD.COMDNAME,
FEDMAXOF(ATACCD.CPUUSE, ATACCD.MEMUSE)
```

:

```
where ATACCD.COMDNAME in ('SYSTEM','LOCAL_SERVICE','NETWORK_SERVICE')
```

```
group by ATACCD.COMDNAME
```

下表のようなデータがあるときに上記集約クエリーを実行すると、group by 指定を無視して集約したプロセッサ使用率 (ATACCD.CPUUSE)の合計値が最大となる 2016/05/10 13:15 のインターバルが選択され、該当インターバルのコマンド毎のメモリ使用量(ATACCD.MEMUSE)の合計値が抽出されます。

インターバル	ATACCD.CMDNAME	ATACCD.CPUUSE	ATACCD.MEMUSE	group by指定を無視して集約したATACCD.CPUUSEの合計	ATACCD.MEMUSEの合計
2016/05/10 13:00	SYSTEM	1	1300000	5.05	8892000
	SYSTEM	0.5	656000		
	SYSTEM	0.7	6936000		
	LOCAL_SERVICE	1.25	2696000		12860000
	LOCAL_SERVICE	0.1	10164000		
	NETWORK_SERVICE	1.5	12345000		12345000
2016/05/10 13:15	SYSTEM	0.3	1300000	12.85	7956000
	SYSTEM	0.5	656000		
	SYSTEM	0.4	6000000		
	LOCAL_SERVICE	1.20	2696000		152949000
	LOCAL_SERVICE	8.95	150253000		
	NETWORK_SERVICE	1.5	10598000		10598000
2016/05/10 13:30	SYSTEM	0.8	1300000	8.5	9214000
	SYSTEM	0.5	656000		
	SYSTEM	0.6	7258000		
	LOCAL_SERVICE	1	2696000		15196000
	LOCAL_SERVICE	4.3	12500000		
	NETWORK_SERVICE	1.3	10396000		10396000

group by 句を指定していない場合、式 1 の合計値が最大となるインターバルの、式 2 の合計値を取得します。

なお、この関数は Flatfile Maintenance のデータ集約でのみ使用可能です。

INT オプション

【構文】

INT グループ演算関数(..)

【説明】

重複したインターバルの値を計算から除外します。このオプションは 1 インターバルに 1 件しか出力されないレコードと 1 インターバルに複数件出力されるレコードを組み合わせてグルーピングを行う場合に必要となる場合があります。

表 A(TBLA)

日付	時刻	VALA
20020101	1000	90
20020101	1100	80

表 B(TBLB)

日付	時刻	NAME	VALB
20020101	1000	AAA	50
20020101	1000	BBB	40
20020101	1100	AAA	30

上記の表 A と表 B から

select DATE,TIME,TBLA.VALA,TBLB.VALB from 'Site'.'System' order by DATE,TIME を実行した結果は、

DATE	TIME	TBLA.VALA	TBLB.VALB
20020101	1000	90	50
20020101	1000	90	40
20020101	1100	80	30

となります。以下のクエリー

select DATE,SUM(TBLA.VALA),SUM(TBLB.VALB) from 'Site'.'System' group by DATE を実行した結果は、

DATE	SUM(TBLA.VALA)	SUM(TBLB.VALB)
20020101	260	120

となります。

結合の結果 10:00 の TBLA のレコードが複数回計算される為 SUM(TBLA.VALA)の結果は(90+90+80=)260 となります。

INT を使用して

select DATE,INT SUM(TBLA.VALA),SUM(TBLB.VALB) from 'Site!.'System' group by DATE とすると結果は

DATE	INT SUM(TBLA.VALA)	SUM(TBLB.VALB)
20020101	170	120

となります。

10:00 の TBLA のレコードは 1 回だけ合計計算の対象となります。

2.1.9. 数値操作関数について

FIX(..)

【構文】

FIX(expr)

expr には数値を返す式を指定します。

【説明】

expr の小数点以下を切り捨てた値を返します。

FLOOR(..),CEIL(..),ROUND(..)

【構文】

FLOOR(expr[,pos])

CEIL(expr[,pos])

ROUND(expr[,pos])

expr には数値を返す式、pos には整数を指定します。

【説明】

FLOOR は数値の切り捨て、CEIL は数値の切上げ、ROUND は数値の四捨五入を行います。

pos が 0 の場合は、小数部以下を処理し、整数部を残します。

pos が 1 以上の場合は、小数部第 pos+1 位以下を処理し、小数部第 pos 位までを残します。

pos が -1 以下の場合は、整数部第-pos 位以下を処理し、整数部第-pos+1 位までを残します。

結果は浮動小数点形式で返されます。

2.1.10. 日時操作関数について

クエリーでは日付は 8 桁の 10 進整数、時刻は 4 桁の 10 進整数として扱っています。その為、数値演算子による演算を行っても暦や時計に基づいた繰り上げ下げ等を行われません(10 進演算と同じ結果となります)。

日付・時刻を操作する関数として以下の 2 つがあります。

DATEADD(..)

【構文】

DATEADD(date-expr,interval,dist-expr)

date-expr には日付を返す式、dist-expr には整数を指定してください。interval は以下のいずれかを指定してください。
'y'or'Y'or'm'or'M'or'd'or'D'

【説明】

date-expr から dist-expr 離れた日付を求めます。interval が'y'か'Y'の時は dist-expr は年の、'm' か'M'の時は月の、'd' か'D'の時は日の間隔と認識します。dist-expr が負の時は dist-expr 年(or 月 or 日)以前となります。

【例】

DATEADD(20020101,'Y',1) → 2003/01/01

DATEADD(20020101,'M',-1) → 2001/12/01

DATEADD(20020101,'D',-1) → 2001/12/31

ROUNDT(..)

【構文】

ROUNDT(time-expr,round-unit)

time-expr には時刻を返す式、round-unit には正の整数を指定してください。

【説明】

時刻値を任意の時(分)で丸めます。分単位で丸めたい場合はその値を、時間単位で丸めたい場合は時間単位に 100 を掛けた数値を round-unit に指定します。

【例】

ROUNDT(1045, 20) → 1040 (20 分単位の丸め)

ROUNDT(1700,400) → 1600 (4 時間単位の丸め)

WEEKNUM(..)

【構文 1】

WEEKNUM(date-expr,base)

【構文 2】

WEEKNUM(date-expr)

date-expr には日付を返す式を指定してください。base は次の【説明】をご覧ください。

【説明】

date-expr がその年の第何週目かを返します。

構文 1 における base は週の始まりの曜日の指定で、0-6(0-日、1-月、…6-土)か SUN、MON、…SAT を使用可能です。

構文 2 では週の始まりは日曜として計算します。

date-expr が日付として認識できない場合は 0 を返します。

2.1.11. 文字列操作関数について

TOUPPER(..)

【構文】

TOUPPER(string-expr)
string-expr には文字列を返す式を指定してください。

【説明】

string-expr 内の英小文字をすべて英大文字に変換します。

TOLOWER(..)

【構文】

TOLOWER(string-expr)
string-expr には文字列を返す式を指定してください。

【説明】

string-expr 内の英大文字を全て英小文字に変換します。

SUBSTR(..)

【構文】

SUBSTR(string-expr,offset,length)
string-expr には文字列を返す式を指定してください。offset と length には整数値を返す式を指定してください。

【説明】

string-expr の部分文字列を返します。
offset ≥ 0 の場合、部分文字列の開始位置は string-expr の左端から offset バイト目になります。
offset < 0 の場合、部分文字列の開始位置を右から数えます。
length > 0 の場合、開始位置から length バイト分を、
length = 0 の場合、開始位置から残りすべてを、
length < 0 の場合、開始位置から、元の文字列に -length バイト分残した文字列を取り出します。

【例】

SUBSTR('abcde', 2, 2)	→	'cd'
SUBSTR('abcde',-4, 3)	→	'bcd'
SUBSTR('abcde', 1,-2)	→	'bc'

STRIDX(..)

【構文】

STRIDX(base-expr,search-expr)

base-expr と search-expr には文字列を返す式を指定してください。

【説明】

base-expr 中に search-expr が最初に現れた位置をオフセットで返します。

【例】

STRIDX('abcde', 'cd') → 2

REPSTR(..)

【構文】

REPSTR(expr,file-name)

expr には文字列を返す式を指定してください。

file-name にはシングルクォートされた文字列を指定してください。

【説明】

expr の結果を file-name で指定されたファイル中の変換定義に従って変換します。

file-name がフルパスで指定されなかった場合は CS シリーズ、Flatfile Maintenance、または Performance Navigator のインストールディレクトリからの相対パスでファイルを決定します。

【変換定義の記述について】

変換定義は複数の変換定義行から構成されます。各変換定義行は式の結果をマッチさせるパターン指定と、パターンにマッチした場合に変換する文字列をカンマ(,)で区切って記述してください。パターンにはアスタリスク(*)とクエスチョンマーク(?)が使用可能であり、それぞれ 0 文字以上の任意の文字と任意の 1 文字にマッチします。マッチングはファイル中に記述した順番で行われ、あるパターンにマッチした時点でそれ以降の記述に対するマッチングは停止します。どのパターンにもマッチしなかった場合は式の結果は変換されません(パターン、変換後文字列共にクォートする必要はありません)。半角スペースで始まる行とカンマ(,)を含まない行は無視されます。また、ファイル中の各行の長さは 1000 文字未満にしてください。変換定義ファイルの拡張子は「.def」としてください。

【例】

C:¥iim¥work¥appropriate.def という変換定義ファイルを以下のような内容で作成した場合

```
COMMENT LINE
/foo/bar/*.jsp, BAR の JSP
/foo/*.jsp, FOO の JSP
/doc/*.html, ドキュメント
*, その他
```

この変換定義ファイルを使用する場合は、クエリー中に

... REPSTR(WLOGDET.URL, 'C:¥iim¥work¥appropriate.def')... のように記述します。

上記変換定義例では /foo/bar/*.jsp は foo/*.jsp より上の行に記述する必要があります。

(/foo/bar/*.jsp にマッチする値は /foo/*.jsp にもマッチする為)

RXREPSTR(..)

【構文】

1. RXREPSTR(expr,file-expr)
 2. RXREPSTR(expr,file-expr,index)
 3. RXREPSTR(expr,file-expr,index,default)
- expr,file-expr,default には文字列を返す式を指定してください。
index には整数を返す式を指定してください。

【説明】

expr の結果を file-expr で示されるファイル中の変換定義に従って変換します。
file-expr がフルパスで指定されなかった場合は CS シリーズ、Flatfile Maintenance、または Performance Navigator のインストールディレクトリからの相対パスでファイルを決定します。
変換定義ファイルには複数の変換候補を記述することが可能であり(詳しくは下の説明を参照してください)、index (ほどの変換候補を使用するかを指定する引数です。構文 1.においては index に 0 が指定されたものとみなします。変換定義ファイルから変換候補が見つからなかった場合は default に指定された式の値が返されます。構文 1.と 2.においては default に expr が指定されたものとみなします。なお、Flatfile Maintenance のデータ集約では使用できません。

【変換定義の記述について】

変換定義は複数の変換定義行から構成されます。各変換定義行は式(上記 expr 引数)の結果をマッチさせるパターン指定と、パターンにマッチした場合に変換する文字列(以下“変換文字列”と記述します)をタブ文字で区切って記述してください。

変換文字列は複数記述可能であり、その場合は各変換後文字列をタブ文字で区切って記述してください。パターンは正規表現で記述してください。パターンマッチはファイル中に記述した順番で行われ、あるパターンにマッチした時点でそれ以降の記述に対するマッチングは停止します。パターンマッチはサーチ(パターンを含むかどうか)により判定されます。先頭からのマッチングを行いたい場合は先頭に(文字列の先頭を示す正規表現である)“^”を記述してください。

どの変換後文字列を使用するかは上記の index 引数で指定します。最も左側の変換文字列を 0、次を 1. . . として数えます。index の指定が 0 より小さい場合は(0 が指定されたものとみなし)最も左側の変換後文字列が使用されます。index の指定が変換後文字列の候補数以上の場合は最も右側の変換後文字列が使用されます。パターンが不正な正規表現である場合、または、4096 バイト以上の行が存在する場合は実行時にエラーが発生します。空白やタブで始まる行、または、タブを含まない行はコメントとみなします。変換定義ファイルの拡張子は「.def」としてください。

【例】

C:¥iim¥work¥example.def という変換定義ファイルを以下のような内容で作成した場合
(以下の空白部分は 1 つのタブとします)

```
THIS IS COMMENT LINE.  
^/foo/.* ¥.jsp FOO_JSP FOO  
^/foo/. FOO_OTHER FOO  
.* ¥.jsp OTH_JSP OTH  
.* OTH_OTHER OTH
```

クエリー中の RXREPSTR(WLOGDET.URL,'C:¥iim¥work¥example.def')という式は、WLOGDET.URL の値が '/foo/red.jsp'であれば'FOO_JSP'に、'/bar/red.jsp'であれば'OTH_JSP'に変換されます。

RXREPSTR(WLOGDET.URL,'C:¥iim¥work¥example.def',1)という式は、WLOGDET.URL の値が '/foo/'で始まれば'FOO'に、それ以外であれば'OTH'に変換されます。

XXREPSTR(..)

【構文】

- 1.XXREPSTR(expr,file-expr)
 - 2.XXREPSTR(expr,file-expr,index)
 - 3.XXREPSTR(expr,file-expr,index,default)
- expr,file-expr,default には文字列を返す式を指定してください。
index には整数を返す式を指定してください。

【説明】

expr の結果を file-expr で示されるファイル中の変換定義に従って変換します。
file-expr がフルパスで指定されなかった場合は CS シリーズ、または Performance Navigator のインストールディレクトリからの相対パスでファイルを決定します。

変換定義ファイルには複数の変換候補を記述することが可能であり(詳しくは下の説明を参照してください)、index (ほどの変換候補を使用するかを指定する引数です。構文 1.においては index に 0 が指定されたものとみなします。変換定義ファイルから変換候補が見つからなかった場合は default に指定された式の値が返されます。構文 1.と 2.においては default に expr が指定されたものとみなします。なお、Flatfile Maintenance のデータ集約では使用できません。

【変換定義の記述について】

変換定義は複数の変換定義行から構成されます。各変換定義行は式(上記 expr 引数)の結果をマッチさせるパターン方法(以下“タイプ”と記述します)、マッチパターン、パターンにマッチした場合に変換する文字列(以下“変換文字列”と記述します)をタブ文字で区切って記述してください。

タイプは'l','a','r'のいずれかを指定可能です。

'l'はマッチパターンを文字列で指定し、文字列中には'?'(任意の一文字にマッチ)と'*'(0 文字以上の任意の文字列にマッチ)が使用可能です。

'a'はマッチパターンをハイフン(-)で区切った IP アドレスの範囲で指定します。

'r'はマッチパターンを正規表現で指定します。

マッチングはファイル中に記述した順番で行われ、ある行にマッチした時点でそれ以降の記述に対するマッチングは停止します。'r'を指定した場合のマッチングはサーチ(パターンを含むかどうか)により判定されます。先頭からのマッチングを行いたい場合は先頭に(文字列の先頭を示す正規表現である)“^”を記述してください。

変換文字列は複数記述可能であり、その場合は各変換後文字列をタブ文字で区切って記述してください。どの変換後文字列を使用するかは上記の index 引数で指定します。最も左側の変換文字列を 0、次を 1. . . として数えます。index の指定が 0 より小さい場合は(0 が指定されたものとみなし)最も左側の変換後文字列が使用されます。index の指定が変換後文字列の候補数以上の場合は最も右側の変換後文字列が使用されます。パターンが不正な正規表現である場合、または、4096 バイト以上の行が存在する場合は実行時にエラーが発生します。'l','a','r'で始まらない行やタブ文字を含まない行はコメントとみなします。変換定義ファイルの拡張子は、「.def」としてください。

【例】

```
#'l' を指定
l http://example.com/*   example.com
#'a' を指定
a 192.0.2.50-192.0.2.88 グループ A
#'r' を指定
r user[0-5] ユーザー(0 ~ 5)
```

NETMASK(..)

【構文】

NETMASK(expr,mask)

expr には文字列を返す式を、mask には数値を返す式を指定してください。

【説明】

expr の値がピリオド(.)で区切られた有効な IP アドレス表現で、かつ、mask の値が 1 以上 31 以下ならば、expr から得た IP アドレスの上位 mask ビット分のみを有効とした IP アドレス表現を文字列で返します。

上の条件に当てはまらない場合は expr をそのまま返します。

【例】

NETMASK("172.18.11.9",16)	→	"172.18.0.0"
NETMASK("172.18.12.101",24)	→	"172.18.12.0"
NETMASK("OTHER",24)	→	"OTHER"

DATEVAL(..)

【構文】

DATEVAL(format,date-time-expr)

format には文字列を、date-time-expr には文字列を返す式を指定してください。

【説明】

format の指定に従い date-time-expr を解釈し、日時を表す数値を返します。数値の整数部は日付を 1900 年 1 月 1 日を 1 とした経過日数で表し、数値の小数部は時刻を 24 時間に対する割合で表します。

(1.0 は 1900/1/1 00:00:00、1.5 は 1900/1/1 12:00:00、2.5 は 1900/1/2 12:00:00 になります)

format は date-time-expr をどの様に解釈するかを指定する文字列であり、以下の指定子が使用可能です。

%y	-	2桁の年(70から99は1970年から1999年とし、00から69は2000年から2069年とします)
%Y	-	4桁の年
%m	-	月
%d	-	日
%H	-	時間
%M	-	分
%S	-	秒

例えば date-time-expr が "2006/01/15 12:10:11" のような値をとる場合は、format に "%Y/%m/%d %H:%M:%S" を指定します。

format の指定と date-time-expr の値がマッチしない場合は欠損値(-1)を返します。

また、変換可能な日時の範囲は 1970 年 1 月 1 日 9 時 0 分 0 秒から 2038 年 1 月 19 日 3 時 14 分 7 秒までであり、この範囲外の日時を変換対象とした場合も欠損値(-1)を返します。

なお、Flatfile Maintenance のデータ集約では使用できません。

TOINT(..)

【構文】

TOINT(expr)

expr には文字列を返す式を指定してください。

【説明】

expr の値を整数値に変換した値を返します。

変換可能な範囲は-2147483648 から 2147483647 までの間の整数表現です。

オーバーフローした場合の返り値は不定です。

変換できなかった場合は 0 を返します。

【例】

TOINT('1')	→	1
TOINT('-1')	→	-1
TOINT('1.25')	→	1
TOINT('abc')	→	0

TODBL(..)

【構文】

TODBL(expr)

expr には文字列を返す式を指定してください。

【説明】

expr の値を浮動小数点値に変換した値を返します。

オーバーフローした場合の返り値は不定です。

変換できなかった場合は 0.0 を返します。

【例】

TODBL('1.25')	→	1.25
TODBL('-1.25')	→	-1.25
TODBL('1')	→	1.0
TODBL('abc')	→	0.0

SVTOP(..)

【構文】

SVTOP(string)

string にはネットワークサービス名を示す文字列を指定してください。

【説明】

string に指定されたネットワークサービス名に対応するポート番号を返します。

ただし、string に 'OTHER' もしくは '25', '80' のような数字のみの指定が行われた場合は置換を行いません。なお、Flatfile Maintenance のデータ集約では使用できません。

【例】

SVTOP('SMTP') → '25'

SVTOP('HTTP') → '80'

【備考】

ネットワークサービス名とポート番号の対応は以下のファイルを参照します。優先順位が上位のファイルから走査します。すべてのファイルに該当の文字列が存在しない場合は交換を行いません。

優先順位	ファイル
1	%SYSTEM¥drivers¥etc¥services
2	CSインストールフォルダ ¥xservice.txt (例) C:¥Program Files¥IIM¥CS¥xservice.txt
3	CSインストールフォルダ ¥services.txt (例) C:¥Program Files¥IIM¥CS¥services.txt

xservice.txt は以下の書式に従い変更が可能です。固有のアプリケーションを使用している場合は対応するポート番号をファイルに反映させてください。「#」はコメントアウトとなります。

ネットワークサービス名,ポート番号

PTOSV(..)

【構文】

PTOSV(expr)

expr にはポート番号を指定してください。

【説明】

expr に指定されたポート番号に対応するネットワークサービス名を返します。

ただし、expr に 'OTHER' が指定された場合は置換を行いません。なお、Flatfile Maintenance のデータ集約では使用できません。

【例】

PTOSV('25') → 'SMTP'

PTOSV('80') → 'HTTP'

【備考】

ネットワークサービス名とポート番号の対応は以下のファイルを参照します。優先順位が上位のファイルから走査します。すべてのファイルに該当の文字列が存在しない場合は交換を行いません。

優先順位	ファイル
1	%SYSTEM¥drivers¥etc¥services
2	CSインストールフォルダ ¥xservice.txt (例) C:¥Program Files¥IIM¥CS¥xservice.txt
3	CSインストールフォルダ ¥services.txt (例) C:¥Program Files¥IIM¥CS¥services.txt

xservice.txt は以下の書式に従い変更が可能です。固有のアプリケーションを使用している場合は対応するポート番号をファイルに反映させてください。「#」はコメントアウトとなります。

ネットワークサービス名,ポート番号

2.1.12. 書式化関数について

FMTI(..)

【構文】

FMTI(format-expr,expr)

expr には整数値を返す式を指定してください。

format-expr には文字列定数を指定してください。

【説明】

expr の値を書式付で返します。書式は format-expr で指定します。

expr の書式指定は以下の形式になります。

%[flags][width]d

[flags]は以下の何れかの組み合わせを指定可能です。

flags	意味	省略時
-	[width]で指定されたフィールド幅に結果を左詰する。	右詰
+	+または-の符号を付ける。	負の値のみ-を付加
0	[width]で指定されたフィールド幅まで前に0を付加する。	0の付加無

[width]は最小文字幅を指定します。値の文字幅が[width]で指定された文字幅より少ないと、[flags]の指定に従い0や空白が追加されます。

【例】

FMTI('%02d:00',HOUR) →時間を2桁の幅で、2桁に満たない場合は前に0を付加し、
'09:00','10:00',...のように編集します。

FMTF(..)

【構文】

FMTF(format-expr,expr)

expr には数値を返す式を指定してください。

format-expr には文字列定数を指定してください。

【説明】

expr の値を書式付で返します。書式は format-expr で指定します。expr の書式指定は以下の形式になります。

%[flags][width][precision]e|E|f|g|G

[flags]は以下の何れかの組み合わせを指定可能です。

flag	意味	省略時
-	FMTI (..) に同じ	
+		
0		
*	e,E,fと同時に使用すると強制的に小数点が入る。	後続の数値がある場合少数点が入る。
	g,Gと同時に使用すると強制的に小数点が入り、後続する0が切り捨てられない。	後続の数値がある場合小数点が入り、後続の0は切り捨てられる。

[width]は最小文字幅を指定します。値の文字幅が[width]で指定された文字幅より少ないと、[flags]の指定に従い0や空白が追加されます。

[precision]は精度指定をピリオド(.)に続けて正の10進整数で行います。[precision]の解釈は後続の指定により異なり、次のようになります。

後続の指定	意味	省略時
e,E	小数点以下の表示桁数を指定する。 最後の数字は丸められる。	精度のデフォルトは6。 0を指定するか、後続する数値を指定しないでピリオドを指定すると、少数点は表示されない。
f	小数点以下の表示桁数を指定する。 小数点が表示される場合は、その前に少なくとも数値が1つ表示される。 値は適切な桁数まで丸められる。	
g,G	表示する最大有効桁数を指定する。	6桁の有効桁を表示し、後続の0は切り捨てられる。

最後の指定はデータ型の指定です。

データ型	書式
e	[-]d.dddde[sign]ddd形式。dは1個の10進数、ddddは1個または複数個の10進数、dddは3桁の10進数、signは+または-。
E	指数の前にeではなくEが付くのを除き上に同じ。
f	[-]dddd.dddd形式。ddddは、1個または複数個の10進数。小数点の前の桁数はその数の絶対値によって決定され、小数点の後の桁数は要求された精度によって決定される。
g	またはeの書式のうち、指定された値および精度を表現できる短い方の書式。e書式は値の指数部が-4より小さいか[precision]で指定された数よりも大きい場合にのみ使用される。後続の0は切り捨てられ、小数点は1個または複数の数字が続く場合にのみ表示される。
G	6桁の有効桁を表示し、後続の0は切り捨てられる。

【例】

FMTF('% .4f', 1.12345) → 1.1235
FMTF('%f', 1.1234567) → 1.123457

FMTD(..)

【構文】

FMTD(format-number,expr)
expr には日付を返す式を指定してください。
format-number には 1～5 の整数を指定してください。

【説明】

日付の書式化を行います。format-number の指定により以下の書式形式となります。

format-number	書式
1	YYYYMMDD
2	YYMMDD
3	YYYY/MM/DD
4	YY/MM/DD
5	Y/M/D

FM TT(..)

【構文】

FM TT(format-number,expr)
expr には時刻を返す式を指定してください。
format-number には 1～4 の整数を指定してください。

【説明】

時刻の書式化を行います。format-number の指定により以下の書式形式となります。

format-number	書式
1	HHMM
2	HH:MM
3	H:MM
4	H:M

WTOS(..)

【構文】

WTOS(format-string,expr)

expr には曜日を返す式を指定してください。

format-string には'e'、'E'、'j'、'J'のいずれかを指定してください。

【説明】

曜日の書式化を行います。曜日は内部的に数値として扱っているため文字列としての結果を得たい場合にはこの関数を指定してください。format-string の指定により以下の書式形式となります。

format-number	書式
'e', 'E'	SUN, MON, TUE, WED, THU, FRI, SAT
'j', 'J'	日, 月, 火, 水, 木, 金, 土

なお、Flatfile Maintenance のデータ集約では使用できません。

2.1.13. 条件判定関数について

FIF (..)

【構文】

FIF(cond-expr,expr1,expr2)

cond-expr には真偽値を返す式を、expr1 と expr2 には任意の式を指定してください。

【説明】

cond-expr の判定結果が真ならば expr1 の値を、偽ならば expr2 の値を返します。

FELSE (..)

【構文】

FELSE(cond-expr,expr)

cond-expr には関係演算子を使用した式を、expr には任意の式を指定してください。

【説明】

cond-expr の判定結果が真ならば cond-expr の左辺値を、偽ならば expr の値を返します。

【例】

FELSE(TBL.FLD in ('a','b','c'),'_other')

TBL.FLD が'b'の時は 'b'を

TBL.FLD が'd'の時は '_other_'を返します。

FSW (..)

【構文】

FSW(expr,(expr1)val1[(expr2)val2...],default-expr)

それぞれ任意の式を指定してください。

【説明】

expr が expr1 と一致すれば val1 を(expr2 と一致すれば val2 を...)何れとも一致しなければ default-expr の値を返します。

【例】

FSW(MONTH,(1)'Jan'(2)'Feb'(3)'Mar'(4)'Apr'(5)'May'(6)'Jun'

(7)'Jul'(8)'Aug'(9)'Sep'(10)'Oct'(11)'Nov','Dec')

2.1.14. サブクエリー指定について

サブクエリーを使用することにより別のクエリーの結果を式に使用可能です(サブクエリーを記述する場合の構文は「2.11 sub query 句の指定」を参照してください)。なお、Flatfile Maintenance のデータ集約では使用できません。

サブクエリーの展開指定(SUBQ n)が使用可能な文脈は限られています。まず、

```
expr =|<>|<|>|<=|>= (SUBQ n)
```

```
expr [NOT] IN (SUBQ n)
```

の2つの文脈において使用可能であり、

上の文脈においてはサブクエリーの結果を単一の値に展開します(複数行が抽出された場合は先頭行の値を使用)。

下の文脈においてはサブクエリーの結果を値のリストに展開します。また、PIVOTCOL のソート順指定オプションの括弧内においても使用可能です(後述の「2.2.3 PIVOT オプション指定」を参照してください)。

(サブクエリー指定の使用例については「2.11 sub query 句の指定」を参照してください)

2.1.15. 特殊変数(ES/1 NEO CS シリーズのみ有効)について

#{TOPNUM}

ES/1 NEO CS のメイン画面にある、「環境」メニュー - 「動作環境設定(共通)」- 「グラフ、資源ログ等に表示する項目を制限」の指定をクエリー登録時に反映させます。「すべて出力する」が選択されている場合は、この部分を削除した形でクエリーが実行されます。「集約数で制限する」が選択されている場合は TOP + 指定された集約数でこの部分を置換した形で実行されます。「出力要素数の制限定義を使用する」が選択されている場合は TOP + 出力要素数の制限定義で指定された集約数でこの部分を置換した形で実行されます。

【例】

```
...order by AVG(TBL.FLD) #{TOPNUM}
(後述の order by 句の説明を参照してください)
```

#{SUMUNIT}

ES/1 NEO CS のメイン画面にある、「環境」メニュー- 「動作環境設定(共通)」- 「時間による集約」の指定をクエリー登録時に反映させます。「集約しない」が選択されている場合は 1 で、「指定時間で集約する」が選択されている場合は指定された分の値で、この部分を置換してクエリーが実行されます。

【例】

```
select FMTT(2,ROUNDT(TIME,#{SUMUNIT}))
...group by ROUNDT(TIME,#{SUMUNIT})...
```

#{CPUNUM}

ES/1 NEO CS のメイン画面にある、「環境」メニュー- 「動作環境設定(共通)」- 「CPU 搭載数」の指定をクエリー登録時に反映させます。設定されたサイト/システムの CPU 搭載数でこの部分を置換してクエリーが実行されます(単一のシステムを対象としたクエリーでのみ有効です)。

`${MEMSZ}`

ES/1 NEO CS のメイン画面にある、「環境」メニュー-「動作環境設定(共通)」-「物理メモリ量/スワップメモリ量」-「サイズ指定」の「物理メモリ量」の指定をクエリー登録時に反映させます。設定されたサイト/システムの物理メモリ量でこの部分を置換してクエリーが実行されます。

`${SWPSZ}`

ES/1 NEO CS のメイン画面にある、「環境」メニュー-「動作環境設定(共通)」-「物理メモリ量/スワップメモリ量」-「サイズ指定」の「スワップメモリ量」の指定をクエリー登録時に反映させます。設定されたサイト/システムのスワップメモリ量でこの部分を置換してクエリーが実行されます。

`${ORDB}`, `${ORDBS}`

クエリー登録時に選択したインスタンスプロファイルの「Oracle ドメイン名」「Oracle データベース名」の指定を、クエリー実行時に反映させます。

(このキーワードの使用例については「2.5 filter 句の指定(ES/1 NEO CS シリーズのみ有効)」を参照してください)。

`${SFDB}`, `${SFDBS}`

クエリー登録時に選択したインスタンスプロファイルの「Symfoware RDB システム名」の指定を、クエリー実行時に反映させます。

(このキーワードの使用例については「2.5 filter 句の指定(ES/1 NEO CS シリーズのみ有効)」を参照してください)。

`${SQ8I}`, `${SQ8IS}`

クエリー登録時に選択したインスタンスプロファイルの「SQL Server インスタンス名」の指定を、クエリー実行時に反映させます。

(このキーワードの使用例については「2.5 filter 句の指定(ES/1 NEO CS シリーズのみ有効)」を参照してください)。

`${R3ID}`, `${R3IDS}`

クエリー登録時に選択したインスタンスプロファイルの「SAP ERP インスタンス名」の指定を、クエリー実行時に反映させます。

(このキーワードの使用例については「2.5 filter 句の指定(ES/1 NEO CS シリーズのみ有効)」を参照してください)。

`${UDBDB}`

クエリー登録時に選択したインスタンスプロファイルの「DB2 ノード名」「DB2 データベース名」の指定を、クエリー実行時に反映させます。

(このキーワードの使用例については「2.5 filter 句の指定(ES/1 NEO CS シリーズのみ有効)」を参照してください)。

なお、上記の特殊変数はすべて Flatfile Maintenance のデータ集約では使用できません。

2.2. select 句の指定

■ select 句の構文

```
select selected-expr[,selected-expr ...]
```

■ selected-expr の構文

[PIVOTROW + PIVOTCOL] 式 [AS 指定] [PIVOT オプション指定]

select 句には抽出対象の式を指定します。

2.2.1. AS 指定

抽出結果の列名を指定します。AS の後ろには出力したい列名をシングルクォートで囲って('')指定します。

```
select DATE, AVG(CPU.USE)... (CPU.USE は仮想のテーブル/フィールド名です)
```

の結果は、

DATE	AVG (CPU.USE)	
20020101	9.99	...
...	...	

となりますが、

```
select DATE as '日付', AVG(CPU.USE) as '平均 CPU 使用率'...
```

の結果は、

日付	平均CPU使用率	
20020101	9.99	...
...	...	

となります。

AS 指定は出力結果のカラム行(1 行目)の形式にのみ影響します。

select 句の式がカンマ(,)を含み、結果を表計算アプリケーション等で取り込む場合は、AS 指定の使用を検討してください。カンマ(,)がフィールドセパレータとして認識される場合があるからです。

2.2.2. PIVOTROW、PIVOTCOL 指定

PIVOT 操作(基本動作の解説を参照してください)を行います。PIVOTROW/PIVOTCOL の各キーワードは抽出対象式の前に指定してください。PIVOTROW/PIVOTCOL 指定の無い式の抽出結果の列名は、PIVOTCOL 指定された式の値 + その式の列名、となります。なお、Flatfile Maintenance のデータ集約では使用できません。

PIVOTROW 指定された式、PIVOTCOL 指定された式、無指定の式の何れかの数が 0 の場合は PIVOT 操作を行わずに処理を実行します。

2.2.3. PIVOT オプション指定

PIVOTCOL 指定された式には列への展開時のソート順を指定するオプションを指定可能です。なお、Flatfile Maintenance のデータ集約では使用できません。

以下の何れかの形式が使用可能です。

- (1)PIVOTCOL 式 ORDER BY ASC | DSC
- (2)PIVOTCOL 式 ORDER BY (expr1[,expr2...]SUBQ n) ASC | DSC
- (3)PIVOTCOL 式 ORDER BY ASC | DSC (exprA[,exprB...] | SUBQ n)
- (4)PIVOTCOL 式 ORDER BY (expr1[,expr2...]SUBQ n) ASC | DSC (exprA[,exprB...] | SUBQ n)

上記の exprX は何れも定数で指定してください。

(1)の場合、式の値の昇(降)順で列への展開を行います。

(2)の場合、expr1[expr2...]の順で展開後、残りの要素を値の昇(降)順で展開します。

(3)の場合、式の値の昇(降)順で列への展開を行いますが、exprA[exprB...]の値はexprA[exprB...]の順で後ろに展開します。

(4)の場合、expr1[expr2...]の順、exprA[exprB...]に含まれない値を昇(降)順、exprA[exprB...]の順で展開します。

**(2)～(4)の括弧内はサブクエリーの展開指定(SUBQn)で記述することも可能です。
サブクエリーの結果を定数のリストとして、上記規則に従い展開を行います。**

PIVOTROW/PIVOTCOL 指定の無い式には COLBASE オプションが指定可能です。

```
select PIVOTROW r, PIVOTCOL c, v1, v2 ...
```

において、c の値が A、B、C である時、結果の出力カラムは、

```
r,Av1,Bv1,Cv1,Av2,Bv2,Cv2..
```

となります。COLBASE を使用して、

```
select PIVOTROW r, PIVOTCOL c, v1 COLBASE, v2 COLBASE ...
```

とすると、結果の出力カラムは

```
r,Av1,Av2,Bv1,Bv2,Cv1,Cv2..
```

になります。

2.3. from 句の指定

■ from 句の構文

```
from 'site-name'.'system-name'[, 'site-name'.'system-name'...]
```

site-name サイト名

system-name システム名

from 句にはデータ抽出対象となるサイト／システム名を記述します。

**MF/CS/Flatfile Maintenance のクエリー編集機能において from 句を記述する必要はありません。
from 句はクエリーの実行時に実行対象のサイト／システムに基づいて動的に設定されます。**

2.4. join 句の指定

■ join 句の構文

(1)outjoin ()

(2)join join-expr

(3)outjoin join-expr

join-expr の構文

table-name on table-name ()

|table-name on table-name(field-name = field-name[, ...])

|join-expr on table-name ()

|join-expr on table-name(field-name = field-name[, ...])

| (join-expr) on table-name ()

| (join-expr) on table-name(field-name = field-name[, ...])

join 句は複数の表からのデータ抽出におけるデータの結合方法を指定します。

(結合方法の詳細については「1.4 複数の表からのデータ抽出」を参照してください)

(1)の構文では日時による結合時に一部の表で結合対象データが欠けている組み合わせも抽出します。

(2)、(3)の構文は日時以外の結合条件を明示的に指定する場合に使用します。

(2)の構文では全ての表で結合条件を満たすデータの組み合わせを抽出します。

(3)の構文では一部の表で結合対象を満たすデータが欠けている組み合わせも抽出します。

join 句は省略可能であり、省略した場合には日時が一致するレコードの組み合わせ全てを抽出します。

Flatfile Maintenance のデータ集約では、3 テーブル以上を結合するクエリーにおいて outjoin 句を使用することはできません。

2.5. filter 句の指定 (ES/1 NEO CS シリーズのみ有効)

■ filter 句の構文

- (1)filter ORDB = 'oracle-domain-name'. 'oracle-db-name'
[, 'oracle-domain-name'. 'oracle-db-name'...]
- (2)filter SFDB = 'RDB-system-name'[, 'RDB-system-name'...]
- (3)filter SQ8I = 'instance-name'[, 'instance-name'...]
- (4)filter R3ID = 'instance-name'[, 'instance-name'...]
- (5)filter UDBDB = 'node-name', 'db-alias-name'

filter 句は Oracle・Symfoware・SQL Server 2000/2005/2008・SAP ERP・DB2 の各データを扱う際に抽出動作の対象となるデータを制限します。filter 句は省略可能です。

後述の where 句を用いてもデータを絞り込むことが可能ですが、where 句に記述した場合はデータの結合動作後に絞り込みを行うのに対し、filter 句に記述した場合は結合動作を行う前にデータを絞り込みます。この相違により、filter 句を用いたほうが join 句の記述が簡単になり、抽出動作が速くなる場合があります。以下に例を示します。

(例)

Oracle セッション情報表(表名:ORSESS)と Oracle セッションサマリ表(表名:ORSESSUM)からのデータ抽出

【Oracle セッション情報(ORSESS)】

	日付	時刻	ドメイン名 (DOMAIN)	DB名 (DBNAME)	セッションID (SESSID)	Oracleユーザ名 (ORACLEUSR)	...
(a)	20020101	0000	world	ora1	0	SYSTEM	
(b)	20020101	0000	world	ora2	0	MASTER	

【Oracle セッションサマリ(ORSESSUM)】

	日付	時刻	ドメイン名 (DOMAIN)	DB名 (DBNAME)	セッションID (SESSID)	プロセッサ使用時間 (CPUSEC)	...
(1)	20020101	0000	world	ora1	0	300	
(2)	20020101	0000	world	ora2	0	150	

①where 句を用いた場合

```
select DATE,TIME,ORSESS.DOMAIN,ORSESS.DBNAME,ORSESS.ORACLEUSR,
       ORSESSUM.CPUSEC join ORSESS on ORSESSUM
       (DOMAIN=DOMAIN,DBNAME=DBNAME,SESSID=SESSID)
       where ORSESS.DOMAIN ='world'and ORSESS.DBNAME ='ora1'...
```

この場合の動作は一度日時と join 句で指定されたフィールドが一致するすべてのレコードを結合し、

	DATE	TIME	ORSESS.DOMAIN	ORSESS.DBNAME	ORSESS.ORACLEUSR	ORSESSUM.CPUSEC
(a)+(1)	20020101	0000	world	ora1	SYSTEM	300
(b)+(1)	20020101	0000	world	ora2	MASTER	150

where 条件の判定が真となるレコードを抽出するという動作を行います。

②filter 句を用いた場合

```
select DATE,TIME,ORSESS.DOMAIN,ORSESS.DBNAME,ORSESS.ORACLEUSR,
ORSESSUM.CPUSEC join ORSESS on ORSESSUM (SESSID=SESSID)
filter ORDB = 'world'. 'ora1'...
```

結合動作を行う前に Oracle ドメイン名/データベース名による絞り込みが行われる為、結合段階で以下のようになります。

	DATE	TIME	ORSESS.DOMAIN	ORSESS.DBNAME	ORSESS.ORACLEUSR	ORSESSUM.CPUSEC
(a)+(1)	20020101	0000	world	ora1	SYSTEM	300

join 句の指定は SESSID フィールドのみで済みます。また、結合動作の対象となるレコード数が減る為、多量のデータを処理する場合の処理時間は短くなります。

③特殊変数の使用

前記「2.1.15 特殊変数(ES/1 NEO CS シリーズのみ有効)について」で挙げた特殊変数の、

```
{ORDB},{ORDBS}
{SFDB},{SFDBS}
{SQ8I},{SQ8IS}
{R3ID},{R38IS}
{UDBDB}
```

は、それぞれ filter 句に使用することが可能です。これらのキーワードはクエリーグループ登録時に指定したデータベース名やインスタンス名で置換されます。

..S の付いたキーワードは、複数を対象とする場合に使用します。

(例)

```
filter ORDB = ${ORDB}
```

とするとクエリーグループ登録時に指定した Oracle ドメイン名/データベース名が、クエリーの実行時に '\${ORDB}'の部分に、

```
filter ORDB = 'domain-a'. 'db-a' のように展開されます。
```

複数 DB を対象にしたい場合は、

```
filter ORDB = ${ORDBS} とします。
```

この場合は

```
filter ORDB = 'domain-a'. 'db-a', 'domain-b'. 'db-b'[,...] のように展開されます。
```

{SFDB},{SFDBS},{SQ8I},{SQ8IS},{R3ID},{R3IDS},{UDBDB} も同様に使用します。

```
filter SFDB = ${SFDB}
filter SFDB = ${SFDBS}
filter SQ8I = ${SQ8I}
filter SQ8I = ${SQ8IS}
filter R3ID = ${R3ID}
filter R3ID = ${R3IDS}
filter UDBDB = ${UDBDB}
```

2.6. where 句の指定

■ where 句の構文

where search-condition

search-condition には真偽値を判定する式を記述します。

where 句には抽出操作の対象となるレコードの条件を記述します。複数テーブルから抽出操作を行う場合は結合規則に基づいた結合を行った後の各レコードについて条件判定を行います。where 句は省略可能です。

(例)

```
select DATE,TIME,TBL.FLD ... where TBL.FLD >= 80 ...
```

TBL.FLD の値が 80 以上のレコードを抽出します。

2.7. when 句の指定

■ when 句の構文

when search-condition

search-condition には真偽値を判定する式を記述します。

where 句に記述する条件式のうち、日付(年・月・日)に関する条件は when 句に記述することも可能です。ただし、日付に関連しない条件式を記述するとすべてのレコードがマッチしなくなりますので注意してください。

when 句には PNDATE というキーワードを指定することも可能です。このキーワードを指定すると ES/1 NEO MF-eASSIST Performance Navigator で設定したサイト/システム毎の対象日付に該当する日付のレコードのみを抽出対象とします。

when 句は省略可能です。

(例 1)

```
... when MONTH >= 4 and MONTH <= 9 ...
```

4～9 月のデータのみ対象とします。

(例 2)PNDATE の使用例

```
... when PNDATE ...
```

PNDATE は他の条件と組み合わせて使用することも可能です。

```
... when PNDATE and WEEK = SUN ...
```

2.8. group by 句の指定

■ group by 句の構文

group by expr[EXCEPTABLE][,expr[EXCEPTABLE]...]

expr には任意の式を指定してください(但しこの式中にはグループ演算関数は使用できません)。

group by 句にはグルーピングの単位としたい式を記述します。複数の式をグルーピング単位とするには式をカンマ(,)で区切って記述します。group by 句は省略可能です。

また、group by 句の式の後ろに EXCEPTABLE キーワードを指定することも可能です。EXCEPTABLE キーワードを使用し、グループ演算関数の前に EXCEPT キーワードを使用すると EXCEPTABLE 指定をしていない式のみでグループ化した演算結果を求めることが可能です。なお、EXCEPT/EXCEPTABLE キーワードは、Flatfile Maintenance のデータ集約では使用できません。

(例 1)

```
select DATE,AVG(TBL.FLD) ... group by DATE ...
```

日付単位の TBL.FLD の平均値を求めます。

(例 2) EXCEPT/EXCEPTABLE の使用

【表(TBL)】

日付	時刻	FLD
20020101	1000	50
20020101	1030	40
20020101	1100	80
20020101	1130	90

上記表に対して、

```
select PIVOTROW DATE,PIVOTCOL HOUR,AVG(TBL.FLD) as ' 時平均 ',
       PIVOTROW EXCEPT AVG(TBL.FLD) as ' 日平均 ' from ...
group by DATE,HOUR EXCEPTABLE ...
```

を実行すると、

DATE	10時平均	11時平均	日平均
20020101	45	85	65

が結果として抽出されます。EXCEPT AVG(TBL.FLD)の値は EXCEPTABLE 指定の無い式(DATE)単位でグルーピングした場合の平均値((50+40+80+90) / 4)65 となります。

2.9. having 句の指定

■ having 句の構文

having search-condition

search-condition には真偽値を判定する式を記述します。

having 句にはグルーピングした結果の内、抽出対象となるグループの条件を記述します。having 句は省略可能です。

(例)

```
select DATE,AVG(TBL.FLD) ... group by DATE having AVG(TBL.FLD) >= 80 ...
```

日単位の TBL.FLD の平均値が 80 以上のものを対象とします。

2.10. order by 句の指定

■ order by 句の構文

```
order by expr[ASC|DSC][,expr[ASC|DSC]...][NODUP expr][TOP n][BOTTOM n]
```

各 expr には任意の式を、n には任意の正の整数を指定してください。

order by 句は抽出結果の並び順を指定します。抽出結果は指定された式の値で並び替えられたものになります。式の後ろに ASC を指定すると昇順、DSC を指定すると降順となります。ASC/DSC を省略した場合は昇順になります。複数の式を指定した場合は左側の式から優先的に判定されます。また、TOP n を指定した場合は、並べ替え後の先頭から n 行を抽出し、BOTTOM n を指定した場合は末尾から n 行を抽出します(ただし、n 行目と並べ替えの基準となる式の値が同じ行が存在する場合はその式も抽出されます)。order by 句は省略可能です。省略した場合の結果の並びは不定となります。なお、NODUP expr/TOP n/ BOTTOM n は、Flatfile Maintenance のデータ集約では使用できません。

(例 1)

```
... order by DATE, TIME
```

日時の昇順で並び替え

(例 2)

```
select DATE,AVG(TBL.FLD) ... order by AVG(TBL.FLD) DSC
```

日単位の TBL.FLD の平均値が高いもの順に並び替え

NODUP expr を指定した場合、expr に指定した式の値がより (ソート後の結果において) 上の行と一致する行は抽出されなくなります。NODUP の例を以下に示します。

【例】NODUP を使用しないクエリ

```
select pivotrow DATE,pivotrow TIME,pivotcol SYSTEM,TBL.FLD
from 'SiteA'.'SYS0','SiteA'.'SYS1','SiteA'.'SYS2' order by DATE,TIME
```

による抽出結果が、

DATE	TIME	SYS0	SYS1	SYS2
20120101
20120101	1000	78	87	93
20120101	1015	50	48	43
...
20120102
20120102	1545	90	91	88
20120102	1600	80	76	68
...

のような結果であり、SYS0、SYS1、SYS2 の TBL.FLD の合計値が最大となる行を日毎に抽出 (各日毎の合計がピークのインターバルを抽出) したい場合、NODUP を使用した以下のクエリ、

```
select pivotrow DATE,pivotrow TIME,pivotcol SYSTEM,TBL.FLD
from 'SiteA'.'SYS0','SiteA'.'SYS1','SiteA'.'SYS2'
group by DATE,TIME,SYSTEM exceptable
order by DATE,except SUM(TBL.FLD) dsc nodup DATE
```

により、

DATE	TIME	SYS0	SYS1	SYS2
20120101	1000	78	87	93
20120102	1545	90	91	88
...

の結果を得ることができます（2012/01/01 は 10:00 が、2012/01/02 は 15:45 がそれぞれピークのインターバルと仮定しています）。この結果は前ページのクエリから"nodup DATE"を除いたクエリ、

```
select pivotrow DATE,pivotrow TIME,pivotcol SYSTEM,TBL.FLD
from 'SiteA'.'SYS0','SiteA'.'SYS1','SiteA'.'SYS2'
group by DATE,TIME,SYSTEM exceptable
order by DATE,except SUM(TBL.FLD) dsc
```

により得られる以下の結果、

DATE	TIME	SYS0	SYS1	SYS2	except SUM (TBL.FLD) の値
20120101	1000	78	87	93	258
20120101
20120101	1015	50	48	43	141
...
20120102	1545	90	91	88	269
20120102
20120102	1600	80	76	68	224
...

を上から順に DATE が重複しないように取り出したものと同じです。

2.11. sub query 句の指定

■ sub query 句の構文

(sub-query-expr)[,(sub-query-expr) ...]

■ sub-query-expr の構文

select 式 [join 句][filter 句][where 句][when 句][group by 句][having 句][order by 句]

sub query 句を使用すると別のクエリー(以降の文中'サブクエリー'と表現します)の実行結果を利用することができます。sub query 句はクエリー本体の後ろに括弧で囲んで記述します。複数指定する場合はカンマ(,)で区切ります。sub query 句はクエリー本体とほぼ同様の記述が可能です。相違点は select の後に指定出来るのは 1 つの式のみであり、from 句は記述しない(本体と同様のサイト/システムを対象とする)ということです。なお、sub query 句は、Flatfile Maintenance のデータ集約では使用できません。

サブクエリーの結果をクエリー内で参照するには(SUBQ n)を使用します。n は 1 からの連番で sub query 句の記述順(最も左側にある sub query 句が 1)になります。

サブクエリーの結果を他のサブクエリーから参照することも可能です。

例えば、サブクエリー 2 の結果をサブクエリー 1 で参照し、その(サブクエリー 1 の)結果をクエリーで参照することが可能です。ただし、参照が循環するような場合はエラーになります。

(SUBQ n)は以下の構文の

expr =|<>|<|>|<=|>= (SUBQ n)

expr [NOT] IN (SUBQ n)

PIVOTCOL expr ORDER BY [(SUBQ n)] ASC|DSC [(SUBQ n)]

の何れかにおいて使用可能です。比較の演算子(=、<>、<、>、<=、>=)との組み合わせにおいてはサブクエリーの抽出結果の最上行の単一の値が参照され、IN や PIVOTCOL のソート順指定との組み合わせにおいては抽出結果のすべての行の値が参照されます。

(例 1)サブクエリーの結果の単一の値を参照

```
select DATE,TIME,TBL.FLD from ... where DATE = (SUBQ 1)
order by TIME (select DATE group by DATE order by AVG(TBL.FLD) dsc TOP 1)
```

TBL.FLD の平均が最も高かった日の TBL.FLD の値を時系列に抽出します(sub query 句で'TOP1'を指定しなくても (SUBQ1)の解釈においては最上行の値のみを使用するので結果は同一となります)。

(例 2)サブクエリーの結果の複数の値を参照

【表(TBL)】

日付	時刻	NAME	VAL
20020101	0900	AAA	9
20020101	0900	BBB	70
20020101	0900	CCC	15
20020101	0900	DDD	80
20020101	0900	EEE	20
20020101	1000	AAA	1
20020101	1000	BBB	60
20020101	1000	CCC	20
20020101	1000	DDD	90
20020101	1000	EEE	90

上記表に対して、

```
select PIVOTROW DATE,PIVOTROW TIME,PIVOTCOL TBL.NAME order by (SUBQ 1) asc,
TBL.VAL as " from ... where TBL.NAME in (SUBQ 1) order by DATE,
TIME (select TBL.NAME group by TBL.NAME order by AVG(TBL.VAL) dsc TOP 3)
```

を実行すると、

DATE	TIME	DDD	BBB	EEE
20020101	0900	80	70	20
20020101	1000	90	60	90

が抽出されます。TBL.VAL の平均が高かった上位 3 つを抽出し、where 句による絞り込みと PIVOTCOL によるカラムへの展開順の制御に使用されています。

2.12. option 句の指定

■ option 句の構文

option { loop N;}

N には 2 以上の整数を指定してください。

"option loop N"が指定されている場合、各レコード(複数の表を結合する場合は結合後のレコード)の実体が N 個あるように計算を行います。クエリーの実行時に各レコードにつき N 回の計算が繰り返され、LOOPIDX 変数にその繰り返しが何番目であるかがセットされます(1 番最初の繰り返しが 0 と数えます)。

この繰り返しは sub query 句で指定したクエリーの実行時には行われません。

(例)

【表(TBL)】

日付	時刻	VAL
20070101	0000	5
20070101	0100	6
20070101	0200	10

上記表に対して、

```
select FSW(LOOPIDX,(0)'5 未満'(1)'10 未満','10 以上')as'range',SUM(FSW(LOOPIDX,
(0)FIF(TBL.VAL < 5,1,0)(1)FIF(TBL.VAL >= 5 and TBL.VAL < 10,1,0),FIF (TBL.V
AL >= 10,1,0))) as 'count' from... group by LOOPIDX order by LOOPIDX option
{loop 3;}
```

を実行すると、

range	count
5未満	0
10未満	2
10以上	1

が抽出されます。

実行時には上記表のデータに繰り返し回数の 3 を適用し、

日付	時刻	VAL	LOOPIDXの値
20070101	0000	5	0
20070101	0000	5	1
20070101	0000	5	2
20070101	0100	6	0
20070101	0100	6	1
20070101	0100	6	2
20070101	0200	10	0
20070101	0200	10	1
20070101	0200	10	2

として計算を行います。

```
option { fill_hour on N (X to Y); }
```

N には 1 以上の整数を指定してください。

X には 0 以上 47 以下の整数を指定してください。

Y には 0 以上 47 以下の X より大きい整数を指定してください。

対象期間中の存在しない時間帯のダミーのレコードを作成します。

この指定は ES/1 NEO MF-eASSIST Performance Navigator の"group by HOUR"としているクエリーに対して行ってください。

N には HOUR を使用した select 句中の式の位置を、X と Y にはダミー作成の開始と終了の時間帯を指定してください。

(例)

```
select FMTI('%02d:00', HOUR) as 'hour', ...  
group by HOUR order by HOUR
```

上記のクエリーの結果が

10:00,...

12:00,...

13:00,...

14:00,...

15:00,...

16:00,...

であった場合、以下のように fill_hour を指定すると

```
select FMTI('%02d:00', HOUR) as 'hour', ...  
group by HOUR order by HOUR option { fill_hour on 1 (9 to 17); }
```

09:00,...*

10:00,...

11:00,...*

12:00,...

13:00,...

14:00,...

15:00,...

16:00,...

17:00,...*

という結果が出力されます。(*が作成されるダミーのレコードです。)

```
option { fill_date on N; }
```

N には 1 以上の整数を指定してください。

対象期間中の存在しない日付のダミーのレコードを作成します。

この指定は ES/1 NEO MF-eASSIST Performance Navigator の"group by DATE"、"group by DAY"、"group by MONTH"のように、日付かそれより大きい単位でグルーピングを行うクエリーに対して行ってください。

N には DATE や DAY を使用した select 句中の式の位置を指定してください。

(例)

```
select FMTD(3, DATE) as 'date', ...  
group by DATE order by DATE
```

上記のクエリーの対象期間を 2015/01/01 から 2015/01/31 とした結果が
2015/01/03,...
2015/01/05,...

(中略)

2015/01/28,...
2015/01/29,...

であった場合、以下のように fill_date を指定すると

```
select FMTD(3, DATE) as 'date', ...  
group by DATE order by DATE option { fill_date on 1; }
```

2015/01/01,...*
2015/01/02,...*
2015/01/03,...
2015/01/04,...*
2015/01/05,...

(中略)

2015/01/28,...
2015/01/29,...
2015/01/30,...*
2015/01/31,...*

という結果が出力されます。(*が作成されるダミーのレコードです。)

2.13. runpy 句の指定

■ runpy 句の構文

run_pyscr (module-name, func-name)

run_pyscr (module-name, func-name, argument)

runpy 句を使用するとクエリーの処理の最後に Python スクリプトの実行を挿入することができます。

メモ!

スクリプトはバージョン 2.3 の Python インタプリタにより実行されます。Python バージョン 2.3 より後に導入された構文や機能は使用できません。

runpy 句を指定した場合、処理は以下の順で行います。

- (1)runpy 句部分を除いた箇所の記述を基にクエリーを実行
- (2)上記 1.の結果を CSV ファイルに出力
- (3)runpy 句の指定に従い Python スクリプト(以下単に"スクリプト"と記述)を実行
- (4)(グラフ/グラフィックファイル*作成時のみ)上記 2.の CSV ファイルからグラフ/グラフィックファイルを作成 *gif や png を示します

スクリプトから上記 2.の CSV ファイルを編集することにより、最終的な出力結果の CSV ファイル、グラフ、グラフィックファイルの内容を変更することができます。

構文中の module-name にはスクリプトのモジュール名を文字列定数で指定します。スクリプトは以下のフォルダに配置し、例の場合はいずれも'myscript'を指定します。

・MF シリーズ

MF インストールフォルダ¥Pnavi¥runpyusrlib

(例) C:¥IIM¥MF¥Pnavi¥runpyusrlib¥myscript.py

・CS シリーズ

CS インストールフォルダ¥runpyusrlib

(例) C:¥IIM¥CS¥runpyusrlib¥myscript.py

メモ!

スクリプトのモジュール名の先頭 2 文字は "m_" と異なるものにして下さい。"m_" で始まるモジュール名は ES/1 が内部的に使用するモジュールに割り当てられており、使用した場合は製品の動作に支障をきたす場合があります。

func-name には module-name で指定したスクリプト中の手続き名を文字列定数で指定します。ここで指定した手続きが実行されます。

以下の例の場合は'accumulate'を指定します。

(例)

```
# coding=utf-8
def accumulate():
    """
    クエリーの実行結果を書き換える
    """
    ...
```

argument には手続きに渡す引数を文字列定数で指定します。ここでは 1 つの文字列定数のみ指定可能であり、手続き側は文字列型の値 1 つを引数として受け取るように定義します。argument を指定しない場合、手続き側は引数を受け取らないように定義します。

スクリプトから出力結果を操作するには `_runpy` モジュールをインポートして使用します。`_runpy` モジュールには以下のメソッドがあります。

`_runpy._getOutfile()`

クエリーの出力した CSV ファイル名を返します。ここで得たファイルの内容を読み込むことでクエリーの出力結果を参照できます。また、ファイルの内容を更新することで出力結果を書き換えることができます。

`_runpy._getRowCnt()`

クエリーの出力した CSV ファイル中の行数を返します。この行数にはヘッダ行の 1 行を含みます。

`_runpy._setRowCnt(rowcnt)`

クエリーの出力結果を元の行数と異なる行数に変更した場合には必ず使用してください。引数 `rowcnt` には更新した結果の行数を整数で指定します。引数の行数にはヘッダ行の 1 行を含めます。

(例)

```
# coding=utf-8
import _runpy
def update_result():
    """
    クエリーの実行結果を書き換える
    """
    filename = _runpy._getOutfile()
    # filename の内容を lines に読み込む
    lines = open(filename).readlines()
    # 内容(lines)を更新する
    ...
    # 変更した内容(lines)を filename に書き込む
    outf = open(filename, 'w')
    ...
    outf.close()
    # 更新後の行数を設定
    _runpy._setRowCnt(len(lines))
```

メモ!

出力結果の行数を変更することは(上記の `_runpy._setRowCnt` の呼び出しにより)サポートされますが、出力結果の列数を変更した場合の動作は不定です。スクリプトで列を生成したい場合はクエリー側にあらかじめ `0 as '...'` のような式を記述し、その列の結果を更新するようにしてください。

以下に runpy 句を使用したサンプルを示します。(ここに紹介しているサンプルのスクリプトは前記の module-name についての説明に準じたフォルダにインストールされています。)

(1)

このサンプルは出力結果の第一カラムに出力された日時部分の重複する日を除きます。例えば、

```
"DateTime","Cpu Use%"
2016/01/01 00:00,87
2016/01/01 01:00,56
2016/01/01 02:00,34
...
2016/01/02 00:00,12
2016/01/02 01:00,77
...
```

このような出力を以下のように編集します。

```
"DateTime","Use%"
2016/01/01 00:00,87
01:00,56
02:00,34
...
2016/01/02 00:00,12
01:00,77
...
```

クエリーは以下のように定義します。

(MF)

```
QueryHead=select FMTD(3,DATE) & ' ' & FMTT(2,TIME) as 'DateTime',
PROC.CPU_AVG as 'Cpu Use%'
QueryTail=order by DATE,TIME run_pyscr ('example1', 'cut_date')
```

(CS)

```
QueryHead=select FMTD(3,DATE) & ' ' & FMTT(2,TIME) as 'DateTime',
ATCPU.SYSUSE + ATCPU.USRUSE as 'Cpu Use%'
QueryTail=order by DATE,TIME run_pyscr ('example1', 'cut_date')
```

以下の内容のスクリプトを"example1.py"というファイル名で作成します。

```
# coding=utf-8
import _runpy
def cut_date():
    """
    日付の重複部分を除く
    以下 #で始まる行は次の行に対するコメント
    """
    filename = _runpy._getOutfile()
    L = open(filename).readlines()
    # 結果を保持する配列を作成、ヘッダ行をコピー
    newL = list(L[0])
    # 前の日付を保持する変数
    svdate = None
    # データ行(2行目以降)を処理
    for l in L[1:]:
        # 行の日付を取得
        d = l[:11]
        # 前の日付と比較する
        if d != svdate:
            # 前の日付を更新、行はそのまま
            svdate = d
        else:
            # 行の日付部分を除く
            l = l[12:]
        # 結果を保持する配列へ追加
        newL.append(l)
    # 結果をファイルへ書き込み
    open(filename, 'w').writelines(newL)
```

(2)

このサンプルは各インターバルの値をそれまでの累積値に書き換えます。例えば、

```
"Time","2016/01/01","2016/01/02"  
00:00,12,7  
01:00,9,3  
02:00,33,12  
...
```

このような出力を以下のように編集します。

```
"Time","2016/01/01","2016/01/02"  
00:00,12,7  
01:00,21,10  
02:00,54,22  
...
```

(01:00 の値は 00:00 と 01:00 の値の和、02:00 の値は 00:00、01:00、02:00 の値の和、...)

クエリーは以下のように定義します。

(MF)

```
QueryHead=select PIVOTROW FMTT(2,TIME) as 'Time',PIVOTCOL  
FMTD(3,DATE), PROC.CPU_AVG as "  
QueryTail=order by TIME run_pyscr('example2', 'accumulate')
```

(CS)

```
QueryHead=select PIVOTROW FMTT(2,TIME) as 'Time',PIVOTCOL  
FMTD(3,DATE),ATCPU.USRUSE+ATCPU.SYSUSE as "  
QueryTail=order by TIME run_pyscr('example2', 'accumulate')
```

以下の内容のスクリプトを"example2.py"というファイル名で作成します。

```
# coding=utf-8
import _runpy
def accumulate():
    """
    累積値に書き換える
    以下 #で始まる行は次の行に対するコメント
    """
    filename = _runpy._getOutfile()
    L = open(filename).readlines()
    # 結果を保持する配列を作成、ヘッダ行をコピー
    newL = list(L[0])
    # データ系列数を取得
    colcnt = len(L[0].split(',')) - 1
    # 累積値保持用の配列を用意
    accums = [0.0] * colcnt
    # データ行(2行目以降)を処理
    for l in L[1:]:
        # 末尾の改行を除去
        l = l.rstrip()
        # 各フィールドを配列に分割
        flds = l.split(',')
        # 各フィールドについて繰り返し
        for i in range(colcnt):
            # 欠損値(空文字列)ではない時に累積値の処理を行う
            if flds[i + 1]:
                # フィールドは文字列型なので float で数値化して計算する
                accums[i] += float(flds[i + 1])
                # 累積値を文字列型にして戻す
                flds[i + 1] = str(accums[i])
        # 編集したフィールドを結合して行の形式に戻す
        l = ','.join(flds) + '\n'
        # 結果を保持する配列へ追加
        newL.append(l)
    # 結果をファイルへ書き込み
    open(filename, 'w').writelines(newL)
```

第3章 マクロ

3.1. マクロの使用方法

“マクロ”とはクエリ文中に現れる特殊な形式の記述であり、あらかじめ定義された内容でクエリ文中で展開される要素です。

【例】

以下のクエリ文

```
select `datetime` as 'datetime' ...
```

において、「`datetime`」はマクロであり、その展開形式が

```
FMTD(3, DATE) & ' ' & TIME(2, TIME)
```

と定義されている時、上記のクエリ文はと同じ

```
select FMTD(3, DATE) & ' ' & TIME(2, TIME) as 'datetime' ...
```

意味を示します。

マクロはクエリ文中で、バッククォート+1 文字の英字+0 文字以上の英数字/アンダースコアの組み合わせによって記述します。
(英大文字小文字の区別はありません)

マクロの定義はテキストファイルに記述します。そのテキストファイルは

- ・MF シリーズの場合 Pnavi インストールフォルダ ¥__macro¥
- ・CS シリーズの場合 CS インストールフォルダ ¥__macro¥

の各フォルダに存在します。

それぞれのフォルダには、iimstd_macro.txt と custom_macro.txt というテキストファイルがあり、

- ・iimstd_macro.txt には ES/1 が標準で提供するマクロ
- ・custom_macro.txt には導入環境毎にカスタマイズして使用するマクロ

をそれぞれ定義します。

iimstd_macro.txt は ES/1 をインストールする際に上書きされるので、カスタマイズして使用するマクロは custom_macro.txt に定義して下さい。また、iimstd_macro.txt は CS シリーズにのみ標準提供クエリでよく使用されている内容が定義されています。

マクロを定義する場合は上記テキストファイルに、

```
マクロ名=展開形式
```

と記述します。マクロ名にはクエリ文中に記述するマクロから先頭のバッククォートを除いたもの（英大文字小文字の区別なし）を、展開形式にはマクロがクエリ文中で展開される内容をそれぞれ記述します。複数のマクロを定義する場合は行毎に上記内容を記述します。また、英字以外で始まる行はコメント行とみなします。

【例】

```
; my macro  
datetime=FMTD(3, DATE) & ' ' & TIME(2, TIME)  
weekday_cond=WEEK in (MON, TUE, WED, THU, FRI)
```

局所的なマクロをクエリ文に記述することも可能です。

```
マクロ名=展開形式
```

とするとクエリ文(に限り)で局所マクロ定義に記述したマクロを使用することが可能です。局所マクロ定義は以下の形式で記述します。

```
マクロ名=展開形式
```

【例】

```
@ | datetime="FMTD(3, DATE) & ' ' & TIME(2, TIME)" | @select `datetime ...
```

3.2. マクロ使用についての注意点

マクロを使用する場合は以下の点に注意してください。

同名のマクロが複数個所 (iimstd_macro.txt, custom_macro.txt, 局所マクロ) で定義されている場合の優先順位は、

局所マクロ > custom_macro.txt > iimstd_macro.txt

とします。

ES/1 が標準で提供するマクロは“es1_”というプレフィックスで始まります。custom_macro.txt や局所マクロで“es1_”で始まるマクロを定義することは避けてください。

マクロを定義するテキストファイルの 1 行の最大長は 4096 バイトとします。この制限はコメント行にも適用されます。

局所マクロ定義のマクロ名と展開形式の最大長はいずれも 256 バイトとします。

マクロのネスト (マクロの展開形式中にマクロを使用) はできません。

展開形式中に#{...}、\${...}のような特殊変数を使用することはできません。