

# Db2 Warehouse on Cloud モニタリング

はじめに

**このSILは、Db2 Warehouse on Cloud (以下 Db2 WoC)で、稼働状況の監視やパフォーマンスモニタリングを行う方法をガイドするものです。**

Db2 WoC は クラウドマネージドサービスであるため、オンプレミスのDb2とは異なり、データベースサーバーにログインしてOSコマンドや Db2システム・コマンドを使用できません。

Db2のインスタンスオーナーIDを使用することはできません。  
このため、稼働状況監視やパフォーマンスモニタリングに関して、従来のオンプレミス環境とは異なる方法を使う必要がある場合があります。

このSILでは、Db2 WoCで使用できるコマンドやSQLを使っでの監視やモニタリングを紹介しています。

このSILで紹介しているモニタリングツールや SQL関数は、オンプレでも使用可能であるため、これらをそのままオンプレミスのDb2で実行することは可能です。

# 目次

## 1. どうやってモニタリングする？

- 1) モニタリングの選択肢
- 2) Webコンソールを使おう
- 3) Db2クライアントを使おう

## 2. パフォーマンスに問題がありますか？

- 1) いいえ、問題はありません。=>「3.稼働状況の監視」
- 2) はい、問題があります。=>「4. こんな時、どうする」

## 3. 稼働状況の監視

## 4. こんな時、どうする？

1. どうやってモニタリングする？

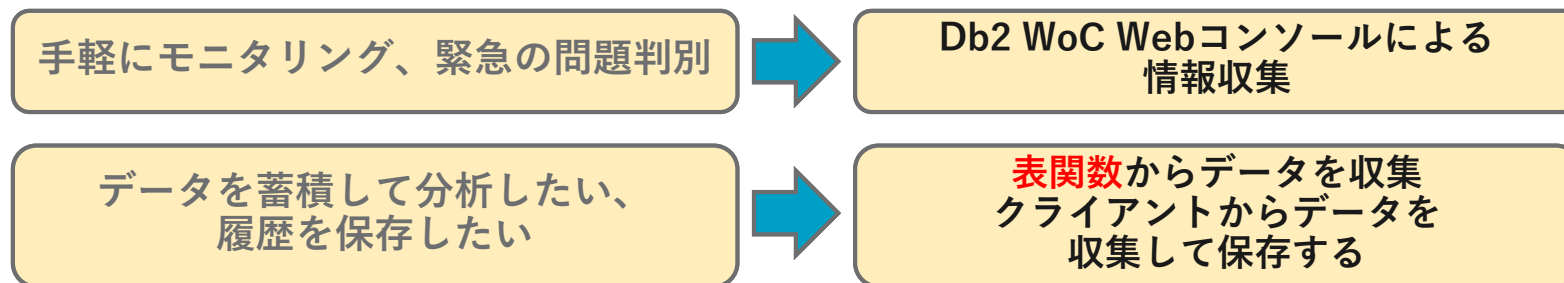


# モニタリングの選択肢

## 適切なモニタリング方法を使おう

Db2 Warehouse が提供する多様なモニタリング機能のすべてをこの資料で網羅することは困難であるため、当資料は「SQLによる取得と分析が可能なMON\_GET表関数」でのモニタリングに重点を置いています。他のモニタリング機能については逆引きトピックの観点から、特に有用性が高い使い方を必要に応じて紹介する形を取っています。

特定のモニタリング機能について重点的に学ぶ場合は、下記の選択肢から適切なモニタリング機能を選び、参考資料を参照してください。



参考資料

[ザ・技術SIL「IBM Db2 Data Management Console 利用ガイド」](#)



## 1.1) Webコンソールを使おう

# Webコンソールにアクセスしよう

**Db2 WoC の Web コンソールでは、さまざまな モニタリング機能を使用することができます。**

ここでは 以下項目について 解説します。

- WebコンソールのURL
- Webコンソール画面
- Webコンソールを使うときに
  - 履歴の蓄積

# WebコンソールのURL

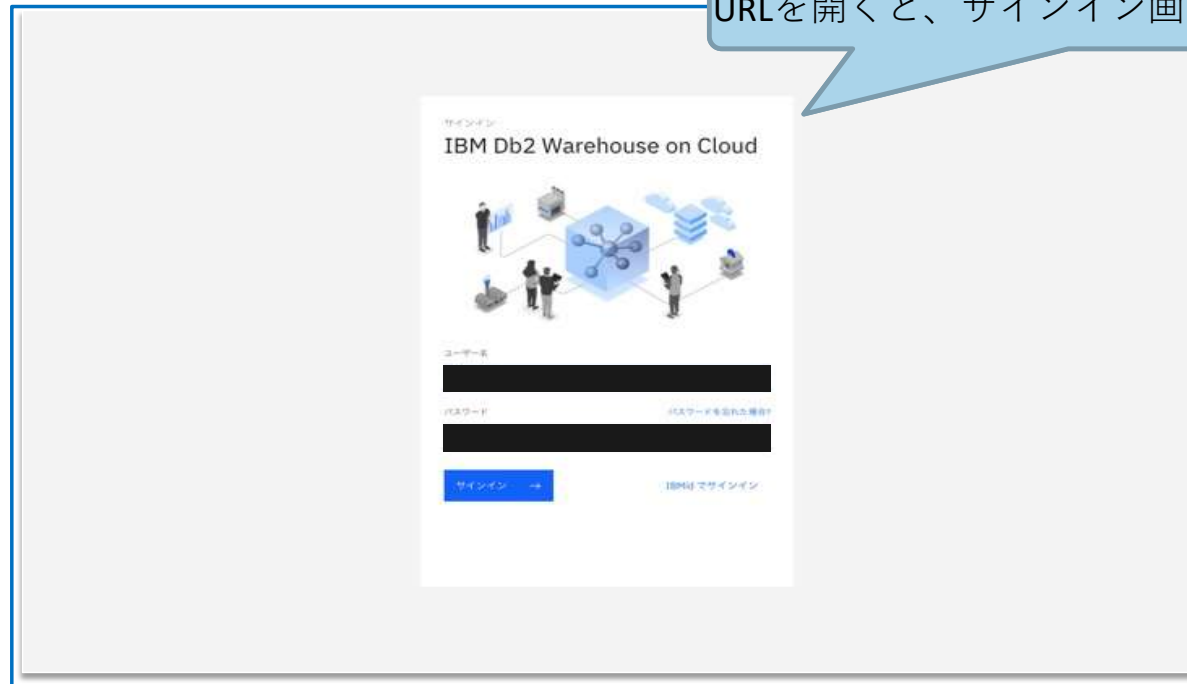
Webコンソールへは、Webブラウザから以下URLで接続することができます。

<https://db2w-xxxxx.ap-north.db2w.cloud.ibm.com/console/>

※赤文字部分はホスト名。

- Db2 Warehouse on Cloud(Db2 WoC) Webコンソールは、Db2 WoCのモニタリングおよび管理を行うためのGUIインタフェースです。
  - Db2 WoCサービスをデプロイすると、コンソールのアドレスとWebコンソールにログインするための認証情報が通知されます。
  - Webコンソールでは、ユーザー登録や、バックアップスケジュールの設定などの運用管理作業も行えますが、本資料では、モニタリング機能に焦点をあてて紹介します。

URLを開くと、サインイン画面が表示される



# Webコンソール画面

画面左上のメニューボタンをクリックすると、メニューが表示されます。

当資料では [ モニター ] メニューにおけるモニタリング機能について解説します。

The screenshot displays the IBM Db2 Warehouse on Cloud Web Console. On the left, a sidebar menu is visible with the 'モニター' (Monitor) option highlighted in a red box. A blue arrow points from this menu item to the 'ダッシュボード' (Dashboard) section of the main interface. The dashboard shows various performance metrics, including '反応性' (Latency) with a value of 1.66k, 'スループット' (Throughput), and 'リソース使用量' (Resource Usage) with CPU at 16% and Memory at 51.33%.

メニュー

- **モニター** . . . **モニタリング情報 確認**
- SQLの実行 . . . **SQL ステートメントの実行**
- データ . . . **データのロード、オブジェクトの操作・管理**
- 管理 . . . **Db2 WoC の管理**

# Webコンソール画面

## - ダッシュボード

[モニター] > [ダッシュボード]タブにて、全体のモニタリング情報が確認できます。



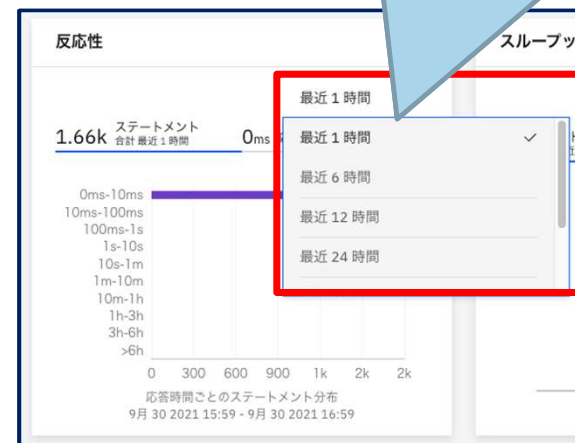
以下期間のモニタリング可能

- 最近1時間
- 最近6時間
- 最近12時間
- 最近24時間
- 最近3日
- 先週
- 先月

※サマリーでは特定の時刻を指定することはできない  
最近1時間より以前の情報を見るためには、後述の「履歴の蓄積」を行うことが必要

以下項目について表示

- 反応性
- スループット
- リソース使用量
- 競合(ロック待ち)
- 消費時間(SQL実行)



# Webコンソール画面

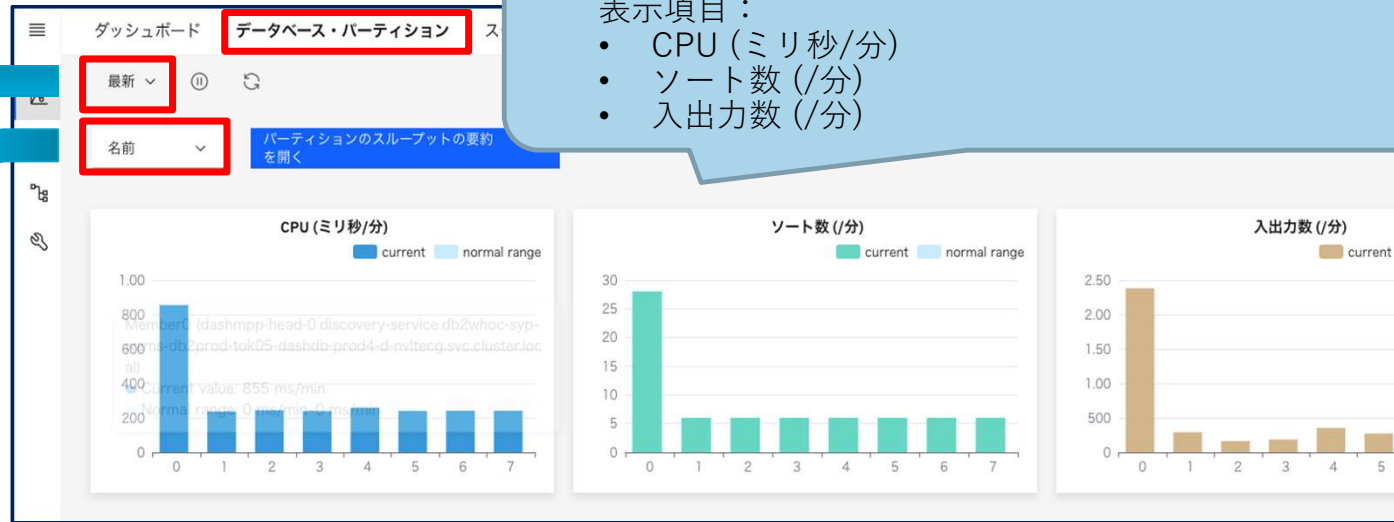
## - データベース・パーティション

[モニター]>[データベース・パーティション]タブにて、  
各データベース・パーティションのCPU, ソート数, 入出力の状態を確認できます。

最新  
最新1時間  
最新6時間  
最新12時間  
最新24時間  
最近3日  
先週  
先月  
カスタム  
Start  
click to select  
End  
click to select  
Close OK

パーティション (データベース・メンバー) 別にグラフ表示  
表示項目:

- CPU (ミリ秒/分)
- ソート数 (/分)
- 入出力数 (/分)



モニタリングする  
期間指定が可能

- 最近1時間
- 最近6時間\*
- 最近12時間\*
- 最近24時間\*
- 最近3日\*
- 先週\*
- 先月\*
- カスタム\*

\*要「履歴の蓄積」

名前

名前

CPU

ソート

I/O

グラフ横軸の表示順を、以下項目でソート可能

- 名前 ...データベース・パーティション名
- CPU ...CPU(ミリ秒/分)
- ソート ...ソート数(/分)
- I/O ...入出力数(/分)

# Webコンソール画面

## - ステートメント (1/3)

[モニター]>[ステートメント]タブにて、ステートメントの実行状況を確認できます。

以下観点でそれぞれのスループットに関するモニタリングが可能です。

- [処理中の実行]タブ
- [パッケージ・キャッシュ]タブ
- [個々の実行]タブ
- [ストアード・プロシージャ]タブ

各タブで様々な観点でステートメントに関する情報がモニタリング可能

The screenshot shows the IBM Db2 Warehouse on Cloud console. The 'Statements' tab is selected. A red box highlights the '処理中の実行' (Executing) sub-tab. Below it, a dropdown menu is set to '最近1時間' (Last 1 hour). A table header is visible with columns: 'クライアント IP アドレス', 'アプリケーション名', 'ユーザー ID', '開始時刻', 'コーディネーター・ステートメント実行時間', 'アクティビティ状態', 'SQL', and 'WLM キュー時間'. A red box highlights the 'WLM キュー時間' column header. A blue arrow points from this box to a '列のカスタマイズ' (Customize Columns) dialog box. This dialog shows a list of columns with checkboxes: 'クライアント IP アドレス', 'アプリケーション名', 'ユーザー ID', '開始時刻', 'コーディネーター・ステートメント実行時間', 'アクティビティ状態', and 'SQL', all of which are checked. At the bottom of the dialog are 'キャンセル' (Cancel) and '保存' (Save) buttons. Another blue arrow points from the '最近1時間' dropdown to a list of monitoring periods: '最新', '最近1時間', '最近6時間', '最近12時間', '最近24時間', '最近3日', '先週', '先月', and 'カスタム'. A blue arrow points from the 'カスタム' option to a 'モニタリングする期間の指定可能' (Monitoring period can be specified) dialog. This dialog lists the following options: '最近1時間', '最近6時間\*', '最近12時間\*', '最近24時間\*', '最近3日\*', '先週\*', '先月\*', and 'カスタム\*'. A note at the bottom says '\*要「履歴の蓄積」' (Requires 'History Accumulation').

モニタリングする期間の指定可能

- 最近1時間
  - 最近6時間\*
  - 最近12時間\*
  - 最近24時間\*
  - 最近3日\*
  - 先週\*
  - 先月\*
  - カスタム\*
- \*要「履歴の蓄積」

[列の表示/非表示] ボタンで表示する列のカスタマイズが可能



# Webコンソール画面

## - ステートメント (2/3)

[モニター] > [ステートメント] タブの

[処理中の実行] タブでは 現在 実行されているSQLステートメント、

[パッケージ・キャッシュ] タブでは パッケージ・キャッシュ内のSQLステートメントに関するメトリックのモニタリングができます。

- [処理中の実行] タブ

実行されている SQLステートメントを確認可能  
SQL以外のユーティリティによる処理も表示される

クライアント IP アドレス	アプリケーション名	ユーザー ID	開始時刻	コーディネーター・ステートメント実行時間	アクティビティ状態	SQL	WLM キュー時間
172.30.146.128	db2bp	OGAWA	2021-12-08 15:46:50	0:02:32.356	EXECUTING	<a href="#">update t1 set c1=? where c1=?</a>	0.000

- [パッケージ・キャッシュ] タブ

パッケージ・キャッシュ内の  
SQLステートメントに関する情報が確認できる

SQL	実行数	ステートメント実行時間 (ミリ秒)	CPU 時間 (ミリ秒)	読み取り行数	戻された行数
<a href="#">WITH T(C1, C2, C3, C4, C5, C6, C7, C8, C9, C10) AS (VALUES (CAS...</a>	2	0.000	0:00.004	0	0

# Webコンソール画面

## - ステートメント (3/3)

[モニター] > [ステートメント] タブの  
[個々の実行] タブでは 過去に実行されたSQLステートメントのメトリック、  
[ストアード・プロシージャ] タブでは 実行されたルーチンに関する情報  
を確認することができます。

- [個々の実行] タブ

過去に実行されたSQLステートメントのメトリックが確認できる

SQL	ステートメントの実行時間	CPU 時間	ワークロードのキュー時間	クライアント IP アドレス	アプリケーション名	セッション許可 ID	ワークロード名
-----	--------------	--------	--------------	----------------	-----------	------------	---------

- [ストアード・プロシージャ] タブ

呼び出されたルーチンが一覧表示される

ルーチン・スキーマ	ルーチン名	呼び出された回数	論理読み取り数/分	物理読み取り数/分	ルーチン ID	読み取り数/分	読み取り行数/分	返された行数
AUDIT	LOAD	4,088	0.00	0.00	68,315	0.00	0.00	0.1

# Webコンソール画面

## - ロック (1/3)

[モニター]>[ロック]タブにて、ロックに関する情報を確認できます。

以下観点でそれぞれのロックに関するモニタリングが可能です。

- [ブロックしている接続と待機している接続]タブ
- [接続の統計]タブ
- [ロックされたオブジェクトと待機している接続]タブ
- [ロックされたオブジェクトの検索]タブ
- [ロック・イベント・モニター]タブ

各タブで様々な観点で  
ロックに関する情報がモニタリング可能



モニタリングする 期間指定が可能

- 最近1時間
  - 最近6時間\*
  - 最近12時間\*
  - 最近24時間\*
  - 最近3日\*
  - 先週\*
  - 先月\*
  - カスタム\*
- \*要「履歴の蓄積」

[列の表示/非表示] ボタンで  
表示する列のカスタマイズが可能

# Webコンソール画面 - ロック (2/3)

[モニター]>[ロック]タブの  
[ブロックしている接続と待機している接続]ではロックによってブロック待機している接続、  
[接続の統計]タブでは各アプリケーションのロックに関する統計を確認できます。

- [ブロックしている接続と待機している接続]タブ

ロックを保持、  
またはロック待機している  
アプリケーションを確認可能

タイプ	アプリケーション・ハンドル	アプリケーション名	ユーザーID	ロック待機時間	保持されているロックの数	ロック・モード	イベントのタイム・スタンプ	待機してい
BLOCKER	4730	db2bp	OGAWAZ	0:02:46.737	8	U	2021-12-08 15:38:33	:

- [接続の統計]タブ

アプリケーションごとの  
接続やロックに関する統計が確認できる

アプリケーション名	アプリケーション・ハンドル	接続開始時刻	CPU 時間/分	使用メモリー・プール (KB)	アイドル時間	デッドロック検出数/分	保留ロック数
db2bp	4867	2021-11-22 15:15:25	0.000	4,800.00	0:29.637	0	:

# Webコンソール画面

## - ロック (3/3)

[モニター] > [ロック]タブの  
[ロックされたオブジェクトと待機している接続] / [ロックされたオブジェクトの検索]タブでは  
ロックされているオブジェクトや、ロック待機の接続数・待機時間などの情報  
を確認できます。

- [ロックされたオブジェクトと待機している接続]タブ

The screenshot shows the 'Locks' tab in the IBM Db2 Warehouse on Cloud console. The 'Locks' tab is selected, and the sub-tab 'Locks on objects and waiting connections' is active. A table displays the following data:

アプリケーション・ハンドル	アプリケーション名	ユーザーID	オブジェクト・タイプ	表スキーマ	表名	ロック名	待機に費やされた時間	待機している接続の数	データ・バー
4730	db2bp	OGAWA2	ROW	OGAWA	T1	02009B00 00000000 00000000 58	0:02:29.000	1	:

ロックされている  
オブジェクトを確認可能

ロック待機している接続数や  
待機時間も確認可能

- [ロックされたオブジェクトの検索]タブ

The screenshot shows the 'Locks' tab in the IBM Db2 Warehouse on Cloud console. The 'Locks' tab is selected, and the sub-tab 'Search for locked objects' is active. A table displays the following data:

アプリケーション・ハンドル	アプリケーション名	セッション許可ID	オブジェクト・タイプ	表スキーマ	表名	ロック名	待機している接続の数	待機
458811	db2fw4	DB2INST1	TABLE	IBM_RTMON	OSMET RICS	0A000A00 00000000 00000000 54	0	:
458811	db2fw4	DB2INST1	TABLE	IBM_RTMON	WLSTA TS	0A000700 00000000 00000000 54	0	:

ロックされているオブジェクトが確認可能

# Webコンソール画面

## - アプリケーション (1/3)

[モニター]>[アプリケーション]タブより、アプリケーションのモニタリングができます。

以下のタブ項目でスループットに関するモニタリングが可能です。

- [消費量上位者] タブ
- [接続] タブ
- [作業単位 (UOW) ] タブ
- [ユーティリティー・イベント・モニター] タブ

各タブで様々な観点でアプリケーションに関する情報がモニタリング可能

ダッシュボード データベース・パーティション ステートメント ロック **アプリケーション** スループット パフォーマンス ストレージ

消費量上位者 接続 作業単位 (UOW) ユーティリティー・イベント・モニター

最終収集: 2021-09-30 17

アプリケーション・ハンドルまたはアプリケーション名の検索

サーバー・リソース	アプリケーション・ハンドル	アプリケーション名	ランク	最上位の UOW ID	最上位の UOW の値	最上位の UOW を除く平均	セッション許
Estimated SQL cost	63639	UC_MYMON	2	1	89,758 timerons	--	DSADM

列のカスタマイズ

列の検索 デフォルトに戻す

- サーバー・リソース
- アプリケーション・ハンドル
- アプリケーション名
- ランク
- 最上位の UOW ID
- 最上位の UOW の値
- 最上位の UOW を除く平均

キャンセル 保存

[列の表示/非表示] ボタンで表示する列のカスタマイズが可能

最新

- 最新
- 最近 1 時間
- 最近 6 時間
- 最近 12 時間
- 最近 24 時間
- 先週
- 先月
- カスタム

Start  
click to select

End  
click to select

Close OK

[接続][ユーティリティー・イベント・モニター] タブでは、モニタリングする期間指定が可能

- 最近1時間
- 最近6時間\*
- 最近12時間\*
- 最近24時間\*
- 最近3日\*
- 先週\*
- 先月\*
- カスタム\*

\*要「履歴の蓄積」

# Webコンソール画面

## - アプリケーション (2/3)

[モニター] > [アプリケーション] タブの [消費量上位者] タブにて、  
リソースを多く消費しているアプリケーションに関する情報を確認することができます。

ダッシュボード データベース・パーティション ステートメント ロック アプリケーション スループット バッファ・プール ストレージ

消費量上位者 接続 作業単位 (UOW) ユーティリティ・イベント・モニター

リソースを多く消費しているアプリケーションが一覧表示される

サーバー・リソース	アプリケーション・ハンドル	アプリケーション名	ランク	最上位の UOW ID	最上位の UOW の値	最上位の UOW を除く平均	セッション許可 ID	
Memory	38312	UC_REPO_dba	3		1,984 KB	320 KB	DSADM	C Li ⋮
CPU time	23454	UC_MYMON	1		--	--	DSADM	C Li ⋮
Estimated SQL cost	23454	UC_MYMON	1	1	74,799 timerons	--	DSADM	C Li ⋮
Num locks held	23454	UC_MYMON	1	1	1	--	DSADM	C Li ⋮
Longest UOW	4877	UC_REPO_run	2	8878	4d:22:55:11.33 2	01:08:11.749	DSADM	C Li ⋮
Remote table wait time			--		0	0		⋮
Log space used			--		0	0		⋮
Rows written			--		0	0		⋮
Remote table rows read			--		0	0		⋮
Lock wait time			--		0	0		⋮

# Webコンソール画面

## - アプリケーション (3/3)

[モニター]>[アプリケーション]タブの  
[接続] タブでは接続されている アプリケーション、  
[作業単位 (UOW)]タブでは 作業単位 (UOW) に関する情報  
が確認できます。

- [接続]タブ

接続されているアプリケーションが一覧表示される

アプリケーション名	クライアント・ユーザー ID	クライアント IP アドレス	アプリケーション・ハンドル	接続開始時刻	ワークロード名	サービス・スーパークラス名
db2bp	DB2INST1		4867	2021-11-22 15:15:25	SYSDEFAULTADM WORKLOAD	SYSDEFAULTUSERCLASS

- [作業単位 (UOW)]タブ

各アプリケーションの作業単位 (UOW) の情報が確認できる

アプリケーション・ハンドル	アプリケーション名	UOW ID	UOW 状態	完了したアクティビティ数	異常終了したアクティビティ数	拒否されたアクティビティ数
4867	db2bp	20679 0	UOWWAIT	0	0	( :)



# Webコンソール画面

## - スループット (1/3)

[モニター]>[スループット]タブより、スループットのモニタリングができます。

以下タブ項目でそれぞれのスループットに関するモニタリングが可能です。

- [接続の要約]タブ
- [パーティションの要約]タブ
- [WLMワークロードの要約]タブ
- [WLMサービス・クラスの要約]タブ
- [費やされたオペレーティング・システム時間]タブ
- [パーティションのスキュー]タブ

各タブで様々な観点でスループットに関する情報がモニタリング可能

アプリケーション名	アプリケーション・ハンドル	CPU時間/分	完了アクティビティ数/分	読み取り行数/分	変更行数/分	戻された行数/分	論理読み取り数/分	データベースからの
db2bp		30	0:00.019	18	534	14,329,09	0	0

列のカスタマイズ

列の検索  デフォルトに戻す

- アプリケーション名
- アプリケーション・ハンドル
- CPU時間/分
- 完了アクティビティ数/分
- 読み取り行数/分
- 変更行数/分
- 戻された行数/分
- データベースからの

キャンセル 保存

モニタリングする 期間指定が可能

- 最近1時間
  - 最近6時間\*
  - 最近12時間\*
  - 最近24時間\*
  - 最近3日\*
  - 先週\*
  - 先月\*
  - カスタム\*
- \*要「履歴の蓄積」

[列の表示/非表示] ボタンで表示する列のカスタマイズが可能

# Webコンソール画面

## - スループット(2/3)

[モニター] > [スループット] タブの

[接続の要約] タブでは アプリケーション ごとのスループット、

[パーティションの要約] タブでは データベースのメンバー (パーティション) ごとのスループット  
がモニタリングができます。

- [接続の要約] タブ

IBM Db2 Warehouse on Cloud

ダッシュボード データベース・パーティション ステートメント ロック アプリケーション スループット

接続の要約

パーティションの要約 WLM ワークロードの要約 WLM サービス・クラスの要約 費やされたオペレータ

最新

アプリケーション・ハンドルとアプリケーション名の検索

アプリケーション名	アプリケーション・ハンドル	CPU 時間/分	完了アクティビティ数/分	読み取り行数/分	変更行数/分	戻された行数/分	論理読み取り数
DS_ConnMgt_	73	0.000	0	0	0	0	:
DS_ConnMgt_	74	0.000	0	0	0	0	:

アプリケーションごとの  
スループットに関する情報が確認可能

- [パーティションの要約] タブ

IBM Db2 Warehouse on Cloud

ダッシュボード データベース・パーティション ステートメント ロック アプリケーション スループット バッファ・プール ストレージ

接続の要約

パーティションの要約

WLM ワークロードの要約 WLM サービス・クラスの要約

最新

データベース・メンバーの検索

データベース・メンバー	CPU 時間/分	完了したアクティビティ数/分	読み取り行数/分	変更行数/分	戻された行数/分	論理読み取り数/分	直接読み取り/分
7	0:00.055	0.00	105.00	117.00	0.00	98.00	0.00
6	0:00.055	0.00	132.00	137.00	0.00	113.00	0.00

データベースのメンバー (パーティション) ごとの  
スループットに関する情報が確認可能

# Webコンソール画面

## - スループット (3/3)

[モニター] > [スループット] タブの

[WLMワークロードの要約] タブでは ワークロードごとのスループット、

[WLM サービス・クラスの要約] タブでは サービス・クラスごとのスループットのモニタリングができます。

- [WLMワークロードの要約] タブ

ワークロード名	合計 CPU 時間/分	完了したアクティビティ数/分	読み取り行数/分	変更行数/分	戻された行数/分	論理読み取り数/分	直接読み取り/分
SYSDEFAULTUSERCLAS	0:00.049	4	36	23	18	34	0
SYSDEFAULTADM							

- [WLMサービス・クラスの要約] タブ

サービス・クラス名	サービス・サブクラス名	合計 CPU 時間/分	完了したアクティビティ数/分	読み取り行数/分	変更行数/分	戻された行数/分	論理読み取
SYSDEFAULTUSERCLA	SYSLOADSUBCLASS	0.000	0	0	0	0	
SYSDEFAULTUSERCLA							

# Webコンソール画面

## - バッファークール

[モニター]> [バッファークール]タブより、  
バッファークールのモニタリングができます。

バッファークールごとに  
ヒット率、読み取り数、書き込み数等を確認可能

バッファークール	ヒット率	論理読み取り数/分	物理読み取り数/分	ページ数	セルフ・チューニングが有効	書き込み数/分	直接読み取り/分	直接書き込み/分
IBMDEFAULTBP	100.00%	6,645	0	327,349	NO	0	1,779	1,33
CONSOLEPOOL	0.00%	0	0	1,000	YES	0	0	
BP4CONSOLE	100.00%	8	0	1,000	YES	0	0	

モニタリングする 期間指定が可能

- 最近1時間
- 最近6時間\*
- 最近12時間\*
- 最近24時間\*
- 最近3日\*
- 先週\*
- 先月\*
- カスタム\*

\*要「履歴の蓄積」

[列の表示/非表示] ボタンで  
表示する列のカスタマイズが可能

# Webコンソール画面

## - ストレージ (1/3)

[モニター]>[ストレージ]タブより、ストレージのモニタリングができます。

以下タブにて、それぞれストレージのモニタリングが可能です。

- ・ [表のパフォーマンス]タブ . . . 各テーブルのパフォーマンスに関する情報が表示される
- ・ [ストレージ]タブ . . . 各テーブルの物理/論理サイズに関する情報が表示される

各タブでストレージに関するモニタリングが可能

ダッシュボード データベース・パーティション ステートメント ロック アプリケーション スループット バッファ・プール **ストレージ**

表のパフォーマンス ストレージ

最新

最終収集: 2021-11-15 15:17:29

フィルター基準: 表 スキーマまたは表名の検索

スキーマ名	表名	表タイプ	表スペース名	表スキャン数/分	読み取り行数/分	アクセス/分
AUDIT	EXECUTE	USER_TABLE	TS1	0	0	0

列のカスタマイズ

列の検索 デフォルトに戻す

- スキーマ名
- 表名
- 表タイプ
- 表スペース名
- 表スキャン数/分
- 読み取り行数/分
- 挿入行数/分

キャンセル 保存

モニタリングする 期間指定が可能

- ・ 最近1時間
- ・ 最近6時間\*
- ・ 最近12時間\*
- ・ 最近24時間\*
- ・ 最近3日\*
- ・ 先週\*
- ・ 先月\*
- ・ カスタム\*

\*要「履歴の蓄積」

[列の表示/非表示] ボタンで表示する列のカスタマイズが可能

最新

最新

最近1時間

最近6時間

最近12時間

最近24時間

最近3日

先週

先月

カスタム

Start  
click to select

End  
click to select

Close OK

# Webコンソール画面

## - ストレージ (2/3)

[モニター]> [ストレージ]タブの [表のパフォーマンス] タブでは、  
表/スキーマ ごとの パフォーマンスに関するモニタリングができます。

スキーマ名	表名	表タイプ	表スペース名	表スキャン数/分	読み取り行数/分	アクセス/分
TEST3Q21	DTBL01	USER_TABLE	USERSPACE1	0	0	0
TEST3Q21	COMMENTTBL01	USER.				
SUMI	MIKOMI	USER.				

フィルター基準: 表 ^ Q スキーマまたは表名の検索

スキーマ名 表 ✓

<4867><D

スキーマ

一覧の表示を 表/スキーマ ごとに切り替えることができる

スキーマ名	表スキャン数/分	読み取り行数/分	アクセス/分
<4867><DB2INST1>	0	0	0
ASN	0	0	0
AUDIT	0	0	0
PL_DB_ADM	0	0	0

表/スキーマ ごとに対するアクセス数や読み取り行数などの情報が一覧表示される

# Webコンソール画面

## - ストレージ(3/3)

[モニター]>[ストレージ]タブの[ストレージ]タブでは、  
表/スキーマごとの物理/論理サイズに関するモニタリングができます。

表/スキーマごとの論理・物理サイズを確認することが可能

スキーマ名	表名	論理サイズ (KB)	物理サイズ (KB)	収集時間
セールス	担当者	1,920	1,920	2021-09-29 09:35:24
XYINGCDL	DrugInPB			
XYINGCDL	DRUG1nM50			
XYINGCDL	DRUG1nL60			
XYINGCDL	DRUG1nL45			

フィルター基準: 表 ^ スキーマまたは表名の検索

スキーマ名	論理サイズ (KB)	物理サイズ (KB)	収集時間
ASN	1,024	1,024	2021-11-30 17:27:06
AUDIT	85,415,648	85,418,112	2021-11-30 17:27:06
BI_DB_ADM	45,312	47,104	2021-11-30 17:27:06
BLUADMIN	6,144	6,144	2021-11-30 17:27:06

一覧の表示を 表/スキーマ ごとに切り替えることができる

# Webコンソールを使うときに (1/2)

Webコンソールでモニタリングをする際には、履歴モニターを“ON”に設定しておく必要があります。

[管理] > [設定] タブの [モニター・プロファイル] タブにて、履歴モニターのON/OFFを設定できます。

IBM Db2 Warehouse on Cloud

コンピュートとストレージ バックアップ ワークロード 許可 ユーザーの管理 接続 **設定** 鍵の管理 レプリケーション

モニター・プロファイル イベント・モニター・プロファイル レプリケーション

コレクション  
履歴とリアルタイムのデータの収集間隔

データ収集間隔 ①  
5 minutes

**履歴モニター** ①  On  
時間の経過に伴うシステムのパフォーマンスをモニターします。

履歴データの合計: 0 Bytes

保存ポリシー

履歴データ:	0 Bytes	履歴照会統計データ:	0 Bytes
平均日次データ使用量:	0 Bytes	平均日次データ使用量:	0 Bytes

モニター・データを保持する期間  
7 days (1 week)

照会統計を保持する期間  
7 days (1 week)

リセット **保存**

履歴モニターの ON/OFF を設定する

設定が完了したら [保存] をクリックする



# Webコンソールを使うときに (2/2)

Db2 WoCでは 4つのイベント・モニターの設定を行います。

## <イベント・モニター>

- Activity (アクティビティ・イベント・モニター) … [モニター]>[ステートメント]>[個々の実行]のモニタリングに必要。
- Locking (ロック・イベント・モニター) … [モニター]>[ロック]>[ロック・イベント・モニター] のモニタリングに必要。
- Utility (ユーティリティ・イベント・モニター) … [モニター]>[アプリケーション]>[ユーティリティ・イベント・モニター]のモニタリングに必要。
- Statistics (統計イベント・モニター) … [モニター]>[ダッシュボード] の [反応性] のモニタリングに必要。

[管理] > [設定]タブの [イベント・モニター・プロファイル] タブにて、各イベント・モニターの有効化/無効化が可能です。

The screenshot shows the IBM Db2 Warehouse on Cloud console. The '設定' (Settings) tab is selected in the top navigation bar. Under 'イベント・モニター・プロファイル' (Event Monitor Profiles), the 'RTMON\_EVMON\_STATS' monitor is configured. The 'Statistics' tab is active, and the '状況' (Status) is set to '使用可能' (Enabled). The '保存' (Save) button is highlighted in blue. A callout box points to the '保存' button with the text '設定が完了したら [保存]をクリックする' (Click [Save] when settings are complete). Another callout box points to the 'Statistics' tab with the text 'イベント・モニター Activity, Locking, Utility, Statistics それぞれの有効化/無効化を設定する' (Set the activation/deactivation for each event monitor: Activity, Locking, Utility, Statistics).

## 1.2) Db2クライアントを使おう

# Db2クライアントのセットアップとDb2WoCへの接続

## **Db2 クライアントを使用した Db2 WoC への接続方法を紹介します。**

以下手順で、Db2 クライアントのセットアップと Db2WoC への接続を実施します。

- Db2 クライアントの導入
- SSL通信用証明書 の準備
- Db2 クライアントの構成
- Db2 クライアントからDb2WoCへの接続

各手順の詳細は、次ページ以降 を ご参照ください。

# Db2クライアントの導入

Db2 Warehouse on Cloud データベースに接続するためのクライアントモジュールを入手します。

ドライバー・パッケージは、Webコンソールの [管理]メニュー > [接続]タブ から取得でき、Linux, Mac, PowerLinux, Windows用が用意されています。  
パッケージには、JDBC, ODBC, CLI等のプロトコルのドライバーが含まれています。  
Windowsの場合は、ドライバー・パッケージではなく、IBM Data Server Clientを導入することも可能です。

IBM Db2 Warehouse on Cloud

コンピュートとストレージ バックアップ ワークロード 許可 ユーザーの管理 **接続** 設定 鍵の管理 レプリケーション

アプリとクライアントを IBM Db2 Warehouse on Cloud に接続

Linux Mac PowerLinux Windows

千原

ダウンロード

1.5.tar.gz (70 MB)

インストール

3. dsdriver ディレクトリーで次のコマンドを実行して、Java ドライバーと ODBC/CLI ドライバーを抽出します。

```
./installDSDriver
```

installDSDriver コマンドにより、dsdriver ディレクトリーに db2profile スクリプト・ファイルと db2cshrc スクリプト・ファイルが作成されます。

4. ご使用のシェル環境に合わせてスクリプト・ファイルを実行します。

Bash シェルまたは Korn シェル:

```
../db2profile
```

C シェル:

```
../db2cshrc
```

接続構成

SSL を使用する

ホスト名: [REDACTED]  
ポート番号: 50001  
データベース名: BLUDB  
ユーザー ID: bluadmin  
バージョン: Db2 と互換性のあるバージョン (5.0 以降)

SSL 証明書のダウンロード

JDBC ストリング

```
jdbc:db2://[REDACTED]:50001/BLUDB:user=bluadmin; password=cyouz_password;ssl[REDACTED]on=ttue;
```

詳細情報

Db2 ドライバー・パッケージ (IBM Knowledge Center)

**IBM Data Server クライアント・パッケージ**

Db2 データベースへの CLPPlus の接続 (IBM Knowledge Center)

Linux, Mac, PowerLinux, Windows それぞれのクライアント導入手順が確認できる

[IBM Data Server クライアント・パッケージ] をクリックすると、ドライバー・パッケージのダウンロードサイトへ移動する



# Db2クライアントの構成

クライアント・ツールから Db2 Warehouse on Cloud に SSL接続するには、クライアント環境を構成する必要があります。

クライアント

接続にはクライアントに  
ドライバー・パッケージが必要

IBM Data Server ドライバー

ドライバー構成ファイル

db2dsdriver.cfg

ドライバー構成ファイル  
db2dsdriver.cfg には、データベースに関する項目を追加する  
(次ページにて 追加方法 記載)

Db2 WoC

Db2 Warehouse

ネットワークがファイアウォールの背後にある場合、

- ・ 標準プロトコル : ポート番号 50000
- ・ SSL 接続 : ポート番号 50001

で通信が許可されている必要がある

## Db2クライアントからDb2WoCへの接続 (1/2)

**Db2WoC に接続するため、db2dsdriver.cfgドライバー構成ファイルに接続先 データベースに関するエントリーを追加します。**

以下のようにdb2cliコマンドを使用すると、db2dsdriver.cfgドライバー構成ファイルにエントリー追加ができます。

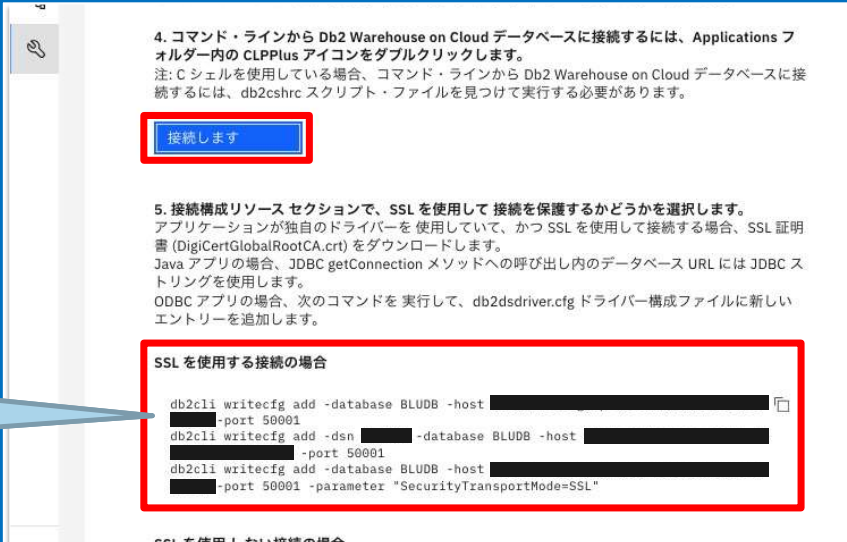
```
db2cli writecfg add -database BLUDB -host <hostname> -port 50001
```

```
db2cli writecfg add -dsn <alias> -database BLUDB -host <hostname> -port 50001
```

```
db2cli writecfg add -database BLUDB -host <hostname> -port 50001 -parameter "SecurityTransportMode=SSL"
```

- <hostname> : サーバーのホスト名
- <alias> : 選択した別名  
※データベース名「BLUDB」と同じ名前にすることはできない。

Webコンソールの [管理]メニュー > [接続]タブでも、手順の [接続します] をクリックすると、上記コマンドが確認できる



4. コマンド・ラインから Db2 Warehouse on Cloud データベースに接続するには、Applications フォルダー内の CLPPlus アイコンをダブルクリックします。  
注: C シェルを使用している場合、コマンド・ラインから Db2 Warehouse on Cloud データベースに接続するには、db2cshrc スクリプト・ファイルを見つけて実行する必要があります。

**接続します**

5. 接続構成リソース セクションで、SSL を使用して 接続を保護するかどうかを選択します。  
アプリケーションが独自のドライバーを使用していて、かつ SSL を使用して接続する場合、SSL 証明書 (DigiCertGlobalRootCA.crt) をダウンロードします。  
Java アプリの場合、JDBC getConnection メソッドへの呼び出し内のデータベース URL には JDBC ストリングを使用します。  
ODBC アプリの場合、次のコマンドを実行して、db2dsdriver.cfg ドライバー構成ファイルに新しいエントリーを追加します。

**SSL を使用する接続の場合**

```
db2cli writecfg add -database BLUDB -host [redacted] -port 50001
db2cli writecfg add -dsn [redacted] -database BLUDB -host [redacted] -port 50001
db2cli writecfg add -database BLUDB -host [redacted] -port 50001 -parameter "SecurityTransportMode=SSL"
```

SSL を使用しない接続の場合

## Db2クライアントからDb2WoCへの接続 (2/2)

コマンド・プロンプトから **db2cli validate** コマンドを発行して、接続テストが可能です。

接続テストには、以下コマンドを実行します。

```
db2cli validate -dsn <alias> -connect -user <userid> -passwd <password>
```

- <alias> : 前手順で作成した別名
- <userid> : Db2WoC ユーザー ID
- <password> : Db2WoC ユーザー・パスワード

実行結果 例 (Mac)

```
=====  
Client information for the current copy:  
=====
```

```
Client Package Type      : IBM Data Server Driver Package  
Client Version (level/bit): DB2 v11.1.3.3 (s1807091300/64-bit)  
Client Platform         : Darwin  
Install/Instance Path   : /Applications/dsdriver  
DB2DSDRIVER_CFG_PATH value: <not-set>  
db2dsdriver.cfg Path     : /Applications/dsdriver/cfg/db2dsdriver.cfg  
DB2CLIINIPATH value     : <not-set>  
db2cli.ini Path         : /Applications/dsdriver/cfg/db2cli.ini  
db2diag.log Path       : /Applications/dsdriver/db2dump/db2diag.log
```

```
~~~(省略)~~~
```

```
=====  
Connection attempt for data source name "dashdb":  
=====
```

```
[SUCCESS]
```

```
=====  
The validation is completed.  
=====
```

接続が成功したら、  
[SUCCESS] が表示される



## Db2 クライアントからのコマンド/SQL実行 (1/2)

**Db2クライアントから コマンド・SQLを実行するには、データベースへの接続して CLPPlus を開始します。**

以下コマンドを実行すると、

db2dsdriver.cfgドライバー構成ファイルのエントリーを使用して データベースへ接続し、CLPPlus が開始されます。

・Linuxの場合：

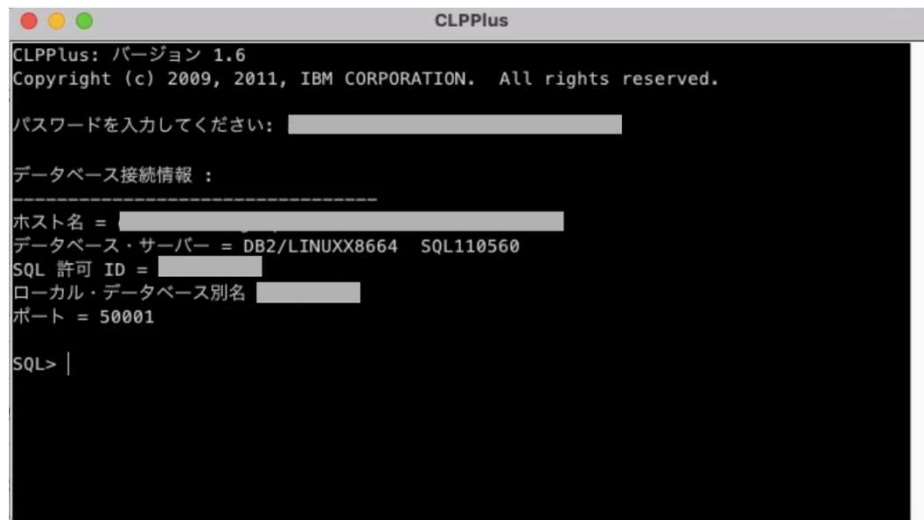
```
clpplus -nw <userid>@<alias>
```

・Windows/Macの場合：

```
clpplus <userid>@<alias>
```

- ・ <userid> : Db2WoCユーザーID
- ・ <alias> : 設定したDB別名

CLPPlus ウィンドウで Db2WoC ユーザー・パスワードを入力すると、データベース情報が表示され 接続されます。



```
CLPPlus
CLPPlus: バージョン 1.6
Copyright (c) 2009, 2011, IBM CORPORATION. All rights reserved.

パスワードを入力してください: ██████████

データベース接続情報 :
-----
ホスト名 = ██████████
データベース・サーバー = DB2/LINUX8664 SQL110560
SQL 許可 ID = ██████████
ローカル・データベース別名 ██████████
ポート = 50001

SQL> |
```

## Db2 クライアントからのコマンド/SQL実行 (2/2)

データベースへの接続が完了したら、  
SQL プロンプトでコマンド/SQL を実行することができます。

- ・ SQLステートメント実行

```
SQL> SELECT * FROM SAMP.CUSTOMER;
```

CUST_ID	NAME	GENDER
50	田中	F
30	山田	M
10	鈴木	F
20	林	M
40	古川	M

- ・ ローカルPC (ここではMac) 上のSQLファイル (/Users/ogawa/sample.sql) 実行

```
SQL> start /Users/ogawa/sample.sql
```

CUST_ID	NAME	GENDER
20	林	M
40	古川	M
10	鈴木	F
50	田中	F
30	山田	M

## 2. パフォーマンスに問題がありますか？

パフォーマンスに問題がありますか？

- いいえ、問題はありません。 => [「3.稼働状況の監視」](#)を行っておきましょう。
- はい、問題があります。 => [「4. こんな時、どうする」](#)を見て、問題判別してみましょう。

### 3. 稼働状況の監視

# 目次

稼働状況の監視	目次
---------	----

No.	Title
1	Db2 Warehouseの稼働環境を調べる
2	Db2 Warehouseの構成情報を調べる(DBM構成パラメータ、DB構成パラメータ)
3	ディスクの構成を知る(表スペース、トランザクションログパス)
4	日常的に収集すべき基本モニタリング項目
4-1	CPU使用率
4-2	メモリ使用率
4-3	メモリー使用量(メモリーの種類ごと)
4-4	ストレージ使用状況
4-5	トランザクションログ領域の使用率
4-6	スループット
4-7	レスポンスタイム
4-8	エラーログ(db2diag.log)の確認

3.稼働状況の監視	1. Db2 Warehouseの稼働環境を調べる
概要	Db2 Warehouse 稼働環境は、ENV_GET_SYSTEM_RESOURCES 表関数で取得することができます。
結果の例	次ページの結果サンプルを参照してください。
詳細情報のリンク	[Db2 WoC] ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す <a href="https://www.ibm.com/docs/ja/db2woc?topic=erv-env-get-system-resources-table-function-return-system-information">https://www.ibm.com/docs/ja/db2woc?topic=erv-env-get-system-resources-table-function-return-system-information</a>
備考	

## サンプル

## Db2 Warehouse稼働環境を調べる (CPUコア数、メモリサイズ、DBパーティション数)

ENV\_GET\_SYSTEM\_RESOURCES 表関数は、システム上のメンバーに関連した情報を戻します。

データベースには 1 つ以上のメンバー (DBパーティション) が存在することがあります。

NV\_GET\_SYSTEM\_RESOURCES 表関数は、メンバーごとの環境情報を表示します。

ここでは、メンバーごとの ホスト名 (HOST\_NAME)、CPU数 (CPU\_TOTAL)、メモリー量 (MEMORY\_TOTAL) を取得します。**CPU数、メモリー量はサーバーの合計となります。**

## ステートメント

```
SELECT
  MEMBER,
  VARCHAR(HOST_NAME,12) AS HOST_NAME,
  CPU_TOTAL,
  MEMORY_TOTAL
FROM TABLE(SYSPROC.ENV_GET_SYSTEM_RESOURCES())
ORDER BY MEMBER;
```

### ■各項目の説明

- MEMBER : データベースのメンバーID (パーティション番号)
- HOST\_NAME : ホスト名
- CPU\_TOTAL : 合計CPU数
- MEMORY\_TOTAL : 合計メモリー数

## 出力結果

メンバーごとの  
環境情報が表示される

MEMBER	HOST_NAME	CPU_TOTAL	MEMORY_TOTAL
0	dashmpp-head	40	386662
1	dashmpp-head	40	386662
2	dashmpp-head	40	386662
3	dashmpp-head	40	386662
4	dashmpp-head	40	386662
5	dashmpp-head	40	386662
6	dashmpp-head	40	386662
7	dashmpp-head	40	386662

各メンバーのホスト名、CPU数、メモリー数が返される

この例は、サーバー1台(ホスト名dashmpp-head)に、8つのデータパーティション(メンバー)が稼働している構成である  
CPUとメモリサイズは、メンバーに対する割り当てではなく、サーバー全体での合計となる

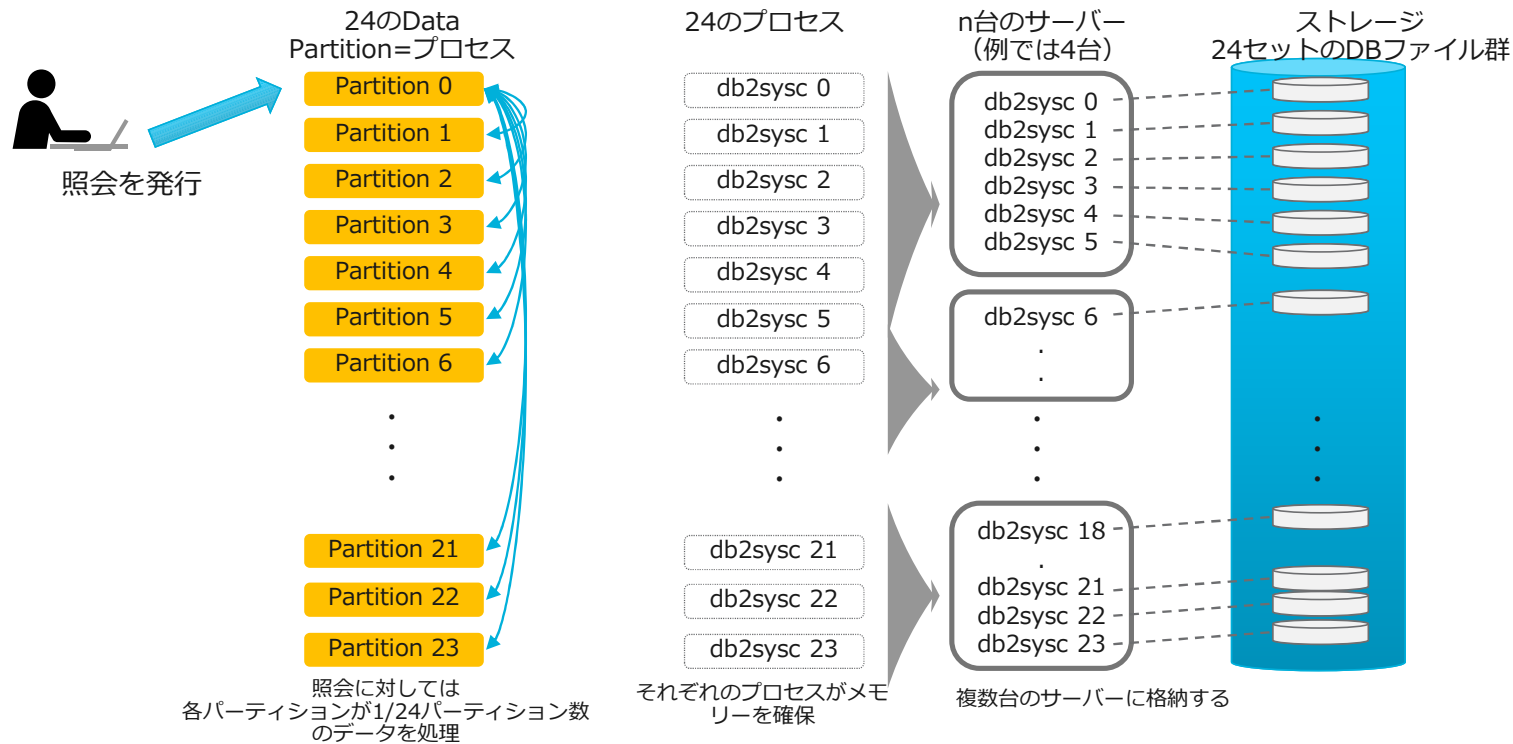


# 参考：Db2 WoCの構成

- Db2 WoCは、複数パーティションで並列にデータを処理するデータベースです。
  - パーティションの数は、Db2 WoCのプランによって異なります。

プラン	Flex One	Flex	Flex Performance
パーティション	1	マルチ	マルチ
ストレージ	40GB~4TB)	960GB~96TB	2.4TB~96TB
仮想VPC	6~28	16 ~160	48~576

- 下の絵は、24パーティションのDb2 WoC構成を表しています。
  - それぞれのパーティションは、割り当てられたデータを処理します。データは重ならないので、シェアード・ナッシングと呼ばれます。
  - パーティションを増やすことによって、処理能力を増強することができます。



3.稼働状況の監視	2. DB2 Warehouseの構成情報を調べる (DB構成パラメーター・DBM構成パラメーター)
概要	DB構成パラメーター/DBM構成パラメーターは、 <code>get db cfg / get dbm cfg</code> コマンドで取得することができます。
結果の例	次ページの結果サンプルを参照してください。
詳細情報のリンク	[Db2 Warehouse] 構成パラメーターによる Db2 データベース・マネージャーの構成 <a href="https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-configuring-db2-database-manager-configuration">https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-configuring-db2-database-manager-configuration</a>
備考	

## サンプル 1

## DB構成パラメーター・DBM構成パラメーターを確認する

get db cfg / get dbm cfg で取得される値は、現在の設定値です。  
現在の値と再活動化に適用される値を知りたい場合は、show detail オプションをつけます。

- DB構成パラメーター 確認

```
$ db2 get db cfg for <DB名>
```

```
Database Configuration for Database IBMWOC
```

```
Database configuration release level      = 0x1500
Database release level                   = 0x1500

Database territory                        = US
Database code page                        = 1208
Database code set                         = UTF-8
Database country/region code             = 1
Database collating sequence              = IDENTITY
Alternate collating sequence              (ALT_COLLATE) =
Number compatibility                      = OFF
Varchar2 compatibility                   = OFF
Date compatibility                        = OFF
Database page size                        = 32768
...
```

- DBM構成パラメーター確認

```
$ db2 get dbm cfg
```

```
Database Manager Configuration
```

```
Node type = Client
```

```
Database manager configuration release level      = 0x1400
```

```
Federated Database System Support      (FEDERATED) = NO
Transaction processor monitor name      (TP_MON_NAME) =
```

```
Default charge-back account      (DFT_ACCOUNT_STR) =
```

```
...
```

3.稼働状況の監視	3. ディスクの構成を知る(表スペース、トランザクションログパス)
-----------	-----------------------------------

概要	
<p>表スペースの情報は、MON_GET_TABLESPACE表関数で取得することができます。 トランザクションログパスの情報は、DB構成パラメーターで取得できます。</p>	
結果の例	
<p>次ページの結果サンプルを参照してください。</p>	
詳細情報のリンク	
<p>[Db2 WoC] MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=functions-mon-get-tablespace-get-table-space-metrics">https://www.ibm.com/docs/ja/db2woc?topic=functions-mon-get-tablespace-get-table-space-metrics</a></p>	
<p>[Db2 Warehouse] 構成パラメーターによる Db2 データベース・マネージャーの構成  <a href="https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-configuring-db2-database-manager-configuration">https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-configuring-db2-database-manager-configuration</a>  [Db2 Warehouse] logpath - ログ・ファイルのロケーション構成パラメーター  <a href="https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-logpath-location-log-files">https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-logpath-location-log-files</a>  [Db2 Warehouse] newlogpath - データベース・ログ・パスの変更構成パラメーター  <a href="https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-newlogpath-change-database-log-path">https://www.ibm.com/docs/ja/db2-warehouse?topic=parameters-newlogpath-change-database-log-path</a></p>	
備考	

## サンプル 1

## 表スペースの情報を取得する

MON\_GET\_TABLESPACE表関数を使用して表スペースの情報を取得します。  
ここでは各DBメンバーに存在する表スペースの情報をそれぞれ返しています。

この表関数はパラメーターとして、tblsp\_name（表スペース名）と member（メンバー）が指定でき、tblsp\_nameがNULL または空ストリングである場合、データベース内のすべての表スペースのメトリックが取得されます。  
また、member は-1を指定すると現行のデータベース・メンバー、-2を指定するとすべてのアクティブなデータベース・メンバーでの表スペースの情報が取得されます。  
member で NULL 値を指定すると、-1 が暗黙的に設定されます。

### ステートメント

```
SELECT
  VARCHAR(TBSP_NAME, 30) as TBSP_NAME,
  TBSP_ID, MEMBER,
  TBSP_TYPE,
  TBSP_CONTENT_TYPE,
  TBSP_STATE,
  TBSP_TOTAL_PAGES,
  TBSP_USABLE_PAGES,
  TBSP_USED_PAGES,
  TBSP_FREE_PAGES,
  TBSP_PAGE_SIZE,
  TBSP_EXTENT_SIZE,
  TBSP_PREFETCH_SIZE,
  TBSP_NUM_CONTAINERS
FROM TABLE(MON_GET_TABLESPACE('', -2)) AS t
ORDER BY TBSP_ID, MEMBER
```

#### ■各項目の説明

• TBSP_NAME	: 表スペース名
• TBSP_ID	: 表スペースID
• MEMBER	: データベース・メンバー
• TBSP_TYPE	: 表スペースのタイプ (DMS, SMS)
• TBSP_CONTENT_TYPE	: 表スペースのコンテンツ・タイプ (ANY, LARGE, SYSTEMP, USRTEMP)
• TBSP_STATE	: 表スペースの状態
• TBSP_TOTAL_PAGES	: 表スペースの合計ページ数
• TBSP_USABLE_PAGES	: 表スペースの使用可能ページ数
• TBSP_USED_PAGES	: 表スペースの使用ページ数
• TBSP_FREE_PAGES	: 表スペースのフリー・ページ数
• TBSP_PAGE_SIZE	: 表スペースのページ・サイズ
• TBSP_EXTENT_SIZE	: 表スペースのエクステント・サイズ
• TBSP_PREFETCH_SIZE	: 表スペースのプリフェッチ・サイズ
• TBSP_NUM_CONTAINERS	: 表スペースのコンテナ数 :

ここではパラメーターは

• tblsp\_name (表スペース名) : NULL  
• member (メンバー) : -2

を指定し、  
アクティブなデータベース・メンバー内のすべての表スペースの情報を取得する

サンプル 1

表スペースの情報を取得する (つづき)

出力結果

メンバー,表スペースごとの情報が表示される

TBSP_NAME	TBSP_ID	MEMBER	TBSP_TYPE	TBSP_CONTENT_TYPE	TBSP_STATE	TBSP_TOTAL_PAGES	TBSP_USABLE_PAGES	TBSP_USED_PAGES	TBSP_FREE_PAGES	TBSP_PAGE_SIZE	TBSP_EXTENT_SIZE	TBSP_PREFETCH_SIZE	TBSP_NUM_CONTAINERS
SYSCATSPACE	0	0	DMS	ANY	NORMAL	23552	23548	23044	504	32768	4	4	1
TEMPSPACE1	1	0	SMS	SYSTEMP	NORMAL	2	2	2	0	32768	4	4	0
TEMPSPACE1	1	1	SMS	SYSTEMP	NORMAL	1	1	1	0	32768	4	4	0
TEMPSPACE1	1	2	SMS	SYSTEMP	NORMAL	1	1	1	0	32768	4	4	0
TEMPSPACE1	1	3	SMS	SYSTEMP	NORMAL	1	1	1	0	32768	4	4	0
TEMPSPACE1	1	4	SMS	SYSTEMP	NORMAL	1	1	1	0	32768	4	4	0
TEMPSPACE1	1	5	SMS	SYSTEMP	NORMAL	1	1	1	0	32768	4	4	0
TEMPSPACE1	1	6	SMS	SYSTEMP	NORMAL	1	1	1	0	32768	4	4	0
TEMPSPACE1	1	7	SMS	SYSTEMP	NORMAL	1	1	1	0	32768	4	4	0
...													

メンバー,表スペースごとの

- TBSP\_TYPE
- TBSP\_CONTENT\_TYPE (SYSTEMP, USRTEMP)
- TBSP\_STATE
- TBSP\_TOTAL\_PAGES
- TBSP\_USABLE\_PAGES
- TBSP\_USED\_PAGES
- TBSP\_FREE\_PAGES
- TBSP\_PAGE\_SIZE
- TBSP\_EXTENT\_SIZE
- TBSP\_PREFETCH\_SIZE
- TBSP\_NUM\_CONTAINERS

が返される

: 表スペースのタイプ (DMS, SMS)

: 表スペースのコンテンツ・タイプ (ANY, LARGE,

: 表スペースの状態

: 表スペースの合計ページ数

: 表スペースの使用可能ページ数

: 表スペースの使用ページ数

: 表スペースのフリー・ページ数

: 表スペースのページ・サイズ

: 表スペースのエクステント・サイズ

: 表スペースのプリフェッチ・サイズ

: 表スペースのコンテナ数



## サンプル2

## トランザクションログパスを取得する

get db cfg コマンドから DB構成パラメーター を取得し、トランザクション・ログ・パスを確認することができます。

アクティブ・ログ・パスは **logpath (Path to log files)** の値を確認します。  
また アーカイブ・ロギングの場合、アーカイブ・ログ・パスは **logarchmeth\*** の値で確認します。  
※ 循環ロギングの場合は logarchmeth\* は OFFになります。

### ステートメント

```
get db cfg
```

### 出力結果

```
~~~(省略)~~~
Log file size (4KB)                (LOGFILSIZ) = 150000
Number of primary log files        (LOGPRIMARY) = 10
Number of secondary log files     (LOGSECOND) = 20
Changed path to log files         (NEWLOGPATH) =
Path to log files                  = /head/db2inst1/NODE0000/SQL00001/LOGSTREAM0000/
Overflow log path                  (OVERFLOWLOGPATH) =
Mirror log path                    (MIRRORLOGPATH) =
First active log file              = S0000518.LOG
~~~(省略)~~~
First log archive method           (LOGARCHMETH1) = DISK:/scratch/db2archive/archive_log/
Archive compression for logarchmeth1 (LOGARCHCOMPR1) = OFF
Options for logarchmeth1           (LOGARCHOPT1) =
Second log archive method          (LOGARCHMETH2) = OFF
Archive compression for logarchmeth2 (LOGARCHCOMPR2) = OFF
Options for logarchmeth2           (LOGARCHOPT2) =
Failover log archive path          (FAILARCHPATH) =
~~~(省略)~~~
```

3.稼働状況の監視	<b>4.日常的に収集すべき基本モニタリング項目</b> 4-1. CPU使用率
<b>概要</b>	<p>ホスト全体のCPU使用率は、Webコンソールから確認できます。</p> <p>また、サービス・サブクラスごとのCPU使用率は、MON_GET_SERVICE_SUBCLASS_STATS / MON_GET_SERVICE_SUPERCLASS_STATS表関数、ワークロードごとのCPU使用率は、MON_GET_WORKLOAD_STATS表関数 / MON_SAMPLE_WORKLOAD_METRICS表関数 のCPU_UTILIZATIONで取得できます。</p>
<b>結果の例</b>	<p>次ページの結果サンプルを参照してください。</p>
<b>詳細情報のリンク</b>	<p>[Db2 WoC] MON_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-service-subclass-stats-return-statistics-service-subclasses">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-service-subclass-stats-return-statistics-service-subclasses</a></p> <p>[Db2 WoC] MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラス・メトリックのサンプルの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-service-class-metrics-get-sample-service-class-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-service-class-metrics-get-sample-service-class-metrics</a></p> <p>[Db2 WoC] MON_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す  <a href="https://www.ibm.com/docs/ja/db2woc?topic=functions-mon-get-workload-stats-return-workload-statistics">https://www.ibm.com/docs/ja/db2woc?topic=functions-mon-get-workload-stats-return-workload-statistics</a></p> <p>[Db2 WoC] MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics</a></p>
<b>備考</b>	



## サンプル1

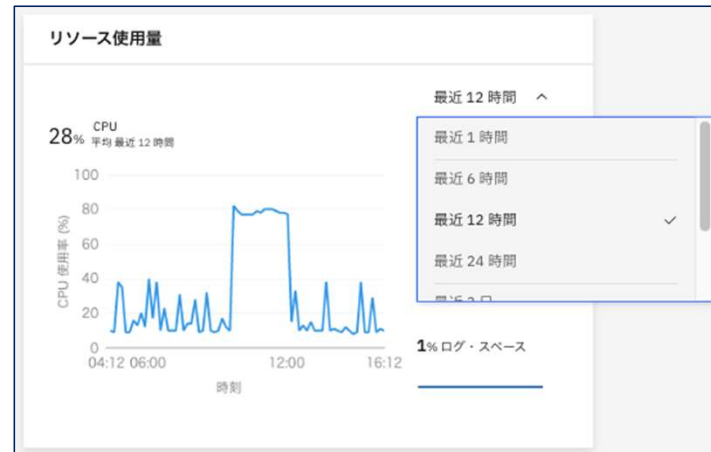
## Webコンソールから ホスト全体のCPU使用率を確認する

Webコンソールの[モニター]メニュー > [ダッシュボード]タブにて、CPU使用率が表示されています。  
期間は 最近1時間、最近6時間、最近12時間、最近24時間、最近3日、先週、先月 を 選択できます。

## 出力結果



[リソース使用量]にて、  
CPU使用率が表示されている



表示期間は以下選択可能

- 最近1時間
- 最近6時間
- 最近12時間
- 最近24時間
- 最近3日
- 先週
- 先月

## サンプル2

## サービスクラスごとのCPU使用率を取得する

MON\_GET\_SERVICE\_SUBCLASS\_STATS 関数で サービス・サブクラスごとの基本統計が取得されます。  
CPU\_UTILIZATIONから サービスクラスごとのCPU使用率を確認します。

### ステートメント

```
SELECT
  SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SERVICE_SUPERCLASS_NAME,
  SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SERVICE_SUBCLASS_NAME,
  SUBSTR(CHAR(MEMBER),1,4) AS MEMBER,
  DECIMAL(CPU_UTILIZATION,3,2) AS CPU_UTILIZATION
FROM TABLE(MON_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME, MEMBER
```

#### ■各項目の説明

- SERVICE\_SUPERCLASS\_NAME : サービス・スーパークラス名
- SERVICE\_SUBCLASS\_NAME : サービス・サブクラス名
- MEMBER : データベース・メンバー
- CPU\_UTILIZATION : CPU使用率

### 出力結果

SERVICE_SUPERCLASS_NAME	SERVICE_SUBCLASS_NAME	MEMBER	CPU_UTILIZATION
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	0	0.58
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	1	0.15
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	2	0.13
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	3	0.11
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	4	0.12
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	5	0.13
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	6	0.15
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	7	0.14
...			

サービスクラス、メンバーごとのCPU使用率が表示される

### サンプル3

### 指定した期間（秒）のサービスクラスごとのCPU使用率をリアルタイムで取得する

MON\_SAMPLE\_SERVICE\_CLASS\_METRICS表関数から、各サービス・クラスの 指定時間内（ここでは300秒）のCPU 使用率を取得します。  
指定時間を変更したい場合は 5つ目の表関数パラメーター にて 秒単位で指定します。  
CPU\_UTILIZATIONから サービスクラスごとのCPU使用率を確認します。

#### ステートメント

```
SELECT  
  SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SERVICE_SUPERCLASS_NAME,  
  SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SERVICE_SUBCLASS_NAME,  
  SUBSTR(CHAR(MEMBER),1,4) AS MEMBER,  
  DECIMAL(CPU_UTILIZATION,3,2) AS CPU_UTILIZATION  
FROM TABLE(MON_SAMPLE_SERVICE_CLASS_METRICS(NULL, NULL, '', '', 300, -2))  
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME, MEMBER
```

#### ■各項目の説明

- SERVICE\_SUPERCLASS\_NAME : サービス・スーパークラス名
- SERVICE\_SUBCLASS\_NAME : サービス・サブクラス名
- MEMBER : データベース・メンバー
- CPU\_UTILIZATION : CPU使用率

300秒間のCPU使用率を取得するように指定

#### 出力結果

SERVICE_SUPERCLASS_NAME	SERVICE_SUBCLASS_NAME	MEMBER	CPU_UTILIZATION
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	0	0.11
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	1	0.01
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	2	0.01
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	3	0.01
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	4	0.01
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	5	0.01
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	6	0.01
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	7	0.01
...			

サービスクラス、メンバーごとのCPU使用率が表示される

#### サンプル4

#### ワークロードごとのCPU使用率を取得する

MON\_GET\_WORKLOAD\_STATS 関数でワークロードの基本統計が取得されます。  
CPU\_UTILIZATIONからワークロードごとのCPU使用率を確認します。

#### ステートメント

```
SELECT  
  SUBSTR(WORKLOAD_NAME,1,18) AS WORKLOAD_NAME,  
  SUBSTR(CHAR(MEMBER),1,4) AS MEMBER,  
  DECIMAL(CPU_UTILIZATION,3,2) AS CPU_UTILIZATION  
FROM TABLE(MON_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2)) AS WLSTATS  
ORDER BY WORKLOAD_NAME, MEMBER
```

#### ■各項目の説明

- WORKLOAD\_NAME : ワークロード名
- MEMBER : データベース・メンバー
- CPU\_UTILIZATION : CPU使用率

#### 出力結果

WORKLOAD_NAME	MEMBER	CPU_UTILIZATION
CONSOLE_WORKLOAD	0	0.66
CONSOLE_WORKLOAD	1	0.07
CONSOLE_WORKLOAD	2	0.09
CONSOLE_WORKLOAD	3	0.05
CONSOLE_WORKLOAD	4	0.06
CONSOLE_WORKLOAD	5	0.06
CONSOLE_WORKLOAD	6	0.06
CONSOLE_WORKLOAD	7	0.07
...		

メンバー、ワークロードごとのCPU使用率が表示される

## サンプル5

## 指定した期間（秒）のワークロードごとのCPU使用率をリアルタイムで取得する

MON\_SAMPLE\_WORKLOAD\_METRICS 表関数から、各ワークロードの 指定時間内（ここでは300秒）のCPU 使用率を取得します。  
指定時間を変更したい場合は、4つ目の表関数パラメーター で秒単位で指定します。  
CPU\_UTILIZATIONから ワークロードごとのCPU使用率を確認します。

### ステートメント

```
SELECT  
  SUBSTR(WORKLOAD_NAME,1,18) AS WORKLOAD_NAME,  
  SUBSTR(CHAR(MEMBER),1,4) AS MEMBER,  
  DECIMAL(ACT_THROUGHPUT,3,2) AS ACT_THROUGHPUT  
FROM TABLE(MON_SAMPLE_WORKLOAD_METRICS(NULL, CURRENT SERVER, '', 300, -2)) AS T  
ORDER BY WORKLOAD_NAME, MEMBER
```

#### ■各項目の説明

- WORKLOAD\_NAME : ワークロード名
- MEMBER : データベース・メンバー
- CPU\_UTILIZATION : CPU使用率

300秒間のCPU使用率を取得するように指定

### 出力結果

WORKLOAD_NAME	MEMBER	CPU_UTILIZATION
CONSOLE_WORKLOAD	0	0.06
CONSOLE_WORKLOAD	1	0.02
CONSOLE_WORKLOAD	2	0.02
CONSOLE_WORKLOAD	3	0.02
CONSOLE_WORKLOAD	4	0.02
CONSOLE_WORKLOAD	5	0.02
CONSOLE_WORKLOAD	6	0.02
CONSOLE_WORKLOAD	7	0.02
...		

メンバー、ワークロードごとのCPU使用率が表示される

3.稼働状況の監視	4.日常的に収集すべき基本モニタリング項目 4-2. メモリ使用率
概要	
<p>全体のメモリ使用率は、Webコンソールの[Webコンソールの[モニター]メニュー &gt; [ダッシュボード]タブ から確認できます。 また、ENV_GET_SYSTEM_RESOURCES 表関数から システム上のサーバーごとの合計メモリー、メモリーの空き容量を確認することができます。</p>	
結果の例	
<p>次ページの結果サンプルを参照してください。</p>	
詳細情報のリンク	
<p>[Db2 WoC] ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す  <a href="https://www.ibm.com/docs/ja/db2woc?topic=erv-env-get-system-resources-table-function-return-system-information">https://www.ibm.com/docs/ja/db2woc?topic=erv-env-get-system-resources-table-function-return-system-information</a></p>	
備考	

## サンプル1

## Webコンソールからメモリー使用率を確認する

Webコンソールの[モニター]メニュー > [ダッシュボード]タブにて、メモリー使用率が表示されています。  
期間は最近1時間、最近6時間、最近12時間、最近24時間、最近3日、先週、先月を選択できます。

## 出力結果



[リソース使用量]にて、  
メモリー使用率が表示されている



表示期間は以下選択可能

- 最近1時間
- 最近6時間
- 最近12時間
- 最近24時間
- 最近3日
- 先週
- 先月

## サンプル2

## 合計メモリ、メモリ空き容量からメモリ使用率を算出する

ENV\_GET\_SYSTEM\_RESOURCES 表関数からシステム上のサーバーごとのメモリ情報を取得することができます。  
合計メモリ数 (MEMORY\_TOTAL)、メモリの空き容量数 (MEMORY\_FREE) を取得し、この2つの値を使用してメモリ使用率 (MEMORY\_UTILIZATION) を算出します。

### <メモリ使用率 算出式>

メモリ使用率(%) = 100 \* ((合計メモリ - メモリの空き容量数) / 合計メモリ数)

### ステートメント

```
SELECT
  VARCHAR(HOST_NAME,12) AS HOST_NAME,
  AVG(DECFLOAT(MEMORY_TOTAL)) AS MEMORY_TOTAL,
  AVG(DECFLOAT(MEMORY_FREE)) AS MEMORY_FREE,
  CASE WHEN AVG(MEMORY_TOTAL) > 0
    THEN DEC(100*(AVG(DECFLOAT(MEMORY_TOTAL-MEMORY_FREE))/AVG(DECFLOAT(MEMORY_TOTAL))),5,2)
    ELSE NULL
  END AS MEMORY_UTILIZATION
FROM TABLE(SYSPROC.ENV_GET_SYSTEM_RESOURCES())
GROUP BY HOST_NAME
```

#### ■各項目の説明

- HOST\_NAME : ホスト名
- MEMORY\_TOTAL : 合計物理メモリ
- MEMORY\_FREE : 物理メモリの空き容量
- MEMORY\_UTILIZATION : メモリ使用率  
(MEMORY\_TOTAL, MEMORY\_FREE で算出)

メンバーごとに行が返されるので、HOST\_NAME ごとにグループ化する

### 出力結果

HOST_NAME	MEMORY_TOTAL	MEMORY_FREE	MEMORY_UTILIZATION
dashmpp-head	386631	168573	56.4

算出されたメモリ使用率が表示される



3.稼働状況の監視	4.日常的に収集すべき基本モニタリング項目 4-3 メモリー使用量(メモリーの種類ごと)
概要	
	Db2の使用するメモリーの種類ごとの内訳は、MON_GET_MEMORY_POOL表関数で取得できます。
結果の例	
	次ページの結果サンプルを参照してください。
詳細情報のリンク	
	[Db2 WoC] MON_GET_MEMORY_POOL表関数 - システム情報を戻す <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-memory-pool-get-memory-pool-information">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-memory-pool-get-memory-pool-information</a>
備考	

## サンプル

## メモリーの種類ごとの使用量を取得する

メモリーの種類ごとの使用量は、MON\_GET\_MEMORY\_POOL 表関数 から取得することができます。  
表関数から メモリー・プールの使用量 (MEMORY\_POOL\_USED) と、メモリープールの最高水準点 (MEMORY\_POOL\_USED\_HWM)

ここでは、すべてのメモリー・セットについて 情報を取得するため 1つ目の表関数パラメーター に **NULL** を指定しています。  
特定のメモリー・セットを指定することも可能です。

また、現行のデータベース・メンバーでのメモリー領域に関する情報を取得するように、3つ目の表関数パラメーター に **-1** を指定しています。  
すべてのアクティブ・メンバーについての情報を取得したい場合は、**-2** を指定します。  
また、各メンバーの FCM メモリーに関して返されるメトリックは、同じ共有メモリー・セットについての情報になります。  
FCM メモリーのメトリックを調べる場合は、1つのメンバーを指定して データを調べるようにしてください。

## ステートメント

```
SELECT
  VARCHAR(MEMORY_SET_TYPE, 20) AS MEMORY_SET_TYPE,
  VARCHAR(MEMORY_POOL_TYPE, 20) AS MEMORY_POOL_TYPE,
  MEMORY_POOL_ID,
  MEMBER,
  VARCHAR(DB_NAME, 20) AS DBNAME,
  SUM(MEMORY_POOL_USED) AS MEMORY_POOL_USED,
  SUM(MEMORY_POOL_USED_HWM) AS MEMORY_POOL_USED_HWM
FROM TABLE(MON_GET_MEMORY_POOL(NULL, CURRENT_SERVER, -1))
GROUP BY MEMORY_SET_TYPE, MEMORY_POOL_TYPE, MEMORY_POOL_ID, MEMBER, DB_NAME
ORDER BY MEMORY_POOL_ID, MEMBER
```

### ■各項目の説明

- MEMORY\_SET\_TYPE : メモリー・セット タイプ
- MEMORY\_POOL\_TYPE : メモリー・プール タイプ
- MEMORY\_POOL\_ID : メモリー・プールID
- MEMBER : データベース・メンバー
- DB\_NAME : データベース名
- MEMORY\_POOL\_USED : メモリー・プール使用量
- MEMORY\_POOL\_USED\_HWM : メモリー・プール最高水準点

**NULL** を指定し すべてのメモリー・セットでの  
メモリー・プール使用量を取得する

**-1** を指定し、現行データベース・メンバーの  
メモリー・プール使用量を取得する

サンプル

メモリーの種類ごとの使用量を取得する（つづき）

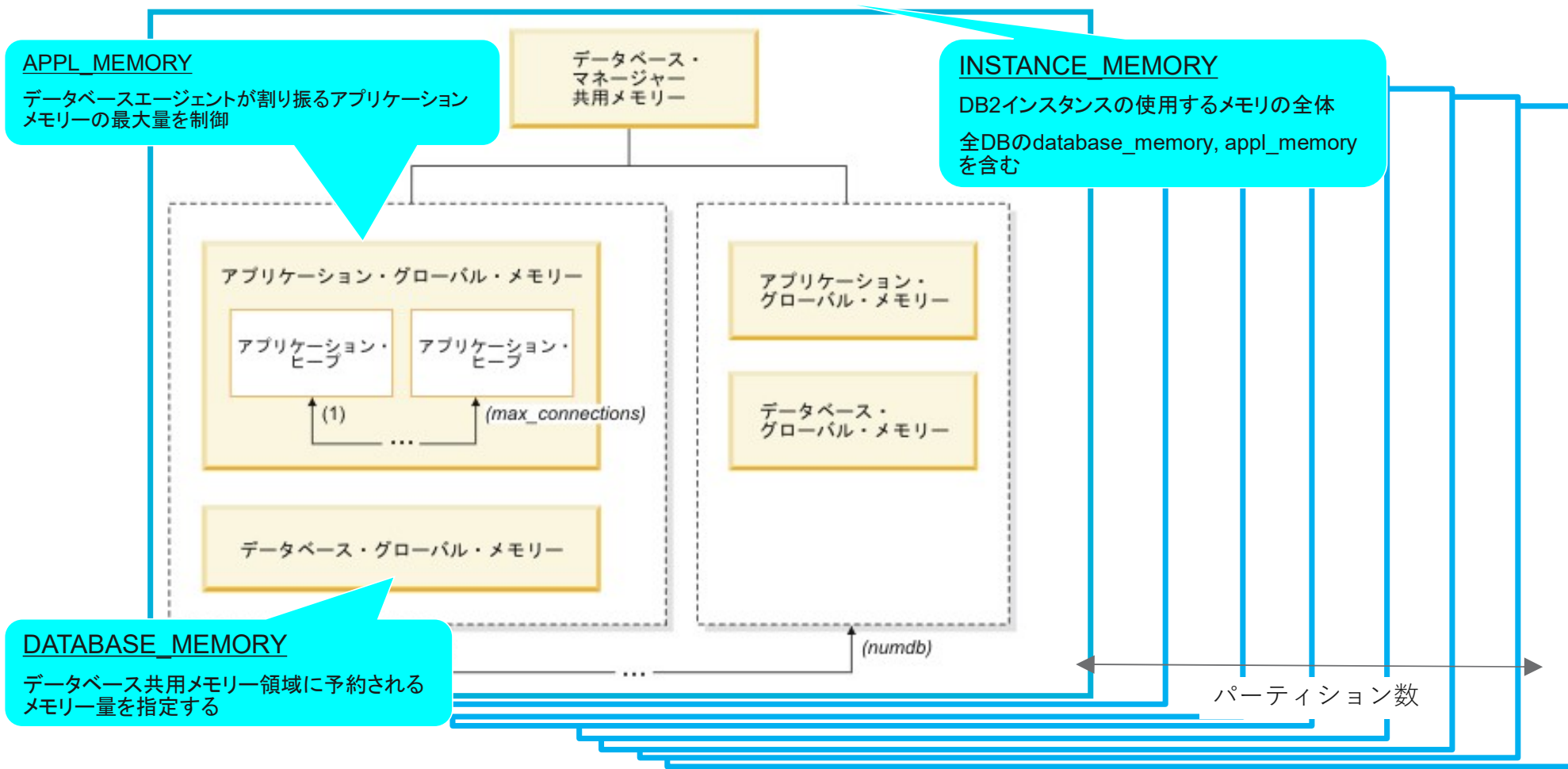
出力結果

MEMORY_SET_TYPE	MEMORY_POOL_TYPE	MEMORY_POOL_ID	MEMBER	DBNAME	MEMORY_POOL_USED	MEMORY_POOL_USED_HWM
APPLICATION	APPLICATION	1	0	BLUDB	9792	32768
DATABASE	DATABASE	2	0	BLUDB	746432	751424
DATABASE	LOCK_MGR	4	0	BLUDB	427520	427520
DATABASE	UTILITY	5	0	BLUDB	256	416384
DATABASE	PACKAGE_CACHE	7	0	BLUDB	288448	292608
...						

各種メモリーの 使用量と最高水準が表示される

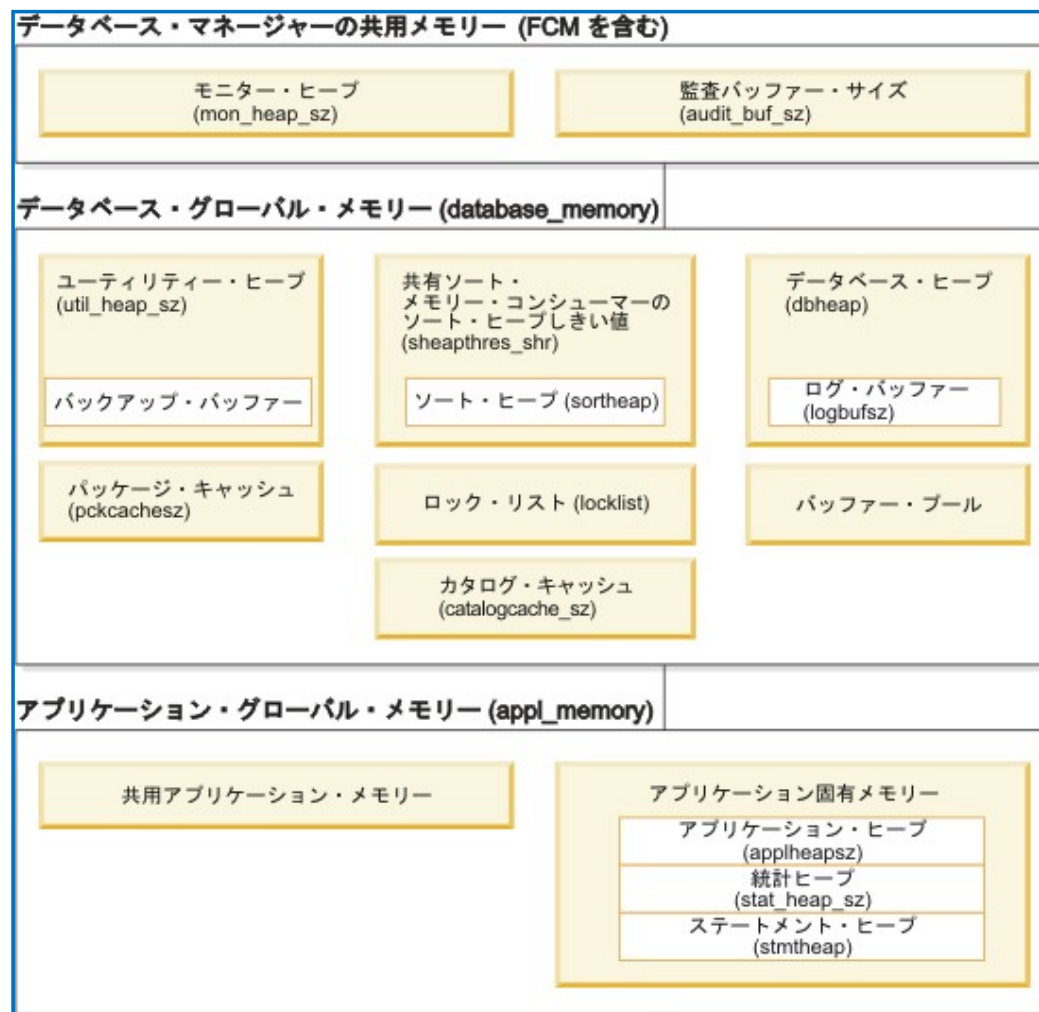
# 参考：Db2 WoCのメモリーモデル(全体)

- Db2 WoCは、複数のパーティション(メンバー)を持つ並列システムです。(Flex Oneタイプではパーティションは1つだけです)
  - パーティションごとに、db2syscプロセスが起動されます。
  - パーティション全体のメモリはinstance\_memoryです。
  - instance\_memory内に、database\_memoryとappl\_memoryが割り振られています。



## 参考：Db2のメモリーの構成要素(内訳)

- 各instance\_memory内は、以下の図のように割り振られています。



3.稼働状況の監視	<b>4.日常的に収集すべき基本モニタリング項目</b> <b>4-4.ストレージ使用状況</b>
<b>概要</b>	<p>ストレージの全体使用率は、Webコンソールの[Webコンソールの[モニター]メニュー&gt;[ダッシュボード]タブ]から確認できます。</p> <p>また、MON_GET_CONTAINER 表関数で コンテナ・ファイル・システムの使用状況、ADMIN_GET_STORAGE_PATHSで自動ストレージ・パスの使用状況を確認することができます。</p>
<b>結果の例</b>	<p>次ページの結果サンプルを参照してください。</p>
<b>詳細情報のリンク</b>	<p>[Db2WoC] MON_GET_CONTAINER 表関数 - 表スペース・コンテナ・メトリックの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-container-get-table-space-container-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-container-get-table-space-container-metrics</a></p> <p>[Db2 WoC] ADMIN_GET_STORAGE_PATHS 表関数 - 自動ストレージ・パス情報の取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=aracp-admin-get-storage-paths-retrieve-automatic-storage-path-information">https://www.ibm.com/docs/ja/db2woc?topic=aracp-admin-get-storage-paths-retrieve-automatic-storage-path-information</a></p>
<b>備考</b>	

## サンプル1

## Webコンソールからストレージ使用率を確認する

Webコンソールの[モニター]メニュー > [ダッシュボード]タブにて、ストレージ使用率が表示されます。  
期間は 最近1時間、最近6時間、最近12時間、最近24時間、最近3日、先週、先月 を 選択できます。

## 出力結果



[リソース使用量]にて、  
ストレージ使用率が表示されている



表示期間は以下選択可能

- 最近1時間
- 最近6時間
- 最近12時間
- 最近24時間
- 最近3日
- 先週
- 先月

## サンプル2

## コンテナ・ファイル・システムの使用状況を確認する

MON\_GET\_CONTAINER 表関数 から、コンテナ・ファイル・システムの使用状況を確認することができます。  
ファイル・システムの合計サイズ (**fs\_total\_size**) と ファイル・システムの使用サイズ (**fs\_used\_size**) を使用して、ファイル・システムの使用率を算出します。

### <ファイル・システム使用率 算出式>

ファイル・システム使用率(%) = 100 \* (ファイル・システム使用サイズ / ファイル・システム合計サイズ)

また、表関数パラメーターの2つ目は デフォルト値が **-1** で、現行データベース・メンバーの情報を返します。  
すべてのデータベース・メンバーの情報を取得したい場合は、**-2** を指定します。

### ステートメント

```
SELECT
  MEMBER,
  VARCHAR(CONTAINER_NAME, 65) AS CONTAINER_NAME,
  FS_ID,
  FS_USED_SIZE,
  FS_TOTAL_SIZE,
  CASE WHEN FS_TOTAL_SIZE > 0
    THEN DEC(100*(FLOAT(FS_USED_SIZE)/FLOAT(FS_TOTAL_SIZE)),5,2)
    ELSE NULL
  END AS UTILIZATION
FROM TABLE(MON_GET_CONTAINER(NULL,-1)) AS t
ORDER BY MEMBER, UTILIZATION DESC
```

- 各項目の説明
  - MEMBER : データベース・メンバー
  - CONTAINER\_NAME : コンテナ名
  - FS\_ID : ファイル・システム識別番号
  - FS\_USED\_SIZE : ファイル・システム 使用スペース量
  - FS\_TOTAL\_SIZE : ファイル・システム合計サイズ
  - UTILIZATION : ファイル・システムの使用率 (FS\_TOTAL\_SIZE, FS\_USED\_SIZE で算出)

**-1** を指定し、現行データベース・メンバーの ストレージ・パスの使用状況を取得する

### 出力結果

MEMBER	CONTAINER_NAME	FS_ID	FS_USED_SIZE	FS_TOTAL_SIZE	UTILIZATION
0	/head/db2inst1/NODE0000/BLUDB/T0000000/C0000000.CAT	64807	38445965312	126277451776	30.44
0	/head/db2inst1/NODE0000/BLUDB/T0000002/C0000000.LRG	64807	38445965312	126277451776	30.44
0	/head/db2inst1/NODE0000/BLUDB/T0000003/C0000000.LRG	64807	38445965312	126277451776	30.44
0	/head/db2inst1/NODE0000/BLUDB/T0000004/C0000000.LRG	64807	38445965312	126277451776	30.44
0	/head/db2inst1/NODE0000/BLUDB/T0000005/C0000000.LRG	64807	38445965312	126277451776	30.44
...					

各コンテナ・ファイルの 算出された ファイル・システム使用率が返される



### サンプル3

### 自動ストレージ・パスの使用状況を確認する

ADMIN\_GET\_STORAGE\_PATHS 表関数 から、各データベース・ストレージ・グループの自動ストレージ・パスの 合計サイズ、使用サイズ を取得することができます。ファイル・システム合計サイズ (fs\_total\_size) と ファイル・システム使用サイズ (fs\_used\_size) を使用して、ファイル・システムの使用率を算出します。

#### <ファイル・システム使用率 算出式>

ファイル・システム使用率(%) = 100 \* (ファイル・システム使用サイズ / ファイル・システム合計サイズ)

また、表関数パラメーターの2つ目は デフォルト値が-1で、現行データベース・メンバーの情報を返します。すべてのデータベース・メンバーの情報を取得したい場合は、-2 を指定します。

#### ステートメント

```
SELECT
  DBPARTITIONNUM,
  SUBSTR(STORAGE_GROUP_NAME,1,15) AS STORAGE_GROUP_NAME,
  SUBSTR(DB_STORAGE_PATH,1,15) AS DB_STORAGE_PATH,
  FS_USED_SIZE,
  FS_USED_SIZE,
  CASE WHEN FS_TOTAL_SIZE > 0
    THEN DEC(100*(FLOAT(FS_USED_SIZE)/FLOAT(FS_TOTAL_SIZE)),5,2)
    ELSE NULL
  END AS UTILIZATION
FROM TABLE(ADMIN_GET_STORAGE_PATHS(NULL,-1))
ORDER BY DBPARTITIONNUM;
```

#### ■各項目の説明

- DBPARTITIONNUM : データ・パーティション番号
- STORAGE\_GROUP\_NAME : ストレージ・グループ名
- DB\_STORAGE\_PATH : ストレージ・パス
- FS\_USED\_SIZE : ファイル・システム 使用スペース量
- FS\_TOTAL\_SIZE : ファイル・システム合計サイズ
- UTILIZATION : ファイル・システムの使用率  
(FS\_TOTAL\_SIZE, FS\_USED\_SIZE で算出)

-1 を指定し、現行データベース・メンバーの ストレージ・パスの使用状況を取得する

#### 出力結果

DBPARTITIONNUM	STORAGE_GROUP_NAME	DB_STORAGE_PATH	FS_USED_SIZE	FS_USED_SIZE	UTILIZATION
0	IBMSTOGROUP	/head	38300966912	38300966912	30.33
0	IBMDASHDBTEMPSG	/local	92655616	92655616	0.01

各ストレージ・パスの 算出された ファイル・システム使用率が返される

3.稼働状況の監視	<b>4.日常的に収集すべき基本モニタリング項目</b> 4-5.トランザクションログ領域の使用率
<b>概要</b>	トランザクションログ領域の使用率は、Webコンソールの[Webコンソールの[モニター]メニュー>[ダッシュボード]タブ から確認できます。 また、MON_TRANSACTION_LOG_UTILIZATION 管理ビューからも、接続されているデータベースの トランザクション・ログ使用率を確認できます。
<b>結果の例</b>	次ページの結果サンプルを参照してください。
<b>詳細情報のリンク</b>	[Db2 WoC] MON_TRANSACTION_LOG_UTILIZATION 管理ビュー - ログ使用率に関する情報の検索 <a href="https://www.ibm.com/docs/ja/db2woc?topic=mv-mon-transaction-log-utilization-retrieve-log-utilization-information">https://www.ibm.com/docs/ja/db2woc?topic=mv-mon-transaction-log-utilization-retrieve-log-utilization-information</a>
<b>備考</b>	

## サンプル1

## Webコンソールからトランザクション・ログ使用率を確認する

Webコンソールの[モニター]メニュー > [ダッシュボード]タブにて、ログ・スペースの使用率が表示されます。  
期間は 最近1時間、最近6時間、最近12時間、最近24時間、最近3日、先週、先月 を 選択できます。

## 出力結果



[リソース使用量]にて、  
ログ・スペース使用率が表示されている



表示期間は以下選択可能

- 最近1時間
- 最近6時間
- 最近12時間
- 最近24時間
- 最近3日
- 先週
- 先月

## サンプル2

## 接続されているデータベースのメンバーごとのトランザクション・ログ使用率を確認する

MON\_TRANSACTION\_LOG\_UTILIZATION 管理ビューから、接続されているデータベースのメンバーごとのトランザクション・ログ使用率 (LOG\_UTILIZATION\_PERCENT) を取得することができます。無限ロギングが設定されている場合は、LOG\_UTILIZATION\_PERCENT の値はNULLになります。

### ステートメント

```
SELECT
  MEMBER AS MEMBER,
  LOG_UTILIZATION_PERCENT AS LOG_UTILIZATION_PERCENT
FROM SYSIBMADM.MON_TRANSACTION_LOG_UTILIZATION
ORDER BY MEMBER
```

#### ■各項目の説明

- MEMBER : データベース・メンバー
- LOG\_UTILIZATION\_PERCENT : 使用中の合計ログ・スペースのパーセンテージ

### 出力結果

メンバーごとの  
ログの使用率が表示される

MEMBER	LOG_UTILIZATION_PERCENT
0	2.48
1	0.51
2	0.51
3	0.51
4	0.51
5	0.51
6	0.51
7	0.51

3.稼働状況の監視	4.日常的に収集すべき基本モニタリング項目 4-6. スループット
-----------	--------------------------------------

<b>概要</b>	<p>スループットは、Webコンソールから確認できます。</p> <p>また、サービス・サブクラスごとのCPU使用率は、MON_GET_SERVICE_SUBCLASS_STATS / MON_GET_SERVICE_SUPERCLASS_STATS表関数、ワークロードごとのCPU使用率は、MON_GET_WORKLOAD_STATS表関数 / MON_SAMPLE_WORKLOAD_METRICS表関数のUOW_THROUGHPUTで取得できます。</p>
<b>結果の例</b>	<p>次ページの結果サンプルを参照してください。</p>
<b>詳細情報のリンク</b>	<p>[Db2 WoC] MON_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-service-subclass-stats-return-statistics-service-subclasses">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-service-subclass-stats-return-statistics-service-subclasses</a>          [Db2 WoC] MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラス・メトリックのサンプルの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-service-class-metrics-get-sample-service-class-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-service-class-metrics-get-sample-service-class-metrics</a></p> <p>[Db2 WoC] MON_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す  <a href="https://www.ibm.com/docs/ja/db2woc?topic=functions-mon-get-workload-stats-return-workload-statistics">https://www.ibm.com/docs/ja/db2woc?topic=functions-mon-get-workload-stats-return-workload-statistics</a>          [Db2 WoC] MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics</a></p>
<b>備考</b>	

## サンプル1

## Webコンソールからスループットを確認する(1)

Webコンソールの[モニター]メニュー > [ダッシュボード]タブにて、全体のスループット情報が表示されます。  
期間は **最近1時間**、**最近6時間**、**最近12時間**、**最近24時間**、**最近3日**、**先週**、**先月** を 選択できます。

## 出力結果



[スループット]にて、  
ログ・スペース使用率が表示されている



表示期間は以下選択可能

- 最近1時間
- 最近6時間
- 最近12時間
- 最近24時間
- 最近3日
- 先週
- 先月



### サンプル3

### サービス・クラスごとのスループットを取得する

MON\_GET\_SERVICE\_SUBCLASS\_STATS 関数で サービス・サブクラスの基本統計が取得されます。  
ACT\_THROUGHPUTから 統計の最後のリセット以降のアクティビティ・スループットを確認します。

#### ステートメント

```
SELECT
  SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
  SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
  SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
  DECIMAL(ACT_THROUGHPUT,8,2) AS ACT_THROUGHPUT
FROM TABLE(MON_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME, MEMBER
```

#### ■各項目の説明

- SERVICE\_SUPERCLASS\_NAME : サービス・スーパークラス名
- SERVICE\_SUBCLASS\_NAME : サービス・サブクラス名
- MEMBER : データベース・メンバー
- ACT\_THROUGHPUT : アクティビティ・スループット

#### 出力結果

サービス・クラス、  
メンバーごとの  
アクティビティ・  
スループットが  
表示される

SUPERCLASS_NAME	SUBCLASS_NAME	MEMB	ACT_THROUGHPUT
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	0	26.50
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	1	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	2	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	3	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	4	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	5	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	6	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	7	0.00
...			



#### サンプル4

#### 指定した期間（300秒）のサービスクラスごとのスループットをリアルタイムで取得する

MON\_SAMPLE\_SERVICE\_CLASS\_METRICS表関数から、各サービス・クラスの指定時間内（ここでは300秒）のアクティビティ・スループットを取得します。指定時間を変更したい場合は、5つ目の表関数パラメーターで秒単位で指定します。

サービスクラスごとのアクティビティ・スループットは、ACT\_THROUGHPUT列を確認します。

#### ステートメント

```
SELECT
  SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
  SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
  SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
  DECIMAL(UOW_THROUGHPUT,3,2) AS UOW_THROUGHPUT
FROM TABLE(MON_SAMPLE_SERVICE_CLASS_METRICS(null, null, '', '', 300, -2)) AS t
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME, MEMBER
```

#### ■各項目の説明

- SERVICE\_SUPERCLASS\_NAME : サービス・スーパークラス名
- SERVICE\_SUBCLASS\_NAME : サービス・サブクラス名
- MEMBER : データベース・メンバー
- ACT\_THROUGHPUT : アクティビティ・スループット

300秒間のCPU使用率を取得するように指定

#### 出力結果

サービス・クラス、  
メンバーごとの  
指定した期間（300秒）の  
アクティビティ・  
スループットが  
表示される

SUPERCLASS_NAME	SUBCLASS_NAME	MEMB	ACT_THROUGHPUT
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	0	10.48
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	1	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	2	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	3	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	4	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	5	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	6	0.00
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	7	0.00
...			

## サンプル5

## ワークロードごとのスループットを取得する

MON\_GET\_WORKLOAD\_STATS 関数でワークロードの基本統計が取得されます。  
ワークロードごとのアクティビティ・スループットは、ACT\_THROUGHPUT 列を確認します。

### ステートメント

```
SELECT  
  SUBSTR(WORKLOAD_NAME,1,18) AS WORKLOAD_NAME,  
  SUBSTR(CHAR(MEMBER),1,4) AS MEMBER,  
  DECIMAL(ACT_THROUGHPUT,8,2) AS ACT_THROUGHPUT  
FROM TABLE(MON_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2)) AS WLSTATS  
ORDER BY WORKLOAD_NAME, MEMBER
```

#### ■各項目の説明

- WORKLOAD\_NAME : ワークロード名
- MEMBER : データベース・メンバー
- ACT\_THROUGHPUT : アクティビティ・スループット

### 出力結果

ワークロード、  
メンバーごとの  
アクティビティ・  
スループットが  
表示される

WORKLOAD_NAME	MEMBER	ACT_THROUGHPUT
SYSDEFAULTUSERWORK	0	0.06
SYSDEFAULTUSERWORK	1	0.00
SYSDEFAULTUSERWORK	2	0.00
SYSDEFAULTUSERWORK	3	0.00
SYSDEFAULTUSERWORK	4	0.00
SYSDEFAULTUSERWORK	5	0.00
SYSDEFAULTUSERWORK	6	0.00
SYSDEFAULTUSERWORK	7	0.00
...		

## サンプル6

## 指定した期間（秒）のワークロードごとのスループットをリアルタイムで取得する

MON\_SAMPLE\_WORKLOAD\_METRICS 表関数から、各ワークロードの指定時間内（ここでは300秒）のアクティビティ・スループットを取得します。指定時間を変更したい場合は、4つ目の表関数パラメーターで秒単位で指定します。

ワークロードごとのアクティビティ・スループットはACT\_THROUGHPUTを確認します。

### ステートメント

```
SELECT  
  SUBSTR(WORKLOAD_NAME,1,18) AS WORKLOAD_NAME,  
  SUBSTR(CHAR(MEMBER),1,4) AS MEMBER,  
  DECIMAL(ACT_THROUGHPUT,3,2) AS ACT_THROUGHPUT  
FROM TABLE(MON_SAMPLE_WORKLOAD_METRICS(NULL, CURRENT SERVER, '', 300, -2)) AS T  
ORDER BY WORKLOAD_NAME, MEMBER
```

#### ■各項目の説明

- WORKLOAD\_NAME : ワークロード名
- MEMBER : データベース・メンバー
- ACT\_THROUGHPUT : アクティビティ・スループット

300秒間のCPU使用率を取得するように指定。

### 出力結果

ワークロード、  
メンバーごとの  
指定した期間（300秒）の  
作業スループットが  
表示される

WORKLOAD_NAME	MEMBER	ACT_THROUGHPUT
SYSDEFAULTUSERWORK	0	0.14
SYSDEFAULTUSERWORK	1	0
SYSDEFAULTUSERWORK	2	0
SYSDEFAULTUSERWORK	3	0
SYSDEFAULTUSERWORK	4	0
SYSDEFAULTUSERWORK	5	0
SYSDEFAULTUSERWORK	6	0
SYSDEFAULTUSERWORK	7	0
...		

3.稼働状況の監視	<b>4.日常的に収集すべき基本モニタリング項目</b> 4-7. レスポンスタイム
<b>概要</b>	レスポンスタイムは、Webコンソール・ステートメント両方から確認することができます。
<b>結果の例</b>	次ページの結果サンプルを参照してください。
<b>詳細情報のリンク</b>	[Db2 WoC] MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ・ステートメント・メトリックの取得 <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-pkg-cache-stmt-table-function-get-package-cache-statement-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-pkg-cache-stmt-table-function-get-package-cache-statement-metrics</a>
<b>備考</b>	

## サンプル1

## Webコンソールからレスポンスタイムを確認する

Webコンソールの[モニター]メニュー > [ステートメント]タブにて、実行されたステートメントの情報が表示されています。  
[個々の実行]タブにて、SQLステートメントそれぞれの実行時間を、**ステートメントの実行時間**列で確認することができます。  
表示期間は **最近1時間**、**最近6時間**、**最近12時間**、**最近24時間**、**最近3日**、**先週**、**先月** を選択できます。

## 出力結果

The screenshot shows the IBM Web Console interface for monitoring SQL statements. The main table displays the following data:

SQL	ステートメントの実行時間	CPU 時間	準備時間	アクティビティ待機時間	ロックを待機した時間
<a href="#">SELECT * FROM SYSCAT.TABLES</a>	0:00.040	0:00.035	0:00.053	0:00.004	0.000
<a href="#">SELECT COUNT(*) FROM SYSIBM.SYSTABLES WITH UR</a>	0:00.002	0:00.001	0:00.003	0:00.001	0.000
<a href="#">SELECT COUNT(*) FROM SYSIBM.SYSTABLES WITH UR</a>	0:00.002	0:00.001	0:00.003	0:00.001	0.000
<a href="#">SELECT COUNT(*) FROM SYSIBM.SYSTABLES WITH UR</a>	0:00.002	0:00.001	0:00.003	0.000	0.000

The date range selector on the left is set to 2021-12-16 00:00:00 - 2021-12-16 13:00:00. The '個々の実行' (Individual Execution) tab is selected. The table header 'ステートメントの実行時間' is highlighted with a red box.

表示期間は以下選択可能。

- 最近1時間
- 最近6時間
- 最近12時間
- 最近24時間
- 最近3日
- 先週
- 先月
- カスタム

SQLステートメント それぞれの実行時間をステートメントの実行時間列で確認可能。

## サンプル2

## ステートメントのレスポンスタイムを確認する

MON\_GET\_PKG\_CACHE\_STMT 表関数 から、SQLステートメントの 情報を取得することができます。

ここでは、ステートメント・テキスト(STMT\_TEXT)、メトリックを使用したステートメント実行回数(NUM\_EXEC\_WITH\_METRICS)、ステートメント実行時間の合計(STMT\_EXEC\_TIME) を取得しています。

また、NUM\_EXEC\_WITH\_METRICS と STMT\_EXEC\_TIME を使用して、ステートメント実行1回あたりの 平均実行時間を算出しています。

### <平均実行時間 算出式>

ステートメント 平均実行時間 = ステートメント実行時間の合計/メトリックを使用したステートメント実行回数

また、表関数パラメーターの1つ目では ステートメントの種類を指定します。

引数がNULL、空ストリングの場合、すべての SQL ステートメントについての情報が返され、"D"の場合は 動的、"S"の場合は静的 SQLステートメントが返されます。

表関数パラメーターの4つ目は デフォルト値が -1で、現行データベース・メンバーでのステートメント情報を返します。

すべてのデータベース・メンバーの情報を取得したい場合は、-2を指定します。

## ステートメント

```
SELECT
  SUBSTR(STMT_TEXT,1,300) AS STMT_TEXT,
  NUM_EXEC_WITH_METRICS,
  STMT_EXEC_TIME,
  ROUND(FLOAT(STMT_EXEC_TIME)/FLOAT(NUM_EXEC_WITH_METRICS),2) as AVG_STMT_EXEC_TIME
FROM TABLE(MON_GET_PKG_CACHE_STMT ('D', NULL, NULL, -2)) as T
```

### ■各項目の説明

- STMT\_TEXT : SQLステートメント・テキスト
- NUM\_EXEC\_WITH\_METRICS : メトリックを使用した SQLステートメント実行回数
- STMT\_EXEC\_TIME : ステートメントの合計 実行時間
- AVG\_STMT\_EXEC\_TIME : ステートメント実行 1回あたりの 平均実行時間 (NUM\_EXEC\_WITH\_METRICS, STMT\_EXEC\_TIME で算出)

-2 を指定し、すべてのデータベース・メンバーの ステートメントの情報を取得する。

'D' を指定し、動的SQLステートメントの情報を取得する。

## 出力結果

STMT_TEXT	NUM_EXEC_WITH_METRICS	STMT_EXEC_TIME	AVG_STMT_EXEC_TIME
/* IBM_DSM */ select 1 from DSSHSV1.OBJECT_STORE where 1 > 2	1	0	0.00
/* IBM_DSSNAP */ drop event monitor RTMON_EVMON_ACTIVITIES_1639570457931	1	1	1.00
/* IBM_DSSNAP */ create index IBM_RTMON."idx_activity_stmt_1639546049527" on IBM_RTMON.ACTIVITY_STMT_1639546049527 (ACTIVITY_ID, ACTIVITY_SECONDARY_ID, PARTITION_NUMBER, APPL_ID, UOW_ID)	1	68	68.00
...			

算出した 実行1回あたりの平均実行時間が返される



3.稼働状況の監視	<b>4.日常的に収集すべき基本モニタリング項目</b> 4-8.エラーログ(db2diag.log)の確認
<b>概要</b>	db2diag.logには 管理通知ログ・メッセージが記録され、トラブルシューティングの際に使用されます。 db2diag.logの中身は、PD_GET_DIAG_HIST表関数で確認できます。
<b>結果の例</b>	次ページの結果サンプルを参照してください。
<b>詳細情報のリンク</b>	[Db2 v11.5] PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す <a href="https://www.ibm.com/docs/ja/db2/11.5?topic=mrpv-pd-get-diag-hist-return-records-from-facility">https://www.ibm.com/docs/ja/db2/11.5?topic=mrpv-pd-get-diag-hist-return-records-from-facility</a>
<b>備考</b>	

サンプル

エラーログ(db2diag.log)を確認する

PD\_GET\_DIAG\_HIST表関数の 1つ目のパラメーターに 'MAIN' を指定すると、db2diag ログ・レコードを取得することができます。  
 ここでは 2021/12/8 のログを確認するため、開始時刻,終了時刻として 4,5つ目のパラメーターに、  
 '2021-12-08-00.00.00', '2021-12-09-00.00.00' をTIMESTAMP型に変換した値を指定しています。

ステートメント

```
SELECT
  FULLREC
FROM TABLE (PD_GET_DIAG_HIST( 'MAIN', '', '', TIMESTAMP('2021-12-08-00.00.00',9), TIMESTAMP('2021-12-09-00.00.00',9), -2) )
ORDER BY TIMESTAMP
```

出力結果

FULLREC		
2021-12-08-00.00.47.805210+000	I7132303E604	LEVEL: Error
PID : 195701	TID : 140596677175040	PROC : db2sysc 0
INSTANCE: db2inst1	NODE : 000	DB : BLUDB
APPHDL : 0-36803	APPID: 172.xx.xxx.xxx.xxxxx.211208000044	
UOWID : 27	ACTID: 1	
AUTHID : DSADM	HOSTNAME: dashmpp-head-0	
EDUID : 638279	EDUNAME: db2agntp (METADB) 0	
FUNCTION: DB2 UDB, runtime interpreter, sqlri_tfcls, probe:191		
RETCODE : ZRC=0xFFFFFC48=-952		
SQL0952N Processing was cancelled due to an interrupt.		
2021-12-08-00.00.47.812034+000	I7132908E605	LEVEL: Error
PID : 195701	TID : 140596677175040	PROC : db2sysc 0
INSTANCE: db2inst1	NODE : 000	DB : BLUDB
APPHDL : 0-36803	APPID: 172.xx.xxx.xxx.xxxxx.211208000044	
UOWID : 27	ACTID: 1	
AUTHID : DSADM	HOSTNAME: dashmpp-head-0	
EDUID : 638279	EDUNAME: db2agntp (METADB) 0	
FUNCTION: DB2 UDB, runtime interpreter, sqlricls_complex, probe:5504		
RETCODE : ZRC=0x8012006D=-2146303891=SQLR_CA_BUILT		
"SQLCA has already been built"		
...		





## 4. こんな時、どうする？

## パフォーマンス問題の発生パターン

**Db2 WoC におけるパフォーマンス問題には様々なものが考えられます。**

パフォーマンス問題の例：

- トランザクションの応答が遅く、性能要件を満たさない
- ピーク時の資源使用率が著しく高く、将来の負荷増大に耐えられない
- バッチ・プログラムの完了が遅延し、朝のオンライン開始に影響
- LOADなど、ユーティリティーの完了が遅延する



**これらの問題発生の原因箇所を突き止めて解消し、本来あるべき性能を発揮できるようにしたい。**

# 問題の原因と発見

## 問題の原因

- **クライアント側の問題**
  - アプリケーション内での遅延
  - 不適當なSQL実行
- **不適當なデータベース設計**
  - 表設計
  - 索引設計
  - 分散キー
- **偏りのあるデータ配置**
- **資源の不足**
- **不適切なDBチューニング**
  - バッファプール
  - ソートヒープ
  - ロック
- **非効率的なSQL処理**
  - 最適でないアクセスプラン



## 発見のための方法

- **コンソールからの監視**
- **SQL(表関数)モニタリング**
  - mon\_get
- **アクティビティ・イベント・モニター**
  - statement
- **エクस्पライン**

# トラブル発生！ 何から始めますか？ ①

## 【ステップ1】 まず、状況をきちんと把握する



アプリケーション  
の  
処理が遅い！？

### ●問題の現状を把握する

急に遅くなったのか？ 徐々に遅くなったのか？  
ある時間帯だけ遅いのか？  
何をしたら改善するのか？（時間帯、タイミング、・・・）

### ●手がかりを探す

DB2のログ（db2diag.log）、・・・  
アプリケーションのログ  
エラーは無いかな？

### ●正常時との違いを把握する

遅い時と正常な時で何が違うのか？（処理内容、環境、・・・）  
何か変化はあったか？（データ量、アクセス数、処理内容、  
新規アプリケーションのリリース、  
・・・）

## 【ポイント】

問題の発生状況を正しく把握することで、  
正しい方向へ問題判別作業を進めることができる

注意：事実に基づいて判断すること。  
そのために、正常時から判断材料として基本的なモニタリング・データを取得しておくことが望ましい

# トラブル発生！ 何から始めますか？ ②

## 【ステップ2】 問題が再現する場合の問題箇所の特定

アプリケーションの画面の処理が遅い

↓ (アプリケーションの設計書より関連するモジュール (群) を特定する)

どのアプリケーション (モジュール) が遅いのかを調べる

↓ アプリケーション内の各モジュールの実行時間をロギングする機能が重要となる  
(開始タイムスタンプと終了タイムスタンプを記録しておき、引き算をして実行時間を計算する)  
あらかじめアプリケーションやフレームワーク等でロギング機能を実装しておくことが望ましい

そのモジュールの中で、データベース処理が遅いのか、  
それ以外の処理が遅いのかを調べる



データベースの処理が遅い場合、どのSQLが遅いのかを調べる

アプリケーションのロギング機能は、ログ・レベルを変更できるようにして、  
詳細なレベルでは各SQLの処理時間まで記録できることが望ましい

# トラブル発生！ 何から始めますか？ ③

## ■ 【ステップ3】 問題判別のための資料収集

- **まずは全体像を把握するための資料を収集（問題発生状況下で取得）**
  - システムレベル
    - CPUネック？ I/Oネック？ どこにも問題なさそう？
  - DBレベル(mon\_get\_databaseなど)
    - データベース全体の状況サマリー
- **特定のアプリやステートメントに問題があるならそれらに特化した情報収集**
  - SQLステートメントモニター
  - 表の情報
  - エクスプレイン (アクセスプランの情報)

サーバーサイドでは問題が見つからないこともある  
=> クライアント (アプリケーション)の問題？  
ネットワークの問題？

### パラメータチューニング可能か？

メモリ配分の変更

### 表、索引のメンテナンス

索引は有効か？

分散キーは適切か？

辞書の再作成や、データの再投入は有効か？

### SQLチューニング/アプリケーション変更

SQLの修正

### WLMの構成

ワークロードマネージャ(WLM)により資源配分の適正化を図る

# Db2 Warehouse 運用時に押さえるべき5つの重要ポイント

本章では、運用時に必用となる重要な調査項目を  
実際の収集コマンドと結果の解釈の方法を中心に説明しています。

<b>1</b>	<b>DB2運用時に押さえるべき重要ポイント</b>
1-1	性能編① SQL実行数を調査する
1-2	性能編②データベース全体でのボトルネックの所在を知りたい
1-3	性能編③ 負荷の高いSQLを手早く把握する
1-4	リソース編① DBサーバーのCPU使用率を知りたい
1-5	リソース編② 表スペースの使用率を知りたい
1-6	リソース編③ メモリーの使用量を把握する
1-7	リソース編④ ログの使用量を知りたい
<b>2</b>	<b>いろいろな切り口でのDB2 Warehouse モニタリング</b>
2-1	データベースの処理状況をリアルタイムでサマリーする
2-2	表や索引サイズの実態を把握する
2-3	自動保守の実行状況を知りたい
2-4	ロック待ち/ロックタイムアウト/デッドロックの原因となるSQLを特定する
2-5	表のパーティションごとのレコード数の偏りを調査する

4. こんな時、どうする？	<b>1. DB2運用時に押さえるべき重要ポイント</b> 1-1 性能編① SQL実行数を調査する
<b>概要</b>	データベースが処理したSQLの数は、ACT_COMPLETED_TOTAL モニターエレメントで把握することができます。 ACT_COMPLETED_TOTALは、ワークロードやパッケージ、コネクションなど様々なオブジェクトの単位で取得することが可能です。
<b>詳細</b>	ここではOLTP的な負荷をかけたデータベースに対して、MONGETデータ取得ツールで収集した結果をサマリーしています。 結果を取得するSQLの実例は、次ページの「サンプル」を参照してください。
<b>詳細情報のリンク</b>	[Db2 WoC] MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得 <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics</a>
<b>備考</b>	



**サンプル 1****指定した期間（秒）のSQL実行数をリアルタイムで取得する**

MON\_SAMPLE\_WORKLOAD\_METRICS表関数を使用して、リアルタイムのSQL実行数を取得するSQLです。

下記の例では、30秒間のトータルSQL処理数（ACT\_COMPLETED\_TOTAL）と毎秒のSQL処理件数（ACT\_THROUGHPUT）を取得しています。

リアルタイムでSQL処理数の推移を監視する場合、下記のようなSQLをスクリプトから連続して実行することで実現可能です。

**ステートメント**

```
SELECT
  CURRENT_TIMESTAMP           AS CURRENT_TIMESTAMP,
  DEC(SUM(ACT_THROUGHPUT),8,2) AS ACT_THROUGHPUT,
  SUM(ACT_COMPLETED_TOTAL)    AS ACT_COMPLETED_TOTAL
FROM TABLE(MON_SAMPLE_WORKLOAD_METRICS(NULL, NULL, NULL, 30, -2))
GROUP BY CURRENT_TIMESTAMP;
```

**出力結果**

CURRENT_TIMESTAMP	ACT_THROUGHPUT	ACT_COMPLETED_TOTAL
2021-10-08 06:47:41.821875	1.06	32

4. こんな時、どうする?	1-2 性能編② データベース全体でのボトルネックの所在を知りたい
---------------	-----------------------------------

<b>概要</b>	
	データベース全体のボトルネックは、MONREPORT.DBUSUMMARYの結果から把握することが出来ます。 全体の処理時間中の待機時間の確認、ロックやBufferpool、ログのDiskI/Oなどの待機時間の割合から、ボトルネックが発生していないかどうか確認することが出来ます。
<b>モニター結果の例</b>	
	この結果を取得するSQLの実例は、次ページの「サンプル」を参照してください。
<b>詳細情報のリンク</b>	
	[Db2 WoC] MONREPORT モジュールを使用して生成されるレポート <a href="https://www.ibm.com/docs/ja/db2woc?topic=interfaces-reports-generated-using-monreport-module">https://www.ibm.com/docs/ja/db2woc?topic=interfaces-reports-generated-using-monreport-module</a>
<b>備考</b>	

サンプル

指定した期間（秒）のデータベースのボトルネックをレポート形式で取得するSQL

メトリックを使用してレポート形式の処理状況を取得できるMONREPORTモジュールは、指定した期間の活動状況をサマリーするモニタリング機能です。その中でも、MONREPORT.DBSUMMARYモジュールは、データベース全体の処理状況を一覧性のある形で手軽に取得できる点で非常に優れています。この出力結果例では、データベース全体の待機状況を示すセクションを抜粋しています。

ステートメント

CALL MONREPORT.DBSUMMARY(10)

出力結果

Wait times

-- Wait time as a percentage of elapsed time --

	%	Wait time/Total time
For requests	0	7/21185
For activities	0	4/10016

データベース内部での「待ち」の発生状況。  
SQLあたりとリクエストあたりの数字

-- Time waiting for next client request --

CLIENT\_IDLE\_WAIT\_TIME = 74081  
CLIENT\_IDLE\_WAIT\_TIME per second = 7408

クライアント要求を待機していた時間

-- Detailed breakdown of TOTAL\_WAIT\_TIME --

	%	Total
TOTAL_WAIT_TIME	100	7

TOTAL\_WAIT\_TIME:  
合計待機時間とその内訳

I/O wait time

POOL_READ_TIME	0	0
POOL_WRITE_TIME	0	0
DIRECT_READ_TIME	0	0
DIRECT_WRITE_TIME	0	0
LOG_DISK_WAIT_TIME	0	0
LOCK_WAIT_TIME	0	0
AGENT_WAIT_TIME	0	0
Network and FCM		
TCPIP_SEND_WAIT_TIME	28	2

待機時間の詳細

この出力結果では、LOCK待ちが25%、LOGのDiskへの書きみに73%の待ちが発生している。ログ領域のDiskI/Oに問題がないかどうか検討する。



4. こんな時、どうする?	1-3 性能編③ 負荷の高いSQLを手早く把握する
---------------	---------------------------

**概要**

性能に影響する負荷の高いSQLを特定するためには、MONREPORT.PKGCACHEモジュールや、MON\_GET\_PKG\_CACHE\_STMT表関数を使用することが出来ます。MONREPORT.PKGCACHEモジュールは、最も手軽に情報を取得することが出来ます。MONGETデータ収集ツールの拡張モニタリング項目として、MON\_GET\_PKG\_CACHE\_STMT表関数の情報を蓄積し、そのデータをもとに解析することも可能です。SQLの実行数に応じて蓄積する容量について考慮してください。

**モニター結果の例**

以下の順番で解析を行うことが可能です。

- ① MONREPORT.PKGCACHEモジュールを取得
- ② MON\_GET\_PKG\_CACHE\_STMT表関数から負荷の高いSQL全文を特定
- ③ 負荷の高いSQLのアクセスプランを取得

#	TOTAL_CPU_TIME	STMT_TEXT
1	105598211	SELECT c_discount, c_last, c_credit, w_tax FROM CUSTOMER, WAREH
2	10254123	SELECT i_price, i_name, i_data FROM item WHERE i_id = ?

MONREPORT.PKGCACHEモジュールでは、CPU実行時間、実行回数、待機時間などの上位10SQLを出力する。

**詳細情報のリンク**

[Db2 WoC]PKGCACHE プロシージャ - パッケージ・キャッシュ・メトリックの要約レポートの生成  
<https://www.ibm.com/docs/ja/db2woc?topic=mm-pkgcache-procedure-generate-summary-report-package-cache-metrics>

[Db2 WoC]EXPLAIN\_FROM\_SECTION プロシージャ - パッケージ・キャッシュまたはパッケージ・キャッシュ・イベント・モニター情報を使用したステートメントの Explain  
<https://www.ibm.com/docs/ja/db2woc?topic=er-explain-from-section-procedure-explain-statement-using-package-cache-package-cache-event-monitor-information>

**備考**



## サンプル1

## パッケージ・キャッシュ上の負荷の高いSQLのサマリーを取得するSQL

メトリックを使用してレポート形式の処理状況を取得できるMONREPORTモジュールは、指定した期間の活動状況をサマリーするモニタリング機能です。その中でも、MONREPORT.PKGCACHEモジュールは、パッケージキャッシュ上の負荷の高いSQLを一覧性のある形で手軽に取得できる点で非常に優れています。また動的SQLと静的SQLの両方の情報を取得することが出来ます。この出力結果例では、過去30分間にメトリックが更新されたパッケージキャッシュ内の動的SQLと静的SQLから、TOTAL\_CUP\_TIMEの高い上位のSQLを表示しているセクションを抜粋しています。

### ステートメント

```
CALL MONREPORT.PKGCACHE(30)
```

引数の30は、過去30分間にメトリックが更新されたパッケージキャッシュ上の、動的・静的SQLを取得することを指定している。引数を指定しない場合には、パッケージキャッシュ上の全てのSQLが対象になる。

### 出力結果

```
Part 1 - Summaries by 'top' metrics
```

```
Top 10 statements by TOTAL_CPU_TIME
```

```
-----  
# TOTAL_ STMT_TEXT  
CPU_TIME
```

```
-----  
1 126913270 /* IBM_DSSNAP */ select reg_var_value from table(env_get_reg_va  
2 106788763 /* IBM_DSSNAP */ SELECT L.APPLICATION_HANDLE AS APPLICATION_HAN  
...  
10 46421546 select count(*) as conn_count from table(MON_GET_CONNECTION(cas
```

CPU時間が多い順にSQLステートメントがリストされる  
実行回数のトータルCPU時間となる。

```
-----  
Part 2 - EXECUTABLE_IDs for statements in Part 1
```

```
# EXECUTABLE_ID
```

```
-----  
1 x'01000000000000008ABE090000000000000000020020211206024014605758'  
2 x'01000000000000000741502000000000000000000020020211110132601256453'  
3 x'0100000000000000EA1402000000000000000000020020211110132546427351'  
4 x'0100000000000000095190A000000000000000000020020211207070847590293'  
5 x'010000000000000098BE090000000000000000000020020211206024017335968'  
6 x'0100000000000000301502000000000000000000020020211110132548204182'  
7 x'0100000000000000D51402000000000000000000020020211110132508304692'  
8 x'01000000000000006C8003000000000000000000020020211115095259468388'  
9 x'0100000000000000C50000000000000000000000020020211104153241669027'  
10 x'010000000000000042000000000000000000000020020211104153125102824'  
...  
-----
```

Pckcacheレポート上の  
番号(#)とEXECUTABLE\_IDの紐付けがリストされる。

※ EXECUTABLE\_ID:実行された  
SQL ステートメント・セクションを一意的に識別する  
バイナリー・トークン

EXECUTABLE\_IDからSQL全文を取得するSQLは次ページを参照

## サンプル 2

## EXECUTABLE\_IDからSQL全文を取得するSQL

MON\_GET\_PKG\_CACHE\_STMT表関数を使用することで、MONREPORT.PKGCACHEモジュールで取得したEXECUTABLE\_IDからSQL全文を取得することができます。  
MON\_GET\_PKG\_CACHE\_STMT表関数の 第2引数に EXECUTABLE\_IDを指定します。

ステートメント	出力結果
<pre>SELECT   SUBSTR(STMT_TEXT,1,300) AS STMT_TEXT FROM TABLE(MON_GET_PKG_CACHE_STMT(null,x'0100000000000000EA1402000000000000000 020020211110132546427351',null,-2));</pre>	<pre>STMT_TEXT UPDATE EMPLOYEE SET BONUS=10000 WHERE PERF_RATING=1</pre> <p>MONREPORT.PKGCACHEモジュールで出力された EXECUTABLE_IDを指定する</p>

## サンプル3

## EXECUTABLE\_IDからアクセスプランを取得するSQL

負荷の高いSQLを特定した後、解析のためにアクセスプランを取得する必要があります。  
EXPLAIN\_FROM\_SECTIONプロシージャを使用することで、指定したEXECUTABLE\_IDのExplain情報をExplain表に格納します。

ステートメント	出力結果
<pre>CALL EXPLAIN_FROM_SECTION( x'010000000000000003015020000000000000000000020020211110132548204182','M',null, 0,'EXPSHEMA',?,?,?,?)</pre> <p>Explain表のスキーマ名を指定しま す</p>	<p>出力パラメーターの値</p> <p>-----</p> <p>パラメーター名: EXPLAIN_SCHEMA パラメーター値: EXPSHEMA</p> <p>パラメーター名: EXPLAIN_REQUESTER パラメーター値: EXPSHEMA</p> <p>パラメーター名: EXPLAIN_TIME パラメーター値: 2021-12-10-02.45.11.849424</p> <p>パラメーター名: SOURCE_NAME パラメーター値: P1864868625</p> <p>パラメーター名: SOURCE_SCHEMA パラメーター値: SYSIBMADM</p> <p>(省略)</p>

4. こんな時、どうする?	1-4 リソース編① DBサーバーのCPU使用率を知りたい
概要	<p>MON_SAMPLE_WORKLOAD_METRICS表関数、またはENV_GET_SYSTEM_RESOURCES表関数で、CPU使用率の情報を取得することが可能です。  MON_SAMPLE_WORKLOAD_METRICS表関数の実行例は、「4. こんな時、どうする?」の「2-1 データベースの処理状況をリアルタイムでサマリーする」のサンプル2をご参照ください。</p>
モニター結果の例	<p>表関数を使用したCPU使用率取得方法は、次のページの「サンプル」を参照してください。</p>
詳細情報のリンク	<p>[Db2 WoC]ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す  <a href="https://www.ibm.com/docs/ja/db2woc?topic=erv-env-get-system-resources-table-function-return-system-information">https://www.ibm.com/docs/ja/db2woc?topic=erv-env-get-system-resources-table-function-return-system-information</a>  [Db2 WoC]MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics</a></p>
備考	

**サンプル****CPU使用率の監視をする**

ENV\_GET\_SYSTEM\_RESOURCES表関数を使用することで、CPU使用率の情報をリアルタイムに取得することが可能です。  
当関数では、CPU使用率以外のメモリーやOSの情報を取得することも可能です。

**ステートメント**

```
SELECT  
  CURRENT_TIMESTAMP AS CURRENT_TIMESTAMP,  
  SUBSTR(HOST_NAME,1,8) AS HOST_NAME,  
  CPU_TOTAL,  
  CPU_USAGE_TOTAL  
FROM TABLE(SYSPROC.ENV_GET_SYSTEM_RESOURCES())
```

**出力結果**

CURRENT_TIMESTAMP	HOST_NAME	CPU_TOTAL	CPU_USAGE_TOTAL
2021/10/19 8:12:16	dashmpp-	8	7



4. こんな時、どうする?

1-5 リソース編② 表スペースの使用率を知りたい

#### 概要

DMS表スペースの場合、TBSP\_UTILIZATION 管理ビュー、MON\_TBSP\_UTILIZATION表関数でモニタリングすることが可能です。  
SMS表スペースの場合、表スペースのコンテナが作成されているファイルシステムの使用率を監視することによって、表スペース使用率をモニタリングすることが出来ます。  
自動ストレージ表スペースの場合、自動ストレージパスが定義されているファイルシステムの使用率監視、  
または ADMIN\_GET\_STORAGE\_PATHS表関数を使用することによってモニタリングすることが可能です。

#### モニター結果の例

ここではDMS表スペースのサイズ、使用率、AUTORESIZE YESの場合の指定サイズをそれぞれの表スペースごとに表示しています。

この結果を取得するSQLの実例は、次ページの「サンプル1」を参照してください。

TBSP_NAME	TBSP_TOTAL_SIZE_KB	TBSP_UTILIZATION_PERCENT	TBSP_MAX_SIZE
DMSAUTO	10240	3.79	104857600
DMSTBS	10240	3.79	-

DMS表スペースの使用率が表示される

#### 詳細情報のリンク

[Db2 v11.5] TBSP\_UTILIZATION 管理ビュー - 表スペースの構成および使用率に関する情報の検索

<https://www.ibm.com/docs/ja/db2/11.5?topic=views-tbsp-utilization>

[Db2 WoC] MON\_TBSP\_UTILIZATION - すべての表スペースとすべてのデータベース・パーティションに関するモニタリング・メトリックの取得

<https://www.ibm.com/docs/ja/db2woc?topic=mv-mon-tbsp-utilization-retrieve-monitoring-metrics-all-table-spaces-all-database-partitions>

[Db2 WoC]ADMIN\_GET\_STORAGE\_PATHS 表関数 - 自動ストレージ・パス情報の取得

<https://www.ibm.com/docs/ja/db2woc?topic=aracp-admin-get-storage-paths-retrieve-automatic-storage-path-information>

#### 備考

### サンプル1

### DMS表スペースの使用率監視

MON\_TBSP\_UTILIZATION 管理ビューを使用することで、DMS表スペースの使用率を監視することが出来ます。  
 DMS表スペースがAUTORESIZE YESで固定値でMAXSIZEを指定して作成されていた場合、TBSP\_MAX\_SIZE (バイト数) 列によって、  
 最大サイズの値が分かるため、SQLを加工することで、MAXサイズまでの使用率を監視することも可能です。  
 SMS表スペースのレコードが不要の場合には、where条件で、TBSP\_TYPE = 'DMS' を指定してください。

ステートメント	出力結果																				
<pre>SELECT   SUBSTR(TBSP_NAME,1,12) AS TBSP_NAME,   TBSP_TOTAL_SIZE_KB,   TBSP_UTILIZATION_PERCENT,   TBSP_MAX_SIZE FROM SYSIBMADM.TBSP_UTILIZATION</pre>	<table border="1"> <thead> <tr> <th>TBSP_NAME</th> <th>TBSP_TOTAL_SIZE_KB</th> <th>TBSP_UTILIZATION_PERCENT</th> <th>TBSP_MAX_SIZE</th> </tr> </thead> <tbody> <tr> <td>USERSPACE1</td> <td>20777856</td> <td>36</td> <td>-1</td> </tr> <tr> <td>DSMSPACE</td> <td>131072</td> <td>85</td> <td>-1</td> </tr> <tr> <td>TS4MONITOR</td> <td>45184</td> <td>27</td> <td>524288000</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table> <div style="border: 1px solid #00a0e3; background-color: #e6f2ff; padding: 5px; margin-top: 10px; display: inline-block;">             SMSの場合には、使用率が-1と表示される         </div>	TBSP_NAME	TBSP_TOTAL_SIZE_KB	TBSP_UTILIZATION_PERCENT	TBSP_MAX_SIZE	USERSPACE1	20777856	36	-1	DSMSPACE	131072	85	-1	TS4MONITOR	45184	27	524288000	...	...	...	...
TBSP_NAME	TBSP_TOTAL_SIZE_KB	TBSP_UTILIZATION_PERCENT	TBSP_MAX_SIZE																		
USERSPACE1	20777856	36	-1																		
DSMSPACE	131072	85	-1																		
TS4MONITOR	45184	27	524288000																		
...	...	...	...																		

### サンプル2

### 自動ストレージパスの使用率監視

ADMIN\_GET\_STORAGE\_PATHS表関数を使用することによって、自動ストレージパスが定義されたファイルシステムの使用率を取得することが出来ます。  
 以下のSQLは、FS\_USED\_SIZEとFS\_TOTAL\_SIZEから使用率を計算したSQLです。このSQLは、ファイルシステムごとに1レコードの出力が得られます。

ステートメント	出力結果						
<pre>SELECT   SUBSTR(STORAGE_GROUP_NAME,1,15) AS STORAGE_GROUP_NAME,   SUBSTR(DB_STORAGE_PATH,1,15) AS DB_STORAGE_PATH,   ROUND(DECFLOAT(FS_USED_SIZE)/DECFLOAT(FS_TOTAL_SIZE)*100,2) AS   FS_USED_PERCENT FROM TABLE(ADMIN_GET_STORAGE_PATHS(NULL, -1))</pre>	<table border="1"> <thead> <tr> <th>STORAGE_GROUP_NAME</th> <th>DB_STORAGE_PATH</th> <th>FS_USED_PERCENT</th> </tr> </thead> <tbody> <tr> <td>IBMSTOGROUP</td> <td>/head</td> <td>80</td> </tr> </tbody> </table>	STORAGE_GROUP_NAME	DB_STORAGE_PATH	FS_USED_PERCENT	IBMSTOGROUP	/head	80
STORAGE_GROUP_NAME	DB_STORAGE_PATH	FS_USED_PERCENT					
IBMSTOGROUP	/head	80					

4. こんな時、どうする？	1-6 リソース編③ メモリーの使用量を把握する
---------------	--------------------------

概要	<p>Db2が使用するメモリーは、インスタンス共有メモリー、データベースメモリー、アプリケーションメモリー、エージェント・プライベート・メモリーに分類されます。メモリー領域の使用量は、MON_GET_MEMORY_SET 表関数 から確認することができます。</p> <p>メモリー・プールの使用量は MON_GET_MEMORY_POOL 表関数から 確認することができます。MON_GET_MEMORY_POOL表関数を使用した、メモリー・プール使用量 取得の実行例は、「3. 稼働状況の監視」の「4-3. メモリー使用量(メモリーの種類ごと)」のサンプルをご参照ください。</p> <p>また、applheap不足が発生した際は、MON_GET_MEMORY_POOL 表関数 からapplhandleごとのメモリー・プール使用量を取得し、アプリケーション・メモリーの使用量 増大の原因となっているセッションを特定します。</p>
詳細情報のリンク	<p>[Db2 WoC] MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-memory-set-get-memory-set-information">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-memory-set-get-memory-set-information</a></p> <p>[Db2 WoC] MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-memory-pool-get-memory-pool-information">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-memory-pool-get-memory-pool-information</a></p>
備考	

## サンプル 1

## 現行のデータベース・メンバーのメモリー領域 使用状況 取得

各メモリー領域の使用容量は、MON\_GET\_MEMORY\_SET 表関数 から取得することができます。

ここでは、現行のデータベース・メンバーでのメモリー領域に関する情報を取得するように、3つ目の表関数パラメーター に **-1** を指定しています。  
すべてのアクティブ・メンバーについての情報を取得したい場合は、**-2** を指定します。  
また、各メンバーの FCM メモリーに関して返されるメトリックは、同じ共有メモリー・セットについての情報になります。  
FCM メモリーのメトリックを調べる場合は、1つのメンバーを指定して データを調べるようにしてください。

### ステートメント

```
SELECT
  VARCHAR(MEMORY_SET_TYPE, 20) as SET_TYPE,
  MEMBER,
  VARCHAR(DB_NAME, 20) AS DBNAME,
  MEMORY_SET_SIZE,
  MEMORY_SET_COMMITTED,
  MEMORY_SET_USED,
  MEMORY_SET_USED_HWM
FROM TABLE(MON_GET_MEMORY_SET(NULL, CURRENT_SERVER, -1))
ORDER BY SET_TYPE, MEMBER
```

### 出力結果

SET_TYPE	MEMBER	DBNAME	MEMORY_SET_SIZE	MEMORY_SET_COMMITTED	MEMORY_SET_USED	MEMORY_SET_USED_HWM
APPLICATION	0	BLUDB	60544	37696	35072	342208
DATABASE	0	BLUDB	24941184	14361152	12081344	14124736
DBMS	0		747008	665536	664512	665536
FCM	0		4786176	1511040	1511040	2519360
FMP	0		22784	1280	1280	1280
PRIVATE	0		231936	228544	185984	344896

## サンプル 2

## アプリケーション・メモリーの使用量が多いセッションを特定する

applheapが不足するなど問題が発生した場合は、  
MON\_GET\_MEMORY\_POOL表関数を使用して、現行データベース・メンバーにおいて、アプリケーション・メモリーの使用量が多いセッションを特定します。

アプリケーション・ハンドル (APPLICATION\_HANDLE) ごとのメモリー使用量 (MEMORY\_POOL\_USED) を取得し、  
メモリー使用量が多い順にソート、上位10個のリストを確認します。

ここでは、現行のデータベース・メンバーでのメモリー領域に関する情報を取得するように、3つ目の表関数パラメーターに **-1** を指定しています。  
また、アプリケーション・メモリー・セットの情報を取得するため、  
WHERE句でメモリー・セット・タイプ (MEMORY\_SET\_TYPE) が **'APPLICATION'** のデータを返すように絞り込みを行います。

### ステートメント

```
SELECT
  APPLICATION_HANDLE,
  MEMORY_POOL_USED
FROM TABLE(
  MON_GET_MEMORY_POOL(NULL, CURRENT_SERVER, -1))
WHERE APPLICATION_HANDLE IS NOT NULL AND MEMORY_SET_TYPE='APPLICATION'
ORDER BY MEMORY_POOL_USED DESC LIMIT 10
```

#### ■各項目の説明

APPLICATION\_HANDLE  
MEMORY\_POOL\_USED

:アプリケーション・ハンドル  
:メモリー・プール使用量

### 出力結果

APPLICATION_HANDLE	MEMORY_POOL_USED
34307	3200
57	1024
4309	704
3088	320
10522	192
7793	192
7838	192
7990	192
6218	128
5099	128

メモリー使用量が多いセッションの  
アプリケーション・ハンドルが確認できる

4. こんな時、どうする?	1-7 リソース編④ ログの使用量を知りたい
概要	<p>ログファイルの使用状況は、MON_GET_TRANSACTION_LOG表関数で確認することが出来ます。</p>
詳細情報のリンク	<p>[Db2 WoC] MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-transaction-log-table-function-get-log-information">https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-transaction-log-table-function-get-log-information</a></p>
備考	

**サンプル****MON\_GET\_TRANSACTION\_LOG表関数によってログ情報を取得するSQL**

MON\_GET\_TRANSACTION\_LOG表関数を使用してログ情報関連を取得することができます。  
以下のSQLはMEMBER毎に、ログファイル使用率を出しています。

**ステートメント**

```
SELECT  
  MEMBER,  
  CASE WHEN TOTAL_LOG_AVAILABLE < 100  
    THEN null  
    ELSE CAST (100*TOTAL_LOG_USED/TOTAL_LOG_AVAILABLE AS DECIMAL(4,1))  
  END AS LOG_USED_PERCENT  
FROM TABLE(MON_GET_TRANSACTION_LOG(-1))  
ORDER BY MEMBER ASC
```

**出力結果**

MEMBER	LOG_USED_PERCENT
0	4.0

4. こんな時、どうする?	2-1. データベースの処理状況をリアルタイムでサマリーする
---------------	--------------------------------

**概要**

MON\_GET表関数は、スナップショットと同様に「ある時点」でのモニター項目の値を一括して取得するモニタリング機能です。そのため、複数回の取得を行った上で差分や時系列の変化を分析します。

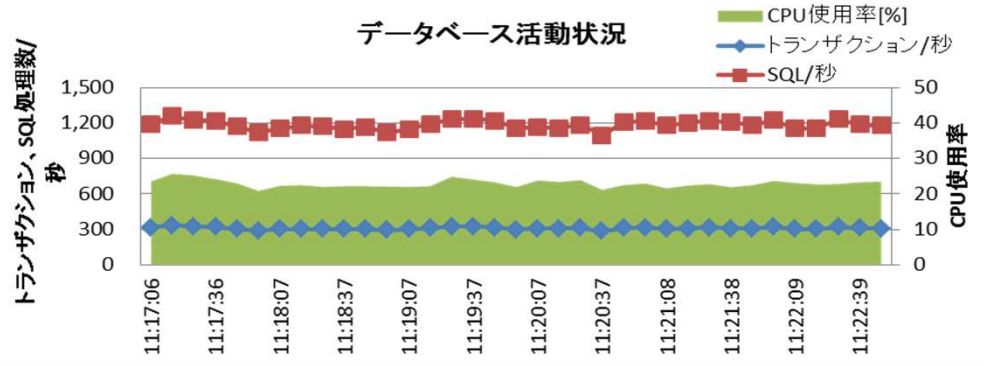
それに対して、「一定期間の処理状況」を収集でき、リアルタイムでの調査に活用できるMONREPORTモジュールやMON\_SAMPLE\_WORKLOAD\_METRICS表関数も提供されています。これらのモニタリング機能では、SQL実行時に「何秒間サンプリングするか」を指定してすることで、SQLが指定された期間待機し、その間の処理状況を収集、出力します。

また、対話的なインターフェースからデータベースの処理状況をモニタリングする場合には、db2topを使用することも有効です。「dオプション」でデータベースの活動状況サマリーを出力できます。

**モニター結果の例**

ここではOLTP的な負荷をかけたデータベースに対して、MON\_SAMPLE\_WORKLOAD\_METRICS（サンプルその2）を使用して、10秒ごとの処理状況を収集した結果をグラフ化しています。

SQLの処理スループットが1200/秒程度、トランザクションの処理スループットは300件/秒程度で推移しており、その間のCPU使用率は20-25%程度で、大きな変動が無いことが読み取れます。



**詳細情報のリンク**

[Db2 WoC] DBSUMMARY procedure - Generate a summary report of system and application performance metrics  
<https://www.ibm.com/docs/ja/db2woc?topic=mm-dbsummary-procedure-generate-summary-report-system-application-performance-metrics>  
 [Db2WoC] MON\_SAMPLE\_WORKLOAD\_METRICS - サンプルの取得  
<https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-sample-workload-metrics-get-sample-workload-metrics>

**備考**





## サンプル 1

## 指定した期間（秒）のデータベース処理状況をレポート形式で取得するSQL

メトリックを使用してレポート形式の処理状況を取得できるMONREPORTモジュールは、指定した期間の活動状況をサマリーするモニタリング機能です。その中でも、MONREPORT.DBSUMMARYモジュールは、データベース全体の処理状況を一覧性のある形で手軽に取得できる点で非常に優れています。この出力結果例では、データベース全体の活動状況を示すセクションを抜粋しています。MONREPORT.DBSUMMARYモジュールの出力全体は長大であるため、詳細については参考資料で紹介したMONREPORTの説明資料を参照してください。

### ステートメント

```
CALL MONREPORT.DBSUMMARY(10)
```

### 出力結果

```
=====  
Part 1 - System performance  
Work volume and throughput  
-----
```

```
Per second Total  
-----
```

```
TOTAL_APP_COMMITS 6 68  
ACT_COMPLETED_TOTAL 10 101  
APP_RQSTS_COMPLETED_TOTAL 25 256  
TOTAL_CPU_TIME = 381501  
TOTAL_CPU_TIME per request = 1490  
Row processing  
ROWS_READ/ROWS_RETURNED = 12 (2724/223)  
ROWS_MODIFIED = 807  
Wait times  
-----
```

```
-- Wait time as a percentage of elapsed time --  
% Wait time/Total time  
-----
```

```
For requests 7 981/13292  
For activities 1 162/10582  
-- Time waiting for next client request --  
  
. . .
```

サンプリング期間中のコミット数、SQL完了数、リクエスト数（カーソルOPEN、FETCHなど）の毎秒とトータル

CPU使用時間と、リクエストあたりのCPU使用時間

SELECT行数に対する内部読み込み数と更新レコード数  
SQL処理効率の指標として使用できる

データベース内部での「待ち」の発生状況。  
SQLあたりとリクエストあたりの数字

## サンプル2

## 指定した期間（秒）のデータベース処理状況をリアルタイムで取得するSQL

MON\_SAMPLE\_WORKLOAD\_METRICS表関数を使用して、一定期間のデータベース処理状況を取得するSQLです。  
下記の例では、サンプリング期間を10秒に指定して主要なパフォーマンス指標をピックアップするSQLを、繰り返し実行することで、10秒ごとの処理状況を時系列で取得できるようにしています。  
なお、パーティション・データベース環境やDB2 pureScale環境など、一つのデータベースが複数のサーバー上にまたがって存在する環境では、CPU\_UTILは各サーバーのCPU使用率の合計となります。

### ステートメント

```
SELECT
  SUBSTR(DB_NAME,1,12)          AS DB_NAME,
  CURRENT_TIMESTAMP            AS TIMESTAMP,
  DECIMAL(SUM(UOW_THROUGHPUT),8,1) AS UOW_THROUGHPUT,
  DECIMAL(SUM(ACT_THROUGHPUT),8,1) AS ACT_THROUGHPUT,
  DECIMAL(AVG(UOW_LIFETIME_AVG),6,1) AS UOW_LIFETIME_AVG,
  SUM(ACT_COMPLETED_TOTAL)       AS ACT_COMPLETED_TOTAL,
  INT(SUM(TOTAL_CPU_TIME/1000))  AS CPU_TIME_MS,
  DECIMAL(SUM(CPU_UTILIZATION),3,1) AS CPU_UTIL
FROM TABLE(MON_SAMPLE_WORKLOAD_METRICS
  (NULL, NULL, NULL, 10, -2)) AS T
WHERE ACT_COMPLETED_TOTAL <> 0
GROUP BY SUBSTR(DB_NAME,1,12), CURRENT_TIMESTAMP
```

### 出力結果

DB_NAME	TIMESTAMP	UOW_THROUGHPUT	ACT_THROUGHPUT	UOW_LIFETIME_AVG	ACT_COMPLETED_TOTAL	CPU_TIME_MS	CPU_UTIL
BLUDB	44:28.0	2.1	1.7	1	18	15	0

毎秒のトランザクション件数 (UOW\_THROUGHPUT) と、SQL処理件数 (ACT\_THROUGHPUT) をデータベース単位で集計している

トランザクションの平均生存時間 (ms)

サンプル期間中のトータルSQL実行数

データベース毎のCPU使用時間(ms)とCPU使用率

#### 4. こんな時、どうする?

#### 2-2. 表や索引サイズの実態を把握する

##### 概要

表、索引の容量は、システム・カタログ表のADMINTABINFO管理ビューによって確認することができます。  
RUNSTATSを実施することなく、常に最新の情報を取得することが可能です。  
なお、SYSCAT.TABLESのFPAGESやNLEAFでも容量の確認は可能ですが、最新の情報を取得するためにはRUNSTATSが必要となります。

##### モニター結果の例

ADMINTABINFO管理ビューをSELECTすることで、  
表が占有しているエクステントをKB換算で表示します。  
データ、索引、LONG、LOB、XMLそれぞれの容量を取得することが可能です。  
この結果を取得するSQLの実例は、次ページの「サンプル」を参照してください。

TABSCHEMA	TABNAME	DATA	INDEX	LONG	LOB	XML	...
SYSIBM	SYSTABLES	192	112	0	2096	0	
SYSIBM	SYSCOLUMNS	1008	528	0	0	0	
SYSIBM	SYSINDEXES	128	80	0	64	0	
SYSIBM	SYSVIEWS	64	48	0	560	0	
SYSIBM	SYSVIEWDEP	48	96	0	0	0	
SYSIBM	SYSPLAN	64	48	0	1328	0	
SYSIBM	SYSPLANDEP	48	64	0	64	0	
SYSIBM	SYSSECTION	64	64	0	15936	0	
SYSIBM	SYSSTMT	64	64	0	560	0	

単位はKB

##### 詳細情報のリンク

[Db2 WoC] ADMINTABINFO 管理ビューおよび ADMIN\_GET\_TAB\_INFO 表関数 - 表のサイズおよび状態に関する情報の検索  
<https://www.ibm.com/docs/ja/db2woc?topic=aracp-admin-get-tab-info-retrieve-table-size-state-information>

##### 備考

**サンプル****表ごとに物理サイズをリストするSQL**

各表が占有しているエクステントをKB換算で表示します。  
 データ、索引、LONG、LOB、XMLそれぞれの容量を取得することが可能です。  
 TABSCHEMAやTABNAMEを条件にして絞り込むことも可能です。

**ステートメント**

```
SELECT
  SUBSTR(TABSCHEMA,1,10) TABSCHEMA,
  SUBSTR(TABNAME,1,20) TABNAME,
  DATA_OBJECT_P_SIZE DATA,
  INDEX_OBJECT_P_SIZE INDEX,
  LONG_OBJECT_P_SIZE LONG,
  LOB_OBJECT_P_SIZE LOB,
  XML_OBJECT_P_SIZE XML
FROM SYSIBMADM.ADMINTABINFO;
```

**出力結果**

TABSCHEMA	TABNAME	DATA	INDEX	LONG	LOB	XML
SYSIBM	SYSTABLES	1152	1152	0	2355328	0
SYSIBM	SYSCOLUMNS	10752	4992	0	512	0
SYSIBM	SYSINDEXES	896	768	0	512	0
SYSIBM	SYSVIEWS	256	256	0	2176	0
SYSIBM	SYSVIEWDEP	384	768	0	0	0

4. こんな時、どうする?

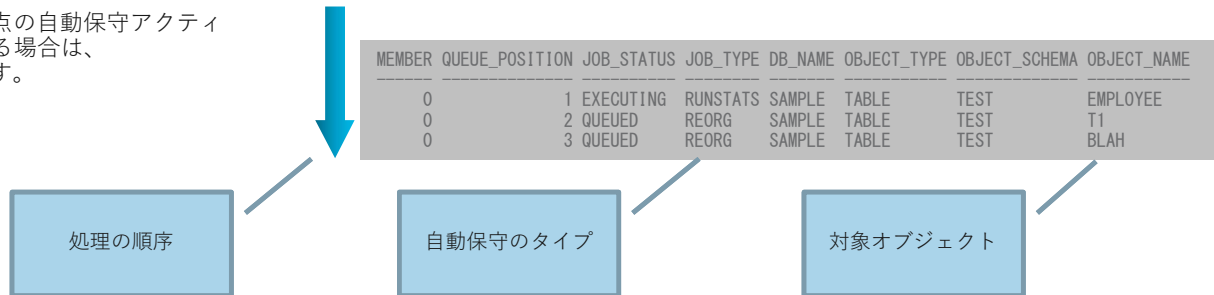
2-3 自動保守の実行状況を知りたい

### 概要

MON\_GET\_AUTO\_MAINT\_QUEUE 表関数を利用することで、自動保守(RUNSTATS、REORG、BACKUP、STATSPROFILE)の現時点のアクティビティの状況(キューに入っている自動保守ジョブ、実行中の自動保守ジョブ)をリストすることができます。さらに接続しているデータベースの自動統計収集(RUNSTATS)の状況のみ取得したい場合は、MON\_GET\_AUTO\_RUNSTATS\_QUEUE表関数が利用できます。

### モニター結果の例

MON\_GET表関数を利用することで、右のように現時点の自動保守アクティビティを確認できます。過去の自動保守常用を確認する場合は、db2diag.logやLIST HISTORYを参照する必要があります。



### 詳細情報のリンク

[Db2WoC] MON\_GET\_AUTO\_MAINT\_QUEUE 表関数 - 自動保守ジョブに関する情報の取得

<https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-auto-maint-queue-table-function-get-information-automatic-maintenance-jobs>

[Db2WoC] MON\_GET\_AUTO\_RUNSTATS\_QUEUE 表関数 - 評価のためにキューに入っているオブジェクトに関する情報の取得

<https://www.ibm.com/docs/ja/db2woc?topic=mpf-mon-get-auto-runstats-queue-table-function-retrieve-information-about-objects-queued-evaluation>

### 備考

## サンプル1

## 現時点の全ての自動保守ジョブの登録、実行情報をリストするSQL

MON\_GET\_AUTO\_MAINT\_QUEUE表関数を使用して、現時点の全ての自動保守ジョブの登録、実行情報をリストするSQLです。  
下記の例では、キュー内に存在するジョブの状況、タイプ、オブジェクト名などを優先度順にリストしています。

### ステートメント

```
SELECT  
  MEMBER,  
  QUEUE_POSITION,  
  JOB_STATUS,  
  JOB_TYPE,  
  VARCHAR(DB_NAME, 10) AS DB_NAME,  
  OBJECT_TYPE,  
  VARCHAR(OBJECT_SCHEMA, 10) AS OBJECT_SCHEMA,  
  VARCHAR(OBJECT_NAME, 10) AS OBJECT_NAME  
FROM TABLE(MON_GET_AUTO_MAINT_QUEUE()) AS t  
ORDER BY MEMBER, QUEUE_POSITION ASC;
```

#### ■各項目の説明

MEMBER : DBメンバー  
QUEUE\_POSITION : キュー内のジョブの位置  
JOB\_STATUS : ジョブ状況 (QUEUED, EXECUTING)  
JOB\_TYPE : ジョブタイプ (RUNSTATS, REORG, BACKUP, STATSPROFILE)  
DB\_NAME : 対象DB名  
OBJECT\_TYPE : オブジェクト種類 (DATABASE, TABLE, NICKNAME, VIEW)  
OBJECT\_SCHEMA : オブジェクトスキーマ  
OBJECT\_NAME : オブジェクト名

### 出力結果

MEMBER	QUEUE_POSITION	JOB_STATUS	JOB_TYPE	DB_NAME	OBJECT_TYPE	OBJECT_SCHEMA	OBJECT_NAME
0	0	EXECUTING	RUNSTATS	BLUDB	TABLE	BLUADMIN	ACTIVITYST

## サンプル2

## 現時点の接続先DBの自動統計収集ジョブの登録、実行情報をリストするSQL

MON\_GET\_AUTO\_RUNSTATS\_QUEUE表関数を使用して、現時点の接続先の自動統計収集ジョブの登録、実行情報をリストするSQLです。下記の例では、DB内のキュー内に存在するRUNSTATSジョブの状況、タイプ、オブジェクト名などを優先度順にリストしています。

### ステートメント

```
SELECT QUEUE_POSITION,  
       OBJECT_TYPE,  
       OBJECT_STATUS,  
       VARCHAR(OBJECT_SCHEMA, 10) AS OBJECT_SCHEMA,  
       VARCHAR(OBJECT_NAME, 10) AS OBJECT_NAME  
FROM TABLE(MON_GET_AUTO_RUNSTATS_QUEUE())  
ORDER BY QUEUE_POSITION ASC;
```

#### ■各項目の説明

QUEUE\_POSITION : キュー内のジョブの位置  
OBJECT\_TYPE : オブジェクト種類 (DATABASE, TABLE, NICKNAME, VIEW)  
OBJECT\_STATUS :  
    EVOLUTION\_PENDING - オブジェクトを調べて統計更新が必要か判断する必要有り  
    JOB\_SUBMITTED - オブジェクトの統計更新が必要と判断し、ジョブをサブミット  
OBJECT\_SCHEMA : オブジェクトスキーマ  
OBJECT\_NAME : オブジェクト名

### 出力結果

QUEUE_POSITION	OBJECT_TYPE	OBJECT_STATUS	OBJECT_SCHEMA	OBJECT_NAME
1	TABLE	JOB_SUBMITTED	BLUADMIN	ACTIVITYST
2	TABLE	EVALUATION_PENDING	OGAWA	T1
3	TABLE	EVALUATION_PENDING	OGAWA	T2
...				

4. こんな時、どうする?	2-4 ロック待ち/ロックタイムアウト/デッドロックの原因となるSQLを特定する
---------------	--

概要	<p>ロックによる長時間待ち/ロックタイムアウト/デッドロックの原因となるSQLは、ロック・イベント・モニターの出力結果によって確認できます。</p> <p>ロック・イベント・モニターの取得手順は以下のとおりです。</p> <ol style="list-style-type: none"> <li>(1) イベント・モニターを定義する。</li> <li>(2) イベント・モニターを活動化する。</li> <li>(3) イベント・モニターでキャプチャーするロッキング・イベントのレベルを指定する。</li> <li>(4) (データベースに対する処理が行われ、ロックタイムアウト、長時間ロック、デッドロックが発生する。)</li> <li>(5) イベント・モニター・キャプチャー・レベルのリセット (必要に応じて)</li> <li>(6) イベント・モニターを非活動化する。 (必要に応じて)</li> <li>(7) モニター・データを参照する。</li> </ol> <p>ロック・イベント・モニターの書き出し先には、未フォーマット表と通常表があり、未フォーマット表はイベント書き出し負荷が低いというメリットがある一方、通常表はSQLによって照会できる利点があります。ここでは、通常表への出力を扱います。</p>
手順およびモニター結果の例	<p>詳細は、以下のページを参照してください。</p>
詳細情報のリンク	
備考	<p>[Db2 WoC] ロック・イベントおよびデッドロック・イベントのモニター  <a href="https://www.ibm.com/docs/ja/db2woc?topic=events-lock-deadlock-event-monitoring">https://www.ibm.com/docs/ja/db2woc?topic=events-lock-deadlock-event-monitoring</a></p>



## 手順 その1

## ロック・イベント・モニターの出力を通常表に出力/レポートする

ロック・イベント・モニターは、長時間待たされていたロック、ロックタイムアウト、デッドロックの情報をキャプチャーできます。

### 手順

- (1) イベント・モニターを定義する。  
CREATE EVENT MONITOR *lcktevmon* FOR LOCKING WRITE TO TABLE  
(*lcktevmon* は任意のイベント・モニター名)
- (2) イベント・モニターを活動化する。  
SET EVENT MONITOR *lcktevmon* STATE 1
- (3) イベント・モニターでキャプチャーするロッキング・イベントのレベルを指定する。  
ロック・イベント・モニターでは、①指定した時間以上経過したロック待機、②ロックタイムアウト、③デッドロック の3種類のイベントをキャプチャー可能ですが、実際にイベントを収集するためには それぞれ設定が必要です。  
キャプチャーするかないかの設定は、WORKLOAD単位またはDB全体で行うことができ、WORKLOADを使用している環境においては、システム全体に影響を与えないWORKLOADによる設定が推奨されます。(ワークロード・マネージャを明示的に使用していない環境でもデフォルト・ワークロードSYSDEFAULTUSERWORKLOADに対する定義として使用可能です。)
- ①ロック待機
  - ・WORKLOADの例：  
ALTER WORKLOAD *sysdefaultuserworkload* COLLECT LOCK WAIT DATA WITH HISTORY AND VALUES FOR LOCKS WAITING MORE THAN 15 SECONDS
  - ・DB全体 (構成パラメータ設定)の例：  
UPDATE DB CFG USING MON\_LOCKWAIT HIST\_AND\_VALUES MON\_LW\_THRESH 15000000  
(MON\_LW\_THRESHで指定する待ち時間はマイクロ秒)
- ②ロックタイムアウト
  - ・WORKLOADの例：  
ALTER WORKLOAD *sysdefaultuserworkload* COLLECT LOCK TIMEOUT DATA WITH HISTORY AND VALUES
  - ・DB全体 (構成パラメータ設定)の例：  
UPDATE DB CFG USING MON\_LOCKTIMEOUT HIST\_AND\_VALUES
- ③デッドロック
  - ・WORKLOADの例：  
ALTER WORKLOAD *sysdefaultuserworkload* COLLECT DEADLOCK DATA WITH HISTORY AND VALUES
  - ・DB全体 (構成パラメータ設定)の例：  
UPDATE DB CFG USING MON\_DEADLOCK HIST\_AND\_VALUES
- (4) (データベースに対する処理が行われ、長時間ロック、ロックタイムアウト、デッドロックが発生する。)

## 手順 2

## ロック・イベント・モニターの出力を通常表に出力/レポートする

(5) イベント・モニター・キャプチャー・レベルのリセット (必要に応じて)

①ロック待機

・ WORKLOADの例:

```
ALTER WORKLOAD sysdefaultuserworkload COLLECT LOCK WAIT DATA NONE
```

・ DB全体 (構成パラメータ設定)の例:

```
UPDATE DB CFG USING MON_LOCKWAIT NONE  
(MON_LW_THRESHで指定する待ち時間はマイクロ秒)
```

②ロックタイムアウト

・ WORKLOADの例:

```
ALTER WORKLOAD sysdefaultuserworkload COLLECT LOCK TIMEOUT DATA NONE
```

・ DB全体 (構成パラメータ設定)の例:

```
UPDATE DB CFG USING MON_LOCKTIMEOUT NONE
```

③デッドロック

・ WORKLOADの例:

```
ALTER WORKLOAD sysdefaultuserworkload COLLECT DEADLOCK DATA NONE
```

・ DB全体 (構成パラメータ設定)の例:

```
UPDATE DB CFG USING MON_DEADLOCK NONE
```

(6) イベント・モニターを非活動化する。 (必要に応じて)

```
SET EVENT MONITOR lcktevmon STATE 0
```

(7) モニター・データを参照する。

ロック・イベント・モニターによって以下の5つの表が作成されるので、必要な情報をSQLで照会する。

```
LOCK_lcktevmon  
LOCK_PARTICIPANTS_lcktevmon  
LOCK_PARTICIPANT_ACTIVITIES_lcktevmon  
LOCK_ACTIVITY_VALUES_lcktevmon  
CONTROL_lcktevmon
```

## 詳細情報のリンク

[Db2 WoC] ロック・イベント・モニター用に表に書き込まれる情報

<https://www.ibm.com/docs/ja/db2woc?topic=monitors-information-written-tables-locking-event-monitor>

## サンプル 1

## ロック・イベント表からDEADLOCKイベントの概要情報を取得する

ロック・イベント表の組み合わせによって、ロック・イベント・レポートと同等な情報の取得が可能です。  
まず、ロック発生種類を指定して、ロック発生時刻と、ロック参加アプリケーションのサマリ情報を得るSQLを実行します。

### ステートメント

```
SELECT int(1.EVENT_ID)          as EVENT_ID,  
       1.EVENT_TIMESTAMP,  
       substr(1.EVENT_TYPE,1,10) as TYPE,  
       p.PARTICIPANT_NO,  
       substr(p.APPL_ID,1,28)    as APPLID,  
       substr(p.APPL_NAME,1,12)  as APPLNAME,  
       int(p.APPLICATION_HANDLE) as APPLHANDLE,  
       1.ROLLED_BACK_PARTICIPANT_NO as VICTIM  
from LOCK_lcktevmon 1, LOCK_PARTICIPANTS_lcktevmon p  
where 1.EVENT_TYPE='DEADLOCK' AND 1.EVENT_ID=p.EVENT_ID
```

### 出力結果

EVENT_ID	EVENT_TIMESTAMP	TYPE	PARTICIPANT_NO	APPLID	APPLNAME	APPLHANDLE	VICTIM
2	2021/12/8 16:29	DEADLOCK	2	172.xx.xx.xx.xxxxx.211208052	db2bp	4342	2
1	2021/12/8 16:20	DEADLOCK	1	172.xx.xx.xx.xxxxx.211208070	db2bp	5714	2
2	2021/12/8 16:29	DEADLOCK	1	172.xx.xx.xx.xxxxx.211208070	db2bp	5714	2
1	2021/12/8 16:20	DEADLOCK	2	172.xx.xx.xx.xxxxx.211208052	db2bp	4342	2

DEADLOCKイベントが2件(ID=1, 2) あり

## サンプル 2

## ロック・イベント表からDEADLOCKイベントの概要情報を取得する (1/2)

サマリー情報を元に、詳細を調査すべきデッドロックのEVENT\_IDを特定し、DEADLOCK原因のオブジェクト、SQL情報を得るSQLを実行します。  
このSQLは、デッドロックに参加したアプリケーション数分のレコードを出力し、それぞれのアプリケーションについて実行時の情報や、ロックを保持しているSQL、ロックを要求しているSQLを一覧できるように作成されています。  
特定のEVENT\_IDを絞り込み条件として指定する必要があるため、下記SQLの「**P.EVENT\_ID=1**」を適宜変更して実行してください。  
また、SQL中に4カ所記述されている表の参照では、表名の末尾に付いている「**\_lcktevmon**」は、イベントモニターの名前によって変動します。  
こちらも、作成したイベントモニターの名前にあわせて適宜変更してください。

### ステートメント

```
WITH
  PAST_ACT AS (SELECT EVENT_ID, ACTIVITY_TYPE, PARTICIPANT_NO, STMT_TEXT,EFFECTIVE_ISOLATION, STMT_FIRST_USE_TIME
               FROM LOCK_PARTICIPANT_ACTIVITIES_lcktevmon WHERE ACTIVITY_TYPE='PAST'),
  CURR_ACT AS (SELECT EVENT_ID, ACTIVITY_TYPE, PARTICIPANT_NO, STMT_TEXT,EFFECTIVE_ISOLATION, STMT_FIRST_USE_TIME
               FROM LOCK_PARTICIPANT_ACTIVITIES_lcktevmon WHERE ACTIVITY_TYPE='CURRENT'),
  VICTIM AS (SELECT EVENT_ID, ROLLED_BACK_PARTICIPANT_NO FROM LOCK_lcktevmon)
SELECT INT(P.EVENT_ID)           AS EVENT_ID,
       P.PARTICIPANT_NO,
       CASE WHEN P.PARTICIPANT_NO=VICTIM.ROLLED_BACK_PARTICIPANT_NO THEN 'VICTIM' ELSE '' END AS VICTIM ,
       INT(P.APPLICATION_HANDLE) AS APPLHANDLE,
       SUBSTR(P.APPL_ID,1,25)     AS APPLID,
       SMALLINT(P.LOCK_MODE)     AS LOCKMODE,
       SMALLINT(P.LOCK_MODE_REQUESTED) AS REQUESTED_MODE,
       SUBSTR(P.LOCK_OBJECT_TYPE,1,10) AS OBJ_TYPE,
       LOCK_ESCALATION           AS LOCK_ESCL,
       SUBSTR(PAST_ACT.STMT_TEXT, 1, 60) AS LOCK_HOLDING_SQL,
       PAST_ACT.EFFECTIVE_ISOLATION AS LOCK HOLDER_ISO,
       PAST_ACT.STMT_FIRST_USE_TIME AS LOCK HOLDER_STMT_START,
       SUBSTR(P.TABLE_NAME,1,15) AS LOCK_REQUESTED_TABLE,
       SUBSTR(CURR_ACT.STMT_TEXT, 1, 60) AS LOCK_REQUESTED_SQL,
       CURR_ACT.EFFECTIVE_ISOLATION AS LOCK_REQUESTER_ISO
FROM LOCK_PARTICIPANTS_lcktevmon P, CURR_ACT, PAST_ACT, VICTIM
WHERE P.EVENT_ID=1
      AND P.EVENT_ID=PAST_ACT.EVENT_ID AND P.EVENT_ID=CURR_ACT.EVENT_ID AND P.EVENT_ID=VICTIM.EVENT_ID
      AND P.PARTICIPANT_NO=PAST_ACT.PARTICIPANT_NO AND P.PARTICIPANT_NO=CURR_ACT.PARTICIPANT_NO
ORDER BY P.PARTICIPANT_NO
```

ACTIVITY表への参照は  
WITH句を利用して切り出している  
PAST\_ACTがロックを持つACTIVITY、  
CURR\_ACTがロックを要求する  
ACTIVITYを表す

デッドロックに関するアプリケーションの情報  
ロールバックされたアプリがどちら側か、  
ロックエスカレーションの有無などが確認できる

デッドロックに関するSQLの情報  
実行済みでロックを保持しているSQLと、  
ロックを要求したSQLの両方が出力される

## サンプル 2

## ロック・イベント表からDEADLOCKイベントの概要情報を取得する (2/2)

前ページのSQLを実行した結果のサンプルです。横幅が長いので、3つの表に分割して表示しています。実際の取得時には、CSVファイルにEXPORTして、EXCELなどから参照されることをおすすめします。

### 出力結果

EVENT_ID	PARTICIPANT_NO	VICTIM	APPLHANDLE	APPLID	LOCKMODE	REQUESTED_MODE	OBJ_TYPE	LOCK_ESCL
1	1		5714	172.xx.xx.xx.xxxxx.211208	5	5	CATALOG	NO



ロックエスカレーション  
発生有無

LOCK_HOLDING_SQL	LOCK HOLDER_ISO	LOCK HOLDER_STMT_START
update t1 set c1=6 where c1=4	CS	2021/12/8 16:19



すでに実行されていて、ロックを保持しているSQL文。  
分離レベルCSで実行されている

LOCK_REQUESTED_TABLE	LOCK_REQUESTED_SQL	LOCK_REQUESTER_ISO
	update t1 set c1=4 where c1=1	RR

ロックを要求しているSQL文。  
分離レベルRRで実行されている

4. こんな時、どうする?	2-5 表のパーティションごとのレコード数の偏りを調査する
---------------	-------------------------------

<b>概要</b>	<p>Db2 WoCは、複数パーティションによる並列処理を行っており、表のレコードは、分散キーの値によってパーティションに配分されています。分散キーの選択が悪いと、パーティションにレコードが均等に配分されず、特定のパーティションに偏ることがあります。レコードが偏ると、並列処理のメリットを得ることができないので、表内のレコードが偏っていないか知っておくことは重要です。</p>
<b>モニター結果の例</b>	<p>詳細は、以下のページを参照してください。</p>
<b>詳細情報のリンク</b>	<p>[Db2WoC] DBPARTITIONNUM スカラー関数  <a href="https://www.ibm.com/docs/ja/db2woc?topic=functions-dbpartitionnum">https://www.ibm.com/docs/ja/db2woc?topic=functions-dbpartitionnum</a></p> <p>[Db2WoC] MON_TBSP_UTILIZATION - すべての表スペースとすべてのデータベース・パーティションに関するモニタリング・メトリックの取得  <a href="https://www.ibm.com/docs/ja/db2woc?topic=mv-mon-tbsp-utilization-retrieve-monitoring-metrics-all-table-spaces-all-database-partitions">https://www.ibm.com/docs/ja/db2woc?topic=mv-mon-tbsp-utilization-retrieve-monitoring-metrics-all-table-spaces-all-database-partitions</a></p>
<b>備考</b>	

## サンプル1

## 表単位で各DBパーティションのレコード件数を確認する

パーティションごとの表データのレコード件数をカウントし、データの分散具合を確認します。  
DBPARTITIONNUM スカラー関数を使用して、行のデータベース・パーティション番号を取得することができます。

ここでは以下ステートメントで作成された表 (SAMP.CUSTOMER) のDBパーティションごとのレコード件数を確認します。  
SAMP.CUSTOMER表の分散キーは CUST\_ID列としています。

### <表作成ステートメント>

```
CREATE TABLE SAMP.CUSTOMER(  
  CUST_ID INT,  
  NAME VARCHAR(80),  
  GENDER CHAR(5)  
)DISTRIBUTE BY HASH(CUST_ID)
```

## ステートメント

```
SELECT  
  DBPARTITIONNUM(CUST_ID) AS DBP_NUM,  
  COUNT(*) AS CNT  
FROM SAMP.CUSTOMER  
GROUP BY DBPARTITIONNUM(CUST_ID)  
ORDER BY DBPARTITIONNUM(CUST_ID)
```

### ■各項目の説明

- ・ DBP\_NUM : DBパーティション番号
- ・ CNT : レコード件数

DBPARTITIONNUM(列名)の列は、表のどれかの列を指定する  
どの列を指定しても同じ結果となる

SAMP.CUSTOMER を指定し、その表のDBパーティションごとのレコード件数を確認する

## 出力結果

DBP_NUM	CNT
0	13
1	15
2	22
3	12
4	15
5	10
6	11
7	13

CNT 列で各DBパーティションのレコード件数が確認できる

## サンプル 2

## 表単位で分散状況 (MAX、MIN、AVG) を確認する

DBパーティション番号を戻す DBPARTITIONNUM(列名) 関数を利用して、MAX、MIN、AVGの件数を確認します。

ここでは 以下 ステートメントで作成された表 (SAMP.CUSTOMER) の DBパーティションごとのレコード件数を確認します。  
SAMP.CUSTOMER表の分散キーは CUST\_ID列としています。

<表作成ステートメント>

```
CREATE TABLE SAMP.CUSTOMER(  
  CUST_ID INT,  
  NAME VARCHAR(80),  
  GENDER CHAR(5)  
)DISTRIBUTE BY HASH(CUST_ID)
```

## ステートメント

```
WITH ROW_CNT AS (  
  SELECT  
    DBPARTITIONNUM(CUST_ID) as DBP_NUM,  
    COUNT(*) AS CNT  
  FROM SAMP.CUSTOMER  
  GROUP BY DBPARTITIONNUM(CUST_ID)  
)  
SELECT  
  MAX(CNT) AS MAX_CNT,  
  MIN(CNT) AS MIN_CNT,  
  AVG(CNT) AS AVG_CNT  
FROM ROW_CNT
```

WITH句にて、各DBパーティションの件数を求めている

DBPARTITIONNUM(列名)の列は、表のどれかの列を指定する  
どの列を指定しても 同じ結果となる

SAMP.CUSTOMER を指定し、その表のDBパーティションごとのレコード件数を確認する

### ■各項目の説明

- ・ MAX\_CNT : DBパーティションごとのレコード件数の最大値
- ・ MIN\_CNT : DBパーティションごとのレコード件数の最小値
- ・ AVG\_CNT : DBパーティションごとのレコード件数の平均値

## 出力結果

MAX_CNT	MIN_CNT	AVG_CNT
22	10	13

表のデータの偏りの傾向がわかる  
MAXとMINがAVGに近ければ、最適な性能が望める



### サンプル 3

### 表スペース単位で分散状況を確認する

表スペース利用状況を返すMON\_TBSP\_UTILIZATION管理ビューを利用して、DBパーティションごとの表スペース状況を確認することができます。

#### ステートメント

```
SELECT
  SUBSTR(TBSP_NAME,1,16) AS TBSP_NAME,
  MEMBER,
  TBSP_TOTAL_SIZE_KB,
  TBSP_USABLE_SIZE_KB,
  TBSP_UTILIZATION_PERCENT
FROM SYSIBMADM.MON_TBSP_UTILIZATION
ORDER BY TBSP_NAME, MEMBER
```

#### ■各項目の説明

- ・ TBSP\_NAME : 表スペース名
- ・ MEMBER : データベース・メンバー
- ・ TBSP\_TOTAL\_SIZE\_KB : 表スペースの合計サイズ (KB)
- ・ TBSP\_USABLE\_SIZE\_KB : 使用可能な表スペースの合計サイズ (KB)
- ・ TBSP\_UTILIZATION\_PERCENT : 表スペースの使用率 (%)

#### 出力結果

TBSP_NAME	MEMBER	TBSP_TOTAL_SIZE_KB	TBSP_USABLE_SIZE_KB	TBSP_UTILIZATION_PERCENT
AUDITSPACE	0	24397824	24397696	99.9
AUDITSPACE	1	19483008	19482880	100
AUDITSPACE	2	14102016	14101888	100
AUDITSPACE	3	25210368	25210240	100
AUDITSPACE	4	15347328	15347200	99.8
AUDITSPACE	5	15714688	15714560	99.8
AUDITSPACE	6	44740096	44739968	100
AUDITSPACE	7	15674752	15674624	99.8
DSMSPACE	0	131072	130944	88.1
...				

DBパーティション番号

表スペース毎にDBパーティションのサイズを出力

## 参考資料

- 本SILでは、モニタリングの方法を解説しています。  
モニタリング情報を基にしたチューニングについては、以下の資料も参照してください。
  - Db2 Warehouseパフォーマンス・チューニングガイド  
<https://ibm.box.com/s/591ww5g0jmyqxr377p9qdc7l76pyirl>
  - IAS/Db2 Warehouse SQLチューニングガイド  
<https://ibm.box.com/s/533bu1kklt7brhhdbx7nvzvl7qysell>

EOF